



Inhaltsverzeichnis

1. Modul:Convert/sandbox	2
2. Modul:Convert/sandbox/Doku	63
3. Modul:Convert/sandbox/testcases	64



Modul:Convert/sandbox

When making a change, copy the current modules to the sandbox pages, then edit the sandbox copies.

If wanted, use the purge link just above to update the following status report.

- [Module:Convert](#) • [Module:Convert/sandbox](#) • [same content](#)
- [Module:Convert/data](#) • [Module:Convert/data/sandbox](#) • [same content](#)
- [Module:Convert/text](#) • [Module:Convert/text/sandbox](#) • [same content](#)
- [Module:Convert/extra](#) • [Module:Convert/extra/sandbox](#) • [same content](#)
- [Module:Convert/wikidata](#) • [Module:Convert/wikidata/sandbox](#) • [same content](#)
- [Module:Convert/wikidata/data](#) • [Module:Convert/wikidata/data/sandbox](#) • [same content](#)

Use the following template to test the results (example `{{convert/sandbox|123|lb|kg}}`):

- [Template:Convert/sandbox](#) • invokes the sandbox modules and shows all warnings

Testscases:

- [Module:Convert/sandbox/testcases](#) • templates to be tested, with expected outputs
- [Module talk:Convert/sandbox/testcases](#) • view test results
- [Template:Convert/testcases#Sandbox testcases](#) • more tests

It is not necessary to save a module before viewing test results. For example, [Module:Convert/sandbox](#) could be edited. While still editing that page, paste

[Vorlage:Nowrap](#)

into the page title box under "Preview page with this template", then click "Show preview".

```
-- Convert a value from one unit of measurement to another.
-- Example: {{convert|123|lb|kg}} --> 123 pounds (56 kg)
-- See [[:en:Template:Convert/Transwiki guide]] if copying to another wiki.

local MINUS = '-' -- Unicode U+2212 MINUS SIGN (UTF-8: e2 88 92)
local abs = math.abs
local floor = math.floor
local format = string.format
local log10 = math.log10
local ustring = mw.ustring
local ulen = ustring.len
local usub = ustring.sub

-- Configuration options to keep magic values in one location.
-- Conversion data and message text are defined in separate modules.
local config, maxsigfig
local numdot -- must be '.' or ',' or a character which works in a regex
local numsep, numsep_remove, numsep_remove2
local data_code, all_units
local text_code
local varname -- can be a code to use variable names that depend on value
local from_en_table -- to translate an output string of en digits to local lang
```



```
local to_en_table    -- to translate an input string of digits in local language
-- Use translation_table in convert/text to change the following.
local en_default     -- true uses lang=en unless convert has lang=local or
local group_method = 3    -- code for how many digits are in a group
local per_word = 'per'   -- for units like "liters per kilometer"
local plural_suffix = 's' -- only other useful value is probably '' to disable p
local omitsep        -- true to omit separator before local symbol/name

-- All units should be defined in the data module. However, to cater for quick cl
-- and experiments, any unknown unit is looked up in an extra data module, if it
-- That module would be transcluded in only a small number of pages, so there sh
-- little server overhead from making changes, and changes should propagate quick
local extra_module   -- name of module with extra units
local extra_units    -- nil or table of extra units from extra_module

-- Some options in the invoking template can set variables used later in the mod
local currency_text  -- for a user-defined currency symbol: {{convert|12|$/ha|$=€

local function from_en(text)
    -- Input is a string representing a number in en digits with '.' decimal
    -- without digit grouping (which is done just after calling this).
    -- Return the translation of the string with numdot and digits in local
    if numdot ~= '.' then
        text = text:gsub('%.', numdot)
    end
    if from_en_table then
        text = text:gsub('%d', from_en_table)
    end
    return text
end

local function to_en(text)
    -- Input is a string representing a number in the local language with
    -- an optional numdot decimal mark and numsep digit grouping.
    -- Return the translation of the string with '.' mark and en digits,
    -- and no separators (they have to be removed here to handle cases like
    -- numsep = '.' and numdot = ',' with input "1.234.567,8").
    if to_en_table then
        text = ustring.gsub(text, '%d', to_en_table)
    end
    if numsep_remove then
        text = text:gsub(numsep_remove, '')
    end
    if numsep_remove2 then
        text = text:gsub(numsep_remove2, '')
    end
    if numdot ~= '.' then
        text = text:gsub(numdot, '.')
    end
    return text
end

local function decimal_mark(text)
    -- Return ',' if text probably is using comma for decimal mark, or has no
    -- Return '.' if text probably is using dot for decimal mark.
    -- Otherwise return nothing (decimal mark not known).
    if not text:find('[.,]') then return '' end
    text = text:gsub('^%-', ''):gsub('%+%/+%/+$', ''):gsub('[Ee]%-?%+$', '')
    local decimal =
        text:match('^0?([.,])%d+$') or
        text:match('%d([.,])%d?%d?$') or
        text:match('%d([.,])%d%d%d+$')
    if decimal then return decimal end
    if text:match('%.%d+%.') then return ',' end
end
```

```
        if text:match('%,%d+',) then return '.' end
end

local add_warning, with_separator -- forward declarations
local function to_en_with_check(text, parms)
    -- Version of to_en() for a wiki using numdot = ',' and numsep = '.' to
    -- text (an input number as a string) which might have been copied from
    -- For example, in '1.234' the '.' could be a decimal mark or a group sep
    -- From viwiki.
    if to_en_table then
        text = ustring.gsub(text, '%d', to_en_table)
    end
    if decimal_mark(text) == '.' then
        local original = text
        text = text:gsub(',', '') -- for example, interpret "1,234.5" as
        if parms then
            add_warning(parms, 0, 'cvt_enwiki_num', original, with_se
        end
    else
        if numsep_remove then
            text = text:gsub(numsep_remove, '')
        end
        if numsep_remove2 then
            text = text:gsub(numsep_remove2, '')
        end
        if numdot ~= '.' then
            text = text:gsub(numdot, '.')
        end
    end
    end
    return text
end

local function omit_separator(id)
    -- Return true if there should be no separator before id (a unit symbol
    -- For zhwiki, there should be no separator if id uses local characters.
    -- The following kludge should be a sufficient test.
    if omitsep then
        if id:sub(1, 2) == '-{' then -- for "-{...}-" content language v
            return true
        end
        if id:byte() > 127 then
            local first = usub(id, 1, 1)
            if first ~= 'À' and first ~= '°' and first ~= 'µ' then
                return true
            end
        end
    end
    return id:sub(1, 1) == '/' -- no separator before units like "/ha"
end

local spell_module -- name of module that can spell numbers
local speller -- function from that module to handle spelling (set if need
local wikidata_module, wikidata_data_module -- names of Wikidata modules
local wikidata_code, wikidata_data -- exported tables from those modules (set if

local function set_config(args)
    -- Set configuration options from template #invoke or defaults.
    config = args
    maxsigfig = config.maxsigfig or 14 -- maximum number of significant fig
    local data_module, text_module
    local sandbox = config.sandbox and ('/' .. config.sandbox) or ''
    data_module = "Module:Convert/data" .. sandbox
    text_module = "Module:Convert/text" .. sandbox
    extra_module = "Module:Convert/extra" .. sandbox
```

```
wikidata_module = "Module:Convert/wikidata" .. sandbox
wikidata_data_module = "Module:Convert/wikidata/data" .. sandbox
spell_module = "Module:ConvertNumeric"
data_code = mw.loadData(data_module)
text_code = mw.loadData(text_module)
all_units = data_code.all_units
local translation = text_code.translation_table
if translation then
    numdot = translation.numdot
    numsep = translation.numsep
    if numdot == ',' and numsep == '.' then
        if text_code.all_messages.cvt_enwiki_num then
            to_en = to_en_with_check
        end
    end
end
if translation.group then
    group_method = translation.group
end
if translation.per_word then
    per_word = translation.per_word
end
if translation.plural_suffix then
    plural_suffix = translation.plural_suffix
end
varname = translation.varname
from_en_table = translation.from_en
local use_workaround = true
if use_workaround then
    -- 2013-07-05 workaround bug by making a copy of the request
    -- mw.usttring.gsub fails with a table (to_en_table) as the
    -- if the table is accessed via mw.loadData.
    local source = translation.to_en
    if source then
        to_en_table = {}
        for k, v in pairs(source) do
            to_en_table[k] = v
        end
    end
else
    to_en_table = translation.to_en
end
if translation.lang == 'en default' then
    en_default = true -- for hiwiki
end
omitsep = translation.omitsep -- for zhwiki
end
numdot = config.numdot or numdot or '.' -- decimal mark before fraction
numsep = config.numsep or numsep or ',' -- group separator for numbers
-- numsep should be ',' or '.' or '' or '&nbsp;' or a Unicode character.
-- numsep_remove must work in a regex to identify separators to be removed
if numsep ~= '' then
    numsep_remove = (numsep == '.') and '%.' or numsep
end
if numsep ~= ',' and numdot ~= ',' then
    numsep_remove2 = ',' -- so numbers copied from enwiki will work
end
end

local function collection()
    -- Return a table to hold items.
    return {
        n = 0,
        add = function (self, item)
            self.n = self.n + 1
        end
    }
end
```

```
                self[self.n] = item
            end,
        }
    end

    local function divide(enumerator, denominator)
        -- Return integers quotient, remainder resulting from dividing the two
        -- given numbers, which should be unsigned integers.
        local quotient, remainder = floor(enumerator / denominator), enumerator % denominator
        if not (0 <= remainder and remainder < denominator) then
            -- Floating point limits may need this, as in {{convert|160.02|Yn
            remainder = 0
        end
        return quotient, remainder
    end

    local function split(text, delimiter)
        -- Return a numbered table with fields from splitting text.
        -- The delimiter is used in a regex without escaping (for example, '.' w
        -- Each field has any leading/trailing whitespace removed.
        local t = {}
        text = text .. delimiter -- to get last item
        for item in text:gmatch('%s*(.)%s*' .. delimiter) do
            table.insert(t, item)
        end
        return t
    end

    local function strip(text)
        -- If text is a string, return its content with no leading/trailing
        -- whitespace. Otherwise return nil (a nil argument gives a nil result).
        if type(text) == 'string' then
            return text:match("^%s*(.)%s*$")
        end
    end

    local function table_len(t)
        -- Return length (<100) of a numbered table to replace #t which is
        -- documented to not work if t is accessed via mw.loadData().
        for i = 1, 100 do
            if t[i] == nil then
                return i - 1
            end
        end
    end

    local function wanted_category(catkey, catsort, want_warning)
        -- Return message category if it is wanted in current namespace,
        -- otherwise return ''.
        local cat
        local title = mw.title.getCurrentTitle()
        if title then
            local nsdefault = '0' -- default namespace: '0' = article; '0,10
            local namespace = title.namespace
            for _, v in ipairs(split(config.nscat or nsdefault, ',')) do
                if namespace == tonumber(v) then
                    cat = text_code.all_categories[want_warning and
                    if catsort and catsort ~= '' and cat:sub(-2) ==
                    cat = cat:sub(1, -3) .. '|' .. mw.text.nc
                end
                break
            end
        end
    end

end
```




```
        return parms.test == 'preview'
    end
    local success, revid = pcall(function ()
        return (parms.frame):preprocess('{{REVISIONID}}')
    end)
    return success and (revid == '')
end
local want_warning = is_warning and
    not config.warnings and -- show unobtrusive warnings if
    not msg.nowarn         -- but use msg settings, not sta
local title = string.gsub(msg[1] or 'Missing message', '%d+', 'p
local text = want_warning and '*' or msg[2] or 'Missing message'
local cat = wanted_category(msg[3], mcode[2], want_warning)
local anchor = msg[4] or ''
local fmtkey = ispreview() and 'cvt_format_preview' or
    (want_warning and 'cvt_format2' or msg.format or 'cvt_fo
local fmt = text_code.all_messages[fmtkey] or 'convert: bug'
return subparm(fmt, title:gsub('\"', '&quot;'), text, cat, anchor)
end
return 'Convert internal error: unknown message'
end
end

function add_warning(parms, level, key, text1, text2) -- for forward declaration
-- If enabled, add a warning that will be displayed after the convert res
-- A higher level is more verbose: more kinds of warnings are displayed.
-- To reduce output noise, only the first warning is displayed.
if level <= (tonumber(config.warnings) or 1) then
    if parms.warnings == nil then
        parms.warnings = message(parms, { key, text1, text2 }, t
    end
end
end

local function spell_number(parms, inout, number, numerator, denominator)
-- Return result of spelling (number, numerator, denominator), or
-- return nil if spelling is not available or not supported for given tex
-- Examples (each value must be a string or nil):
--   number  numerator  denominator  output
--   -----  -
--   "1.23"   nil          nil          one point two three
--   "1"      "2"          "3"          one and two thirds
--   nil      "2"          "3"          two thirds
if not speller then
    local function get_speller(module)
        return require(module).spell_number
    end
    local success
    success, speller = pcall(get_speller, spell_module)
    if not success or type(speller) ~= 'function' then
        add_warning(parms, 1, 'cvt_no_spell', 'spell')
        return nil
    end
end
local case
if parms.spell_upper == inout then
    case = true
    parms.spell_upper = nil -- only uppercase first word in a multip
end
local sp = not parms.opt_sp_us
local adj = parms.opt_adjectival
return speller(number, numerator, denominator, case, sp, adj)
end

-----
-- BEGIN: Code required only for built-in units.
```

```
-- LATER: If need much more code, move to another module to simplify this module
local function speed_of_sound(altitude)
  -- This is for the Mach built-in unit of speed.
  -- Return speed of sound in metres per second at given altitude in feet.
  -- If no altitude given, use default (zero altitude = sea level).
  -- Table gives speed of sound in miles per hour at various altitudes:
  -- altitude = -17,499 to 402,499 feet
  -- mach_table[a + 4] = s where
  -- a = (altitude / 5000) rounded to nearest integer (-3 to 80)
  -- s = speed of sound (mph) at that altitude
  -- LATER: Should calculate result from an interpolation between the next
  -- lower and higher altitudes in table, rather than rounding to nearest.
  -- From: http://www.aerospaceweb.org/question/atmosphere/q0112.shtml
  local mach_table = {
    799.5, 787.0, 774.2, 761.207051,
    748.0, 734.6, 721.0, 707.0, 692.8, 678.3, 663.5, 660.1, 660.1, 660.1, 660.1,
    660.1, 660.1, 660.1, 662.0, 664.3, 666.5, 668.9, 671.1, 673.4, 675.6, 677.9,
    677.9, 683.7, 689.9, 696.0, 702.1, 708.1, 714.0, 719.9, 725.8, 731.7,
    737.3, 737.7, 737.7, 736.2, 730.5, 724.6, 718.8, 712.9, 707.0, 701.1,
    695.0, 688.9, 682.8, 676.6, 670.4, 664.1, 657.8, 652.9, 648.3, 643.4,
    639.1, 634.4, 629.6, 624.8, 620.0, 615.2, 613.2, 613.2, 613.2, 613.2,
    614.4, 615.3, 616.7, 619.8, 623.4, 629.7, 635.0, 641.1, 650.6, 660.1,
    672.5, 674.3, 676.1, 677.9, 679.7, 681.5, 683.3, 685.1, 686.8, 688.6
  }
  altitude = altitude or 0
  local a = (altitude < 0) and -altitude or altitude
  a = floor(a / 5000 + 0.5)
  if altitude < 0 then
    a = -a
  end
  if a < -3 then
    a = -3
  elseif a > 80 then
    a = 80
  end
  return mach_table[a + 4] * 0.44704 -- mph converted to m/s
end
-- END: Code required only for built-in units.
-----

local function add_style(parms, class)
  -- Add selected template style to parms if not already present.
  parms.templatestyles = parms.templatestyles or {}
  if not parms.templatestyles[class] then
    parms.templatestyles[class] = parms.frame:extensionTag({
      name = 'templatestyles', args = { src = text_code.titles
    })
  end
end

local function get_styles(parms)
  -- Return string of required template styles, empty if none.
  if parms.templatestyles then
    local t = {}
    for _, v in pairs(parms.templatestyles) do
      table.insert(t, v)
    end
    return table.concat(t)
  end
  return ''
end

local function get_range(word)
  -- Return a range (string or table) corresponding to word (like "to"),
```



```
-- or return nil if not a range word.
local ranges = text_code.ranges
return ranges.types[word] or ranges.types[ranges.aliases[word]]
end

local function check_mismatch(unit1, unit2)
  -- If unit1 cannot be converted to unit2, return an error message table.
  -- This allows conversion between units of the same type, and between
  -- Nm (normally torque) and ftlb (energy), as in gun-related articles.
  -- This works because Nm is the base unit (scale = 1) for both the
  -- primary type (torque), and the alternate type (energy, where Nm = J).
  -- A match occurs if the primary types are the same, or if unit1 matches
  -- the alternate type of unit2, and vice versa. That provides a whitelist
  -- of which conversions are permitted between normally incompatible types
  if unit1.utype == unit2.utype or
      (unit1.utype == unit2.alttype and unit1.alttype == unit2.utype) then
    return nil
  end
  return { 'cvt_mismatch', unit1.utype, unit2.utype }
end

local function override_from(out_table, in_table, fields)
  -- Copy the specified fields from in_table to out_table, but do not
  -- copy nil fields (keep any corresponding field in out_table).
  for _, field in ipairs(fields) do
    if in_table[field] then
      out_table[field] = in_table[field]
    end
  end
end

local function shallow_copy(t)
  -- Return a shallow copy of table t.
  -- Do not need the features and overhead of the Scribunto mw.clone().
  local result = {}
  for k, v in pairs(t) do
    result[k] = v
  end
  return result
end

local unit_mt = {
  -- Metatable to get missing values for a unit that does not accept SI pre
  -- Warning: The boolean value 'false' is returned for any missing field
  -- so __index is not called twice for the same field in a given unit.
  __index = function(self, key)
    local value
    if key == 'name1' or key == 'sym_us' then
      value = self.symbol
    elseif key == 'name2' then
      value = self.name1 .. plural_suffix
    elseif key == 'name1_us' then
      value = self.name1
      if not rawget(self, 'name2_us') then
        -- If name1_us is 'foot', do not make name2_us by
        self.name2_us = self.name2
      end
    elseif key == 'name2_us' then
      local raw1_us = rawget(self, 'name1_us')
      if raw1_us then
        value = raw1_us .. plural_suffix
      else
        value = self.name2
      end
    end
  end
}
```

```
        elseif key == 'link' then
            value = self.name1
        else
            value = false
        end
        rawset(self, key, value)
        return value
    end
}

local function prefixed_name(unit, name, index)
    -- Return unit name with SI prefix inserted at correct position.
    -- index = 1 (name1), 2 (name2), 3 (name1_us), 4 (name2_us).
    -- The position is a byte (not character) index, so use Lua's sub().
    local pos = rawget(unit, 'prefix_position')
    if type(pos) == 'string' then
        pos = tonumber(split(pos, ',')[index])
    end
    if pos then
        return name:sub(1, pos - 1) .. unit.si_name .. name:sub(pos)
    end
    return unit.si_name .. name
end

local unit_prefixes_mt = {
    -- Metatable to get missing values for a unit that accepts SI prefixes.
    -- Before use, fields si_name, si_prefix must be defined.
    -- The unit must define _symbol, _name1 and
    -- may define _sym_us, _name1_us, _name2_us
    -- (_sym_us, _name2_us may be defined for a language using sp=us
    -- to refer to a variant unrelated to U.S. units).
    __index = function (self, key)
        local value
        if key == 'symbol' then
            value = self.si_prefix .. self._symbol
        elseif key == 'sym_us' then
            value = rawget(self, '_sym_us')
            if value then
                value = self.si_prefix .. value
            else
                value = self.symbol
            end
        elseif key == 'name1' then
            value = prefixed_name(self, self._name1, 1)
        elseif key == 'name2' then
            value = rawget(self, '_name2')
            if value then
                value = prefixed_name(self, value, 2)
            else
                value = self.name1 .. plural_suffix
            end
        elseif key == 'name1_us' then
            value = rawget(self, '_name1_us')
            if value then
                value = prefixed_name(self, value, 3)
            else
                value = self.name1
            end
        elseif key == 'name2_us' then
            value = rawget(self, '_name2_us')
            if value then
                value = prefixed_name(self, value, 4)
            elseif rawget(self, '_name1_us') then
                value = self.name1_us .. plural_suffix
            end
        end
    end
}
```

```
        else
            value = self.name2
        end
    elseif key == 'link' then
        value = self.name1
    else
        value = false
    end
    rawset(self, key, value)
    return value
end
}

local unit_per_mt = {
    -- Metatable to get values for a per unit of form "x/y".
    -- This is never called to determine a unit name or link because per unit
    -- are handled as a special case.
    -- Similarly, the default output is handled elsewhere, and for a symbol
    -- this is only called from get_default() for default_exceptions.
    __index = function (self, key)
        local value
        if key == 'symbol' then
            local per = self.per
            local unit1, unit2 = per[1], per[2]
            if unit1 then
                value = unit1[key] .. '/' .. unit2[key]
            else
                value = '/' .. unit2[key]
            end
        elseif key == 'sym_us' then
            value = self.symbol
        elseif key == 'scale' then
            local per = self.per
            local unit1, unit2 = per[1], per[2]
            value = (unit1 and unit1.scale or 1) * self.scalemultipl
        else
            value = false
        end
        rawset(self, key, value)
        return value
    end
}

local function make_per(unitcode, unit_table, ulookup)
    -- Return true, t where t is a per unit with unit codes expanded to unit
    -- or return false, t where t is an error message table.
    local result = {
        unitcode = unitcode,
        utype = unit_table.utype,
        per = {}
    }
    override_from(result, unit_table, { 'invert', 'iscomplex', 'default', 'l
    result.symbol_raw = (result.symbol or false) -- to distinguish between a
    local prefix
    for i, v in ipairs(unit_table.per) do
        if i == 1 and v == '' then
            -- First unit symbol can be empty; that gives a nil first
        elseif i == 1 and text_code.currency[v] then
            prefix = currency_text or v
        else
            local success, t = ulookup(v)
            if not success then return false, t end
            result.per[i] = t
        end
    end
end
```

```
end
local multiplier = unit_table.multiplier
if not result.utype then
  -- Creating an automatic per unit.
  local unit1 = result.per[1]
  local utype = (unit1 and unit1.utype or prefix or '') .. '/' .. unit1
  local t = data_code.per_unit_fixups[utype]
  if t then
    if type(t) == 'table' then
      utype = t.utype or utype
      result.link = result.link or t.link
      multiplier = multiplier or t.multiplier
    else
      utype = t
    end
  end
  result.utype = utype
end
result.scalemultiplier = multiplier or 1
result.vprefix = prefix or false -- set to non-nil to avoid calling __infix
return true, setmetatable(result, unit_per_mt)
end

local function lookup(parms, unitcode, what, utable, fails, depth)
  -- Return true, t where t is a copy of the unit's converter table,
  -- or return false, t where t is an error message table.
  -- Parameter 'what' determines whether combination units are accepted:
  -- 'no_combination' : single unit only
  -- 'any_combination' : single unit or combination or output multiple
  -- 'only_multiple' : single unit or output multiple only
  -- Parameter unitcode is a symbol (like 'g'), with an optional SI prefix
  -- If, for example, 'kg' is in this table, that entry is used;
  -- otherwise the prefix ('k') is applied to the base unit ('g').
  -- If unitcode is a known combination code (and if allowed by what),
  -- a table of output multiple unit tables is included in the result.
  -- For compatibility with the old template, an underscore in a unitcode is
  -- replaced with a space so usage like {{convert|350|board_feet}} works.
  -- Wikignomes may also put two spaces or "&nbsp;" in combinations, so
  -- replace underscore, "&nbsp;", and multiple spaces with a single space
  utable = utable or parms.unittable or all_units
  fails = fails or {}
  depth = depth and depth + 1 or 1
  if depth > 9 then
    -- There are ways to mistakenly define units which result in infinite
    -- recursion when lookup() is called. That gives a long delay and
    -- confusing error messages, so the depth parameter is used as a
    return false, { 'cvt_lookup', unitcode }
  end
  if unitcode == nil or unitcode == '' then
    return false, { 'cvt_no_unit' }
  end
  unitcode = unitcode:gsub('_', ' '):gsub('&nbsp;', ' '):gsub(' +', ' ')
  local function call_make_per(t)
    return make_per(unitcode, t,
      function (ucode) return lookup(parms, ucode, 'no_combination',
        utable, fails, depth)
      )
  end
  local t = utable[unitcode]
  if t then
    if t.shouldbe then
      return false, { 'cvt_should_be', t.shouldbe }
    end
    if t.sp_us then
      parms.opt_sp_us = true
    end
  end
end
```



```
end
local target = t.target -- nil, or unitcode is an alias for this
if target then
    local success, result = lookup(parms, target, what, utable)
    if not success then return false, result end
    override_from(result, t, { 'customary', 'default', 'link' })
    local multiplier = t.multiplier
    if multiplier then
        result.multiplier = tostring(multiplier)
        result.scale = result.scale * multiplier
    end
    return true, result
end
if t.per then
    return call_make_per(t)
end
local combo = t.combination -- nil or a table of unitcodes
if combo then
    local multiple = t.multiple
    if what == 'no_combination' or (what == 'only_multiple' and
        return false, { 'cvt_bad_unit', unitcode }
    end
    -- Recursively create a combination table containing the
    -- converter table of each unitcode.
    local result = { utype = t.utype, multiple = multiple, combination = {} }
    local cvt = result.combination
    for i, v in ipairs(combo) do
        local success, t = lookup(parms, v, multiple and
            if not success then return false, t end
            cvt[i] = t
        end
        return true, result
    end
    local result = shallow_copy(t)
    result.unitcode = unitcode
    if result.prefixes then
        result.si_name = ''
        result.si_prefix = ''
        return true, setmetatable(result, unit_prefixed_mt)
    end
    return true, setmetatable(result, unit_mt)
end
local SIprefixes = text_code.SIprefixes
for plen = SIprefixes[1] or 2, 1, -1 do
    -- Look for an SI prefix; should never occur with an alias.
    -- Check for longer prefix first ('dam' is decametre).
    -- SIprefixes[1] = prefix maximum #characters (as seen by mw.ust)
    local prefix = usub(unitcode, 1, plen)
    local si = SIprefixes[prefix]
    if si then
        local t = utable[usub(unitcode, plen+1)]
        if t and t.prefixes then
            local result = shallow_copy(t)
            result.unitcode = unitcode
            result.si_name = parms.opt_sp_us and si.name_us or si.name
            result.si_prefix = si.prefix or prefix
            result.scale = t.scale * 10 ^ (si.exponent * t.prefix)
            return true, setmetatable(result, unit_prefixed_mt)
        end
    end
end
end
-- Accept user-defined combinations like "acre+m2+ha" or "acre m2 ha" for
-- If '+' is used, each unit code can include a space, and any error is 1
-- If ' ' is used and if each space-separated word is a unit code, it is
```

```
-- but errors are not fatal so the unit code can be looked up as an extra
local err_is_fatal
local combo = collection()
if unitcode:find('+', 1, true) then
    err_is_fatal = true
    for item in (unitcode .. '+'):gmatch('%s*(.)%s*%+') do
        if item ~= '' then
            combo:add(item)
        end
    end
end
elseif unitcode:find('%s') then
    for item in unitcode:gmatch('%S+') do
        combo:add(item)
    end
end
end
if combo.n > 1 then
    local function lookup_combo()
        if what == 'no_combination' or what == 'only_multiple' then
            return false, { 'cvt_bad_unit', unitcode }
        end
        local result = { combination = {} }
        local cvt = result.combination
        for i, v in ipairs(combo) do
            local success, t = lookup(parms, v, 'only_multiple')
            if not success then return false, t end
            if i == 1 then
                result.utype = t.utype
            else
                local mismatch = check_mismatch(result, t)
                if mismatch then
                    return false, mismatch
                end
            end
            cvt[i] = t
        end
        return true, result
    end
    local success, result = lookup_combo()
    if success or err_is_fatal then
        return success, result
    end
end
end
-- Accept any unit with an engineering notation prefix like "e6cuft"
-- (million cubic feet), but not chained prefixes like "e3e6cuft",
-- and not if the unit is a combination or multiple,
-- and not if the unit has an offset or is a built-in.
-- Only en digits are accepted.
local exponent, baseunit = unitcode:match('^e(%d+)(.*)')
if exponent then
    local engscale = text_code.eng_scales[exponent]
    if engscale then
        local success, result = lookup(parms, baseunit, 'no_combination')
        if success and not (result.offset or result.builtin or result.builtup) then
            result.unitcode = unitcode -- 'e6cuft' not 'cuft'
            result.defkey = unitcode -- key to lookup default
            result.engscale = engscale
            result.scale = result.scale * 10 ^ tonumber(exponent)
            return true, result
        end
    end
end
end
end
-- Look for x/y; split on right-most slash to get scale correct (x/y/z is
local top, bottom = unitcode:match('^(.)/([/]+)$')
if top and not unitcode:find('e%d') then
```

```
-- If valid, create an automatic per unit for an "x/y" unit code
-- The unitcode must not include extraneous spaces.
-- Engineering notation (apart from at start and which has been s
-- is not supported so do not make a per unit if find text like
local success, result = call_make_per({ per = {top, bottom} })
if success then
    return true, result
end
end
if not parms.opt_ignore_error and not get_range(unitcode) then
    -- Want the "what links here" list for the extra_module to show c
    -- where an extra unit is used, so do not require it if invoked f
    -- or if looking up a range word which cannot be a unit.
    if not extra_units then
        local success, extra = pcall(function () return require(
            if success and type(extra) == 'table' then
                extra_units = extra
            end
        end
    end
    if extra_units then
        -- A unit in one data table might refer to a unit in the
        -- switch between them, relying on fails or depth to term
        if not fails[unitcode] then
            fails[unitcode] = true
            local other = (utable == all_units) and extra_uni
            local success, result = lookup(parms, unitcode, v
            if success then
                return true, result
            end
        end
    end
end
if to_en_table then
    -- At fawiki it is common to translate all digits so a unit like
    local en_code = ustring.gsub(unitcode, '%d', to_en_table)
    if en_code ~= unitcode then
        return lookup(parms, en_code, what, utable, fails, depth)
    end
end
return false, { 'cvt_unknown', unitcode }
end

local function valid_number(num)
    -- Return true if num is a valid number.
    -- In Scribunto (different from some standard Lua), when expressed as a s
    -- overflow or other problems are indicated with text like "inf" or "nan"
    -- which are regarded as invalid here (each contains "n").
    if type(num) == 'number' and tostring(num):find('n', 1, true) == nil then
        return true
    end
end

local function hyphenated(name, parts)
    -- Return a hyphenated form of given name (for adjectival usage).
    -- The name may be linked and the target of the link must not be changed
    -- Hypothetical examples:
    -- [[long ton|ton]] → [[long ton|ton]] (no change)
    -- [[tonne|long ton]] → [[tonne|long-ton]]
    -- [[metric ton|long ton]] → [[metric ton|long-ton]]
    -- [[long ton]] → [[long ton|long-ton]]
    -- Input can also have multiple links in a single name like:
    -- [[United States customary units|U.S.]] [[US gallon|gallon]]
    -- [[mile]]s per [[United States customary units|U.S.]] [[quart]]
    -- [[long ton]]s per [[short ton]]
```

```
-- Assume that links cannot be nested (never like "[[abc[[def]]ghi]]").
-- This uses a simple and efficient procedure that works for most cases.
-- Some units (if used) would require more, and can later think about
-- adding a method to handle exceptions.
-- The procedure is to replace each space with a hyphen, but
-- not a space after ')' [for "(pre-1954&nbsp;US) nautical mile"], and
-- not spaces immediately before '(' or in '(...)' [for cases like
-- "British thermal unit (ISO)" and "Calorie (International Steam Table)"]
if name:find(' ', 1, true) then
    if parts then
        local pos
        if name:sub(1, 1) == '(' then
            pos = name:find(')', 1, true)
            if pos then
                return name:sub(1, pos+1) .. name:sub(pos+1)
            end
        elseif name:sub(-1) == ')' then
            pos = name:find('(', 1, true)
            if pos then
                return name:sub(1, pos-2):gsub(' ', '-')
            end
        end
        return name:gsub(' ', '-')
    end
    parts = collection()
    for before, item, after in name:gmatch('([^\]]*)(%[[^\]]*%])') do
        if item:find(' ', 1, true) then
            local prefix
            local plen = item:find('|', 1, true)
            if plen then
                prefix = item:sub(1, plen)
                item = item:sub(plen + 1, -3)
            else
                prefix = item:sub(1, -3) .. '|'
                item = item:sub(3, -3)
            end
            item = prefix .. hyphenated(item, parts) .. ']]'
        end
        parts:add(before:gsub(' ', '-') .. item .. after:gsub(' ', '-'))
    end
    if parts.n == 0 then
        -- No link like "[[...]]" was found in the original name
        parts:add(hyphenated(name, parts))
    end
    return table.concat(parts)
end
return name
end

local function hyphenated_maybe(parms, want_name, sep, id, inout)
    -- Return s, f where
    --   s = id, possibly modified
    --   f = true if hyphenated
    -- Possible modifications: hyphenate; prepend '-'; append mid text.
    if id == nil or id == '' then
        return ''
    end
    local mid = (inout == (parms.opt_flip and 'out' or 'in')) and parms.mid
    if want_name then
        if parms.opt_adjectival then
            return '-' .. hyphenated(id) .. mid, true
        end
        if parms.opt_add_s and id:sub(-1) ~= 's' then
            id = id .. 's' -- for nowiki
        end
    end
end
```

```
        end
    end
    return sep .. id .. mid
end

local function use_minus(text)
    -- Return text with Unicode minus instead of '-', if present.
    if text:sub(1, 1) == '-' then
        return MINUS .. text:sub(2)
    end
    return text
end

local function digit_groups(parms, text, method)
    -- Return a numbered table of groups of digits (left-to-right, in local
    -- Parameter method is a number or nil:
    --   3 for 3-digit grouping (default), or
    --   2 for 3-then-2 grouping (only for digits before decimal mark).
    local len_right
    local len_left = text:find('.', 1, true)
    if len_left then
        len_right = #text - len_left
        len_left = len_left - 1
    else
        len_left = #text
    end
    local twos = method == 2 and len_left > 5
    local groups = collection()
    local run = len_left
    local n
    if run < 4 or (run == 4 and parms.opt_comma5) then
        if parms.opt_gaps then
            n = run
        else
            n = #text
        end
    elseif twos then
        n = run % 2 == 0 and 1 or 2
    else
        n = run % 3 == 0 and 3 or run % 3
    end
    while run > 0 do
        groups:add(n)
        run = run - n
        n = (twos and run > 3) and 2 or 3
    end
    if len_right then
        if groups.n == 0 then
            groups:add(0)
        end
        if parms.opt_gaps and len_right > 3 then
            local want4 = not parms.opt_gaps3 -- true gives no gap
            local isfirst = true
            run = len_right
            while run > 0 do
                n = (want4 and run == 4) and 4 or (run > 3 and 3)
                if isfirst then
                    isfirst = false
                    groups[groups.n] = groups[groups.n] + 1
                else
                    groups:add(n)
                end
                run = run - n
            end
        end
    end
end
```

```
        else
            groups[groups.n] = groups[groups.n] + 1 + len_right
        end
    end
end
local pos = 1
for i, length in ipairs(groups) do
    groups[i] = from_en(text:sub(pos, pos + length - 1))
    pos = pos + length
end
return groups
end

function with_separator(parms, text) -- for forward declaration above
-- Input text is a number in en digits with optional '.' decimal mark.
-- Return an equivalent, formatted for display:
--   with a custom decimal mark instead of '.', if wanted
--   with thousand separators inserted, if wanted
--   digits in local language
-- The given text is like '123' or '123.' or '12345.6789'.
-- The text has no sign (caller inserts that later, if necessary).
-- When using gaps, they are inserted before and after the decimal mark.
-- Separators are inserted only before the decimal mark.
-- A trailing dot (as in '123.') is removed because their use appears to
-- be accidental, and such a number should be shown as '123' or '123.0'.
-- It is useful for convert to suppress the dot so, for example, '4000.'
-- is a simple way of indicating that all the digits are significant.
if text:sub(-1) == '.' then
    text = text:sub(1, -2)
end
if #text < 4 or parms.opt_nocomma or numsep == '' then
    return from_en(text)
end
local groups = digit_groups(parms, text, group_method)
if parms.opt_gaps then
    if groups.n <= 1 then
        return groups[1] or ''
    end
    local nowrap = '<span style="white-space: nowrap">'
    local gap = '<span style="margin-left: 0.25em">'
    local close = '</span>'
    return nowrap .. groups[1] .. gap .. table.concat(groups, close)
end
return table.concat(groups, numsep)
end

-- An input value like 1.23e12 is displayed using scientific notation (1.23×1012)
-- That also makes the output use scientific notation, except for small values.
-- In addition, very small or very large output values use scientific notation.
-- Use format(fmtpower, significand, '10', exponent) where each argument is a string
local fmtpower = '%s<span style="margin:0 .15em 0 .25em">×</span>%s<sup>%s</sup>'

local function with_exponent(parms, show, exponent)
-- Return wikitext to display the implied value in scientific notation.
-- Input uses en digits; output uses digits in local language.
return format(fmtpower, with_separator(parms, show), from_en('10'), use_
end

local function make_sigfig(value, sigfig)
-- Return show, exponent that are equivalent to the result of
-- converting the number 'value' (where value >= 0) to a string,
-- rounded to 'sigfig' significant figures.
-- The returned items are:
--   show: a string of digits; no sign and no dot;
--   there is an implied dot before show.
```

```
-- exponent: a number (an integer) to shift the implied dot.
-- Resulting value = tonumber('.') .. show) * 10^exponent.
-- Examples:
-- make_sigfig(23.456, 3) returns '235', 2 (.235 * 10^2).
-- make_sigfig(0.0023456, 3) returns '235', -2 (.235 * 10^-2).
-- make_sigfig(0, 3) returns '000', 1 (.000 * 10^1).
if sigfig <= 0 then
    sigfig = 1
elseif sigfig > maxsigfig then
    sigfig = maxsigfig
end
if value == 0 then
    return string.rep('0', sigfig), 1
end
local exp, fracpart = math.modf(log10(value))
if fracpart >= 0 then
    fracpart = fracpart - 1
    exp = exp + 1
end
local digits = format('%.*f', 10^(fracpart + sigfig))
if #digits > sigfig then
    -- Overflow (for sigfig=3: like 0.9999 rounding to "1000"; need
    digits = digits:sub(1, sigfig)
    exp = exp + 1
end
assert(#digits == sigfig, 'Bug: rounded number has wrong length')
return digits, exp
end

-- Fraction output format.
local fracfmt = {
    { -- Like {{frac}} (fraction slash).
        '<span class="frac" role="math">{SIGN}<span class="num">{NUM}</span></span>',
        '<span class="frac" role="math">{SIGN}{WHOLE}<span class="sr-only" style = "font-size: 0.8em; vertical-align: middle; margin-left: 0.1em; margin-right: 0.1em;">/</span>',
        style = 'frac',
    },
    { -- Like {{sfrac}} (stacked fraction, that is, horizontal bar).
        '<span class="sfrac" role="math">{SIGN}<span class="num">{NUM}</span></span>',
        '<span class="sfrac" role="math">{SIGN}{WHOLE}<span class="sr-only" style = "font-size: 0.8em; vertical-align: middle; margin-left: 0.1em; margin-right: 0.1em;">/</span>',
        style = 'sfrac',
    },
}

local function format_fraction(parms, inout, negative, wholestr, numstr, denstr,
-- Return wikitext for a fraction, possibly spelled.
-- Inputs use en digits and have no sign; output uses digits in local language
local wikitext
if not style then
    style = parms.opt_fraction_horizontal and 2 or 1
end
if wholestr == '' then
    wholestr = nil
end
local substitute = {
    SIGN = negative and MINUS or '',
    WHOLE = wholestr and with_separator(parms, wholestr),
    NUM = from_en(numstr),
    DEN = from_en(denstr),
}
wikitext = fracfmt[style][wholestr and 2 or 1]:gsub('{{%u+}}', substitute)
if do_spell then
    if negative then
        if wholestr then
            wholestr = '-' .. wholestr
        end
    end
end
end
```

```

        else
            numstr = '-' .. numstr
        end
    end
    local s = spell_number(parms, inout, wholestr, numstr, denstr)
    if s then
        return s
    end
end
add_style(parms, fracfmt[style].style)
return wikitext
end

local function format_number(parms, show, exponent, isnegative)
    -- Parameter show is a string or a table containing strings.
    -- Each string is a formatted number in en digits and optional '.' decimal
    -- A table represents a fraction: integer, numerator, denominator;
    -- if a table is given, exponent must be nil.
    -- Return t where t is a table with fields:
    --   show = wikitext formatted to display implied value
    --         (digits in local language)
    --   is_scientific = true if show uses scientific notation
    --   clean = unformatted show (possibly adjusted and with inserted '.')
    --         (en digits)
    --   sign = '' or MINUS
    --   exponent = exponent (possibly adjusted)
    -- The clean and exponent fields can be used to calculate the
    -- rounded absolute value, if needed.
    --
    -- The value implied by the arguments is found from:
    --   exponent is nil; and
    --   show is a string of digits (no sign), with an optional dot;
    --   show = '123.4' is value 123.4, '1234' is value 1234.0;
    -- or:
    --   exponent is an integer indicating where dot should be;
    --   show is a string of digits (no sign and no dot);
    --   there is an implied dot before show;
    --   show does not start with '0';
    --   show = '1234', exponent = 3 is value 0.1234*10^3 = 123.4.
    --
    -- The formatted result:
    -- * Is for an output value and is spelled if wanted and possible.
    -- * Includes a Unicode minus if isnegative and not spelled.
    -- * Uses a custom decimal mark, if wanted.
    -- * Has digits grouped where necessary, if wanted.
    -- * Uses scientific notation if requested, or for very small or large values
    --   (which forces result to not be spelled).
    -- * Has no more than maxsigfig significant digits
    --   (same as old template and {{#expr}}).
    local xhi, xlo -- these control when scientific notation (exponent) is used
    if parms.opt_scientific then
        xhi, xlo = 4, 2 -- default for output if input uses e-notation
    elseif parms.opt_scientific_always then
        xhi, xlo = 0, 0 -- always use scientific notation (experimental)
    else
        xhi, xlo = 10, 4 -- default
    end
    local sign = isnegative and MINUS or ''
    local maxlen = maxsigfig
    local tfrac
    if type(show) == 'table' then
        tfrac = show
        show = tfrac.wholestr
        assert(exponent == nil, 'Bug: exponent given with fraction')
    end
end
```

```

end
if not tfrac and not exponent then
    local integer, dot, decimals = show:match('^(%d*)(%?.?)(.*)')
    if integer == '0' or integer == '' then
        local zeros, figs = decimals:match('^([0]*)([^\0]?.*')
        if #figs == 0 then
            if #zeros > maxlen then
                show = '0.' .. zeros:sub(1, maxlen)
            end
        elseif #zeros >= xlo then
            show = figs
            exponent = -#zeros
        elseif #figs > maxlen then
            show = '0.' .. zeros .. figs:sub(1, maxlen)
        end
    elseif #integer >= xhi then
        show = integer .. decimals
        exponent = #integer
    else
        maxlen = maxlen + #dot
        if #show > maxlen then
            show = show:sub(1, maxlen)
        end
    end
end
end
if exponent then
    local function zeros(n)
        return string.rep('0', n)
    end
    if #show > maxlen then
        show = show:sub(1, maxlen)
    end
    if exponent > xhi or exponent <= -xlo or (exponent == xhi and show:sub(1, 1) == '0') then
        -- When xhi, xlo = 10, 4 (the default), scientific notation
        -- rounded value satisfies: value >= 1e9 or value < 1e-4
        -- except if show is '1000000000' (1e9), for example:
        -- {{convert|1000000000|m|m|sigfig=10}} → 1,000,000,000 m
        local significand
        if #show > 1 then
            significand = show:sub(1, 1) .. '.' .. show:sub(2, #show)
        else
            significand = show
        end
        return {
            clean = '.' .. show,
            exponent = exponent,
            sign = sign,
            show = sign .. with_exponent(parms, significand,
            is_scientific = true,
        }
    end
    if exponent >= #show then
        show = show .. zeros(exponent - #show) -- result has no
    elseif exponent <= 0 then
        show = '0.' .. zeros(-exponent) .. show
    else
        show = show:sub(1, exponent) .. '.' .. show:sub(exponent+1, #show)
    end
end
end
local formatted_show
if tfrac then
    show = tostring(tfrac.value) -- to set clean in returned table
    formatted_show = format_fraction(parms, 'out', isnegative, tfrac)
else

```

```
        if isnegative and show:match('^0.?0*$') then
            sign = '' -- don't show minus if result is negative but
        end
        formatted_show = sign .. with_separator(parms, show)
        if parms.opt_spell_out then
            formatted_show = spell_number(parms, 'out', sign .. show)
        end
    end
    return {
        clean = show,
        sign = sign,
        show = formatted_show,
        is_scientific = false, -- to avoid calling __index
    }
end

local function extract_fraction(parms, text, negative)
    -- If text represents a fraction, return
    -- value, altvalue, show, denominator
    -- where
    -- value is a number (value of the fraction in argument text)
    -- altvalue is an alternate interpretation of any fraction for the hands
    -- unit where "12.1+3/4" means 12 hands 1.75 inches
    -- show is a string (formatted text for display of an input value,
    -- and is spelled if wanted and possible)
    -- denominator is value of the denominator in the fraction
    -- Otherwise, return nil.
    -- Input uses en digits and '.' decimal mark (input has been translated)
    -- Output uses digits in local language and local decimal mark, if any.
    -----
    -- Originally this function accepted x+y/z where x, y, z were any valid
    -- numbers, possibly with a sign. For example '1.23e+2+1.2/2.4' = 123.5,
    -- and '2-3/8' = 1.625. However, such usages were found to be errors or
    -- misunderstandings, so since August 2014 the following restrictions apply:
    -- x (if present) is an integer or has a single digit after decimal mark
    -- y and z are unsigned integers
    -- e-notation is not accepted
    -- The overall number can start with '+' or '-' (so '12+3/4' and '+12+3/4'
    -- and '-12-3/4' are valid).
    -- Any leading negative sign is removed by the caller, so only inputs
    -- like the following are accepted here (may have whitespace):
    -- negative = false      false      true (there was a leading '-')
    -- text      = '2/3'      '+2/3'      '2/3'
    -- text      = '1+2/3'    '+1+2/3'    '1-2/3'
    -- text      = '12.3+1/2' '+12.3+1/2' '12.3-1/2'
    -- Values like '12.3+1/2' are accepted, but are intended only for use
    -- with the hands unit (not worth adding code to enforce that).
    -----
    local leading_plus, prefix, numstr, slashes, denstr =
        text:match('^%s*(%+?)%s*(.-)%s*(%d+)%s*(/)%s*(%d+)%s*$')
    if not leading_plus then
        -- Accept a single U+2044 fraction slash because that may be past
        leading_plus, prefix, numstr, denstr =
            text:match('^%s*(%+?)%s*(.-)%s*(%d+)%s*%s*(%d+)%s*$')
        slashes = '/'
    end
    local numerator = tonumber(numstr)
    local denominator = tonumber(denstr)
    if numerator == nil or denominator == nil or (negative and leading_plus) then
        return nil
    end
    local whole, wholestr
    if prefix == '' then
        wholestr = ''
    end
end
```

```

    whole = 0
else
    -- Any prefix must be like '12+' or '12-' (whole number and fract
    -- '12.3+' and '12.3-' are also accepted (single digit after deci
    -- because '12.3+1/2 hands' is valid (12 hands 3½ inches).
    local num1, num2, frac_sign = prefix:match('^(%d+)(%.?%d?)%s*([+%-])')
    if num1 == nil then return nil end
    if num2 == '' then -- num2 must be '' or like '.1' but not '.' end
        wholestr = num1
    else
        if #num2 ~= 2 then return nil end
        wholestr = num1 .. num2
    end
    if frac_sign ~= (negative and '-' or '+') then return nil end
    whole = tonumber(wholestr)
    if whole == nil then return nil end
end
local value = whole + numerator / denominator
if not valid_number(value) then return nil end
local altvalue = whole + numerator / (denominator * 10)
local style = #slashes -- kludge: 1 or 2 slashes can be used to select s
if style > 2 then style = 2 end
local wikitext = format_fraction(parms, 'in', negative, leading_plus .. v
return value, altvalue, wikitext, denominator
end

local function extract_number(parms, text, another, no_fraction)
    -- Return true, info if can extract a number from text,
    -- where info is a table with the result,
    -- or return false, t where t is an error message table.
    -- Input can use en digits or digits in local language and can
    -- have references at the end. Accepting references is intended
    -- for use in infoboxes with a field for a value passed to convert.
    -- Parameter another = true if the expected value is not the first.
    -- Before processing, the input text is cleaned:
    -- * Any thousand separators (valid or not) are removed.
    -- * Any sign is replaced with '-' (if negative) or '' (otherwise).
    -- That replaces Unicode minus with '-'.
    -- If successful, the returned info table contains named fields:
    -- value = a valid number
    -- altvalue = a valid number, usually same as value but different
    -- if fraction used (for hands unit)
    -- singular = true if value is 1 or -1 (to use singular form of units)
    -- clean = cleaned text with any separators and sign removed
    -- (en digits and '.' decimal mark)
    -- show = text formatted for output, possibly with ref strip marker
    -- (digits in local language and custom decimal mark)
    -- The resulting show:
    -- * Is for an input value and is spelled if wanted and possible.
    -- * Has a rounded value, if wanted.
    -- * Has digits grouped where necessary, if wanted.
    -- * If negative, a Unicode minus is used; otherwise the sign is
    -- '+' (if the input text used '+'), or is '' (if no sign in input).
    text = strip(text or '')
    local reference
    local pos = text:find('\127', 1, true)
    if pos then
        local before = text:sub(1, pos - 1)
        local remainder = text:sub(pos)
        local refs = {}
        while #remainder > 0 do
            local ref, spaces
            ref, spaces, remainder = remainder:match('^(\\127[^\127]*t
            if ref then

```

```
                table.insert(refs, ref)
            else
                refs = {}
                break
            end
        end
        if #refs > 0 then
            text = strip(before)
            reference = table.concat(refs)
        end
    end
    local clean = to_en(text, parms)
    if clean == '' then
        return false, { another and 'cvt_no_num2' or 'cvt_no_num' }
    end
    local isnegative, propersign = false, '' -- most common case
    local singular, show, denominator
    local value = tonumber(clean)
    local altvalue
    if value then
        local sign = clean:sub(1, 1)
        if sign == '+' or sign == '-' then
            propersign = (sign == '+') and '+' or MINUS
            clean = clean:sub(2)
        end
        if value < 0 then
            isnegative = true
            value = -value
        end
    else
        local valstr
        for _, prefix in ipairs({ '-', MINUS, '&minus;' }) do
            -- Including '-' sets isnegative in case input is a fract
            local plen = #prefix
            if clean:sub(1, plen) == prefix then
                valstr = clean:sub(plen + 1)
                if valstr:match('^%s') then -- "- 1" is invalid
                    return false, { 'cvt_bad_num', text }
                end
                break
            end
        end
        if valstr then
            isnegative = true
            propersign = MINUS
            clean = valstr
            value = tonumber(clean)
        end
        if value == nil then
            if not no_fraction then
                value, altvalue, show, denominator = extract_fra
            end
            if value == nil then
                return false, { 'cvt_bad_num', text }
            end
            if value <= 1 then
                singular = true -- for example, "½ mile" or "one
            end
        end
    end
    if not valid_number(value) then -- for example, "1e310" may overflow
        return false, { 'cvt_invalid_num' }
    end
    if show == nil then
```

```
-- clean is a non-empty string with no spaces, and does not represent
-- and value = tonumber(clean) is a number >= 0.
-- If the input uses e-notation, show will be displayed using a
-- we use the number as given so it might not be normalized scientific
-- The input value is spelled if specified so any e-notation is ignored
-- that allows input like 2e6 to be spelled as "two million" which
-- because the spell module converts '2e6' to '2000000' before spelling
local function rounded(value, default, exponent)
    local precision = parms.opt_precision
    if precision then
        local fmt = '%.' .. format('%d', precision) .. 'f'
        local result = fmt:format(tonumber(value) + 2e-14)
        if not exponent then
            singular = (tonumber(result) == 1)
        end
        return result
    end
    return default
end
singular = (value == 1)
local scientific
local significand, exponent = clean:match('^([%d.]+)[Ee]([+%-]?%d)')
if significand then
    show = with_exponent(parms, rounded(significand, scientific, exponent))
    scientific = true
else
    show = with_separator(parms, rounded(value, clean))
end
show = propersign .. show
if parms.opt_spell_in then
    show = spell_number(parms, 'in', propersign .. rounded(value, clean))
    scientific = false
end
if scientific then
    parms.opt_scientific = true
end
end
if isnegative and (value ~= 0) then
    value = -value
    altvalue = -(altvalue or value)
end
return true, {
    value = value,
    altvalue = altvalue or value,
    singular = singular,
    clean = clean,
    show = show .. (reference or ''),
    denominator = denominator,
}
end

local function get_number(text)
    -- Return v, f where:
    -- v = nil (text is not a number)
    -- or
    -- v = value of text (text is a number)
    -- f = true if value is an integer
    -- Input can use en digits or digits in local language or separators,
    -- but no Unicode minus, and no fraction.
    if text then
        local number = tonumber(to_en(text))
        if number then
            local _, fracpart = math.modf(number)
            return number, (fracpart == 0)
        end
    end
end
```

```
        end
    end
end

local function gcd(a, b)
    -- Return the greatest common denominator for the given values,
    -- which are known to be positive integers.
    if a > b then
        a, b = b, a
    end
    if a <= 0 then
        return b
    end
    local r = b % a
    if r <= 0 then
        return a
    end
    if r == 1 then
        return 1
    end
    return gcd(r, a)
end

local function fraction_table(value, denominator)
    -- Return value as a string or a table:
    -- * If result is a string, there is no fraction, and the result
    --   is value formatted as a string of en digits.
    -- * If result is a table, it represents a fraction with named fields:
    --   wholestr, numstr, denstr (strings of en digits for integer, numerate
    -- The result is rounded to the nearest multiple of (1/denominator).
    -- If the multiple is zero, no fraction is included.
    -- No fraction is included if value is very large as the fraction would
    -- be unhelpful, particularly if scientific notation is required.
    -- Input value is a non-negative number.
    -- Input denominator is a positive integer for the desired fraction.
    if value <= 0 then
        return '0'
    end
    if denominator <= 0 or value > 1e8 then
        return format('%.2f', value)
    end
    local integer, decimals = math.modf(value)
    local numerator = floor((decimals * denominator) +
        0.5 + 2e-14) -- add fudge for some common cases of bad rounding
    if numerator >= denominator then
        integer = integer + 1
        numerator = 0
    end
    local wholestr = tostring(integer)
    if numerator > 0 then
        local div = gcd(numerator, denominator)
        if div > 1 then
            numerator = numerator / div
            denominator = denominator / div
        end
        return {
            wholestr = (integer > 0) and wholestr or '',
            numstr = tostring(numerator),
            denstr = tostring(denominator),
            value = value,
        }
    end
    return wholestr
end
```

```
local function preunits(count, preunit1, preunit2)
  -- If count is 1:
  --   ignore preunit2
  --   return p1
  -- else:
  --   preunit1 is used for preunit2 if the latter is empty
  --   return p1, p2
  -- where:
  --   p1 is text to insert before the input unit
  --   p2 is text to insert before the output unit
  --   p1 or p2 may be nil to mean "no preunit"
  -- Using '+' gives output like "5+ feet" (no space before, but space after)
  local function withspace(text, wantboth)
    -- Return text with space before and, if wantboth, after.
    -- However, no space is added if there is a space or '&nbsp;' or
    -- at that position ('-' is for adjectival text).
    -- There is also no space if text starts with '&'
    -- (e.g. '&deg;' would display a degree symbol with no preceding
    local char = text:sub(1, 1)
    if char == '&' then
      return text -- an html entity can be used to specify the
    end
    if not (char == ' ' or char == '-' or char == '+') then
      text = ' ' .. text
    end
    if wantboth then
      char = text:sub(-1, -1)
      if not (char == ' ' or char == '-' or text:sub(-6, -1) == '&nbsp;') then
        text = text .. ' '
      end
    end
    return text
  end
  local PLUS = '+'
  preunit1 = preunit1 or ''
  local trim1 = strip(preunit1)
  if count == 1 then
    if trim1 == '' then
      return nil
    end
    if trim1 == '+' then
      return PLUS
    end
    return withspace(preunit1, true)
  end
  preunit1 = withspace(preunit1)
  preunit2 = preunit2 or ''
  local trim2 = strip(preunit2)
  if trim1 == '+' then
    if trim2 == '' or trim2 == '+' then
      return PLUS, PLUS
    end
    preunit1 = PLUS
  end
  if trim2 == '' then
    if trim1 == '' then
      return nil, nil
    end
    preunit2 = preunit1
  elseif trim2 == '+' then
    preunit2 = PLUS
  elseif trim2 == '&#32;' then -- trick to make preunit2 empty
    preunit2 = nil
  end
end
```

```
        else
            preunit2 = whitespace(preunit2)
        end
        return preunit1, preunit2
    end
end

local function range_text(range, want_name, parms, before, after, inout, options)
    -- Return before .. rtext .. after
    -- where rtext is the text that separates two values in a range.
    local rtext, adj_text, exception
    options = options or {}
    if type(range) == 'table' then
        -- Table must specify range text for ('off' and 'on') or ('input
        -- and may specify range text for 'adj=on',
        -- and may specify exception = true.
        rtext = range[want_name and 'off' or 'on'] or
            range[((inout == 'in') == (parms.opt_flip == true)
        adj_text = range['adj']
        exception = range['exception']
    else
        rtext = range
    end
    if parms.opt_adjectival then
        if want_name or (exception and parms.abbr_org == 'on') then
            rtext = adj_text or rtext:gsub(' ', '-'):gsub('&nbsp;', ' ')
        end
    end
    if rtext == '-' and (options.spaced or after:sub(1, #MINUS) == MINUS) then
        rtext = '&nbsp;- '
    end
    return before .. rtext .. after
end

local function get_composite(parms, iparm, in_unit_table)
    -- Look for a composite input unit. For example, {{convert|1|yd|2|ft|3|in
    -- would result in a call to this function with
    -- iparm = 3 (parms[iparm] = "2", just after the first unit)
    -- in_unit_table = (unit table for "yd"; contains value 1 for number of
    -- Return true, iparm, unit where
    -- iparm = index just after the composite units (7 in above example)
    -- unit = composite unit table holding all input units,
    -- or return true if no composite unit is present in parms,
    -- or return false, t where t is an error message table.
    local default, subinfo
    local composite_units, count = { in_unit_table }, 1
    local fixups = {}
    local total = in_unit_table.valinfo[1].value
    local subunit = in_unit_table
    while subunit.subdivs do -- subdivs is nil or a table of allowed subdivi
        local subcode = strip(parms[iparm+1])
        local subdiv = subunit.subdivs[subcode] or subunit.subdivs[(all_u
        if not subdiv then
            break
        end
        local success
        success, subunit = lookup(parms, subcode, 'no_combination')
        if not success then return false, subunit end -- should never oc
        success, subinfo = extract_number(parms, parms[iparm])
        if not success then return false, subinfo end
        iparm = iparm + 2
        subunit.inout = 'in'
        subunit.valinfo = { subinfo }
        -- Recalculate total as a number of subdivisions.
        -- subdiv[1] = number of subdivisions per previous unit (integer
```



```
        total = total * subdiv[1] + subinfo.value
        if not default then -- set by the first subdiv with a default de
            default = subdiv.default
        end
        count = count + 1
        composite_units[count] = subunit
        if subdiv.unit or subdiv.name then
            fixups[count] = { unit = subdiv.unit, name = subdiv.name
        end
    end
end
if count == 1 then
    return true -- no error and no composite unit
end
for i, fixup in pairs(fixups) do
    local unit = fixup.unit
    local name = fixup.name
    if not unit or (count > 2 and name) then
        composite_units[i].fixed_name = name
    else
        local success, alternate = lookup(parms, unit, 'no_combin
        if not success then return false, alternate end -- shoul
        alternate.inout = 'in'
        alternate.valinfo = fixup.valinfo
        composite_units[i] = alternate
    end
end
return true, iparm, {
    utype = in_unit_table.utype,
    scale = subunit.scale, -- scale of last (least significant) unit
    valinfo = { { value = total, clean = subinfo.clean, denominator =
    composite = composite_units,
    default = default or in_unit_table.default
}
end

local function translate_parms(parms, kv_pairs)
    -- Update fields in parms by translating each key:value in kv_pairs to te
    -- used by this module (may involve translating from local language to En
    -- Also, checks are performed which may display warnings, if enabled.
    -- Return true if successful or return false, t where t is an error messa
    currency_text = nil -- local testing can hold module in memory; must cle
    if kv_pairs.adj and kv_pairs.sing then
        -- For enwiki (before translation), warn if attempt to use adj ar
        -- as the latter is a deprecated alias for the former.
        if kv_pairs.adj ~= kv_pairs.sing and kv_pairs.sing ~= '' then
            add_warning(parms, 1, 'cvt_unknown_option', 'sing=' .. kv
        end
        kv_pairs.sing = nil
    end
end
kv_pairs.comma = kv_pairs.comma or config.comma -- for plwiki who want c
for loc_name, loc_value in pairs(kv_pairs) do
    local en_name = text_code.en_option_name[loc_name]
    if en_name then
        local en_value = text_code.en_option_value[en_name]
        if en_value == 'INTEGER' then -- altitude_ft, altitude_n
            en_value = nil
            if loc_value == '' then
                add_warning(parms, 2, 'cvt_empty_option',
            else
                local minimum
                local number, is_integer = get_number(loc
                if en_name == 'sigfig' then
                    minimum = 1
                elseif en_name == 'frac' then
```



```
        minimum = 2
        if number and number < 0 then
            parms.opt_fraction_horize
            number = -number
        end
    else
        minimum = -1e6
    end
    if number and is_integer and number >= m
        en_value = number
    else
        local m
        if en_name == 'frac' then
            m = 'cvt_bad_frac'
        elseif en_name == 'sigfig' then
            m = 'cvt_bad_sigfig'
        else
            m = 'cvt_bad_altitude'
        end
        add_warning(parms, 1, m, loc_name)
    end
end
elseif en_value == 'TEXT' then -- $, input, qid, qual, s
    en_value = loc_value ~= '' and loc_value or nil
    if not en_value and (en_name == '$' or en_name ==
        add_warning(parms, 2, 'cvt_empty_option',
    elseif en_name == '$' then
        -- Value should be a single character like
        currency_text = (loc_value == 'euro') and
    elseif en_name == 'input' then
        -- May have something like {{convert|input
        -- with optional fields. In that case, wa
        parms.input_text = loc_value -- keep inp
    end
else
    en_value = en_value[loc_value]
    if en_value and en_value:sub(-1) == '?' then
        en_value = en_value:sub(1, -2)
        add_warning(parms, -1, 'cvt_deprecated',
    end
    if en_value == nil then
        if loc_value == '' then
            add_warning(parms, 2, 'cvt_empty_
        else
            add_warning(parms, 1, 'cvt_unknow
        end
    elseif en_value == '' then
        en_value = nil -- an ignored option like
    elseif type(en_value) == 'string' and en_value:st
        for _, v in ipairs(split(en_value, ','))
            local lhs, rhs = v:match('^(.-)=(
            if rhs then
                parms[lhs] = tonumber(rhs
            else
                parms[v] = true
            end
        end
        en_value = nil
    end
end
parms[en_name] = en_value
else
    add_warning(parms, 1, 'cvt_unknown_option', loc_name ..
end
```

```
end
local abbr_entered = parms.abbr
local cfg_abbr = config.abbr
if cfg_abbr then
    -- Don't warn if invalid because every convert would show that we
    if cfg_abbr == 'on always' then
        parms.abbr = 'on'
    elseif cfg_abbr == 'off always' then
        parms.abbr = 'off'
    elseif parms.abbr == nil then
        if cfg_abbr == 'on default' then
            parms.abbr = 'on'
        elseif cfg_abbr == 'off default' then
            parms.abbr = 'off'
        end
    end
end
end
if parms.abbr then
    if parms.abbr == 'unit' then
        parms.abbr = 'on'
        parms.number_word = true
    end
    parms.abbr_org = parms.abbr -- original abbr, before any flip
elseif parms.opt_hand_hh then
    parms.abbr_org = 'on'
    parms.abbr = 'on'
else
    parms.abbr = 'out' -- default is to abbreviate output only (use
end
if parms.opt_order_out then
    -- Disable options that do not work in a useful way with order=out
    parms.opt_flip = nil -- override adj=flip
    parms.opt_spell_in = nil
    parms.opt_spell_out = nil
    parms.opt_spell_upper = nil
end
if parms.opt_spell_out and not abbr_entered then
    parms.abbr = 'off' -- should show unit name when spelling the out
end
if parms.opt_flip then
    local function swap_in_out(option)
        local value = parms[option]
        if value == 'in' then
            parms[option] = 'out'
        elseif value == 'out' then
            parms[option] = 'in'
        end
    end
    swap_in_out('abbr')
    swap_in_out('lk')
    if parms.opt_spell_in and not parms.opt_spell_out then
        -- For simplicity, and because it does not appear to be r
        -- user cannot set an option to spell the output only.
        parms.opt_spell_in = nil
        parms.opt_spell_out = true
    end
end
if parms.opt_spell_upper then
    parms.spell_upper = parms.opt_flip and 'out' or 'in'
end
if parms.opt_table or parms.opt_tablecen then
    if abbr_entered == nil and parms.lk == nil then
        parms.opt_values = true
    end
end
```

```
        parms.table_align = parms.opt_table and 'right' or 'center'
    end
    if parms.table_align or parms.opt_sortable_on then
        parms.need_table_or_sort = true
    end
    local disp_joins = text_code.disp_joins
    local default_joins = disp_joins['b']
    parms.join_between = default_joins[3] or ';'
    local disp = parms.disp
    if disp == nil then -- special case for the most common setting
        parms.joins = default_joins
    elseif disp == 'x' then
        -- Later, parms.joins is set from the input parameters.
    else
        -- Old template does this.
        local abbr = parms.abbr
        if disp == 'slash' then
            if abbr_entered == nil then
                disp = 'slash-nbsp'
            elseif abbr == 'in' or abbr == 'out' then
                disp = 'slash-sp'
            else
                disp = 'slash-nosp'
            end
        elseif disp == 'sqbr' then
            if abbr == 'on' then
                disp = 'sqbr-nbsp'
            else
                disp = 'sqbr-sp'
            end
        end
        parms.joins = disp_joins[disp] or default_joins
        parms.join_between = parms.joins[3] or parms.join_between
        parms.wantname = parms.joins.wantname
    end
    if (en_default and not parms.opt_lang_local and (parms[1] or ''):find('%d')
        from_en_table = nil
    end
    if en_default and from_en_table then
        -- For hiwiki: localized symbol/name is defined with the US symbol
        -- and is used if output uses localized numbers.
        parms.opt_sp_us = true
    end
    return true
end

local function get_values(parms)
    -- If successful, update parms and return true, v, i where
    --   v = table of input values
    --   i = index to next entry in parms after those processed here
    -- or return false, t where t is an error message table.
    local valinfo = collection() -- numbered table of input values
    local range = collection() -- numbered table of range items (having, for
    local had_nocomma -- true if removed "nocomma" kludge from second param
    local parm2 = strip(parms[2])
    if parm2 and parm2:sub(-7, -1) == 'nocomma' then
        parms[2] = strip(parm2:sub(1, -8))
        parms.opt_nocomma = true
        had_nocomma = true
    end
    local function extractor(i)
        -- If the parameter is not a value, try unpacking it as a range (
        -- However, "-1-2/3" is a negative fraction ( $-1\frac{2}{3}$ ), so it must be
        -- Do not unpack a parameter if it is like "3-1/2" which is somet
```

```
-- used instead of "3+1/2" (and which should not be interpreted a
-- Unpacked items are inserted into the parms table.
-- The tail recursion allows combinations like "1x2 to 3x4".
local valstr = strip(parms[i]) -- trim so any '-' as a negative
local success, result = extract_number(parms, valstr, i > 1)
if not success and valstr and i < 20 then -- check i to limit at
    local lhs, sep, rhs = valstr:match('^(%S+)%s+(%S+)%s+(%S+)'
    if lhs and not (sep == '-' and rhs:match('/')) then
        if sep:find('%d') then
            return success, result -- to reject {{c
        end
        parms[i] = rhs
        table.insert(parms, i, sep)
        table.insert(parms, i, lhs)
        return extractor(i)
    end
    if not valstr:match('%-.*/*') then
        for _, sep in ipairs(text_code.ranges.words) do
            local start, stop = valstr:find(sep, 2, t
            if start then
                parms[i] = valstr:sub(stop + 1)
                table.insert(parms, i, sep)
                table.insert(parms, i, valstr:sub
                return extractor(i)
            end
        end
    end
end
return success, result
end
local i = 1
local is_change
while true do
    local success, info = extractor(i) -- need to set parms.opt_noc
    if not success then return false, info end
    i = i + 1
    if is_change then
        info.is_change = true -- value is after "±" and so is a
        is_change = nil
    end
    valinfo:add(info)
    local range_item = get_range(strip(parms[i]))
    if not range_item then
        break
    end
    i = i + 1
    range:add(range_item)
    if type(range_item) == 'table' then
        -- For range "x", if append unit to some values, append i
        parms.in_range_x = parms.in_range_x or range_item.in_rang
        parms.out_range_x = parms.out_range_x or range_item.out_r
        parms.abbr_range_x = parms.abbr_range_x or range_item.abbr
        is_change = range_item.is_range_change
    end
end
if range.n > 0 then
    if range.n > 30 then -- limit abuse, although 4 is a more likely
        return false, { 'cvt_invalid_num' } -- misleading messag
    end
    parms.range = range
elseif had_nocomma then
    return false, { 'cvt_unknown', parm2 }
end
return true, valinfo, i
```

```
end

local function simple_get_values(parms)
  -- If input is like "{{convert|valid_value|valid_unit|...}}",
  -- return true, i, in_unit, in_unit_table
  -- i = index in parms of what follows valid_unit, if anything.
  -- The valid_value is not negative and does not use a fraction, and
  -- no options requiring further processing of the input are used.
  -- Otherwise, return nothing or return false, parm1 for caller to interpret
  -- Testing shows this function is successful for 96% of converts in articles
  -- and that on average it speeds up converts by 8%.
  local clean = to_en(strip(parms[1] or ''), parms)
  if parms.opt_ri or parms.opt_spell_in or #clean > 10 or not clean:match(
    return false, clean
  end
  local value = tonumber(clean)
  if not value then return end
  local info = {
    value = value,
    altvalue = value,
    singular = (value == 1),
    clean = clean,
    show = with_separator(parms, clean),
  }
  local in_unit = strip(parms[2])
  local success, in_unit_table = lookup(parms, in_unit, 'no_combination')
  if not success then return end
  in_unit_table.valinfo = { info }
  return true, 3, in_unit, in_unit_table
end

local function wikidata_call(parms, operation, ...)
  -- Return true, s where s is the result of a Wikidata operation,
  -- or return false, t where t is an error message table.
  local function worker(...)
    wikidata_code = wikidata_code or require(wikidata_module)
    wikidata_data = wikidata_data or mw.loadData(wikidata_data_module)
    return wikidata_code[operation](wikidata_data, ...)
  end
  local success, status, result = pcall(worker, ...)
  if success then
    return status, result
  end
  if parms.opt_sortable_debug then
    -- Use debug=yes to crash if an error while accessing Wikidata.
    error('Error accessing Wikidata: ' .. status, 0)
  end
  return false, { 'cvt_wd_fail' }
end

local function get_parms(parms, args)
  -- If successful, update parms and return true, unit where
  -- parms is a table of all arguments passed to the template
  -- converted to named arguments, and
  -- unit is the input unit table;
  -- or return false, t where t is an error message table.
  -- For special processing (not a convert), can also return
  -- true, wikitext where wikitext is the final result.
  -- The returned input unit table may be for a fake unit using the specific
  -- unit code as the symbol and name, and with bad_mcode = message code table
  -- MediaWiki removes leading and trailing whitespace from the values of
  -- named arguments. However, the values of numbered arguments include any
  -- whitespace entered in the template, and whitespace is used by some
  -- parameters (example: the numbered parameters associated with "disp=x")
end
```

```
local kv_pairs = {} -- table of input key:value pairs where key is a name
for k, v in pairs(args) do
    if type(k) == 'number' or k == 'test' then -- parameter "test" is special
        parms[k] = v
    else
        kv_pairs[k] = v
    end
end
end
if parms.test == 'wikidata' then
    local ulookup = function (ucode)
        -- Use empty table for parms so it does not accumulate results
        return lookup({}, ucode, 'no_combination')
    end
    return wikidata_call(parms, '_listunits', ulookup)
end
local success, msg = translate_parms(parms, kv_pairs)
if not success then return false, msg end
if parms.input then
    success, msg = wikidata_call(parms, '_adjustparameters', parms, kv_pairs)
    if not success then return false, msg end
end
local success, i, in_unit, in_unit_table = simple_get_values(parms)
if not success then
    if type(i) == 'string' and i:match('^NNN+$') then
        -- Some infoboxes have examples like {{convert|NNN|m}} (3)
        -- Output an empty string for these.
        return false, { 'cvt_no_output' }
    end
    local valinfo
    success, valinfo, i = get_values(parms)
    if not success then return false, valinfo end
    in_unit = strip(parms[i])
    i = i + 1
    success, in_unit_table = lookup(parms, in_unit, 'no_combination')
    if not success then
        in_unit = in_unit or ''
        if parms.opt_ignore_error then -- display given unit code
            in_unit_table = '' -- suppress error message and return empty table
        end
        in_unit_table = setmetatable({
            symbol = in_unit, name2 = in_unit, utype = in_unit,
            scale = 1, default = '', defkey = '', linkey = '',
            bad_mcode = in_unit_table }, unit_mt)
    end
    in_unit_table.valinfo = valinfo
end
if parms.test == 'msg' then
    -- Am testing the messages produced when no output unit is specified
    -- the input unit has a missing or invalid default.
    -- Set two units for testing that.
    -- LATER: Remove this code.
    if in_unit == 'chain' then
        in_unit_table.default = nil -- no default
    elseif in_unit == 'rd' then
        in_unit_table.default = "ft!X!m" -- an invalid expression
    end
end
end
in_unit_table.inout = 'in' -- this is an input unit
if not parms.range then
    local success, inext, composite_unit = get_composite(parms, i, in_unit_table)
    if not success then return false, inext end
    if composite_unit then
        in_unit_table = composite_unit
        i = inext
    end
end
```

```
        end
    end
    if in_unit_table.builtin == 'mach' then
        -- As with old template, a number following Mach as the input unit
        -- That is deprecated: should use altitude_ft=NUMBER or altitude_
        local success, info
        success = tonumber(parms[i]) -- this will often work and will g
        if success then
            info = { value = success }
        else
            success, info = extract_number(parms, parms[i], false, t
        end
        if success then
            i = i + 1
            in_unit_table.altitude = info.value
        end
    end
    local word = strip(parms[i])
    i = i + 1
    local precision, is_bad_precision
    local function set_precision(text)
        local number, is_integer = get_number(text)
        if number then
            if is_integer then
                precision = number
            else
                precision = text
                is_bad_precision = true
            end
        end
        return true -- text was used for precision, good or bad
    end
    end
    if word and not set_precision(word) then
        parms.out_unit = parms.out_unit or word
        if set_precision(strip(parms[i])) then
            i = i + 1
        end
    end
    end
    if parms.opt_adj_mid then
        word = parms[i]
        i = i + 1
        if word then -- mid-text words
            if word:sub(1, 1) == '-' then
                parms.mid = word
            else
                parms.mid = ' ' .. word
            end
        end
    end
    end
    if parms.opt_one_preunit then
        parms[parms.opt_flip and 'preunit2' or 'preunit1'] = preunits(1,
        i = i + 1
    end
    if parms.disp == 'x' then
        -- Following is reasonably compatible with the old template.
        local first = parms[i] or ''
        local second = parms[i+1] or ''
        i = i + 2
        if strip(first) == '' then -- user can enter '&#32;' rather than
            first = ' [&nbsp;]' .. first
            second = '&nbsp;]' .. second
        end
        parms.joins = { first, second }
    elseif parms.opt_two_preunits then
```



```
        local p1, p2 = preunits(2, parms[i], parms[i+1])
        i = i + 2
        if parms.preunit1 then
            -- To simplify documentation, allow unlikely use of adj=
            -- (however, an output unit must be specified with adj=p
            parms.preunit1 = parms.preunit1 .. p1
            parms.preunit2 = p2
        else
            parms.preunit1, parms.preunit2 = p1, p2
        end
    end
end
if precision == nil then
    if set_precision(strip(parms[i])) then
        i = i + 1
    end
end
if is_bad_precision then
    add_warning(parms, 1, 'cvt_bad_prec', precision)
else
    parms.precision = precision
end
for j = i, i + 3 do
    local parm = parms[j] -- warn if find a non-empty extraneous pa
    if parm and parm:match('%S') then
        add_warning(parms, 1, 'cvt_unknown_option', parm)
        break
    end
end
return true, in_unit_table
end

local function record_default_precision(parms, out_current, precision)
    -- If necessary, adjust parameters and return a possibly adjusted precisi
    -- When converting a range of values where a default precision is require
    -- that default is calculated for each value because the result sometimes
    -- depends on the precise input and output values. This function may caus
    -- the entire convert process to be repeated in order to ensure that the
    -- same default precision is used for each individual convert.
    -- If that were not done, a range like 1000 to 1000.4 may give poor resul
    -- because the first output could be heavily rounded, while the second is
    -- For range 1000.4 to 1000, this function can give the second convert th
    -- same default precision that was used for the first.
    if not parms.opt_round_each then
        local maxdef = out_current.max_default_precision
        if maxdef then
            if maxdef < precision then
                parms.do_convert_again = true
                out_current.max_default_precision = precision
            else
                precision = out_current.max_default_precision
            end
        else
            out_current.max_default_precision = precision
        end
    end
end
return precision
end

local function default_precision(parms, invalue, inclean, denominator, outvalue,
    -- Return a default value for precision (an integer like 2, 0, -2).
    -- If denominator is not nil, it is the value of the denominator in inclu
    -- Code follows procedures used in old template.
    local fudge = 1e-14 -- {{Order of magnitude}} adds this, so we do too
    local prec, minprec, adjust
```



```
local subunit_ignore_trailing_zero
local subunit_more_precision -- kludge for "in" used in input like "|2|ft|10|lb"
local composite = in_current.composite
if composite then
    subunit_ignore_trailing_zero = true -- input "|2|st|10|lb" has p
    if composite[#composite].exception == 'subunit_more_precision' th
        subunit_more_precision = true -- do not use standard pre
    end
end
end
if denominator and denominator > 0 then
    prec = math.max(log10(denominator), 1)
else
    -- Count digits after decimal mark, handling cases like '12.345e
    local exponent
    local integer, dot, decimals, expstr = inclean:match('^(%d*)(%?.?)
    local e = expstr:sub(1, 1)
    if e == 'e' or e == 'E' then
        exponent = tonumber(expstr:sub(2))
    end
    if dot == '.' then
        prec = subunit_ignore_trailing_zero and 0 or -integer:mat
    else
        prec = #decimals
    end
    if exponent then
        -- So '1230' and '1.23e3' both give prec = -1, and '0.001
        prec = prec - exponent
    end
end
end
if in_current.istemperature and out_current.istemperature then
    -- Converting between common temperatures (°C, °F, °R, K); not ke
    -- Kelvin value can be almost zero, or small but negative due to
    -- Also, an input value like -300 C (below absolute zero) gives r
    -- Calculate minimum precision from absolute value.
    adjust = 0
    local kelvin = abs((invalue - in_current.offset) * in_current.sca
    if kelvin < 1e-8 then -- assume nonzero due to input or calculat
        minprec = 2
    else
        minprec = 2 - floor(log10(kelvin) + fudge) -- 3 sigfigs
    end
end
else
    if invalue == 0 or outvalue <= 0 then
        -- We are never called with a negative outvalue, but it m
        -- This is special-cased to avoid calculation exceptions.
        return record_default_precision(parms, out_current, 0)
    end
    if out_current.exception == 'integer_more_precision' and floor(in
        -- With certain output units that sometimes give poor res
        -- with default rounding, use more precision when the inp
        -- value is equal to an integer. An example of a poor res
        -- is when input 50 gives a smaller output than input 49.
        -- Experiment shows this helps, but it does not eliminat
        -- surprises because it is not clear whether "50" should
        -- interpreted as "from 45 to 55" or "from 49.5 to 50.5"
        adjust = -log10(in_current.scale)
    elseif subunit_more_precision then
        -- Conversion like "{{convert|6|ft|1|in|cm}}" (where subu
        -- has a non-standard adjust value, to give more output p
        adjust = log10(out_current.scale) + 2
    else
        adjust = log10(abs(invalue / outvalue))
    end
    adjust = adjust + log10(2)
end
```



```
        -- Ensure that the output has at least two significant figures.
        minprec = 1 - floor(log10(outvalue) + fudge)
    end
    if extra then
        adjust = extra.adjust or adjust
        minprec = extra.minprec or minprec
    end
    return record_default_precision(parms, out_current, math.max(floor(prec -
end

local function convert(parms, invalue, info, in_current, out_current)
    -- Convert given input value from one unit to another.
    -- Return output_value (a number) if a simple convert, or
    -- return f, t where
    --   f = true, t = table of information with results, or
    --   f = false, t = error message table.
    local inscale = in_current.scale
    local outscale = out_current.scale
    if not in_current.iscomplex and not out_current.iscomplex then
        return invalue * (inscale / outscale) -- minimize overhead for n
    end
    if in_current.invert or out_current.invert then
        -- Inverted units, such as inverse length, inverse time, or
        -- fuel efficiency. Built-in units do not have invert set.
        if (in_current.invert or 1) * (out_current.invert or 1) < 0 then
            return 1 / (invalue * inscale * outscale)
        end
        return invalue * (inscale / outscale)
    elseif in_current.offset then
        -- Temperature (there are no built-ins for this type of unit).
        if info.is_change then
            return invalue * (inscale / outscale)
        end
        return (invalue - in_current.offset) * (inscale / outscale) + out
    else
        -- Built-in unit.
        local in_builtin = in_current.builtin
        local out_builtin = out_current.builtin
        if in_builtin and out_builtin then
            if in_builtin == out_builtin then
                return invalue
            end
            -- There are no cases (yet) where need to convert from or
            -- built-in unit to another, so this should never occur.
            return false, { 'cvt_bug_convert' }
        end
        if in_builtin == 'mach' or out_builtin == 'mach' then
            -- Should check that only one altitude is given but am pl
            -- in_current.altitude (which can only occur when Mach is
            -- and out_current.altitude cannot occur.
            local alt = parms.altitude_ft or in_current.altitude
            if not alt and parms.altitude_m then
                alt = parms.altitude_m / 0.3048 -- 1 ft = 0.3048
            end
            local spd = speed_of_sound(alt)
            if in_builtin == 'mach' then
                inscale = spd
                return invalue * (inscale / outscale)
            end
            outscale = spd
            local adjust = 0.1 / inscale
            return true, {
                outvalue = invalue * (inscale / outscale),
                adjust = log10(adjust) + log10(2),
            }
        end
    end
end
```

```
    }
elseif in_builtin == 'hand' then
    -- 1 hand = 4 inches; 1.2 hands = 6 inches.
    -- Decimals of a hand are only defined for the first digit
    -- the first fractional digit should be a number of inches
    -- However, this code interprets the entire fractional part
    -- of inches / 10 (so 1.75 inches would be 0.175 hands).
    -- A value like 12.3 hands is exactly 12*4 + 3 inches; but
local integer, fracpart = math.modf(invalue)
local inch_value = 4 * integer + 10 * fracpart -- equivalent
local factor = inscale / outscale
if factor == 4 then
    -- Am converting to inches: show exact result, and
    if parms.abbr_org == nil then
        out_current.username = true
    end
    local show = format('%g', abs(inch_value)) -- show
    if not show:find('e', 1, true) then
        return true, {
            invalue = inch_value,
            outvalue = inch_value,
            clean = show,
            show = show,
        }
    end
end
local outvalue = (integer + 2.5 * fracpart) * factor
local fracstr = info.clean:match('%.(.*)') or ''
local fmt
if fracstr == '' then
    fmt = '%.0f'
else
    fmt = '%.' .. format('%d', #fracstr - 1) .. 'f'
end
return true, {
    invalue = inch_value,
    clean = format(fmt, inch_value),
    outvalue = outvalue,
    minprec = 0,
}
end
end
return false, { 'cvt_bug_convert' } -- should never occur
end

local function user_style(parms, i)
    -- Return text for a user-specified style for a table cell, or '' if none
    -- given i = 1 (input style) or 2 (output style).
    local style = parms[(i == 1) and 'stylein' or 'styleout']
    if style then
        style = style:gsub('"', '')
        if style ~= '' then
            if style:sub(-1) ~= ';' then
                style = style .. ';'
            end
            return style
        end
    end
end
return ''
end

local function make_table_or_sort(parms, invalue, info, in_current, scaled_top)
    -- Set options to handle output for a table or a sort key, or both.
    -- The text sort key is based on the value resulting from converting
```

```
-- the input to a fake base unit with scale = 1, and other properties
-- required for a conversion derived from the input unit.
-- For other modules, return the sort key in a hidden span element, and
-- the scaled value used to generate the sort key.
-- If scaled_top is set, it is the scaled value of the numerator of a per
-- to be combined with this unit (the denominator) to make the sort key.
-- Scaling only works with units that convert with a factor (not temperat
local sortkey, scaled_value
if parms.opt_sortable_on then
    local base = { -- a fake unit with enough fields for a valid con
        scale = 1,
        invert = in_current.invert and 1,
        iscomplex = in_current.iscomplex,
        offset = in_current.offset and 0,
    }
    local outvalue, extra = convert(parms, invalue, info, in_current,
    if extra then
        outvalue = extra.outvalue
    end
    if in_current.istemperature then
        -- Have converted to kelvin; assume numbers close to zero
        -- rounding error and should be zero.
        if abs(outvalue) < 1e-12 then
            outvalue = 0
        end
    end
    if scaled_top and outvalue ~= 0 then
        outvalue = scaled_top / outvalue
    end
    scaled_value = outvalue
    if not valid_number(outvalue) then
        if outvalue < 0 then
            sortkey = '10000000000000000000'
        else
            sortkey = '90000000000000000000'
        end
    elseif outvalue == 0 then
        sortkey = '50000000000000000000'
    else
        local mag = floor(log10(abs(outvalue))) + 1e-14
        local prefix
        if outvalue > 0 then
            prefix = 7000 + mag
        else
            prefix = 2999 - mag
            outvalue = outvalue + 10^(mag+1)
        end
        sortkey = format('%d', prefix) .. format('%015.0f', floor
    end
end
local sortspan
if sortkey and not parms.table_align then
    sortspan = parms.opt_sortable_debug and
        '<span data-sort-value="' .. sortkey .. '♠"><span style='
        '<span data-sort-value="' .. sortkey .. '♠"></span>'
    parms.join_before = sortspan
end
if parms.table_align then
    local sort
    if sortkey then
        sort = ' data-sort-value="' .. sortkey .. ''
        if parms.opt_sortable_debug then
            parms.join_before = '<span style="border:1px sol
        end
    end
end
```

```
        else
            sort = ''
        end
        local style = 'style="text-align:' .. parms.table_align .. '; '
        local joins = {}
        for i = 1, 2 do
            joins[i] = (i == 1 and '' or '\n|') .. style .. user_style
        end
        parms.table_joins = joins
    end
    return sortspan, scaled_value
end

local cvt_to_hand

local function cvtround(parms, info, in_current, out_current)
    -- Return true, t where t is a table with the conversion results; fields
    -- show = rounded, formatted string with the result of converting value
    -- using the rounding specified in parms.
    -- singular = true if result (after rounding and ignoring any negative
    -- is "1", or like "1.00", or is a fraction with value < 1;
    -- (and more fields shown below, and a calculated 'absvalue' field).
    -- or return false, t where t is an error message table.
    -- Input info.clean uses en digits (it has been translated, if necessary)
    -- Output show uses en or non-en digits as appropriate, or can be spelled
    if out_current.builtin == 'hand' then
        return cvt_to_hand(parms, info, in_current, out_current)
    end
    local inval = in_current.builtin == 'hand' and info.altvalue or info.value
    local outvalue, extra = convert(parms, inval, info, in_current, out_current)
    if parms.need_table_or_sort then
        parms.need_table_or_sort = nil -- process using first input value
        make_table_or_sort(parms, inval, info, in_current)
    end
    if extra then
        if not outvalue then return false, extra end
        inval = extra.inval or inval
        outvalue = extra.outvalue
    end
    if not valid_number(outvalue) then
        return false, { 'cvt_invalid_num' }
    end
    local isnegative
    if outvalue < 0 then
        isnegative = true
        outvalue = -outvalue
    end
    local precision, show, exponent
    local denominator = out_current.frac
    if denominator then
        show = fraction_table(outvalue, denominator)
    else
        precision = parms.precision
        if not precision then
            if parms.sigfig then
                show, exponent = make_sigfig(outvalue, parms.sigfig)
            elseif parms.opt_round then
                local n = parms.opt_round
                if n == 0.5 then
                    local integer, fracpart = math.modf(floor(outvalue))
                    if fracpart == 0 then
                        show = format('%.0f', integer)
                    else
                        show = format('%.1f', integer + 1)
                    end
                else
                    show = format('%.1f', integer + 1)
                end
            end
        end
    end
end
```

```

        end
        else
            show = format('%0f', floor((outvalue / r
        end
elseif in_current.builtin == 'mach' then
    local sigfig = info.clean:gsub('^[0.]+', '):gsu
    show, exponent = make_sigfig(outvalue, sigfig)
else
    local inclean = info.clean
    if extra then
        inclean = extra.clean or inclean
        show = extra.show
    end
    if not show then
        precision = default_precision(parms, inve
    end
end
end
end
end
if precision then
    if precision >= 0 then
        local fudge
        if precision <= 8 then
            -- Add a fudge to handle common cases of bad rou
            -- to precisely represent some values. This makes
            -- {{convert|-100.1|C|K}} and {{convert|5555000|C|K}}
            -- Old template uses #expr round, which invokes R
            -- LATER: Investigate how PHP round() works.
            fudge = 2e-14
        else
            fudge = 0
        end
        local fmt = '%.' .. format('%d', precision) .. 'f'
        local success
        success, show = pcall(format, fmt, outvalue + fudge)
        if not success then
            return false, { 'cvt_big_prec', tostring(precision) }
        end
    else
        precision = -precision -- #digits to zero (in addition to
        local shift = 10 ^ precision
        show = format('%0f', outvalue/shift)
        if show ~= '0' then
            exponent = #show + precision
        end
    end
end
end
local t = format_number(parms, show, exponent, isnegative)
if type(show) == 'string' then
    -- Set singular using match because on some systems 0.9999999999
    if exponent then
        t.singular = (exponent == 1 and show:match('^10*$'))
    else
        t.singular = (show == '1' or show:match('^1%.0*$'))
    end
else
    t.fraction_table = show
    t.singular = (outvalue <= 1) -- cannot have 'fraction == 1', but
end
t.raw_absvalue = outvalue -- absolute value before rounding
return true, setmetatable(t, {
    __index = function (self, key)
        if key == 'absvalue' then
            -- Calculate absolute value after rounding, if ne

```

```
        local clean, exponent = rawget(self, 'clean'), rawget(self, 'exponent')
        local value = tonumber(clean) -- absolute value
        if exponent then
            value = value * 10^exponent
        end
        rawset(self, key, value)
        return value
    end
end })
end

function cvt_to_hand(parms, info, in_current, out_current)
    -- Convert input to hands, inches.
    -- Return true, t where t is a table with the conversion results;
    -- or return false, t where t is an error message table.
    if parms.abbr_org == nil then
        out_current.username = true -- default is to show name not symbol
    end
    local precision = parms.precision
    local frac = out_current.frac
    if not frac and precision and precision > 1 then
        frac = (precision == 2) and 2 or 4
    end
    local out_next = out_current.out_next
    if out_next then
        -- Use magic knowledge to determine whether the next unit is inches
        -- The following ensures that when the output combination "hand 1
        -- value is rounded to match the hands value. Also, displaying se
        -- is better as 61.5 implies the value is not 61.4.
        if out_next.exception == 'subunit_more_precision' then
            out_next.frac = frac
        end
    end
    -- Convert to inches; calculate hands from that.
    local dummy_unit_table = { scale = out_current.scale / 4, frac = frac }
    local success, outinfo = cvtround(parms, info, in_current, dummy_unit_table)
    if not success then return false, outinfo end
    local tfrac = outinfo.fraction_table
    local inches = outinfo.raw_absvalue
    if tfrac then
        inches = floor(inches) -- integer part only; fraction added later
    else
        inches = floor(inches + 0.5) -- a hands measurement never shows
    end
    local hands, inches = divide(inches, 4)
    outinfo.absvalue = hands + inches/4 -- supposed to be the absolute rounded
    local inchstr = tostring(inches) -- '0', '1', '2' or '3'
    if precision and precision <= 0 then -- using negative or 0 for precision
        hands = floor(outinfo.raw_absvalue/4 + 0.5)
        inchstr = ''
    elseif tfrac then
        -- Always show an integer before fraction (like "15.0½") because
        inchstr = numdot .. format_fraction(parms, 'out', false, inchstr)
    else
        inchstr = numdot .. from_en(inchstr)
    end
    outinfo.show = outinfo.sign .. with_separator(parms, format('%0f', hands), inchstr)
    return true, outinfo
end

local function evaluate_condition(value, condition)
    -- Return true or false from applying a conditional expression to value,
    -- or throw an error if invalid.
    -- A very limited set of expressions is supported:
```

```
--      v < 9
--      v * 9 < 9
-- where
--      'v' is replaced with value
--      9 is any number (as defined by Lua tonumber)
--      only en digits are accepted
--      '<' can also be '<=' or '>' or '>='
-- In addition, the following form is supported:
--      LHS and RHS
-- where
--      LHS, RHS = any of above expressions.
local function compare(value, text)
    local arithop, factor, compop, limit = text:match('^%s*v%s*([*]?%s*)')
    if arithop == nil then
        error('Invalid default expression', 0)
    elseif arithop == '*' then
        factor = tonumber(factor)
        if factor == nil then
            error('Invalid default expression', 0)
        end
        value = value * factor
    end
    limit = tonumber(limit)
    if limit == nil then
        error('Invalid default expression', 0)
    end
    if compop == '<' then
        return value < limit
    elseif compop == '<=' then
        return value <= limit
    elseif compop == '>' then
        return value > limit
    elseif compop == '>=' then
        return value >= limit
    end
    error('Invalid default expression', 0) -- should not occur
end
local lhs, rhs = condition:match('^(.-%W)and(%W.*)')
if lhs == nil then
    return compare(value, condition)
end
return compare(value, lhs) and compare(value, rhs)
end

local function get_default(value, unit_table)
    -- Return true, s where s = name of unit's default output unit,
    -- or return false, t where t is an error message table.
    -- Some units have a default that depends on the input value
    -- (the first value if a range of values is used).
    -- If '!' is in the default, the first bang-delimited field is an
    -- expression that uses 'v' to represent the input value.
    -- Example: 'v < 120 ! small ! big ! suffix' (suffix is optional)
    -- evaluates 'v < 120' as a boolean with result
    -- 'smallsuffix' if (value < 120), or 'bigsuffix' otherwise.
    -- Input must use en digits and '.' decimal mark.
    local default = data_code.default_exceptions[unit_table.defkey or unit_table.defkey]
    if not default then
        local per = unit_table.per
        if per then
            local function a_default(v, u)
                local success, ucode = get_default(v, u)
                if not success then
                    return '?' -- an unlikely error has occurred
                end
            end
        end
    end
end
```

```
-- Attempt to use only the first unit if a combination
-- This is not bulletproof but should work for most cases
-- Where it does not work, the convert will need to be fixed
local t = all_units[ucode]
if t then
    local combo = t.combination
    if combo then
        -- For a multiple like ftin, the unit is the first one
        local i = t.multiple and table_length(combo, i)
        ucode = combo[i]
    end
else
    -- Try for an automatically generated combination
    local item = ucode:match('^(.-)%+') or ucode
    if all_units[item] then
        return item
    end
end
return ucode
end
local unit1, unit2 = per[1], per[2]
local def1 = (unit1 and a_default(value, unit1) or unit1)
local def2 = a_default(1, unit2) -- 1 because per unit
return true, def1 .. '/' .. def2
end
return false, { 'cvt_no_default', unit_table.symbol }
end
if default:find('!', 1, true) == nil then
    return true, default
end
local t = split(default, '!')
if #t == 3 or #t == 4 then
    local success, result = pcall(evaluate_condition, value, t[1])
    if success then
        default = result and t[2] or t[3]
        if #t == 4 then
            default = default .. t[4]
        end
    end
    return true, default
end
end
return false, { 'cvt_bad_default', unit_table.symbol }
end

local linked_pages -- to record linked pages so will not link to the same page

local function unlink(unit_table)
    -- Forget that the given unit has previously been linked (if it has).
    -- That is needed when processing a range of inputs or outputs when an id
    -- for the first range value may have been evaluated, but only an id for
    -- the last value is displayed, and that id may need to be linked.
    linked_pages[unit_table.unitcode or unit_table] = nil
end

local function make_link(link, id, unit_table)
    -- Return wikilink "[[link|id]]", possibly abbreviated as in examples:
    -- [[Mile|mile]] --> [[mile]]
    -- [[Mile|miles]] --> [[mile]]s
    -- However, just id is returned if:
    -- * no link given (so caller does not need to check if a link was defined)
    -- * link has previously been used during the current convert (to avoid c
    local link_key
    if unit_table then
        link_key = unit_table.unitcode or unit_table
    end
end
```

```
else
    link_key = link
end
if not link or link == '' or linked_pages[link_key] then
    return id
end
linked_pages[link_key] = true
-- Following only works for language en, but it should be safe on other v
-- and overhead of doing it generally does not seem worthwhile.
local l = link:sub(1, 1):lower() .. link:sub(2)
if link == id or l == id then
    return '[' .. id .. ']'
elseif link .. 's' == id or l .. 's' == id then
    return '[' .. id:sub(1, -2) .. ']'s'
else
    return '[' .. link .. '|' .. id .. ']'
end
end
end

local function variable_name(clean, unit_table)
    -- For slwiki, a unit name depends on the value.
    -- Parameter clean is the unsigned rounded value in en digits, as a string
    -- Value          Source      Example for "m"
    -- integer 1:     name1       meter (also is the name of the unit)
    -- integer 2:     var{1}      metra
    -- integer 3 and 4: var{2}    metri
    -- integer else:  var{3}      metrov (0 and 5 or more)
    -- real/fraction: var{4}      metra
    -- var{i} means the i'th field in unit_table.varname if it exists and has
    -- an i'th field, otherwise name2.
    -- Fields are separated with "!" and are not empty.
    -- A field for a unit using an SI prefix has the prefix name inserted,
    -- replacing '#' if found, or before the field otherwise.
    local vname
    if clean == '1' then
        vname = unit_table.name1
    elseif unit_table.varname then
        local i
        if clean == '2' then
            i = 1
        elseif clean == '3' or clean == '4' then
            i = 2
        elseif clean:find('.', 1, true) then
            i = 4
        else
            i = 3
        end
        if i > 1 and varname == 'pl' then
            i = i - 1
        end
        vname = split(unit_table.varname, '!')[i]
    end
    if vname then
        local si_name = rawget(unit_table, 'si_name') or ''
        local pos = vname:find('#', 1, true)
        if pos then
            vname = vname:sub(1, pos - 1) .. si_name .. vname:sub(pos)
        else
            vname = si_name .. vname
        end
    end
    return vname
end
return unit_table.name2
end
```



```
local function linked_id(parms, unit_table, key_id, want_link, clean)
  -- Return final unit id (symbol or name), optionally with a wikilink,
  -- and update unit_table.sep if required.
  -- key_id is one of: 'symbol', 'sym_us', 'name1', 'name1_us', 'name2', 'name2_us'
  local abbr_on = (key_id == 'symbol' or key_id == 'sym_us')
  if abbr_on and want_link then
    local symlink = rawget(unit_table, 'symlink')
    if symlink then
      return symlink -- for exceptions that have the linked symbol
    end
  end
end
local multiplier = rawget(unit_table, 'multiplier')
local per = unit_table.per
if per then
  local paren1, paren2 = '', '' -- possible parentheses around both
  local unit1 = per[1] -- top unit_table, or nil
  local unit2 = per[2] -- bottom unit_table
  if abbr_on then
    if not unit1 then
      unit_table.sep = '' -- no separator in "$2/acre"
    end
    if not want_link then
      local symbol = unit_table.symbol_raw
      if symbol then
        return symbol -- for exceptions that have the linked symbol
      end
    end
    if (unit2.symbol):find('.', 1, true) then
      paren1, paren2 = '(', ')'
    end
  end
  local key_id2 -- unit2 is always singular
  if key_id == 'name2' then
    key_id2 = 'name1'
  elseif key_id == 'name2_us' then
    key_id2 = 'name1_us'
  else
    key_id2 = key_id
  end
  local result
  if abbr_on then
    result = '/'
  elseif omitsep then
    result = per_word
  elseif unit1 then
    result = ' ' .. per_word .. ' '
  else
    result = per_word .. ' '
  end
  if want_link and unit_table.link then
    if abbr_on or not varname then
      result = (unit1 and linked_id(parms, unit1, key_id, want_link, clean))
    else
      result = (unit1 and variable_name(clean, unit1))
    end
    if omit_separator(result) then
      unit_table.sep = ''
    end
    return make_link(unit_table.link, result, unit_table)
  end
  if unit1 then
    result = linked_id(parms, unit1, key_id, want_link, clean)
    if unit1.sep then
```

```
        unit_table.sep = unit1.sep
    end
    elseif omitsep then
        unit_table.sep = ''
    end
    return result .. paren1 .. linked_id(parms, unit2, key_id2, want)
end
if multiplier then
    -- A multiplier (like "100" in "100km") forces the unit to be plu
    multiplier = from_en(multiplier)
    if not omitsep then
        multiplier = multiplier .. (abbr_on and '&nbsp;' or ' ')
    end
    if not abbr_on then
        if key_id == 'name1' then
            key_id = 'name2'
        elseif key_id == 'name1_us' then
            key_id = 'name2_us'
        end
    end
end
else
    multiplier = ''
end
local id = unit_table.fixed_name or ((varname and not abbr_on) and variat
if omit_separator(id) then
    unit_table.sep = ''
end
if want_link then
    local link = data_code.link_exceptions[unit_table.linkey or unit
    if link then
        local before = ''
        local i = unit_table.customary
        if i == 1 and parms.opt_sp_us then
            i = 2 -- show "U.S." not "US"
        end
        if i == 3 and abbr_on then
            i = 4 -- abbreviate "imperial" to "imp"
        end
        local customary = text_code.customary_units[i]
        if customary then
            -- LATER: This works for language en only, but it
            local pertext
            if id:sub(1, 1) == '/' then
                -- Want unit "/USgal" to display as "/U.S
                pertext = '/'
                id = id:sub(2)
            elseif id:sub(1, 4) == 'per ' then
                -- Similarly want "per U.S. gallon", not
                pertext = 'per '
                id = id:sub(5)
            else
                pertext = ''
            end
            -- Omit any "US"/"U.S.IMP"/"imperial" from sta
            local removes = (i < 3) and { 'US&nbsp;', 'US ',
            for _, prefix in ipairs(removes) do
                local plen = #prefix
                if id:sub(1, plen) == prefix then
                    id = id:sub(plen + 1)
                    break
                end
            end
            before = pertext .. make_link(customary.link, cus
        end
    end
end
```

```
        id = before .. make_link(link, id, unit_table)
    end
end
return multiplier .. id
end

local function make_id(parms, which, unit_table)
    -- Return id, f where
    --   id = unit name or symbol, possibly modified
    --   f = true if id is a name, or false if id is a symbol
    -- using the value for index 'which', and for 'in' or 'out' (unit_table.inout)
    -- Result is '' if no symbol/name is to be used.
    -- In addition, set unit_table.sep = ' ' or '&nbsp;' or ''
    -- (the separator that caller will normally insert before the id).
    if parms.opt_values then
        unit_table.sep = ''
        return ''
    end
    local inout = unit_table.inout
    local info = unit_table.valinfo[which]
    local abbr_org = parms.abbr_org
    local adjectival = parms.opt_adjectival
    local lk = parms.lk
    local want_link = (lk == 'on' or lk == inout)
    local username = unit_table.username
    local singular = info.singular
    local want_name
    if username then
        want_name = true
    else
        if abbr_org == nil then
            if parms.wantname then
                want_name = true
            end
            if unit_table.usesymbol then
                want_name = false
            end
        end
        if want_name == nil then
            local abbr = parms.abbr
            if abbr == 'on' or abbr == inout or (abbr == 'mos' and info.value == 1) then
                want_name = false
            else
                want_name = true
            end
        end
    end
    local key
    if want_name then
        if lk == nil and unit_table.builtin == 'hand' then
            want_link = true
        end
        if parms.opt_use_nbsp then
            unit_table.sep = '&nbsp;'
        else
            unit_table.sep = ' '
        end
        if parms.opt_singular then
            local value
            if inout == 'in' then
                value = info.value
            else
                value = info.absvalue
            end
        end
    end
end
```

```
        if value then -- some unusual units do not always set va
            value = abs(value)
            singular = (0 < value and value < 1.0001)
        end
    end
    if unit_table.engscale then
        -- engscale: so "|1|e3kg" gives "1 thousand kilograms" (p
        singular = false
    end
    key = (adjectival or singular) and 'name1' or 'name2'
    if parms.opt_sp_us then
        key = key .. '_us'
    end
else
    if unit_table.builtin == 'hand' then
        if parms.opt_hand_hh then
            unit_table.symbol = 'hh' -- LATER: might want i
        end
    end
    unit_table.sep = '&nbsp;'
    key = parms.opt_sp_us and 'sym_us' or 'symbol'
end
return linked_id(parms, unit_table, key, want_link, info.clean), want_nam
end

local function decorate_value(parms, unit_table, which, number_word)
    -- If needed, update unit_table so values will be shown with extra inform
    -- For consistency with the old template (but different from fmtpower),
    -- the style to display powers of 10 includes "display:none" to allow som
    -- browsers to copy, for example, "103" as "103", rather than as "103".
    local info
    local engscale = unit_table.engscale
    local prefix = unit_table.vprefix
    if engscale or prefix then
        info = unit_table.valinfo[which]
        if info.decorated then
            return -- do not redecorate if repeating convert
        end
        info.decorated = true
        if engscale then
            local inout = unit_table.inout
            local abbr = parms.abbr
            if (abbr == 'on' or abbr == inout) and not parms.number_v
                info.show = info.show ..
                    '<span style="margin-left:0.2em">x<span s
                    from_en('10') ..
                    '</span></span><s style="display:none">^<
                    from_en(tostring(engscale.exponent)) ..
            elseif number_word then
                local number_id
                local lk = parms.lk
                if lk == 'on' or lk == inout then
                    number_id = make_link(engscale.link, engs
                else
                    number_id = engscale[1]
                end
                -- WP:NUMERAL recommends "&nbsp;" in values like
                info.show = info.show .. (parms.opt_adjectival ar
        end
    end
    if prefix then
        info.show = prefix .. info.show
    end
end
```



```
end

local function process_input(parms, in_current)
  -- Processing required once per conversion.
  -- Return block of text to represent input (value/unit).
  if parms.opt_output_only or parms.opt_output_number_only or parms.opt_out
      parms.joins = { ' ', ' ' }
      return ''
  end
  local first_unit
  local composite = in_current.composite -- nil or table of units
  if composite then
      first_unit = composite[1]
  else
      first_unit = in_current
  end
  local id1, want_name = make_id(parms, 1, first_unit)
  local sep = first_unit.sep -- separator between value and unit, set by n
  local preunit = parms.preunit1
  if preunit then
      sep = '' -- any separator is included in preunit
  else
      preunit = ''
  end
  if parms.opt_input_unit_only then
      parms.joins = { ' ', ' ' }
      if composite then
          local parts = { id1 }
          for i, unit in ipairs(composite) do
              if i > 1 then
                  table.insert(parts, (make_id(parms, 1, unit
                      end
                  id1 = table.concat(parts, ' ')
              end
              if want_name and parms.opt_adjectival then
                  return preunit .. hyphenated(id1)
              end
              return preunit .. id1
          end
          if parms.opt_also_symbol and not composite and not parms.opt_flip then
              local join1 = parms.joins[1]
              if join1 == ' (' or join1 == ' [' then
                  parms.joins = { ' [' .. first_unit[parms.opt_sp_us and 's
              end
          end
          if in_current.builtin == 'mach' and first_unit.sep ~= '' then -- '' mean
              local prefix = id1 .. '&nbsp;'
              local range = parms.range
              local valinfo = first_unit.valinfo
              local result = prefix .. valinfo[1].show
              if range then
                  -- For simplicity and because more not needed, handle one
                  local prefix2 = make_id(parms, 2, first_unit) .. '&nbsp;'
                  result = range_text(range[1], want_name, parms, result, p
              end
              return preunit .. result
          end
          if composite then
              -- Simplify: assume there is no range, and no decoration.
              local mid = (not parms.opt_flip) and parms.mid or ''
              local sep1 = '&nbsp;'
              local sep2 = ' '
              if parms.opt_adjectival and want_name then
```

```
        sep1 = '-'
        sep2 = '-'
    end
    if omitsep and sep == '' then
        -- Testing the id of the most significant unit should be
        sep1 = ''
        sep2 = ''
    end
    local parts = { first_unit.valinfo[1].show .. sep1 .. id1 }
    for i, unit in ipairs(composite) do
        if i > 1 then
            table.insert(parts, unit.valinfo[1].show .. sep1
            end
        end
    end
    return table.concat(parts, sep2) .. mid
end
local add_unit = (parms.abbr == 'mos') or
    parms[parms.opt_flip and 'out_range_x' or 'in_range_x'] or
    (not want_name and parms.abbr_range_x)
local range = parms.range
if range and not add_unit then
    unlink(first_unit)
end
local id = range and make_id(parms, range.n + 1, first_unit) or id1
local extra, was_hyphenated = hyphenated_maybe(parms, want_name, sep, id)
if was_hyphenated then
    add_unit = false
end
local result
local valinfo = first_unit.valinfo
if range then
    for i = 0, range.n do
        local number_word
        if i == range.n then
            add_unit = false
            number_word = true
        end
        decorate_value(parms, first_unit, i+1, number_word)
        local show = valinfo[i+1].show
        if add_unit then
            show = show .. first_unit.sep .. (i == 0 and id1
            end
            if i == 0 then
                result = show
            else
                result = range_text(range[i], want_name, parms,
            end
        end
    end
else
    decorate_value(parms, first_unit, 1, true)
    result = valinfo[1].show
end
return result .. preunit .. extra
end

local function process_one_output(parms, out_current)
    -- Processing required for each output unit.
    -- Return block of text to represent output (value/unit).
    local inout = out_current.inout -- normally 'out' but can be 'in' for o
    local id1, want_name = make_id(parms, 1, out_current)
    local sep = out_current.sep -- set by make_id
    local preunit = parms.preunit2
    if preunit then
        sep = '' -- any separator is included in preunit
```

```
else
    preunit = ''
end
if parms.opt_output_unit_only then
    if want_name and parms.opt_adjectival then
        return preunit .. hyphenated(id1)
    end
    return preunit .. id1
end
if out_current.builtin == 'mach' and out_current.sep ~= '' then -- '' me
    local prefix = id1 .. '&nbsp;'
    local range = parms.range
    local valinfo = out_current.valinfo
    local result = prefix .. valinfo[1].show
    if range then
        -- For simplicity and because more not needed, handle one
        result = range_text(range[1], want_name, parms, result, p
    end
    return preunit .. result
end
local add_unit = (parms[parms.opt_flip and 'in_range_x' or 'out_range_x']
    (not want_name and parms.abbr_range_x)) and
    not parms.opt_output_number_only
local range = parms.range
if range and not add_unit then
    unlink(out_current)
end
local id = range and make_id(parms, range.n + 1, out_current) or id1
local extra, was_hyphenated = hyphenated_maybe(parms, want_name, sep, id)
if was_hyphenated then
    add_unit = false
end
local result
local valinfo = out_current.valinfo
if range then
    for i = 0, range.n do
        local number_word
        if i == range.n then
            add_unit = false
            number_word = true
        end
        decorate_value(parms, out_current, i+1, number_word)
        local show = valinfo[i+1].show
        if add_unit then
            show = show .. out_current.sep .. (i == 0 and id)
        end
        if i == 0 then
            result = show
        else
            result = range_text(range[i], want_name, parms, r
        end
    end
else
    decorate_value(parms, out_current, 1, true)
    result = valinfo[1].show
end
if parms.opt_output_number_only then
    return result
end
return result .. preunit .. extra
end

local function make_output_single(parms, in_unit_table, out_unit_table)
    -- Return true, item where item = wikitext of the conversion result
```

```
-- for a single output (which is not a combination or a multiple);
-- or return false, t where t is an error message table.
if parms.opt_order_out and in_unit_table.unitcode == out_unit_table.unitcode
    out_unit_table.valinfo = in_unit_table.valinfo
else
    out_unit_table.valinfo = collection()
    for _, v in ipairs(in_unit_table.valinfo) do
        local success, info = cvtround(parms, v, in_unit_table, out_unit_table)
        if not success then return false, info end
        out_unit_table.valinfo:add(info)
    end
end
return true, process_one_output(parms, out_unit_table)
end

local function make_output_multiple(parms, in_unit_table, out_unit_table)
-- Return true, item where item = wikitext of the conversion result
-- for an output which is a multiple (like 'ftin');
-- or return false, t where t is an error message table.
local inout = out_unit_table.inout -- normally 'out' but can be 'in' for 'ftin'
local multiple = out_unit_table.multiple -- table of scaling factors (with units)
local combos = out_unit_table.combination -- table of unit tables (with units)
local abbr = parms.abbr
local abbr_org = parms.abbr_org
local disp = parms.disp
local want_name = (abbr_org == nil and (disp == 'or' or disp == 'slash'))
                    not (abbr == 'on' or abbr == inout)
local want_link = (parms.lk == 'on' or parms.lk == inout)
local mid = parms.opt_flip and parms.mid or ''
local sep1 = '&nbsp;-'
local sep2 = ''
if parms.opt_adjectival and want_name then
    sep1 = '- '
    sep2 = ' '
end
local do_spell = parms.opt_spell_out
parms.opt_spell_out = nil -- so the call to cvtround does not spell the units
local function make_result(info, isfirst)
    local fmt, outvalue, sign
    local results = {}
    for i = 1, #combos do
        local tfrac, thisvalue, strforce
        local out_current = combos[i]
        out_current.inout = inout
        local scale = multiple[i]
        if i == 1 then -- least significant unit ('in' from 'ftin')
            local decimals
            out_current.frac = out_unit_table.frac
            local success, outinfo = cvtround(parms, info, out_current)
            if not success then return false, outinfo end
            if isfirst then
                out_unit_table.valinfo = { outinfo } --
            end
            sign = outinfo.sign
            tfrac = outinfo.fraction_table
            if outinfo.is_scientific then
                strforce = outinfo.show
                decimals = ''
            elseif tfrac then
                decimals = ''
            else
                local show = outinfo.show -- number as a string
                local p1, p2 = show:find(numdot, 1, true)
                decimals = p1 and show:sub(p2 + 1) or ''
            end
        end
        results[i] = {
            value = thisvalue,
            sign = sign,
            strforce = strforce,
            decimals = decimals,
            tfrac = tfrac,
            is_scientific = outinfo.is_scientific,
            show = outinfo.show,
            fraction_table = outinfo.fraction_table,
        }
    end
    return true, results
end
```

```
end
fmt = '%.' .. ulen(decimals) .. 'f' -- to repro
if decimals == '' then
  if tfrac then
    outvalue = floor(outinfo.raw_absv
  else
    outvalue = floor(outinfo.raw_absv
  end
else
  outvalue = outinfo.absvalue
end
end
if scale then
  outvalue, thisvalue = divide(outvalue, scale)
else
  thisvalue = outvalue
end
local id
if want_name then
  if varname then
    local clean
    if strforce or tfrac then
      clean = '.1' -- dummy value to
    else
      clean = format(fmt, thisvalue)
    end
    id = variable_name(clean, out_current)
  else
    local key = 'name2'
    if parms.opt_adjectival then
      key = 'name1'
    elseif tfrac then
      if thisvalue == 0 then
        key = 'name1'
      end
    elseif parms.opt_singular then
      if 0 < thisvalue and thisvalue <
        key = 'name1'
      end
    else
      if thisvalue == 1 then
        key = 'name1'
      end
    end
    id = out_current[key]
  end
end
else
  id = out_current['symbol']
end
if i == 1 and omit_separator(id) then
  -- Testing the id of the least significant unit s
  sep1 = ''
  sep2 = ''
end
if want_link then
  local link = out_current.link
  if link then
    id = make_link(link, id, out_current)
  end
end
local strval
local spell_inout = (i == #combos or outvalue == 0) and
if strforce and outvalue == 0 then
  sign = '' -- any sign is in strforce
```

```
        strval = strforce -- show small values in scient
elseif tfrac then
    local wholestr = (thisvalue > 0) and tostring(th
    strval = format_fraction(parms, spell_inout, fals
else
    strval = (thisvalue == 0) and from_en('0') or wit
    if do_spell then
        strval = spell_number(parms, spell_inout
    end
end
table.insert(results, strval .. sep1 .. id)
if outvalue == 0 then
    break
end
fmt = '%.0f' -- only least significant unit can have a r
end
local reversed, count = {}, #results
for i = 1, count do
    reversed[i] = results[count + 1 - i]
end
return true, sign .. table.concat(reversed, sep2)
end
local valinfo = in_unit_table.valinfo
local success, result = make_result(valinfo[1], true)
if not success then return false, result end
local range = parms.range
if range then
    for i = 1, range.n do
        local success, result2 = make_result(valinfo[i+1])
        if not success then return false, result2 end
        result = range_text(range[i], want_name, parms, result,
    end
end
return true, result .. mid
end
end

local function process(parms, in_unit_table, out_unit_table)
    -- Return true, s, outunit where s = final wikitext result,
    -- or return false, t where t is an error message table.
    linked_pages = {}
    local success, bad_output
    local bad_input_mcode = in_unit_table.bad_mcode -- nil if input unit is
    local out_unit = parms.out_unit
    if out_unit == nil or out_unit == '' or type(out_unit) == 'function' then
        if bad_input_mcode or parms.opt_input_unit_only then
            bad_output = ''
        else
            local getdef = type(out_unit) == 'function' and out_unit
            success, out_unit = getdef(in_unit_table.valinfo[1].value
            parms.out_unit = out_unit
            if not success then
                bad_output = out_unit
            end
        end
    end
end
if not bad_output and not out_unit_table then
    success, out_unit_table = lookup(parms, out_unit, 'any_combinatio
    if success then
        local mismatch = check_mismatch(in_unit_table, out_unit_t
        if mismatch then
            bad_output = mismatch
        end
    end
else
    bad_output = out_unit_table
end
```



```
end
end
local lhs, rhs
local flipped = parms.opt_flip and not bad_input_mcode
if bad_output then
    rhs = (bad_output == '') and '' or message(parms, bad_output)
elseif parms.opt_input_unit_only then
    rhs = ''
else
    local combos -- nil (for 'ft' or 'ftin'), or table of unit table
    if not out_unit_table.multiple then -- nil/false ('ft' or 'm ft')
        combos = out_unit_table.combination
    end
    local frac = parms.frac -- nil or denominator of fraction for out
    if frac then
        -- Apply fraction to the unit (if only one), or to non-SI
        -- except that if a precision is also specified, the frac
        -- the hand unit; that allows the following result:
        -- {{convert|156|cm|in hand|1|frac=2}} → 156 centimetres
        -- However, the following is handled elsewhere as a special case
        -- {{convert|156|cm|hand in|1|frac=2}} → 156 centimetres
        if combos then
            local precision = parms.precision
            for _, unit in ipairs(combos) do
                if unit.builtin == 'hand' or (not precision and unit.builtin == 'in') then
                    unit.frac = frac
                end
            end
        else
            out_unit_table.frac = frac
        end
    end
    local outputs = {}
    local imax = combos and #combos or 1 -- 1 (single unit) or number of units
    if imax == 1 then
        parms.opt_order_out = nil -- only useful with an output table
    end
    if not flipped and not parms.opt_order_out then
        -- Process left side first so any duplicate links (from left to right)
        -- on right. Example: {{convert|28|e9pc|e9ly|abbr=off|link=|}}
        lhs = process_input(parms, in_unit_table)
    end
    for i = 1, imax do
        local success, item
        local out_current = combos and combos[i] or out_unit_table
        out_current.inout = 'out'
        if i == 1 then
            if imax > 1 and out_current.builtin == 'hand' then
                out_current.out_next = combos[2] -- builtin unit
            end
            if parms.opt_order_out then
                out_current.inout = 'in'
            end
        end
        if out_current.multiple then
            success, item = make_output_multiple(parms, in_unit_table, out_current)
        else
            success, item = make_output_single(parms, in_unit_table, out_current)
        end
        if not success then return false, item end
        outputs[i] = item
    end
    if parms.opt_order_out then
        lhs = outputs[1]
    end
end
```

```
        table.remove(outputs, 1)
    end
    local sep = parms.table_joins and parms.table_joins[2] or parms.
    rhs = table.concat(outputs, sep)
end
if flipped or not lhs then
    local input = process_input(parms, in_unit_table)
    if flipped then
        lhs = rhs
        rhs = input
    else
        lhs = input
    end
end
if parms.join_before then
    lhs = parms.join_before .. lhs
end
local wikitext
if bad_input_mcode then
    if bad_input_mcode == '' then
        wikitext = lhs
    else
        wikitext = lhs .. message(parms, bad_input_mcode)
    end
elseif parms.table_joins then
    wikitext = parms.table_joins[1] .. lhs .. parms.table_joins[2]
else
    wikitext = lhs .. parms.joins[1] .. rhs .. parms.joins[2]
end
if parms.warnings and not bad_input_mcode then
    wikitext = wikitext .. parms.warnings
end
return true, get_styles(parms) .. wikitext, out_unit_table
end

local function main_convert(frame)
    -- Do convert, and if needed, do it again with higher default precision.
    local parms = { frame = frame } -- will hold template arguments, after
    set_config(frame.args)
    local success, result = get_parms(parms, frame:getParent().args)
    if success then
        if type(result) ~= 'table' then
            return tostring(result)
        end
        local in_unit_table = result
        local out_unit_table
        for _ = 1, 2 do -- use counter so cannot get stuck repeating con
            success, result, out_unit_table = process(parms, in_unit
            if success and parms.do_convert_again then
                parms.do_convert_again = false
            else
                break
            end
        end
    end
    -- If input=x gives a problem, the result should be just the user input
    -- (if x is a property like P123 it has been replaced with '').
    -- An unknown input unit would display the input and an error message
    -- with success == true at this point.
    -- Also, can have success == false with a message that outputs an empty
    if parms.input_text then
        if success and not parms.have_problem then
            return result
        end
    end
end
```



```
        local cat
        if parms.tracking then
            -- Add a tracking category using the given text as the ca
            -- There is currently only one type of tracking, but in p
            -- items could be tracked, using different sort keys for
            cat = wanted_category('tracking', parms.tracking)
        end
        return parms.input_text .. (cat or '')
    end
end
return success and result or message(parms, result)
end

local function _unit(unitcode, options)
    -- Helper function for Module:Val to look up a unit.
    -- Parameter unitcode must be a string to identify the wanted unit.
    -- Parameter options must be nil or a table with optional fields:
    --   value = number (for sort key; default value is 1)
    --   scaled_top = nil for a normal unit, or a number for a unit which is
    --               the denominator of a per unit (for sort key)
    --   si = { 'symbol', 'link' }
    --         (a table with two strings) to make an SI unit
    --         that will be used for the look up
    --   link = true if result should be [[linked]]
    --   sort = 'on' or 'debug' if result should include a sort key in a
    --         span element ('debug' makes the key visible)
    --   name = true for the name of the unit instead of the symbol
    --   us = true for the US spelling of the unit, if any
    -- Return nil if unitcode is not a non-empty string.
    -- Otherwise return a table with fields:
    --   text = requested symbol or name of unit, optionally linked
    --   scaled_value = input value adjusted by unit scale; used for sort key
    --   sortspan = span element with sort key like that provided by {{ntsh}}
    --               calculated from the result of converting value
    --               to a base unit with scale 1.
    --   unknown = true if the unitcode was not known
    unitcode = strip(unitcode)
    if unitcode == nil or unitcode == '' then
        return nil
    end
    set_config({})
    linked_pages = {}
    options = options or {}
    local parms = {
        abbr = options.name and 'off' or 'on',
        lk = options.link and 'on' or nil,
        opt_sp_us = options.us and true or nil,
        opt_ignore_error = true, -- do not add pages using this function
        opt_sortable_on = options.sort == 'on' or options.sort == 'debug',
        opt_sortable_debug = options.sort == 'debug',
    }
    if options.si then
        -- Make a dummy table of units (just one unit) for lookup to use
        -- This makes lookup recognize any SI prefix in the unitcode.
        local symbol = options.si[1] or '?'
        parms.unittable = { [symbol] = {
            _name1 = symbol,
            _name2 = symbol,
            _symbol = symbol,
            utype = symbol,
            scale = symbol == 'g' and 0.001 or 1,
            prefixes = 1,
            default = symbol,
            link = options.si[2],
        }}
    end
end
```



```
end
local success, unit_table = lookup(parms, unitcode, 'no_combination')
if not success then
    unit_table = setmetatable({
        symbol = unitcode, name2 = unitcode, utype = unitcode,
        scale = 1, default = '', defkey = '', linkey = '' }, unit
end
local value = tonumber(options.value) or 1
local clean = tostring(abs(value))
local info = {
    value = value,
    altvalue = value,
    singular = (clean == '1'),
    clean = clean,
    show = clean,
}
unit_table.inout = 'in'
unit_table.valinfo = { info }
local sortspan, scaled_value
if options.sort then
    sortspan, scaled_value = make_table_or_sort(parms, value, info, t
end
return {
    text = make_id(parms, 1, unit_table),
    sortspan = sortspan,
    scaled_value = scaled_value,
    unknown = not success and true or nil,
}
end

return { convert = main_convert, _unit = _unit }
```

Modul:Convert/sandbox/Doku

Dies ist die Dokumentationsseite für **Modul:Convert/sandbox**

When making a change, copy the current modules to the sandbox pages, then edit the sandbox copies.

If wanted, use the purge link just above to update the following status report.

- [Module:Convert](#) • [Module:Convert/sandbox](#) • [same content](#)
- [Module:Convert/data](#) • [Module:Convert/data/sandbox](#) • [same content](#)
- [Module:Convert/text](#) • [Module:Convert/text/sandbox](#) • [same content](#)
- [Module:Convert/extra](#) • [Module:Convert/extra/sandbox](#) • [same content](#)
- [Module:Convert/wikidata](#) • [Module:Convert/wikidata/sandbox](#) • [same content](#)
- [Module:Convert/wikidata/data](#) • [Module:Convert/wikidata/data/sandbox](#) • [same content](#)

Use the following template to test the results (example `{{convert/sandbox|123|lb|kg}}`):

- [Template:Convert/sandbox](#) • invokes the sandbox modules and shows all warnings

Testscases:

- [Module:Convert/sandbox/testcases](#) • templates to be tested, with expected outputs
- [Module talk:Convert/sandbox/testcases](#) • view test results
- [Template:Convert/testcases#Sandbox testcases](#) • more tests

It is not necessary to save a module before viewing test results. For example, [Module:Convert/sandbox](#) could be edited. While still editing that page, paste

[Vorlage:Nowrap](#)

into the page title box under "Preview page with this template", then click "Show preview".



Modul:Convert/sandbox/testcases

Die Dokumentation für dieses Modul kann unter *Modul:Convert/sandbox/testcases/Doku* erstellt werden

```

-- Tests for sandboxed convert.
-- [[Module talk:Convert/sandbox/testcases]] contains:
-- {{#invoke:convert/sandbox/testcases|run_tests}}

local tests = {[
-- Table cell items.
{{convert/sandbox|0.16|/l|2|disp=table}}          align="right"|0.16\n|ali
{{convert/sandbox|0.17|/l|2|disp=table}}          align="right"|0.17\n|ali
{{convert/sandbox|9999|/l|2|disp=table}}          align="right"|9,999\n|al
{{convert/sandbox|9999|/l|2|disp=table|cen}}      align="center"|9,999\n|a

-- Error/warning messages.

-- Using "test=msg" with the unit codes shown here patches the units data to temp
{{convert/sandbox|123|chain|test=msg}}            123 chains (<sup class="
{{convert/sandbox|123|rd|test=msg}}              123 rods (<sup class="no

-- Missing/invalid.
{{convert/sandbox}}                              <sup class="noprint Inli
{{convert/sandbox|}}                              <sup class="noprint Inli
{{convert/sandbox| |}}                            <sup class="noprint Inli
{{convert/sandbox|x|m}}                            <sup class="noprint Inli
{{convert/sandbox|12}}                            12<sup class="noprint Ir
{{convert/sandbox|1.2e310|m|mm}}                  <sup class="noprint Inli
{{convert/sandbox|12|ftin|m}}                     12 ftin<sup class="nopri
{{convert/sandbox|12|xyz|m}}                      12 xyz<sup class="nopri
{{convert/sandbox|ft|m}}                          <sup class="noprint Inli
{{convert/sandbox|12|to|ft|m}}                    <sup class="noprint Inli
{{convert/sandbox|X12|ft|m}}                      <sup class="noprint Inli
{{convert/sandbox|12|to|X34|ft|m}}                <sup class="noprint Inli
{{convert/sandbox|1234|ft|kg}}                    1,234 feet (<sup class="
{{convert/sandbox|1|L100km}}                       1 L100km<sup
{{convert/sandbox|1|gallons}}                      1 gallons<sup
{{convert/sandbox|1|gallon}}                      1 gallon<sup
{{convert/sandbox|1|kilogram}}                    1 kilogram<sup
{{convert/sandbox|1|light-years}}                 1 light-year
{{convert/sandbox|1|light-year}}                 1 light-year
{{convert/sandbox|1|meters}}                      1 meters<sup
{{convert/sandbox|1|meter}}                      1 meter<sup
{{convert/sandbox|1|metres}}                      1 metres<sup
{{convert/sandbox|1|metre}}                      1 metre<sup
{{convert/sandbox|1|mpg}}                        1 mpg<sup cl
{{convert/sandbox|1|pt}}                          1 pt<sup cla
{{convert/sandbox|1|qt}}                          1 qt<sup cla
{{convert/sandbox|1|sq feet}}                    1 sq feet<sup
{{convert/sandbox|123|m|m|999}}                  <sup class="noprint Inli
{{convert/sandbox|1.234e-200|m|ft}}              <sup class="noprint Inli
{{convert/sandbox|123|ft|m|1.5}}                 123 feet (37&nbsp;m)<sup
{{convert/sandbox|123|m|ft|junk=}}               123 metres (
{{convert/sandbox|123|m|ft|junk=on}}             123 metres (
{{convert/sandbox|123|m|ft|adj=junk}}            123 metres (
{{convert/sandbox|123|m|ft|adj=}}               123 metres (404&nbsp;ft)
{{convert/sandbox|123|m|ft|adj=on}}             123-metre (404&nbsp;ft)

```



```

{{convert/sandbox|123|m|ft|sing=on}} 123-metre (404&nbsp;ft)
{{convert/sandbox|123|m|ft|adj=off|sing=off}} 123 metres (404&nbsp;ft)
{{convert/sandbox|123|m|ft|adj=on|sing=on}} 123-metre (404&nbsp;ft)
{{convert/sandbox|123|m|ft|adj=off|sing=on}} 123 metres (404&nbsp;ft)
{{convert/sandbox|123|m|ft|adj=on|sing=off}} 123-metre (404&nbsp;ft)
{{convert/sandbox|123|mm|in|sigfig=3}} 123 millimetres (4.84&nbsp;in)
{{convert/sandbox|123|mm|in|sigfig=}} 123 millimetres (4.8&nbsp;in)
{{convert/sandbox|123|mm|in|sigfig=}} 123 millimetres (4.8&nbsp;in)
{{convert/sandbox|123|mm|in|sigfig=-1}} 123 millimetres (4.8&nbsp;in)
{{convert/sandbox|123|mm|in|sigfig=0}} 123 millimetres (4.8&nbsp;in)
{{convert/sandbox|123|ft|m|sigfig=1.5}} 123 feet (37&nbsp;m)<sup class="no
{{convert/sandbox|123|mm|in|sigfig=bogus}} 123 millimetres (4.8&nbsp;in)
{{convert/sandbox|123|mm|in|sortable=off}} 123 millimetres (4.8&nbsp;in)
{{convert/sandbox|123|mm|in|sortable=}} 123 millimetres (4.8&nbsp;in)
{{convert/sandbox|123|mm|in|sortable=bogus}} 123 millimetres (4.8&nbsp;in)
{{convert/sandbox|123|mm|in|debug=yes}} 123 millimetres (4.8&nbsp;in)
{{convert/sandbox|123|mm|in|sortable=on}} <span style="display:none"
{{convert/sandbox|123|mm|in|sortable=on|debug=yes}} <span style="border:1px
{{convert/sandbox|123|mm|in|sortable=on|debug=y}} <span style="border:1px
{{convert/sandbox|1|m/s2|m2}} 1 metre per second square
{{convert/sandbox|1|m2|ha|e}} 1 square metre per hectare
{{convert/sandbox|1|gmol|kgCO2/L}} 1 gram-mole (<sup class="no
{{convert/sandbox|1|$/m2|$/kg}} $1 per square metre (<sup class="no
{{convert/sandbox|1|f/ha|g/L}} f1 per hectare (<sup class="no
{{convert/sandbox|1|J|kJ/km}} 1 joule (<sup class="no
{{convert/sandbox|1|kJ/g|kJ/L}} 1 kilojoule per gram (<sup class="no
{{convert/sandbox|1|g/km|L/h}} 1 gram per kilometre (<sup class="no
{{convert/sandbox|1|N|L/km}} 1 newton (<sup class="no
{{convert/sandbox|1|kPa|m|m/km}} 1 kilopascal per metre (<sup class="no
{{convert/sandbox|1|m|kg/m}} 1 metre (<sup class="no
{{convert/sandbox|1|kg|t/ha}} 1 kilogram (<sup class="no
{{convert/sandbox|1|kg/kW|kg/h}} 1 kilogram per kilowatt (<sup class="no
{{convert/sandbox|1|gmol/s|/sqkm}} 1 gram-mole per second (<sup class="no
{{convert/sandbox|1|/l|PD/sqkm}} 1 per litre (<sup class="no
{{convert/sandbox|1|W|kW/t}} 1 watt (<sup class="no
{{convert/sandbox|1|Pa|Bq}} 1 pascal (<sup class="no
{{convert/sandbox|1|m/s|C}} 1 metre per second (<sup class="no
{{convert/sandbox|1|C-change|tsfc}} 1&nbsp;°C (<sup class="no
{{convert/sandbox|1|s|Nm}} 1 second (<sup class="no
{{convert/sandbox|1|m3|m3/km2}} 1 cubic metre (<sup class="no

-- Examples used at [[Help:Convert messages]].
{{convert/sandbox|123|m|ft}} 123 metres (404&nbsp;ft)
{{convert/sandbox| |m|ft}} <sup class="noprint Inl
{{convert/sandbox|10|x|20|m|ft}} 10 by 20 metres (33&nbsp;ft)
{{convert/sandbox|10|x| |m|ft}} <sup class="noprint Inl
{{convert/sandbox|10|x|20|x| |m|ft}} <sup class="noprint Inl
{{convert/sandbox|1|km|ft}} 1 kilometre (3,300&nbsp;ft)
{{convert/sandbox|km|ft}} <sup class="noprint Inl
{{convert/sandbox|1e290|m|ft}} 1e290 metres (3.3<span s
{{convert/sandbox|1e310|m|ft}} <sup class="noprint Inl
{{convert/sandbox|21455|acre|ha|2}} 21,455 acres (8,682.53&nbsp;ha)
{{convert/sandbox|21455|acre|ha|-2}} 21,455 acres (8,700&nbsp;ha)
{{convert/sandbox|21455|acre|ha|2.5}} 21,455 acres (8,683&nbsp;ha)
{{convert/sandbox|123|m|ft|2}} 123 metres (403.54&nbsp;ft)
{{convert/sandbox|123|m|ft|200}} <sup class="noprint Inl
{{convert/sandbox|123|m|ft|-2}} 123 metres (400&nbsp;ft)
{{convert/sandbox|123|m|ft|-200}} 123 metres (0&nbsp;ft)
{{convert/sandbox|1.234|m|ft}} 1.234 metres (4.05&nbsp;ft)
{{convert/sandbox|1.234e-200|m|ft}} <sup class="noprint Inl
{{convert/sandbox|1.234e-200|m|ft|sigfig=3}} 1.234e-200 metres (4.054
{{convert/sandbox|123|m|ft|sigfig=3}} 123 metres (404&nbsp;ft)
{{convert/sandbox|123|m|ft|sigfig=0}} 123 metres (404&nbsp;ft)
{{convert/sandbox|123|m|ft|sigfig=3.5}} 123 metres (404&nbsp;ft)

```



```

{{convert/sandbox|123|m|ft|sp=us}} 123 meters (404&nbsp;ft)
{{convert/sandbox|123|m|ft|sp=}} 123 metres (404&nbsp;ft)
{{convert/sandbox|123|m|ft|abbr=off}} 123 metres (404 feet)
{{convert/sandbox|123|m|ft|abr=off}} 123 metres (
{{convert/sandbox|12|ft}} 12 feet (3.7&nbsp;m)
{{convert/sandbox|12}} 12<sup class="noprint In
{{convert/sandbox|12|ft}} 12<sup class="noprint In
{{convert/sandbox|12|x|20|ft}} 12 by 20 feet (3.7&nbsp;
{{convert/sandbox|12|x|20}} 12 by 20<sup class="nop
{{convert/sandbox|12|ft|mi}} 12 feet (0.0023&nbsp;mi)
{{convert/sandbox|12|Ft|mi}} 12 Ft<sup class="noprint
{{convert/sandbox|12|ft|m i}} 12 feet (<sup class="nop
{{convert/sandbox|123|psi|Pa}} 123 pounds per square in
{{convert/sandbox|123|psi|ha}} 123 pounds per square in
{{convert/sandbox|21|mpgus|L/km}} 21 miles per US gallon (
{{convert/sandbox|21|mpg|L/km}} 21 mpg<sup class="nop
{{convert/sandbox|21|USpt|L}} 21 US pints (9.9&nbsp;L)
{{convert/sandbox|21|pt|L}} 21 pt<sup class="nop
{{convert/sandbox|123|K|C F}} 123&nbsp;K (-150&nbsp;°C)
{{convert/sandbox|123|C F|K}} 123 C F<sup class="nopri
{{convert/sandbox|12345|ft|mi km}} 12,345 feet (2.3381&nbsp;
{{convert/sandbox|12345|ft|yd+mi+km}} 12,345 feet (4,115&nbsp;
{{convert/sandbox|12345|ft|yd+mi km}} 12,345 feet (<sup class=

```

-- All 2458 "bytype" testcases as at 2013-09-24

-- acceleration

```

{{convert/sandbox|1|ft/s2|lk=on}} 1 [[foot per second squa
{{convert/sandbox|1|m/s2|ft/s2|lk=on}} 1 [[metre per second squ
{{convert/sandbox|1|g0|lk=on}} 1 [[standard gravity]] (
{{convert/sandbox|1|m/s2|g0|lk=on}} 1 [[metre per second squ
{{convert/sandbox|1|km/h/s|lk=on}} 1 [[Acceleration|kilomet
{{convert/sandbox|1|m/s2|km/h/s|lk=on}} 1 [[metre per second squ
{{convert/sandbox|1|km/hs|lk=on}} 1 [[Acceleration|kilomet
{{convert/sandbox|1|m/s2|km/hs|lk=on}} 1 [[metre per second squ
{{convert/sandbox|1|m/s2|lk=on}} 1 [[metre per second squ
{{convert/sandbox|1|mph/s|lk=on}} 1 [[Acceleration|mile pe
{{convert/sandbox|1|m/s2|mph/s|lk=on}} 1 [[metre per second squ
{{convert/sandbox|1|m/s2|standard gravity|lk=on}} 1 [[metre per second squ

```

-- area

```

{{convert/sandbox|1|acre|lk=on}} 1 [[acre]] (0.40&nbsp;[
{{convert/sandbox|1|m2|acre|lk=on}} 1 [[square metre]] (0.00
{{convert/sandbox|1|m2|acre ha|lk=on}} 1 [[square metre]] (0.00
{{convert/sandbox|1|m2|acre m2|lk=on}} 1 [[square metre]] (0.00
{{convert/sandbox|1|m2|acre sqm|lk=on}} 1 [[square metre]] (0.00
{{convert/sandbox|1|m2|acre sqmi|lk=on}} 1 [[square metre]] (0.00
{{convert/sandbox|1|acres|lk=on}} 1 [[acre]] (0.40&nbsp;[
{{convert/sandbox|1|m2|acres|lk=on}} 1 [[square metre]] (0.00
{{convert/sandbox|1|acre-sing|lk=on}} 1 [[acre]] (0.40&nbsp;[
{{convert/sandbox|1|m2|acre-sing|lk=on}} 1 [[square metre]] (0.00
{{convert/sandbox|1|are|lk=on}} 1 [[Hectare#Are|are]] (1
{{convert/sandbox|1|m2|are|lk=on}} 1 [[square metre]] (0.01
{{convert/sandbox|1|arpent|lk=on}} 1 [[arpent]] (0.34&nbsp;[
{{convert/sandbox|1|m2|arpent|lk=on}} 1 [[square metre]] (0.00
{{convert/sandbox|1|cm2|lk=on}} 1 [[square centimetre]]
{{convert/sandbox|1|m2|cm2|lk=on}} 1 [[square metre]] (10,0
{{convert/sandbox|1|m2|cm2 in2|lk=on}} 1 [[square metre]] (10,0
{{convert/sandbox|1|m2|cm2 sqin|lk=on}} 1 [[square metre]] (10,0
{{convert/sandbox|1|m2|Cyriot donum|lk=on}} 1 [[square metre]] (0.00
{{convert/sandbox|1|m2|Cyriot dönüm|lk=on}} 1 [[square metre]] (0.00
{{convert/sandbox|1|m2|Cyriot donum diaeresis|lk=on}} 1 [[square metre]] (0.00
{{convert/sandbox|1|m2|Cyriot donum dots|lk=on}} 1 [[square metre]] (0.00
{{convert/sandbox|1|m2|Cyriot dunam|lk=on}} 1 [[square metre]] (0.00

```



{{convert/sandbox 1 m2 Cypriot dunum lk=on}}	1	[[square metre]] (0.0010 decare)
{{convert/sandbox 1 daa lk=on}}	1	[[decare]] (0.0010 decare)
{{convert/sandbox 1 m2 daa lk=on}}	1	[[square metre]] (0.0010 decare)
{{convert/sandbox 1 dam2 lk=on}}	1	[[Square metre square metre]] (0.0010 decare)
{{convert/sandbox 1 m2 dam2 lk=on}}	1	[[square metre]] (0.0010 decare)
{{convert/sandbox 1 decare lk=on}}	1	[[decare]] (0.0010 decare)
{{convert/sandbox 1 m2 decare lk=on}}	1	[[square metre]] (0.0010 decare)
{{convert/sandbox 1 dm2 lk=on}}	1	[[square decimetre]] (0.0001 decare)
{{convert/sandbox 1 m2 dm2 lk=on}}	1	[[square metre]] (10000 square decimetre)
{{convert/sandbox 1 donum lk=on}}	1	[[Dunam donum]] (0.0010 decare)
{{convert/sandbox 1 m2 donum lk=on}}	1	[[square metre]] (0.0010 decare)
{{convert/sandbox 1 dönüm lk=on}}	1	[[Dunam dönüm]] (0.0010 decare)
{{convert/sandbox 1 m2 dönüm lk=on}}	1	[[square metre]] (0.0010 decare)
{{convert/sandbox 1 m2 donum diaeresis lk=on}}	1	[[square metre]] (0.0010 decare)
{{convert/sandbox 1 m2 donum dots lk=on}}	1	[[square metre]] (0.0010 decare)
{{convert/sandbox 1 dunam lk=on}}	1	[[dunam]] (0.0010 decare)
{{convert/sandbox 1 m2 dunam lk=on}}	1	[[square metre]] (0.0010 decare)
{{convert/sandbox 1 dunum lk=on}}	1	[[Dunam dunum]] (0.0010 decare)
{{convert/sandbox 1 m2 dunum lk=on}}	1	[[square metre]] (0.0010 decare)
{{convert/sandbox 1 foot2 lk=on}}	1	[[square foot]] (0.09290304 square metre)
{{convert/sandbox 1 m2 foot2 lk=on}}	1	[[square metre]] (11 square foot)
{{convert/sandbox 1 m2 foot2 m2 lk=on}}	1	[[square metre]] (11 square foot)
{{convert/sandbox 1 ft2 lk=on}}	1	[[square foot]] (0.09290304 square metre)
{{convert/sandbox 1 m2 ft2 lk=on}}	1	[[square metre]] (11 square foot)
{{convert/sandbox 1 m2 ft2 m2 lk=on}}	1	[[square metre]] (11 square foot)
{{convert/sandbox 1 ha lk=on}}	1	[[hectare]] (2.5 [[acre]])
{{convert/sandbox 1 m2 ha lk=on}}	1	[[square metre]] (0.0001 hectare)
{{convert/sandbox 1 m2 ha acre lk=on}}	1	[[square metre]] (0.0001 hectare)
{{convert/sandbox 1 m2 ha sqmi lk=on}}	1	[[square metre]] (0.0001 hectare)
{{convert/sandbox 1 hm2 lk=on}}	1	[[Square metre square metre]] (0.0001 hectare)
{{convert/sandbox 1 m2 hm2 lk=on}}	1	[[square metre]] (0.0001 hectare)
{{convert/sandbox 1 in2 lk=on}}	1	[[square inch]] (6.5 square centimetre)
{{convert/sandbox 1 m2 in2 lk=on}}	1	[[square metre]] (1,6 square inch)
{{convert/sandbox 1 m2 in2 cm2 lk=on}}	1	[[square metre]] (1,6 square inch)
{{convert/sandbox 1 m2 in2 mm2 lk=on}}	1	[[square metre]] (1,6 square inch)
{{convert/sandbox 1 m2 Iraqi donum lk=on}}	1	[[square metre]] (0.0010 decare)
{{convert/sandbox 1 m2 Iraqi dönüm lk=on}}	1	[[square metre]] (0.0010 decare)
{{convert/sandbox 1 m2 Iraqi donum diaeresis lk=on}}	1	[[square metre]] (0.0010 decare)
{{convert/sandbox 1 m2 Iraqi donum dots lk=on}}	1	[[square metre]] (0.0010 decare)
{{convert/sandbox 1 m2 Iraqi dunam lk=on}}	1	[[square metre]] (0.0010 decare)
{{convert/sandbox 1 m2 Iraqi dunum lk=on}}	1	[[square metre]] (0.0010 decare)
{{convert/sandbox 1 m2 Irish acre lk=on}}	1	[[square metre]] (0.0001 hectare)
{{convert/sandbox 1 km2 lk=on}}	1	[[square kilometre]] (1000000 square metre)
{{convert/sandbox 1 m2 km2 lk=on}}	1	[[square metre]] (1.0 square kilometre)
{{convert/sandbox 1 km² lk=on}}	1	[[square kilometre]] (1000000 square metre)
{{convert/sandbox 1 m2 km² lk=on}}	1	[[square metre]] (1.0 square kilometre)
{{convert/sandbox 1 m2 km2 acre sqmi lk=on}}	1	[[square metre]] (1.0 square kilometre)
{{convert/sandbox 1 m2 km2 mi2 lk=on}}	1	[[square metre]] (1.0 square kilometre)
{{convert/sandbox 1 m2 km2 sqmi lk=on}}	1	[[square metre]] (1.0 square kilometre)
{{convert/sandbox 1 m2 lk=on}}	1	[[square metre]] (11 square foot)
{{convert/sandbox 1 m² lk=on}}	1	[[square metre]] (11 square foot)
{{convert/sandbox 1 m2 m2 ft2 lk=on}}	1	[[square metre]] (1.0 square metre)
{{convert/sandbox 1 m2 m2 sqft lk=on}}	1	[[square metre]] (1.0 square metre)
{{convert/sandbox 1 m2 metric donum lk=on}}	1	[[square metre]] (0.0010 decare)
{{convert/sandbox 1 m2 metric dönüm lk=on}}	1	[[square metre]] (0.0010 decare)
{{convert/sandbox 1 m2 metric donum diaeresis lk=on}}	1	[[square metre]] (0.0010 decare)
{{convert/sandbox 1 m2 metric donum dots lk=on}}	1	[[square metre]] (0.0010 decare)
{{convert/sandbox 1 m2 metric dunam lk=on}}	1	[[square metre]] (0.0010 decare)
{{convert/sandbox 1 mi2 lk=on}}	1	[[square mile]] (2.6 square kilometre)
{{convert/sandbox 1 m2 mi2 lk=on}}	1	[[square metre]] (3.9 square mile)
{{convert/sandbox 1 m2 mi2 ha lk=on}}	1	[[square metre]] (3.9 square mile)
{{convert/sandbox 1 m2 mi2 km2 lk=on}}	1	[[square metre]] (3.9 square mile)
{{convert/sandbox 1 m2 million acre lk=on}}	1	[[square metre]] (2.5 square kilometre)
{{convert/sandbox 1 m2 million acres lk=on}}	1	[[square metre]] (2.5 square kilometre)



```

{{convert/sandbox|1|m2|million hectares|lk=on}} 1 [[square metre]] (1.0&#x10;
{{convert/sandbox|1|mm2|lk=on}} 1 [[square millimetre]]
{{convert/sandbox|1|m2|mm2|lk=on}} 1 [[square metre]] (1,0&#x10;
{{convert/sandbox|1|m2|mm2 in2|lk=on}} 1 [[square metre]] (1,0&#x10;
{{convert/sandbox|1|m2|mm2 sqin|lk=on}} 1 [[square metre]] (1,0&#x10;
{{convert/sandbox|1|nmi2|lk=on}} 1 [[Nautical mile|square
{{convert/sandbox|1|m2|nmi2|lk=on}} 1 [[square metre]] (2.9&#x10;

-- area2
{{convert/sandbox|1|m2|old donum|lk=on}} 1 [[square metre]] (0.0&#x10;
{{convert/sandbox|1|m2|old dönüm|lk=on}} 1 [[square metre]] (0.0&#x10;
{{convert/sandbox|1|m2|old donum diaeresis|lk=on}} 1 [[square metre]] (0.0&#x10;
{{convert/sandbox|1|m2|old donum dots|lk=on}} 1 [[square metre]] (0.0&#x10;
{{convert/sandbox|1|m2|old dunam|lk=on}} 1 [[square metre]] (0.0&#x10;
{{convert/sandbox|1|m2|old dunum|lk=on}} 1 [[square metre]] (0.0&#x10;
{{convert/sandbox|1|pond|lk=on}} 1 [[:nl:pondemaat|ponder
{{convert/sandbox|1|m2|pond|lk=on}} 1 [[square metre]] (0.0&#x10;
{{convert/sandbox|1|pondemaat|lk=on}} 1 [[:nl:pondemaat|ponder
{{convert/sandbox|1|m2|pondemaat|lk=on}} 1 [[square metre]] (0.0&#x10;
{{convert/sandbox|1|pyeong|lk=on}} 1 [[pyeong]] (3.3&#x10;
{{convert/sandbox|1|m2|pyeong|lk=on}} 1 [[square metre]] (0.3&#x10;
{{convert/sandbox|1|rood|lk=on}} 1 [[Rood (unit)|rood]] (
{{convert/sandbox|1|m2|rood|lk=on}} 1 [[square metre]] (0.0&#x10;
{{convert/sandbox|1|m2|sq arp|lk=on}} 1 [[square metre]] (0.0&#x10;
{{convert/sandbox|1|sqfoot|lk=on}} 1 [[square foot]] (0.09&#x10;
{{convert/sandbox|1|m2|sqfoot|lk=on}} 1 [[square metre]] (11&#x10;
{{convert/sandbox|1|m2|sqfoot m2|lk=on}} 1 [[square metre]] (11&#x10;
{{convert/sandbox|1|sqft|lk=on}} 1 [[square foot]] (0.09&#x10;
{{convert/sandbox|1|m2|sqft|lk=on}} 1 [[square metre]] (11&#x10;
{{convert/sandbox|1|m2|sqft m2|lk=on}} 1 [[square metre]] (11&#x10;
{{convert/sandbox|1|sqin|lk=on}} 1 [[square inch]] (6.5&#x10;
{{convert/sandbox|1|m2|sqin|lk=on}} 1 [[square metre]] (1,6&#x10;
{{convert/sandbox|1|m2|sqin cm2|lk=on}} 1 [[square metre]] (1,6&#x10;
{{convert/sandbox|1|m2|sqin mm2|lk=on}} 1 [[square metre]] (1,6&#x10;
{{convert/sandbox|1|sqkm|lk=on}} 1 [[square kilometre]] (
{{convert/sandbox|1|m2|sqkm|lk=on}} 1 [[square metre]] (1.0&#x10;
{{convert/sandbox|1|sqm|lk=on}} 1 [[square metre]] (11&#x10;
{{convert/sandbox|1|sqmi|lk=on}} 1 [[square mile]] (2.6&#x10;
{{convert/sandbox|1|m2|sqmi|lk=on}} 1 [[square metre]] (3.9&#x10;
{{convert/sandbox|1|m2|sqmi acre|lk=on}} 1 [[square metre]] (3.9&#x10;
{{convert/sandbox|1|m2|sqmi ha|lk=on}} 1 [[square metre]] (3.9&#x10;
{{convert/sandbox|1|m2|sqmi ha km2|lk=on}} 1 [[square metre]] (3.9&#x10;
{{convert/sandbox|1|m2|sqmi km2|lk=on}} 1 [[square metre]] (3.9&#x10;
{{convert/sandbox|1|sqnmi|lk=on}} 1 [[Nautical mile|square
{{convert/sandbox|1|m2|sqnmi|lk=on}} 1 [[square metre]] (2.9&#x10;
{{convert/sandbox|1|m2|square verst|lk=on}} 1 [[square metre]] (8.8&#x10;
{{convert/sandbox|1|sqverst|lk=on}} 1 [[Verst|square verst]]
{{convert/sandbox|1|m2|sqverst|lk=on}} 1 [[square metre]] (8.8&#x10;
{{convert/sandbox|1|sqyd|lk=on}} 1 [[square yard]] (0.84&#x10;
{{convert/sandbox|1|m2|sqyd|lk=on}} 1 [[square metre]] (1.2&#x10;
{{convert/sandbox|1|tsubo|lk=on}} 1 [[Japanese units of me
{{convert/sandbox|1|m2|tsubo|lk=on}} 1 [[square metre]] (0.3&#x10;
{{convert/sandbox|1|m2|tsubo sqft|lk=on}} 1 [[square metre]] (0.3&#x10;
{{convert/sandbox|1|verst2|lk=on}} 1 [[Verst|square verst]]
{{convert/sandbox|1|m2|verst2|lk=on}} 1 [[square metre]] (8.8&#x10;
{{convert/sandbox|1|yd2|lk=on}} 1 [[square yard]] (0.84&#x10;
{{convert/sandbox|1|m2|yd2|lk=on}} 1 [[square metre]] (1.2&#x10;

-- density
{{convert/sandbox|1|g/cm3|lk=on}} 1 [[gram]] per [[cubic c
{{convert/sandbox|1|kg/m3|g/cm3|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|g/dm3|lk=on}} 1 [[Density|gram per cub
{{convert/sandbox|1|kg/m3|g/dm3|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|g/L|lk=on}} 1 [[Density|gram per lit

```



```

{{convert/sandbox|1|kg/m3|g/L|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|g/mL|lk=on}} 1 [[Density|gram per mil
{{convert/sandbox|1|kg/m3|g/mL|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|g/mL|lk=on}} 1 [[Density|gram per mil
{{convert/sandbox|1|kg/m3|g/mL|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|kg/dm3|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|kg/m3|kg/dm3|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|kg/L|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|kg/m3|kg/L|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|kg/l|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|kg/m3|kg/l|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|kg/m3|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|lb/cuft|lk=on}} 1 [[Density|pound per cu
{{convert/sandbox|1|kg/m3|lb/cuft|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|lb/cuin|lk=on}} 1 [[Density|pound per cu
{{convert/sandbox|1|kg/m3|lb/cuin|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|lb/cuyd|lk=on}} 1 [[Density|pound per cu
{{convert/sandbox|1|kg/m3|lb/cuyd|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|lb/ft3|lk=on}} 1 [[Density|pound per cu
{{convert/sandbox|1|kg/m3|lb/ft3|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|lb/impgal|lk=on}} 1 [[Density|pound per in
{{convert/sandbox|1|kg/m3|lb/impgal|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|kg/m3|lb/impgal|lb/USgal|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|lb/in3|lk=on}} 1 [[Density|pound per cu
{{convert/sandbox|1|kg/m3|lb/in3|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|lb/U.S.gal|lk=on}} 1 [[Density|pound per U
{{convert/sandbox|1|kg/m3|lb/U.S.gal|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|lb/USbu|lk=on}} 1 [[Bushel|pound per US
{{convert/sandbox|1|kg/m3|lb/USbu|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|lb/USgal|lk=on}} 1 [[Density|pound per US
{{convert/sandbox|1|kg/m3|lb/USgal|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|lb/yd3|lk=on}} 1 [[Density|pound per cu
{{convert/sandbox|1|kg/m3|lb/yd3|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|mcg/dL|lk=on}} 1 [[microgram]] per [[de
{{convert/sandbox|1|kg/m3|mcg/dL|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|mg/L|lk=on}} 1 [[Density|milligram pé
{{convert/sandbox|1|kg/m3|mg/L|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|Mg/m3|lk=on}} 1 [[Tonne|megagram]] per
{{convert/sandbox|1|kg/m3|Mg/m3|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|oz/cuin|lk=on}} 1 [[Density|ounce per cu
{{convert/sandbox|1|kg/m3|oz/cuin|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|oz/in3|lk=on}} 1 [[Density|ounce per cu
{{convert/sandbox|1|kg/m3|oz/in3|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|ug/dL|lk=on}} 1 [[microgram]] per [[de
{{convert/sandbox|1|kg/m3|ug/dL|lk=on}} 1 [[Density|kilogram per
{{convert/sandbox|1|µg/dL|lk=on}} 1 [[microgram]] per [[de
{{convert/sandbox|1|kg/m3|µg/dL|lk=on}} 1 [[Density|kilogram per

-- energy
{{convert/sandbox|1|µerg|lk=on}} 1 [[Erg|microerg]] (0.00
{{convert/sandbox|1|kJ|µerg|lk=on}} 1 [[kilojoule]] (1.0<spa
{{convert/sandbox|1|µeV|lk=on}} 1 [[Electronvolt|microe
{{convert/sandbox|1|kJ|µeV|lk=on}} 1 [[kilojoule]] (6.2<spa
{{convert/sandbox|1|µJ|lk=on}} 1 [[Joule|microjoule]] (
{{convert/sandbox|1|kJ|µJ|lk=on}} 1 [[kilojoule]] (1.0<spa
{{convert/sandbox|1|µLatm|lk=on}} 1 [[Atmosphere (unit)|mi
{{convert/sandbox|1|kJ|µLatm|lk=on}} 1 [[kilojoule]] (9,900,0
{{convert/sandbox|1|µlatm|lk=on}} 1 [[Atmosphere (unit)|mi
{{convert/sandbox|1|kJ|µlatm|lk=on}} 1 [[kilojoule]] (9,900,0
{{convert/sandbox|1|µtonTNT|lk=on}} 1 [[TNT equivalent|micro
{{convert/sandbox|1|kJ|µtonTNT|lk=on}} 1 [[kilojoule]] (0.24 [
{{convert/sandbox|1|µtTNT|lk=on}} 1 [[TNT equivalent|micro
{{convert/sandbox|1|kJ|µtTNT|lk=on}} 1 [[kilojoule]] (0.24 [
{{convert/sandbox|1|µW.h|lk=on}} 1 [[Watt-hour|microwatt

```



{{convert/sandbox 1 kJ μW.h lk=on}}	1	[[kilojoule]] (280,000
{{convert/sandbox 1 μW.h lk=on}}	1	[[Watt-hour microwatt:
{{convert/sandbox 1 kJ μW.h lk=on}}	1	[[kilojoule]] (280,000
{{convert/sandbox 1 μWh lk=on}}	1	[[Watt-hour microwatt:
{{convert/sandbox 1 kJ μWh lk=on}}	1	[[kilojoule]] (280,000
{{convert/sandbox 1 aJ lk=on}}	1	[[Joule attojoule]] (0
{{convert/sandbox 1 kJ aJ lk=on}}	1	[[kilojoule]] (1.0<spa
{{convert/sandbox 1 B.O.T.U. lk=on}}	1	[[Watt-hour Board of T
{{convert/sandbox 1 kJ B.O.T.U. lk=on}}	1	[[kilojoule]] (0.00028
{{convert/sandbox 1 bboe lk=on}}	1	[[barrel of oil equivale
{{convert/sandbox 1 kJ bboe lk=on}}	1	[[kilojoule]] (1.6<spa
{{convert/sandbox 1 BOE lk=on}}	1	[[barrel of oil equivale
{{convert/sandbox 1 kJ BOE lk=on}}	1	[[kilojoule]] (1.6<spa
{{convert/sandbox 1 Btu lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ Btu lk=on}}	1	[[kilojoule]] (0.95&nt
{{convert/sandbox 1 btu lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ btu lk=on}}	1	[[kilojoule]] (0.95&nt
{{convert/sandbox 1 BTU lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ BTU lk=on}}	1	[[kilojoule]] (0.95&nt
{{convert/sandbox 1 BTU-39F lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ BTU-39F lk=on}}	1	[[kilojoule]] (0.94&nt
{{convert/sandbox 1 Btu-39F lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ Btu-39F lk=on}}	1	[[kilojoule]] (0.94&nt
{{convert/sandbox 1 BTU-59F lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ BTU-59F lk=on}}	1	[[kilojoule]] (0.95&nt
{{convert/sandbox 1 Btu-59F lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ Btu-59F lk=on}}	1	[[kilojoule]] (0.95&nt
{{convert/sandbox 1 Btu-60F lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ Btu-60F lk=on}}	1	[[kilojoule]] (0.95&nt
{{convert/sandbox 1 BTU-60F lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ BTU-60F lk=on}}	1	[[kilojoule]] (0.95&nt
{{convert/sandbox 1 BTU-63F lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ BTU-63F lk=on}}	1	[[kilojoule]] (0.95&nt
{{convert/sandbox 1 Btu-63F lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ Btu-63F lk=on}}	1	[[kilojoule]] (0.95&nt
{{convert/sandbox 1 BTU-ISO lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ BTU-ISO lk=on}}	1	[[kilojoule]] (0.95&nt
{{convert/sandbox 1 Btu-ISO lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ Btu-ISO lk=on}}	1	[[kilojoule]] (0.95&nt
{{convert/sandbox 1 Btu-IT lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ Btu-IT lk=on}}	1	[[kilojoule]] (0.95&nt
{{convert/sandbox 1 BTU-IT lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ BTU-IT lk=on}}	1	[[kilojoule]] (0.95&nt
{{convert/sandbox 1 BTU-mean lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ BTU-mean lk=on}}	1	[[kilojoule]] (0.95&nt
{{convert/sandbox 1 Btu-mean lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ Btu-mean lk=on}}	1	[[kilojoule]] (0.95&nt
{{convert/sandbox 1 BTU-th lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ BTU-th lk=on}}	1	[[kilojoule]] (0.95&nt
{{convert/sandbox 1 Btu-th lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ Btu-th lk=on}}	1	[[kilojoule]] (0.95&nt
{{convert/sandbox 1 Cal lk=on}}	1	[[calorie]] (4.2 sp
{{convert/sandbox 1 kJ Cal lk=on}}	1	[[kilojoule]] (0.24&nt
{{convert/sandbox 1 cal lk=on}}	1	[[calorie]] (4.2 sp
{{convert/sandbox 1 kJ cal lk=on}}	1	[[kilojoule]] (240 sp
{{convert/sandbox 1 cal-15 lk=on}}	1	[[Calorie calorie (15°
{{convert/sandbox 1 kJ cal-15 lk=on}}	1	[[kilojoule]] (240 sp
{{convert/sandbox 1 Cal-15 lk=on}}	1	[[Calorie Calorie (15°
{{convert/sandbox 1 kJ Cal-15 lk=on}}	1	[[kilojoule]] (0.24&nt
{{convert/sandbox 1 Cal-IT lk=on}}	1	[[Calorie Calorie (Int
{{convert/sandbox 1 kJ Cal-IT lk=on}}	1	[[kilojoule]] (0.24&nt
{{convert/sandbox 1 cal-IT lk=on}}	1	[[Calorie calorie (Int
{{convert/sandbox 1 kJ cal-IT lk=on}}	1	[[kilojoule]] (240 sp
{{convert/sandbox 1 Cal-th lk=on}}	1	[[Calorie Calorie (the



```

{{convert/sandbox|1|kJ|Cal-th|lk=on}} 1 [[kilojoule]] (0.24&nb
{{convert/sandbox|1|cal-th|lk=on}} 1 [[Calorie|calorie (the
{{convert/sandbox|1|kJ|cal-th|lk=on}} 1 [[kilojoule]] (240&nb
{{convert/sandbox|1|ccatm|lk=on}} 1 [[Atmosphere (unit)|cu
{{convert/sandbox|1|kJ|ccatm|lk=on}} 1 [[kilojoule]] (9,900&
{{convert/sandbox|1|CHU-IT|lk=on}} 1 [[Conversion of units|
{{convert/sandbox|1|kJ|CHU-IT|lk=on}} 1 [[kilojoule]] (0.53&nb
{{convert/sandbox|1|cJ|lk=on}} 1 [[Joule|centijoule]] (
{{convert/sandbox|1|kJ|cJ|lk=on}} 1 [[kilojoule]] (100,000
{{convert/sandbox|1|cm3atm|lk=on}} 1 [[Atmosphere (unit)|cu
{{convert/sandbox|1|kJ|cm3atm|lk=on}} 1 [[kilojoule]] (9,900&
{{convert/sandbox|1|cufootatm|lk=on}} 1 [[Atmosphere (unit)|cu
{{convert/sandbox|1|kJ|cufootatm|lk=on}} 1 [[kilojoule]] (0.35&nb
{{convert/sandbox|1|cufootnaturalgas|lk=on}} 1 [[Conversion of units|
{{convert/sandbox|1|kJ|cufootnaturalgas|lk=on}} 1 [[kilojoule]] (0.00095
{{convert/sandbox|1|cuftatm|lk=on}} 1 [[Atmosphere (unit)|cu
{{convert/sandbox|1|kJ|cuftatm|lk=on}} 1 [[kilojoule]] (0.35&nb
{{convert/sandbox|1|cuftnaturalgas|lk=on}} 1 [[Conversion of units|
{{convert/sandbox|1|kJ|cuftnaturalgas|lk=on}} 1 [[kilojoule]] (0.00095
{{convert/sandbox|1|cuydatm|lk=on}} 1 [[Atmosphere (unit)|cu
{{convert/sandbox|1|kJ|cuydatm|lk=on}} 1 [[kilojoule]] (0.013&
{{convert/sandbox|1|daJ|lk=on}} 1 [[Joule|decajoule]] (2
{{convert/sandbox|1|kJ|daJ|lk=on}} 1 [[kilojoule]] (100&nb
{{convert/sandbox|1|dJ|lk=on}} 1 [[Joule|decijoule]] (0
{{convert/sandbox|1|kJ|dJ|lk=on}} 1 [[kilojoule]] (10,000&
{{convert/sandbox|1|e3BTU|lk=on}} 1&nbsp;thousand [[Britis
{{convert/sandbox|1|kJ|e3BTU|lk=on}} 1 [[kilojoule]] (0.00095
{{convert/sandbox|1|e6BTU|lk=on}} 1&nbsp;million [[Britis
{{convert/sandbox|1|kJ|e6BTU|lk=on}} 1 [[kilojoule]] (9.5<spa
{{convert/sandbox|1|Eh|lk=on}} 1 [[Hartree]] (27&nbsp;
{{convert/sandbox|1|kJ|Eh|lk=on}} 1 [[kilojoule]] (2.3<spa
{{convert/sandbox|1|EJ|lk=on}} 1 [[Joule|exajoule]] (2
{{convert/sandbox|1|kJ|EJ|lk=on}} 1 [[kilojoule]] (1.0<spa
{{convert/sandbox|1|erg|lk=on}} 1 [[erg]] (0.10&nbsp;[[
{{convert/sandbox|1|kJ|erg|lk=on}} 1 [[kilojoule]] (1.0<spa
{{convert/sandbox|1|eV|lk=on}} 1 [[electronvolt]] (0.16
{{convert/sandbox|1|kJ|eV|lk=on}} 1 [[kilojoule]] (6.2<spa
{{convert/sandbox|1|feV|lk=on}} 1 [[Electronvolt|femtoel
{{convert/sandbox|1|kJ|feV|lk=on}} 1 [[kilojoule]] (6.2<spa
{{convert/sandbox|1|fJ|lk=on}} 1 [[Joule|femtojoule]] (
{{convert/sandbox|1|kJ|fJ|lk=on}} 1 [[kilojoule]] (1.0<spa
{{convert/sandbox|1|foe|lk=on}} 1 [[Foe (unit of energy)
{{convert/sandbox|1|kJ|foe|lk=on}} 1 [[kilojoule]] (1.0<spa
{{convert/sandbox|1|ft.lbf|lk=on}} 1 [[Foot-pound (energy)|
{{convert/sandbox|1|kJ|ft.lbf|lk=on}} 1 [[kilojoule]] (740&nb
{{convert/sandbox|1|ft.lbf|lk=on}} 1 [[Foot-pound (energy)|
{{convert/sandbox|1|kJ|ft.lbf|lk=on}} 1 [[kilojoule]] (740&nb
{{convert/sandbox|1|ft.lb-f|lk=on}} 1 [[Foot-pound (energy)|
{{convert/sandbox|1|kJ|ft.lb-f|lk=on}} 1 [[kilojoule]] (740&nb
{{convert/sandbox|1|ftlb|lk=on}} 1 [[Foot-pound (energy)|
{{convert/sandbox|1|kJ|ftlb|lk=on}} 1 [[kilojoule]] (740&nb
{{convert/sandbox|1|ftlbf|lk=on}} 1 [[Foot-pound (energy)|
{{convert/sandbox|1|kJ|ftlbf|lk=on}} 1 [[kilojoule]] (740&nb
{{convert/sandbox|1|ftlb-f|lk=on}} 1 [[Foot-pound (energy)|
{{convert/sandbox|1|kJ|ftlb-f|lk=on}} 1 [[kilojoule]] (740&nb
{{convert/sandbox|1|ftpd|lk=on}} 1 [[foot-poundal]] (0.04
{{convert/sandbox|1|kJ|ftpd|lk=on}} 1 [[kilojoule]] (24,000&
-- energy2
{{convert/sandbox|1|g-cal-15|lk=on}} 1 [[Calorie|calorie (15
{{convert/sandbox|1|kJ|g-cal-15|lk=on}} 1 [[kilojoule]] (240&nb
{{convert/sandbox|1|g-cal-IT|lk=on}} 1 [[Calorie|calorie (Int
{{convert/sandbox|1|kJ|g-cal-IT|lk=on}} 1 [[kilojoule]] (240&nb
{{convert/sandbox|1|g-cal-th|lk=on}} 1 [[Calorie|calorie (the

```



{{convert/sandbox 1 kJ g-cal-th lk=on}}	1	[[kilojoule]] (240
{{convert/sandbox 1 GeV lk=on}}	1	[[Electronvolt gigaele
{{convert/sandbox 1 kJ GeV lk=on}}	1	[[kilojoule]] (6.2<spa
{{convert/sandbox 1 GJ lk=on}}	1	[[Joule gigajoule]] (2
{{convert/sandbox 1 kJ GJ lk=on}}	1	[[kilojoule]] (1.0<spa
{{convert/sandbox 1 g-kcal-15 lk=on}}	1	[[Calorie kilocalorie
{{convert/sandbox 1 kJ g-kcal-15 lk=on}}	1	[[kilojoule]] (0.24&nt
{{convert/sandbox 1 g-kcal-IT lk=on}}	1	[[Calorie kilocalorie
{{convert/sandbox 1 kJ g-kcal-IT lk=on}}	1	[[kilojoule]] (0.24&nt
{{convert/sandbox 1 g-kcal-th lk=on}}	1	[[Calorie kilocalorie
{{convert/sandbox 1 kJ g-kcal-th lk=on}}	1	[[kilojoule]] (0.24&nt
{{convert/sandbox 1 GLatm lk=on}}	1	[[Atmosphere (unit) gi
{{convert/sandbox 1 kJ GLatm lk=on}}	1	[[kilojoule]] (9.9<spa
{{convert/sandbox 1 GLatm lk=on}}	1	[[Atmosphere (unit) gi
{{convert/sandbox 1 kJ GLatm lk=on}}	1	[[kilojoule]] (9.9<spa
{{convert/sandbox 1 g-Mcal-15 lk=on}}	1	[[Calorie megacalorie
{{convert/sandbox 1 kJ g-Mcal-15 lk=on}}	1	[[kilojoule]] (0.00024
{{convert/sandbox 1 g-mcal-15 lk=on}}	1	[[Calorie millicalorie
{{convert/sandbox 1 kJ g-mcal-15 lk=on}}	1	[[kilojoule]] (240,000
{{convert/sandbox 1 g-Mcal-IT lk=on}}	1	[[Calorie megacalorie
{{convert/sandbox 1 kJ g-Mcal-IT lk=on}}	1	[[kilojoule]] (0.00024
{{convert/sandbox 1 g-mcal-IT lk=on}}	1	[[Calorie millicalorie
{{convert/sandbox 1 kJ g-mcal-IT lk=on}}	1	[[kilojoule]] (240,000
{{convert/sandbox 1 g-mcal-th lk=on}}	1	[[Calorie millicalorie
{{convert/sandbox 1 kJ g-mcal-th lk=on}}	1	[[kilojoule]] (240,000
{{convert/sandbox 1 g-Mcal-th lk=on}}	1	[[Calorie megacalorie
{{convert/sandbox 1 kJ g-Mcal-th lk=on}}	1	[[kilojoule]] (0.00024
{{convert/sandbox 1 gTNT lk=on}}	1	[[TNT equivalent gram
{{convert/sandbox 1 kJ gTNT lk=on}}	1	[[kilojoule]] (0.24 [
{{convert/sandbox 1 Gtoe lk=on}}	1	[[Tonne of oil equival
{{convert/sandbox 1 kJ Gtoe lk=on}}	1	[[kilojoule]] (2.4<spa
{{convert/sandbox 1 GtonTNT lk=on}}	1	[[TNT equivalent gigat
{{convert/sandbox 1 kJ GtonTNT lk=on}}	1	[[kilojoule]] (2.4<spa
{{convert/sandbox 1 GtTNT lk=on}}	1	[[TNT equivalent gigat
{{convert/sandbox 1 kJ GtTNT lk=on}}	1	[[kilojoule]] (2.4<spa
{{convert/sandbox 1 GW.h lk=on}}	1	[[Watt-hour gigawatt-h
{{convert/sandbox 1 kJ GW.h lk=on}}	1	[[kilojoule]] (2.8<spa
{{convert/sandbox 1 GW·h lk=on}}	1	[[Watt-hour gigawatt-h
{{convert/sandbox 1 kJ GW·h lk=on}}	1	[[kilojoule]] (2.8<spa
{{convert/sandbox 1 GWh lk=on}}	1	[[Watt-hour gigawatt-h
{{convert/sandbox 1 kJ GWh lk=on}}	1	[[kilojoule]] (2.8<spa
{{convert/sandbox 1 GW-h lk=on}}	1	[[Watt-hour gigawatt-h
{{convert/sandbox 1 kJ GW-h lk=on}}	1	[[kilojoule]] (2.8<spa
{{convert/sandbox 1 Hartree lk=on}}	1	[[Hartree]] (27 f
{{convert/sandbox 1 kJ Hartree lk=on}}	1	[[kilojoule]] (2.3<spa
{{convert/sandbox 1 hJ lk=on}}	1	[[Joule hectojoule]] (
{{convert/sandbox 1 kJ hJ lk=on}}	1	[[kilojoule]] (10 sp
{{convert/sandbox 1 hp.h lk=on}}	1	[[Horsepower horsepowe
{{convert/sandbox 1 kJ hp.h lk=on}}	1	[[kilojoule]] (0.00037
{{convert/sandbox 1 hp·h lk=on}}	1	[[Horsepower horsepowe
{{convert/sandbox 1 kJ hp·h lk=on}}	1	[[kilojoule]] (0.00037
{{convert/sandbox 1 hph lk=on}}	1	[[Horsepower horsepowe
{{convert/sandbox 1 kJ hph lk=on}}	1	[[kilojoule]] (0.00037
{{convert/sandbox 1 impgalatm lk=on}}	1	[[Atmosphere (unit) in
{{convert/sandbox 1 kJ impgalatm lk=on}}	1	[[kilojoule]] (2.2 f
{{convert/sandbox 1 in.lbf lk=on}}	1	[[Foot-pound (energy)
{{convert/sandbox 1 kJ in.lbf lk=on}}	1	[[kilojoule]] (8,900&f
{{convert/sandbox 1 in.lb-f lk=on}}	1	[[Foot-pound (energy)
{{convert/sandbox 1 kJ in.lb-f lk=on}}	1	[[kilojoule]] (8,900&f
{{convert/sandbox 1 in.ozf lk=on}}	1	[[Foot-pound (energy)
{{convert/sandbox 1 kJ in.ozf lk=on}}	1	[[kilojoule]] (140,000
{{convert/sandbox 1 in.oz-f lk=on}}	1	[[Foot-pound (energy)
{{convert/sandbox 1 kJ in.oz-f lk=on}}	1	[[kilojoule]] (140,000
{{convert/sandbox 1 in.lbf lk=on}}	1	[[Foot-pound (energy)



{{convert/sandbox 1 kJ in·lb lk=on}}	1	[[kilojoule]] (8,900&
{{convert/sandbox 1 in·lb·f lk=on}}	1	[[Foot-pound (energy)
{{convert/sandbox 1 kJ in·lb·f lk=on}}	1	[[kilojoule]] (8,900&
{{convert/sandbox 1 in·ozf lk=on}}	1	[[Foot-pound (energy)
{{convert/sandbox 1 kJ in·ozf lk=on}}	1	[[kilojoule]] (140,000
{{convert/sandbox 1 in·oz·f lk=on}}	1	[[Foot-pound (energy)
{{convert/sandbox 1 kJ in·oz·f lk=on}}	1	[[kilojoule]] (140,000
{{convert/sandbox 1 J lk=on}}	1	[[joule]] (0.24
{{convert/sandbox 1 kJ J lk=on}}	1	[[kilojoule]] (1,000&
{{convert/sandbox 1 kbboe lk=on}}	1	[[Barrel of oil equiva
{{convert/sandbox 1 kJ kbboe lk=on}}	1	[[kilojoule]] (1.6<sp
{{convert/sandbox 1 kBOE lk=on}}	1	[[Barrel of oil equiva
{{convert/sandbox 1 kJ kBOE lk=on}}	1	[[kilojoule]] (1.6<sp
{{convert/sandbox 1 kcal lk=on}}	1	[[Calorie kilocalorie]
{{convert/sandbox 1 kJ kcal lk=on}}	1	[[kilojoule]] (0.24&nt
{{convert/sandbox 1 kcal-15 lk=on}}	1	[[Calorie kilocalorie
{{convert/sandbox 1 kJ kcal-15 lk=on}}	1	[[kilojoule]] (0.24&nt
{{convert/sandbox 1 kcal-IT lk=on}}	1	[[Calorie kilocalorie
{{convert/sandbox 1 kJ kcal-IT lk=on}}	1	[[kilojoule]] (0.24&nt
{{convert/sandbox 1 kcal-th lk=on}}	1	[[Calorie kilocalorie
{{convert/sandbox 1 kJ kcal-th lk=on}}	1	[[kilojoule]] (0.24&nt
{{convert/sandbox 1 kerG lk=on}}	1	[[Erg kiloerg]] (0.100
{{convert/sandbox 1 kJ kerG lk=on}}	1	[[kilojoule]] (10,000,
{{convert/sandbox 1 keV lk=on}}	1	[[Electronvolt kiloelé
{{convert/sandbox 1 kJ keV lk=on}}	1	[[kilojoule]] (6.2<sp
{{convert/sandbox 1 kg-cal-15 lk=on}}	1	[[Calorie Calorie (15
{{convert/sandbox 1 kJ kg-cal-15 lk=on}}	1	[[kilojoule]] (0.24&nt
{{convert/sandbox 1 kg-cal-IT lk=on}}	1	[[Calorie Calorie (Int
{{convert/sandbox 1 kJ kg-cal-IT lk=on}}	1	[[kilojoule]] (0.24&nt
{{convert/sandbox 1 kg-cal-th lk=on}}	1	[[Calorie Calorie (the
{{convert/sandbox 1 kJ kg-cal-th lk=on}}	1	[[kilojoule]] (0.24&nt
{{convert/sandbox 1 kgTNT lk=on}}	1	[[TNT equivalent kilog
{{convert/sandbox 1 kJ kgTNT lk=on}}	1	[[kilojoule]] (0.00024
{{convert/sandbox 1 kJ lk=on}}	1	[[kilojoule]] (240&nbs
{{convert/sandbox 1 klatm lk=on}}	1	[[Atmosphere (unit) ki
{{convert/sandbox 1 kJ klatm lk=on}}	1	[[kilojoule]] (0.00996
{{convert/sandbox 1 kLatm lk=on}}	1	[[Atmosphere (unit) ki
{{convert/sandbox 1 kJ kLatm lk=on}}	1	[[kilojoule]] (0.00996
{{convert/sandbox 1 kt(TNT) lk=on}}	1	[[TNT equivalent kilot
{{convert/sandbox 1 kJ kt(TNT) lk=on}}	1	[[kilojoule]] (2.4<sp
{{convert/sandbox 1 ktoe lk=on}}	1	[[Tonne of oil equiva
{{convert/sandbox 1 kJ ktoe lk=on}}	1	[[kilojoule]] (2.4<sp
{{convert/sandbox 1 ktonTNT lk=on}}	1	[[TNT equivalent kilot
{{convert/sandbox 1 kJ ktonTNT lk=on}}	1	[[kilojoule]] (2.4<sp
{{convert/sandbox 1 ktTNT lk=on}}	1	[[TNT equivalent kilot
{{convert/sandbox 1 kJ ktTNT lk=on}}	1	[[kilojoule]] (2.4<sp
{{convert/sandbox 1 kW·h lk=on}}	1	[[Watt-hour kilowatt-h
{{convert/sandbox 1 kJ kW·h lk=on}}	1	[[kilojoule]] (0.00028
{{convert/sandbox 1 kW·h lk=on}}	1	[[Watt-hour kilowatt-h
{{convert/sandbox 1 kJ kW·h lk=on}}	1	[[kilojoule]] (0.00028
{{convert/sandbox 1 kWh lk=on}}	1	[[Watt-hour kilowatt-h
{{convert/sandbox 1 kJ kWh lk=on}}	1	[[kilojoule]] (0.00028
{{convert/sandbox 1 kW-h lk=on}}	1	[[Watt-hour kilowatt-h
{{convert/sandbox 1 kJ kW-h lk=on}}	1	[[kilojoule]] (0.00028
{{convert/sandbox 1 Latm lk=on}}	1	[[Atmosphere (unit) li
{{convert/sandbox 1 kJ Latm lk=on}}	1	[[kilojoule]] (9.9&nbs
{{convert/sandbox 1 latm lk=on}}	1	[[Atmosphere (unit) li
{{convert/sandbox 1 kJ latm lk=on}}	1	[[kilojoule]] (9.9&nbs
{{convert/sandbox 1 m3atm lk=on}}	1	[[Atmosphere (unit) cu
{{convert/sandbox 1 kJ m3atm lk=on}}	1	[[kilojoule]] (0.00996
{{convert/sandbox 1 MBtu lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ MBtu lk=on}}	1	[[kilojoule]] (0.00095
{{convert/sandbox 1 MBtu-39F lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ MBtu-39F lk=on}}	1	[[kilojoule]] (0.00094



```

{{convert/sandbox|1|MBTU-39F|lk=on}} 1 [[British thermal unit
{{convert/sandbox|1|kJ|MBTU-39F|lk=on}} 1 [[kilojoule]] (0.00094
{{convert/sandbox|1|MBTU-59F|lk=on}} 1 [[British thermal unit
{{convert/sandbox|1|kJ|MBTU-59F|lk=on}} 1 [[kilojoule]] (0.00095
{{convert/sandbox|1|MBtu-59F|lk=on}} 1 [[British thermal unit
{{convert/sandbox|1|kJ|MBtu-59F|lk=on}} 1 [[kilojoule]] (0.00095
{{convert/sandbox|1|MBTU-60F|lk=on}} 1 [[British thermal unit
{{convert/sandbox|1|kJ|MBTU-60F|lk=on}} 1 [[kilojoule]] (0.00095
{{convert/sandbox|1|MBtu-60F|lk=on}} 1 [[British thermal unit
{{convert/sandbox|1|kJ|MBtu-60F|lk=on}} 1 [[kilojoule]] (0.00095
{{convert/sandbox|1|MBtu-63F|lk=on}} 1 [[British thermal unit
{{convert/sandbox|1|kJ|MBtu-63F|lk=on}} 1 [[kilojoule]] (0.00095
{{convert/sandbox|1|MBTU-63F|lk=on}} 1 [[British thermal unit
{{convert/sandbox|1|kJ|MBTU-63F|lk=on}} 1 [[kilojoule]] (0.00095
{{convert/sandbox|1|MBTU-ISO|lk=on}} 1 [[British thermal unit
{{convert/sandbox|1|kJ|MBTU-ISO|lk=on}} 1 [[kilojoule]] (0.00095
{{convert/sandbox|1|MBtu-ISO|lk=on}} 1 [[British thermal unit
{{convert/sandbox|1|kJ|MBtu-ISO|lk=on}} 1 [[kilojoule]] (0.00095
{{convert/sandbox|1|MBTU-IT|lk=on}} 1 [[British thermal unit
{{convert/sandbox|1|kJ|MBTU-IT|lk=on}} 1 [[kilojoule]] (0.00095
{{convert/sandbox|1|MBtu-IT|lk=on}} 1 [[British thermal unit
{{convert/sandbox|1|kJ|MBtu-IT|lk=on}} 1 [[kilojoule]] (0.00095
{{convert/sandbox|1|MBtu-mean|lk=on}} 1 [[British thermal unit
{{convert/sandbox|1|kJ|MBtu-mean|lk=on}} 1 [[kilojoule]] (0.00095
{{convert/sandbox|1|MBTU-mean|lk=on}} 1 [[British thermal unit
{{convert/sandbox|1|kJ|MBTU-mean|lk=on}} 1 [[kilojoule]] (0.00095
{{convert/sandbox|1|MBTU-th|lk=on}} 1 [[British thermal unit
{{convert/sandbox|1|kJ|MBTU-th|lk=on}} 1 [[kilojoule]] (0.00095
{{convert/sandbox|1|MBtu-th|lk=on}} 1 [[British thermal unit
{{convert/sandbox|1|kJ|MBtu-th|lk=on}} 1 [[kilojoule]] (0.00095

-- energy3
{{convert/sandbox|1|mcal|lk=on}} 1 [[Calorie|millicalorie]
{{convert/sandbox|1|kJ|mcal|lk=on}} 1 [[kilojoule]] (240,000
{{convert/sandbox|1|Mcal|lk=on}} 1 [[Calorie|megacalorie]
{{convert/sandbox|1|kJ|Mcal|lk=on}} 1 [[kilojoule]] (0.00024
{{convert/sandbox|1|Mcal-15|lk=on}} 1 [[Calorie|megacalorie]
{{convert/sandbox|1|kJ|Mcal-15|lk=on}} 1 [[kilojoule]] (0.00024
{{convert/sandbox|1|mcal-15|lk=on}} 1 [[Calorie|millicalorie]
{{convert/sandbox|1|kJ|mcal-15|lk=on}} 1 [[kilojoule]] (240,000
{{convert/sandbox|1|mcal-IT|lk=on}} 1 [[Calorie|millicalorie]
{{convert/sandbox|1|kJ|mcal-IT|lk=on}} 1 [[kilojoule]] (240,000
{{convert/sandbox|1|Mcal-IT|lk=on}} 1 [[Calorie|megacalorie]
{{convert/sandbox|1|kJ|Mcal-IT|lk=on}} 1 [[kilojoule]] (0.00024
{{convert/sandbox|1|mcal-th|lk=on}} 1 [[Calorie|millicalorie]
{{convert/sandbox|1|kJ|mcal-th|lk=on}} 1 [[kilojoule]] (240,000
{{convert/sandbox|1|Mcal-th|lk=on}} 1 [[Calorie|megacalorie]
{{convert/sandbox|1|kJ|Mcal-th|lk=on}} 1 [[kilojoule]] (0.00024
{{convert/sandbox|1|Merg|lk=on}} 1 [[Erg|megaerg]] (0.108
{{convert/sandbox|1|kJ|Merg|lk=on}} 1 [[kilojoule]] (10,000&
{{convert/sandbox|1|merg|lk=on}} 1 [[Erg|milliery]] (0.00
{{convert/sandbox|1|kJ|merg|lk=on}} 1 [[kilojoule]] (1.0<spa
{{convert/sandbox|1|MeV|lk=on}} 1 [[Electronvolt|megaele
{{convert/sandbox|1|kJ|MeV|lk=on}} 1 [[kilojoule]] (6.2<spa
{{convert/sandbox|1|meV|lk=on}} 1 [[Electronvolt|millie
{{convert/sandbox|1|kJ|meV|lk=on}} 1 [[kilojoule]] (6.2<spa
{{convert/sandbox|1|MJ|lk=on}} 1 [[megajoule]] (0.28&nk
{{convert/sandbox|1|kJ|MJ|lk=on}} 1 [[kilojoule]] (0.0010&
{{convert/sandbox|1|mJ|lk=on}} 1 [[Joule|millijoule]] (
{{convert/sandbox|1|kJ|mJ|lk=on}} 1 [[kilojoule]] (1,000,0
{{convert/sandbox|1|Matm|lk=on}} 1 [[Atmosphere (unit)|mé
{{convert/sandbox|1|kJ|Matm|lk=on}} 1 [[kilojoule]] (9.9<spa
{{convert/sandbox|1|mLatm|lk=on}} 1 [[Atmosphere (unit)|m
{{convert/sandbox|1|kJ|mLatm|lk=on}} 1 [[kilojoule]] (9,900&

```



{{convert/sandbox 1 MLatm lk=on}}	1	[[Atmosphere (unit) mega]]
{{convert/sandbox 1 kJ MLatm lk=on}}	1	[[kilojoule]] (9.9<spa
{{convert/sandbox 1 mLatm lk=on}}	1	[[Atmosphere (unit) mi
{{convert/sandbox 1 kJ mLatm lk=on}}	1	[[kilojoule]] (9,900&
{{convert/sandbox 1 MMBtu lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ MMBtu lk=on}}	1	[[kilojoule]] (9.5<spa
{{convert/sandbox 1 MMBTU-39F lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ MMBTU-39F lk=on}}	1	[[kilojoule]] (9.4<spa
{{convert/sandbox 1 MMBtu-39F lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ MMBtu-39F lk=on}}	1	[[kilojoule]] (9.4<spa
{{convert/sandbox 1 MMBTU-59F lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ MMBTU-59F lk=on}}	1	[[kilojoule]] (9.5<spa
{{convert/sandbox 1 MMBtu-59F lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ MMBtu-59F lk=on}}	1	[[kilojoule]] (9.5<spa
{{convert/sandbox 1 MMBTU-60F lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ MMBTU-60F lk=on}}	1	[[kilojoule]] (9.5<spa
{{convert/sandbox 1 MMBtu-60F lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ MMBtu-60F lk=on}}	1	[[kilojoule]] (9.5<spa
{{convert/sandbox 1 MMBTU-63F lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ MMBTU-63F lk=on}}	1	[[kilojoule]] (9.5<spa
{{convert/sandbox 1 MMBtu-63F lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ MMBtu-63F lk=on}}	1	[[kilojoule]] (9.5<spa
{{convert/sandbox 1 MMBTU-ISO lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ MMBTU-ISO lk=on}}	1	[[kilojoule]] (9.5<spa
{{convert/sandbox 1 MMBtu-ISO lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ MMBtu-ISO lk=on}}	1	[[kilojoule]] (9.5<spa
{{convert/sandbox 1 MMBTU-IT lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ MMBTU-IT lk=on}}	1	[[kilojoule]] (9.5<spa
{{convert/sandbox 1 MMBtu-IT lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ MMBtu-IT lk=on}}	1	[[kilojoule]] (9.5<spa
{{convert/sandbox 1 MMBTU-mean lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ MMBTU-mean lk=on}}	1	[[kilojoule]] (9.5<spa
{{convert/sandbox 1 MMBtu-mean lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ MMBtu-mean lk=on}}	1	[[kilojoule]] (9.5<spa
{{convert/sandbox 1 MMBTU-th lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ MMBTU-th lk=on}}	1	[[kilojoule]] (9.5<spa
{{convert/sandbox 1 MMBtu-th lk=on}}	1	[[British thermal unit
{{convert/sandbox 1 kJ MMBtu-th lk=on}}	1	[[kilojoule]] (9.5<spa
{{convert/sandbox 1 Mt(TNT) lk=on}}	1	[[TNT equivalent megat
{{convert/sandbox 1 kJ Mt(TNT) lk=on}}	1	[[kilojoule]] (2.4<spa
{{convert/sandbox 1 Mtoe lk=on}}	1	[[Tonne of oil equival
{{convert/sandbox 1 kJ Mtoe lk=on}}	1	[[kilojoule]] (2.4<spa
{{convert/sandbox 1 mtonTNT lk=on}}	1	[[TNT equivalent milli
{{convert/sandbox 1 kJ mtonTNT lk=on}}	1	[[kilojoule]] (0.00024
{{convert/sandbox 1 MtonTNT lk=on}}	1	[[TNT equivalent megat
{{convert/sandbox 1 kJ MtonTNT lk=on}}	1	[[kilojoule]] (2.4<spa
{{convert/sandbox 1 MtTNT lk=on}}	1	[[TNT equivalent megat
{{convert/sandbox 1 kJ MtTNT lk=on}}	1	[[kilojoule]] (2.4<spa
{{convert/sandbox 1 mtTNT lk=on}}	1	[[TNT equivalent milli
{{convert/sandbox 1 kJ mtTNT lk=on}}	1	[[kilojoule]] (0.00024
{{convert/sandbox 1 MW.h lk=on}}	1	[[Watt-hour megawatt-h
{{convert/sandbox 1 kJ MW.h lk=on}}	1	[[kilojoule]] (2.8<spa
{{convert/sandbox 1 mW.h lk=on}}	1	[[Watt-hour milliwatt
{{convert/sandbox 1 kJ mW.h lk=on}}	1	[[kilojoule]] (280&nbs
{{convert/sandbox 1 MW·h lk=on}}	1	[[Watt-hour megawatt-h
{{convert/sandbox 1 kJ MW·h lk=on}}	1	[[kilojoule]] (2.8<spa
{{convert/sandbox 1 mW·h lk=on}}	1	[[Watt-hour milliwatt
{{convert/sandbox 1 kJ mW·h lk=on}}	1	[[kilojoule]] (280&nbs
{{convert/sandbox 1 MWh lk=on}}	1	[[Watt-hour milliwatt
{{convert/sandbox 1 kJ MWh lk=on}}	1	[[kilojoule]] (280&nbs
{{convert/sandbox 1 MW·h lk=on}}	1	[[Watt-hour megawatt-h
{{convert/sandbox 1 kJ MW·h lk=on}}	1	[[kilojoule]] (2.8<spa
{{convert/sandbox 1 MW-h lk=on}}	1	[[Watt-hour megawatt-h
{{convert/sandbox 1 kJ MW-h lk=on}}	1	[[kilojoule]] (2.8<spa



```

{{convert/sandbox|1|mW-h|lk=on}} 1 [[Watt-hour|milliwatt-]]
{{convert/sandbox|1|kJ|mW-h|lk=on}} 1 [[kilojoule]] (280&nb
{{convert/sandbox|1|neV|lk=on}} 1 [[Electronvolt|nanoele
{{convert/sandbox|1|kJ|neV|lk=on}} 1 [[kilojoule]] (6.2<spa
{{convert/sandbox|1|nJ|lk=on}} 1 [[Joule|nanojoule]] (2
{{convert/sandbox|1|kJ|nJ|lk=on}} 1 [[kilojoule]] (1.0<spa
{{convert/sandbox|1|PeV|lk=on}} 1 [[Electronvolt|petaele
{{convert/sandbox|1|kJ|PeV|lk=on}} 1 [[kilojoule]] (6,200,0
{{convert/sandbox|1|peV|lk=on}} 1 [[Electronvolt|picoele
{{convert/sandbox|1|kJ|peV|lk=on}} 1 [[kilojoule]] (6.2<spa
{{convert/sandbox|1|PJ|lk=on}} 1 [[Joule|petajoule]] (2
{{convert/sandbox|1|kJ|PJ|lk=on}} 1 [[kilojoule]] (1.0<spa
{{convert/sandbox|1|pJ|lk=on}} 1 [[Joule|picojoule]] (0
{{convert/sandbox|1|kJ|pJ|lk=on}} 1 [[kilojoule]] (1.0<spa
{{convert/sandbox|1|quad|lk=on}} 1 [[Quad (energy)|quadr
{{convert/sandbox|1|kJ|quad|lk=on}} 1 [[kilojoule]] (9.5<spa

-- energy4
{{convert/sandbox|1|Ry|lk=on}} 1 [[Rydberg constant|ryd
{{convert/sandbox|1|kJ|Ry|lk=on}} 1 [[kilojoule]] (4.6<spa
{{convert/sandbox|1|scc|lk=on}} 1 [[Atmosphere (unit)|st
{{convert/sandbox|1|kJ|scc|lk=on}} 1 [[kilojoule]] (9,900&
{{convert/sandbox|1|scf|lk=on}} 1 [[Atmosphere (unit)|st
{{convert/sandbox|1|kJ|scf|lk=on}} 1 [[kilojoule]] (0.35&nk
{{convert/sandbox|1|scfoot|lk=on}} 1 [[Atmosphere (unit)|st
{{convert/sandbox|1|kJ|scfoot|lk=on}} 1 [[kilojoule]] (0.35&nk
{{convert/sandbox|1|scy|lk=on}} 1 [[Atmosphere (unit)|st
{{convert/sandbox|1|kJ|scy|lk=on}} 1 [[kilojoule]] (0.013&
{{convert/sandbox|1|sl|lk=on}} 1 [[Atmosphere (unit)|st
{{convert/sandbox|1|kJ|sl|lk=on}} 1 [[kilojoule]] (9.9&nb
{{convert/sandbox|1|t(TNT)|lk=on}} 1 [[TNT equivalent|tonne
{{convert/sandbox|1|kJ|t(TNT)|lk=on}} 1 [[kilojoule]] (2.4<spa
{{convert/sandbox|1|TeV|lk=on}} 1 [[Electronvolt|teraele
{{convert/sandbox|1|kJ|TeV|lk=on}} 1 [[kilojoule]] (6.2<spa
{{convert/sandbox|1|th|lk=on}} 1 [[Conversion of units|
{{convert/sandbox|1|kJ|th|lk=on}} 1 [[kilojoule]] (0.00024
{{convert/sandbox|1|thm-EC|lk=on}} 1 [[Therm|therm (EC)]] (
{{convert/sandbox|1|kJ|thm-EC|lk=on}} 1 [[kilojoule]] (9.5<spa
{{convert/sandbox|1|thm-UK|lk=on}} 1 [[Therm|therm (UK)]] (
{{convert/sandbox|1|kJ|thm-UK|lk=on}} 1 [[kilojoule]] (9.5<spa
{{convert/sandbox|1|thm-US|lk=on}} 1 [[Therm|therm (US)]] (
{{convert/sandbox|1|kJ|thm-US|lk=on}} 1 [[kilojoule]] (9.5<spa
{{convert/sandbox|1|TJ|lk=on}} 1 [[Joule|terajoule]] (2
{{convert/sandbox|1|kJ|TJ|lk=on}} 1 [[kilojoule]] (1.0<spa
{{convert/sandbox|1|toe|lk=on}} 1 [[tonne of oil equival
{{convert/sandbox|1|kJ|toe|lk=on}} 1 [[kilojoule]] (2.4<spa
{{convert/sandbox|1|tonTNT|lk=on}} 1 [[TNT equivalent|ton e
{{convert/sandbox|1|kJ|tonTNT|lk=on}} 1 [[kilojoule]] (2.4<spa
{{convert/sandbox|1|tTNT|lk=on}} 1 [[TNT equivalent|tonne
{{convert/sandbox|1|kJ|tTNT|lk=on}} 1 [[kilojoule]] (2.4<spa
{{convert/sandbox|1|TtonTNT|lk=on}} 1 [[TNT equivalent|terat
{{convert/sandbox|1|kJ|TtonTNT|lk=on}} 1 [[kilojoule]] (2.4<spa
{{convert/sandbox|1|TtTNT|lk=on}} 1 [[TNT equivalent|terat
{{convert/sandbox|1|kJ|TtTNT|lk=on}} 1 [[kilojoule]] (2.4<spa
{{convert/sandbox|1|TW.h|lk=on}} 1 [[Watt-hour|terawatt-h
{{convert/sandbox|1|kJ|TW.h|lk=on}} 1 [[kilojoule]] (2.8<spa
{{convert/sandbox|1|TW·h|lk=on}} 1 [[Watt-hour|terawatt-h
{{convert/sandbox|1|kJ|TW·h|lk=on}} 1 [[kilojoule]] (2.8<spa
{{convert/sandbox|1|TWh|lk=on}} 1 [[Watt-hour|terawatt-h
{{convert/sandbox|1|kJ|TWh|lk=on}} 1 [[kilojoule]] (2.8<spa
{{convert/sandbox|1|TW-h|lk=on}} 1 [[Watt-hour|terawatt-h
{{convert/sandbox|1|kJ|TW-h|lk=on}} 1 [[kilojoule]] (2.8<spa
{{convert/sandbox|1|U.S.galatm|lk=on}} 1 [[Atmosphere (unit)|U
{{convert/sandbox|1|kJ|U.S.galatm|lk=on}} 1 [[kilojoule]] (2.6&nb

```



```

{{convert/sandbox|1|uerg|lk=on}} 1 [[Erg|microerg]] (0.00
{{convert/sandbox|1|kJ|uerg|lk=on}} 1 [[kilojoule]] (1.0<spa
{{convert/sandbox|1|ueV|lk=on}} 1 [[Electronvolt|microe
{{convert/sandbox|1|kJ|ueV|lk=on}} 1 [[kilojoule]] (6.2<spa
{{convert/sandbox|1|uJ|lk=on}} 1 [[Joule|microjoule]] (
{{convert/sandbox|1|kJ|uJ|lk=on}} 1 [[kilojoule]] (1.0<spa
{{convert/sandbox|1|uLatm|lk=on}} 1 [[Atmosphere (unit)|m
{{convert/sandbox|1|kJ|uLatm|lk=on}} 1 [[kilojoule]] (9,900,0
{{convert/sandbox|1|ulatm|lk=on}} 1 [[Atmosphere (unit)|m
{{convert/sandbox|1|kJ|ulatm|lk=on}} 1 [[kilojoule]] (9,900,0
{{convert/sandbox|1|USgalatm|lk=on}} 1 [[Atmosphere (unit)|US
{{convert/sandbox|1|kJ|USgalatm|lk=on}} 1 [[kilojoule]] (2.6&nb
{{convert/sandbox|1|utonTNT|lk=on}} 1 [[TNT equivalent|micro
{{convert/sandbox|1|kJ|utonTNT|lk=on}} 1 [[kilojoule]] (0.24 [
{{convert/sandbox|1|utTNT|lk=on}} 1 [[TNT equivalent|micro
{{convert/sandbox|1|kJ|utTNT|lk=on}} 1 [[kilojoule]] (0.24 [
{{convert/sandbox|1|uW.h|lk=on}} 1 [[Watt-hour|microwatt
{{convert/sandbox|1|kJ|uW.h|lk=on}} 1 [[kilojoule]] (280,000
{{convert/sandbox|1|uW·h|lk=on}} 1 [[Watt-hour|microwatt
{{convert/sandbox|1|kJ|uW·h|lk=on}} 1 [[kilojoule]] (280,000
{{convert/sandbox|1|uWh|lk=on}} 1 [[Watt-hour|microwatt
{{convert/sandbox|1|kJ|uWh|lk=on}} 1 [[kilojoule]] (280,000
{{convert/sandbox|1|uW-h|lk=on}} 1 [[Watt-hour|microwatt
{{convert/sandbox|1|kJ|uW-h|lk=on}} 1 [[kilojoule]] (280,000
{{convert/sandbox|1|W.h|lk=on}} 1 [[watt-hour]] (3.6&nb
{{convert/sandbox|1|kJ|W.h|lk=on}} 1 [[kilojoule]] (0.28&nb
{{convert/sandbox|1|W·h|lk=on}} 1 [[watt-hour]] (3.6&nb
{{convert/sandbox|1|kJ|W·h|lk=on}} 1 [[kilojoule]] (0.28&nb
{{convert/sandbox|1|Wh|lk=on}} 1 [[watt-hour]] (3.6&nb
{{convert/sandbox|1|kJ|Wh|lk=on}} 1 [[kilojoule]] (0.28&nb
{{convert/sandbox|1|W-h|lk=on}} 1 [[watt-hour]] (3.6&nb
{{convert/sandbox|1|kJ|W-h|lk=on}} 1 [[kilojoule]] (0.28&nb
{{convert/sandbox|1|YJ|lk=on}} 1 [[Joule|yottajoule]] (
{{convert/sandbox|1|kJ|YJ|lk=on}} 1 [[kilojoule]] (1.0<spa
{{convert/sandbox|1|yJ|lk=on}} 1 [[Joule|yoctojoule]] (
{{convert/sandbox|1|kJ|yJ|lk=on}} 1 [[kilojoule]] (1.0<spa
{{convert/sandbox|1|ZJ|lk=on}} 1 [[Joule|zettajoule]] (
{{convert/sandbox|1|kJ|ZJ|lk=on}} 1 [[kilojoule]] (1.0<spa
{{convert/sandbox|1|zJ|lk=on}} 1 [[Joule|zeptojoule]] (
{{convert/sandbox|1|kJ|zJ|lk=on}} 1 [[kilojoule]] (1.0<spa
{{convert/sandbox|1|uerg|lk=on}} 1 [[Erg|microerg]] (0.00
{{convert/sandbox|1|kJ|uerg|lk=on}} 1 [[kilojoule]] (1.0<spa
{{convert/sandbox|1|ueV|lk=on}} 1 [[Electronvolt|microe
{{convert/sandbox|1|kJ|ueV|lk=on}} 1 [[kilojoule]] (6.2<spa
{{convert/sandbox|1|uJ|lk=on}} 1 [[Joule|microjoule]] (
{{convert/sandbox|1|kJ|uJ|lk=on}} 1 [[kilojoule]] (1.0<spa
{{convert/sandbox|1|uLatm|lk=on}} 1 [[Atmosphere (unit)|m
{{convert/sandbox|1|kJ|uLatm|lk=on}} 1 [[kilojoule]] (9,900,0
{{convert/sandbox|1|ulatm|lk=on}} 1 [[Atmosphere (unit)|m
{{convert/sandbox|1|kJ|ulatm|lk=on}} 1 [[kilojoule]] (9,900,0
{{convert/sandbox|1|utonTNT|lk=on}} 1 [[TNT equivalent|micro
{{convert/sandbox|1|kJ|utonTNT|lk=on}} 1 [[kilojoule]] (0.24 [
{{convert/sandbox|1|utTNT|lk=on}} 1 [[TNT equivalent|micro
{{convert/sandbox|1|kJ|utTNT|lk=on}} 1 [[kilojoule]] (0.24 [
{{convert/sandbox|1|uW.h|lk=on}} 1 [[Watt-hour|microwatt
{{convert/sandbox|1|kJ|uW.h|lk=on}} 1 [[kilojoule]] (280,000
{{convert/sandbox|1|uW·h|lk=on}} 1 [[Watt-hour|microwatt
{{convert/sandbox|1|kJ|uW·h|lk=on}} 1 [[kilojoule]] (280,000
{{convert/sandbox|1|uWh|lk=on}} 1 [[Watt-hour|microwatt
{{convert/sandbox|1|kJ|uWh|lk=on}} 1 [[kilojoule]] (280,000
{{convert/sandbox|1|uW-h|lk=on}} 1 [[Watt-hour|microwatt
{{convert/sandbox|1|kJ|uW-h|lk=on}} 1 [[kilojoule]] (280,000
-- energyperlength

```



```

{{convert/sandbox|1|BTU/mi|lk=on}} 1 [[British thermal unit
{{convert/sandbox|1|kJ/km|BTU/mi|lk=on}} 1 [[kilojoule]] per [[ki
{{convert/sandbox|1|kJ/km|lk=on}} 1 [[kilojoule]] per [[ki
{{convert/sandbox|1|kJ/km|kWh/100 km|lk=on}} 1 [[kilojoule]] per [[ki
{{convert/sandbox|1|kWh/km|lk=on}} 1 [[Kilowatt hour|kilowatt
{{convert/sandbox|1|kJ/km|kWh/km|lk=on}} 1 [[kilojoule]] per [[ki
{{convert/sandbox|1|kJ/km|kWh/km kWh/mi|lk=on}} 1 [[kilojoule]] per [[ki
{{convert/sandbox|1|kJ/km|kWh/km MJ/km|lk=on}} 1 [[kilojoule]] per [[ki
{{convert/sandbox|1|kWh/mi|lk=on}} 1 [[Kilowatt hour|kilowatt
{{convert/sandbox|1|kJ/km|kWh/mi|lk=on}} 1 [[kilojoule]] per [[ki
{{convert/sandbox|1|kJ/km|MJ/100 km|lk=on}} 1 [[kilojoule]] per [[ki
{{convert/sandbox|1|MJ/km|lk=on}} 1 [[megajoule]] per [[ki
{{convert/sandbox|1|kJ/km|MJ/km|lk=on}} 1 [[kilojoule]] per [[ki
{{convert/sandbox|1|kJ/km|MJ/km kWh/km|lk=on}} 1 [[kilojoule]] per [[ki
{{convert/sandbox|1|kJ/km|MJ/km kWh/mi|lk=on}} 1 [[kilojoule]] per [[ki

-- energypermass
{{convert/sandbox|1|BTU/lb|lk=on}} 1 [[British thermal unit
{{convert/sandbox|1|kJ/kg|BTU/lb|lk=on}} 1 [[kilojoule per kilogr
{{convert/sandbox|1|cal/g|lk=on}} 1 [[calorie per gram]] (
{{convert/sandbox|1|kJ/kg|cal/g|lk=on}} 1 [[kilojoule per kilogr
{{convert/sandbox|1|Cal/g|lk=on}} 1 [[calorie]] per [[gram
{{convert/sandbox|1|kJ/kg|Cal/g|lk=on}} 1 [[kilojoule per kilogr
{{convert/sandbox|1|GJ/kg|lk=on}} 1 [[Joule|gigajoule per
{{convert/sandbox|1|kJ/kg|GJ/kg|lk=on}} 1 [[kilojoule per kilogr
{{convert/sandbox|1|J/g|lk=on}} 1 [[Joule|joule per gram
{{convert/sandbox|1|kJ/kg|J/g|lk=on}} 1 [[kilojoule per kilogr
{{convert/sandbox|1|kcal/g|lk=on}} 1 [[kilocalorie per gram
{{convert/sandbox|1|kJ/kg|kcal/g|lk=on}} 1 [[kilojoule per kilogr
{{convert/sandbox|1|kJ/g|lk=on}} 1 [[Joule|kilojoule per
{{convert/sandbox|1|kJ/kg|kJ/g|lk=on}} 1 [[kilojoule per kilogr
{{convert/sandbox|1|kJ/kg|lk=on}} 1 [[kilojoule per kilogr
{{convert/sandbox|1|ktonTNT/MT|lk=on}} 1 [[TNT equivalent|kilot
{{convert/sandbox|1|kJ/kg|ktonTNT/MT|lk=on}} 1 [[kilojoule per kilogr
{{convert/sandbox|1|ktTNT/t|lk=on}} 1 [[TNT equivalent|kilot
{{convert/sandbox|1|kJ/kg|ktTNT/t|lk=on}} 1 [[kilojoule per kilogr
{{convert/sandbox|1|MtonTNT/MT|lk=on}} 1 [[TNT equivalent|megat
{{convert/sandbox|1|kJ/kg|MtonTNT/MT|lk=on}} 1 [[kilojoule per kilogr
{{convert/sandbox|1|MtTNT/MT|lk=on}} 1 [[TNT equivalent|megat
{{convert/sandbox|1|kJ/kg|MtTNT/MT|lk=on}} 1 [[kilojoule per kilogr
{{convert/sandbox|1|TJ/kg|lk=on}} 1 [[Joule|terajoule per
{{convert/sandbox|1|kJ/kg|TJ/kg|lk=on}} 1 [[kilojoule per kilogr

-- energypervolume
{{convert/sandbox|1|BTU/cuft|lk=on}} 1 [[British thermal unit
{{convert/sandbox|1|kJ/l|BTU/cuft|lk=on}} 1 [[kilojoule]] per [[li
{{convert/sandbox|1|Cal/USoz|lk=on}} 1 [[calorie]] per [[US f
{{convert/sandbox|1|kJ/l|Cal/USoz|lk=on}} 1 [[kilojoule]] per [[li
{{convert/sandbox|1|kJ/L|lk=on}} 1 [[kilojoule]] per [[li
{{convert/sandbox|1|kJ/l|kJ/L|lk=on}} 1 [[kilojoule]] per [[li
{{convert/sandbox|1|kJ/l|lk=on}} 1 [[kilojoule]] per [[li
{{convert/sandbox|1|kJ/ml|lk=on}} 1 [[kilojoule]] per [[ml
{{convert/sandbox|1|kJ/l|kJ/ml|lk=on}} 1 [[kilojoule]] per [[li

-- exhaustemission
{{convert/sandbox|1|g/km|lk=on}} 1 [[Exhaust gas|gram per
{{convert/sandbox|1|kg/km|g/km|lk=on}} 1 [[Exhaust gas|kilogram
{{convert/sandbox|1|g/mi|lk=on}} 1 [[Exhaust gas|gram per
{{convert/sandbox|1|kg/km|g/mi|lk=on}} 1 [[Exhaust gas|kilogram
{{convert/sandbox|1|kg/km|lk=on}} 1 [[Exhaust gas|kilogram
{{convert/sandbox|1|lb/mi|lk=on}} 1 [[Exhaust gas|pound pé
{{convert/sandbox|1|kg/km|lb/mi|lk=on}} 1 [[Exhaust gas|kilogram
{{convert/sandbox|1|oz/mi|lk=on}} 1 [[Exhaust gas|ounce pé
{{convert/sandbox|1|kg/km|oz/mi|lk=on}} 1 [[Exhaust gas|kilogram

```



```
-- flow
{{convert/sandbox|1|cuft/a|lk=on}} 1 [[Cubic foot per second]]
{{convert/sandbox|1|m3/s|cuft/a|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|cuft/d|lk=on}} 1 [[Cubic foot per second]]
{{convert/sandbox|1|m3/s|cuft/d|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|cuft/h|lk=on}} 1 [[Cubic foot per second]]
{{convert/sandbox|1|m3/s|cuft/h|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|cuft/min|lk=on}} 1 [[Cubic foot#cubic foot per second]]
{{convert/sandbox|1|m3/s|cuft/min|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|cuft/s|lk=on}} 1 [[cubic foot per second]]
{{convert/sandbox|1|m3/s|cuft/s|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|cumi/a|lk=on}} 1 [[Cubic foot per second]]
{{convert/sandbox|1|m3/s|cumi/a|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|cuyd/h|lk=on}} 1 [[Cubic foot per minute]]
{{convert/sandbox|1|m3/s|cuyd/h|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|cuyd/s|lk=on}} 1 [[cubic yard per second]]
{{convert/sandbox|1|m3/s|cuyd/s|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|ft3/a|lk=on}} 1 [[Cubic foot per second]]
{{convert/sandbox|1|m3/s|ft3/a|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|ft3/d|lk=on}} 1 [[Cubic foot per second]]
{{convert/sandbox|1|m3/s|ft3/d|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|ft3/h|lk=on}} 1 [[Cubic foot per second]]
{{convert/sandbox|1|m3/s|ft3/h|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|ft3/s|lk=on}} 1 [[cubic foot per second]]
{{convert/sandbox|1|m3/s|ft3/s|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|Gcuft/a|lk=on}} 1&nbsp;[[1000000000 (number)]]
{{convert/sandbox|1|m3/s|Gcuft/a|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|Gcuft/d|lk=on}} 1&nbsp;[[1000000000 (number)]]
{{convert/sandbox|1|m3/s|Gcuft/d|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|Goilbbl/a|lk=on}} 1 [[Barrel per day|barrel]]
{{convert/sandbox|1|m3/s|Goilbbl/a|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|impgal/h|lk=on}} 1 [[Gallon|imperial gallon]]
{{convert/sandbox|1|m3/s|impgal/h|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|impgal/min|lk=on}} 1 [[Gallon|imperial gallon]]
{{convert/sandbox|1|m3/s|impgal/min|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|m3/s|impgal/min USgal/min|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|impgal/s|lk=on}} 1 [[Imperial gallons per second]]
{{convert/sandbox|1|m3/s|impgal/s|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|kcuft/a|lk=on}} 1&nbsp;thousand [[Cubic foot]]
{{convert/sandbox|1|m3/s|kcuft/a|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|kcuft/d|lk=on}} 1&nbsp;thousand [[Cubic foot]]
{{convert/sandbox|1|m3/s|kcuft/d|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|kcuft/s|lk=on}} 1&nbsp;thousand [[cubic foot]]
{{convert/sandbox|1|m3/s|kcuft/s|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|km3/a|lk=on}} 1 [[Cubic metre per second]]
{{convert/sandbox|1|m3/s|km3/a|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|koilbbl/a|lk=on}} 1 [[Barrel per day|thousand]]
{{convert/sandbox|1|m3/s|koilbbl/a|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|koilbbl/d|lk=on}} 1 [[Barrel per day|thousand]]
{{convert/sandbox|1|m3/s|koilbbl/d|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|L/min|lk=on}} 1 [[Cubic metre per second]]
{{convert/sandbox|1|m3/s|L/min|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|L/s|lk=on}} 1 [[Cubic metre per second]]
{{convert/sandbox|1|m3/s|L/s|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|m3/s|L/s impgal/min|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|m3/a|lk=on}} 1 [[Cubic metre per second]]
{{convert/sandbox|1|m3/s|m3/a|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|m3/d|lk=on}} 1 [[Cubic metre per second]]
{{convert/sandbox|1|m3/s|m3/d|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|m3/h|lk=on}} 1 [[Cubic metre per second]]
{{convert/sandbox|1|m3/s|m3/h|lk=on}} 1 [[cubic metre per second]]
{{convert/sandbox|1|m3/min|lk=on}} 1 [[Cubic metre per second]]
{{convert/sandbox|1|m3/s|m3/min|lk=on}} 1 [[cubic metre per second]]
```




{{convert/sandbox 1 N grf lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 gr-f lk=on}}	1	[[Pound-force grain-force]]
{{convert/sandbox 1 N gr-f lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 kdyn lk=on}}	1	[[Dyne kilodyne]] (0.001)
{{convert/sandbox 1 N kdyn lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 kgf lk=on}}	1	[[kilogram-force]] (9.80665)
{{convert/sandbox 1 N kgf lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 kg-f lk=on}}	1	[[kilogram-force]] (9.80665)
{{convert/sandbox 1 N kg-f lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 kN lk=on}}	1	[[Newton (unit) kilonewton]]
{{convert/sandbox 1 N kN lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 N kN lbf lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 N kN lb-f lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 N kN LTf STf lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 N kN LT-f ST-f lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 N kN STf LTf lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 N kN ST-f LT-f lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 kp lk=on}}	1	[[Kilogram-force kilopond]]
{{convert/sandbox 1 N kp lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 lbf lk=on}}	1	[[pound-force]] (4.4482216152605)
{{convert/sandbox 1 N lbf lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 lb-f lk=on}}	1	[[pound-force]] (4.4482216152605)
{{convert/sandbox 1 N lb-f lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 LTf lk=on}}	1	[[long ton-force]] (1000)
{{convert/sandbox 1 N LTf lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 LT-f lk=on}}	1	[[long ton-force]] (1000)
{{convert/sandbox 1 N LT-f lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 N LTf STf lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 N LT-f ST-f lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 mdyn lk=on}}	1	[[Dyne millidyne]] (1e-05)
{{convert/sandbox 1 N mdyn lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 Mdyn lk=on}}	1	[[Dyne megadyne]] (2e+11)
{{convert/sandbox 1 N Mdyn lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 mgf lk=on}}	1	[[Kilogram-force milligram-force]]
{{convert/sandbox 1 N mgf lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 mg-f lk=on}}	1	[[Kilogram-force milligram-force]]
{{convert/sandbox 1 N mg-f lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 MN lk=on}}	1	[[Newton (unit) meganeuton]]
{{convert/sandbox 1 N MN lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 mN lk=on}}	1	[[Newton (unit) millinewton]]
{{convert/sandbox 1 N mN lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 N mN grf lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 N mN gr-f lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 N MN LTf STf lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 N MN LT-f ST-f lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 N mN ozf lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 N mN oz-f lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 N MN STf LTf lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 N MN ST-f LT-f lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 Mp lk=on}}	1	[[Kilogram-force megapond]]
{{convert/sandbox 1 N Mp lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 mp lk=on}}	1	[[Kilogram-force millipond]]
{{convert/sandbox 1 N mp lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 N lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 N N lbf lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 N N lb-f lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 N N ozf lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 N N oz-f lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 newtons lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 N newtons lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 nN lk=on}}	1	[[Newton (unit) nanoneuton]]
{{convert/sandbox 1 N nN lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 N nN grf lk=on}}	1	[[Newton (unit) newton]]
{{convert/sandbox 1 N nN gr-f lk=on}}	1	[[Newton (unit) newton]]



```

{{convert/sandbox|1|ozf|lk=on}} 1 [[Pound-force|ounce-force]]
{{convert/sandbox|1|N|ozf|lk=on}} 1 [[Newton (unit)|newton]]
{{convert/sandbox|1|oz-f|lk=on}} 1 [[Pound-force|ounce-force]]
{{convert/sandbox|1|N|oz-f|lk=on}} 1 [[Newton (unit)|newton]]
{{convert/sandbox|1|p|lk=on}} 1 [[Kilogram-force|pond]]
{{convert/sandbox|1|N|p|lk=on}} 1 [[Newton (unit)|newton]]
{{convert/sandbox|1|pd|lk=on}} 1 [[poundal]] (0.14&nbsp;N)
{{convert/sandbox|1|N|pd|lk=on}} 1 [[Newton (unit)|newton]]
{{convert/sandbox|1|poundal|lk=on}} 1 [[poundal]] (0.14&nbsp;N)
{{convert/sandbox|1|N|poundal|lk=on}} 1 [[Newton (unit)|newton]]
{{convert/sandbox|1|S/Tf|lk=on}} 1 [[short ton-force]] (907.18473 N)
{{convert/sandbox|1|N|S/Tf|lk=on}} 1 [[Newton (unit)|newton]]
{{convert/sandbox|1|S/T-f|lk=on}} 1 [[short ton-force]] (907.18473 N)
{{convert/sandbox|1|N|S/T-f|lk=on}} 1 [[Newton (unit)|newton]]
{{convert/sandbox|1|N|S/Tf L/Tf|lk=on}} 1 [[Newton (unit)|newton]]
{{convert/sandbox|1|N|S/T-f L/T-f|lk=on}} 1 [[Newton (unit)|newton]]
{{convert/sandbox|1|STf|lk=on}} 1 [[short ton-force]] (907.18473 N)
{{convert/sandbox|1|N|STf|lk=on}} 1 [[Newton (unit)|newton]]
{{convert/sandbox|1|ST-f|lk=on}} 1 [[short ton-force]] (907.18473 N)
{{convert/sandbox|1|N|ST-f|lk=on}} 1 [[Newton (unit)|newton]]
{{convert/sandbox|1|N|STf LTf|lk=on}} 1 [[Newton (unit)|newton]]
{{convert/sandbox|1|N|ST-f LT-f|lk=on}} 1 [[Newton (unit)|newton]]
{{convert/sandbox|1|tf|lk=on}} 1 [[Ton-force#Tonne-force]]
{{convert/sandbox|1|N|tf|lk=on}} 1 [[Newton (unit)|newton]]
{{convert/sandbox|1|t-f|lk=on}} 1 [[Ton-force#Tonne-force]]
{{convert/sandbox|1|N|t-f|lk=on}} 1 [[Newton (unit)|newton]]
{{convert/sandbox|1|uN|lk=on}} 1 [[Newton (unit)|micronewton]]
{{convert/sandbox|1|N|uN|lk=on}} 1 [[Newton (unit)|newton]]
{{convert/sandbox|1|N|uN grf|lk=on}} 1 [[Newton (unit)|newton]]
{{convert/sandbox|1|N|uN gr-f|lk=on}} 1 [[Newton (unit)|newton]]
{{convert/sandbox|1|µN|lk=on}} 1 [[Newton (unit)|micronewton]]
{{convert/sandbox|1|N|µN|lk=on}} 1 [[Newton (unit)|newton]]
{{convert/sandbox|1|N|µN grf|lk=on}} 1 [[Newton (unit)|newton]]
{{convert/sandbox|1|N|µN gr-f|lk=on}} 1 [[Newton (unit)|newton]]

-- fuelconsumption
{{convert/sandbox|1|impgal/mi|lk=on}} 1 [[Imperial unit|imperial]]
{{convert/sandbox|1|l/km|impgal/mi|lk=on}} 1 [[litre]] per [[kilometre]]
{{convert/sandbox|1|l/km|impgal/mi U.S.gal/mi|lk=on}} 1 [[litre]] per [[kilometre]]
{{convert/sandbox|1|l/km|impgal/mi USgal/mi|lk=on}} 1 [[litre]] per [[kilometre]]
{{convert/sandbox|1|km/L|lk=on}} 1 [[kilometre]] per [[litre]]
{{convert/sandbox|1|l/km|km/L|lk=on}} 1 [[litre]] per [[kilometre]]
{{convert/sandbox|1|km/l|lk=on}} 1 [[kilometre]] per [[litre]]
{{convert/sandbox|1|l/km|km/l|lk=on}} 1 [[litre]] per [[kilometre]]
{{convert/sandbox|1|l/km|km/L mpgimp|lk=on}} 1 [[litre]] per [[kilometre]]
{{convert/sandbox|1|l/km|km/l mpgimp|lk=on}} 1 [[litre]] per [[kilometre]]
{{convert/sandbox|1|l/km|km/L mpgU.S.|lk=on}} 1 [[litre]] per [[kilometre]]
{{convert/sandbox|1|l/km|km/l mpgU.S.|lk=on}} 1 [[litre]] per [[kilometre]]
{{convert/sandbox|1|l/km|km/l mpgUS|lk=on}} 1 [[litre]] per [[kilometre]]
{{convert/sandbox|1|l/km|l/100 km|lk=on}} 1 [[litre]] per [[kilometre]]
{{convert/sandbox|1|l/km|L/100 km|lk=on}} 1 [[litre]] per [[kilometre]]
{{convert/sandbox|1|l/km|l/100 km mpgimp|lk=on}} 1 [[litre]] per [[kilometre]]
{{convert/sandbox|1|l/km|L/100 km mpgimp|lk=on}} 1 [[litre]] per [[kilometre]]
{{convert/sandbox|1|l/km|l/100 km mpgU.S.|lk=on}} 1 [[litre]] per [[kilometre]]
{{convert/sandbox|1|l/km|L/100 km mpgU.S.|lk=on}} 1 [[litre]] per [[kilometre]]
{{convert/sandbox|1|l/km|L/100 km mpgUS|lk=on}} 1 [[litre]] per [[kilometre]]
{{convert/sandbox|1|L/100km|lk=on}} 1 [[litre]] per 100 [[kilometre]]
{{convert/sandbox|1|l/km|L/100km|lk=on}} 1 [[litre]] per [[kilometre]]

```



```

{{convert/sandbox|1|l/km|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|l/km|l/km|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|l/km|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|l/km|l/km|impgal/mi|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|l/km|l/km|impgal/mi|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|l/km|l/km|U.S.gal/mi|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|l/km|l/km|U.S.gal/mi|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|l/km|l/km|USgal/mi|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|l/km|l/km|USgal/mi|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|l/km|l/km|usgal/mi|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|mi/impqt|lk=on}} 1 [[mile]] per [[Imperia
{{convert/sandbox|1|l/km|mi/impqt|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|mi/U.S.qt|lk=on}} 1 [[mile]] per [[United
{{convert/sandbox|1|l/km|mi/U.S.qt|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|mi/USqt|lk=on}} 1 [[mile]] per [[United
{{convert/sandbox|1|l/km|mi/USqt|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|mi/usqt|lk=on}} 1 [[mile]] per [[United
{{convert/sandbox|1|l/km|mi/usqt|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|mpgimp|lk=on}} 1 [[mile]] per [[Imperia
{{convert/sandbox|1|l/km|mpgimp|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|l/km|mpgimp|l/100 km|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|l/km|mpgimp|mpgU.S.|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|l/km|mpgimp|mpgUS|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|l/km|mpgimp|mpgus|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|mpgU.S.|lk=on}} 1 [[mile]] per [[United
{{convert/sandbox|1|l/km|mpgU.S.|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|mpgu.s.|lk=on}} 1 [[mile]] per [[United
{{convert/sandbox|1|l/km|mpgu.s.|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|l/km|mpgU.S.|mpgimp|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|mpgus|lk=on}} 1 [[mile]] per [[United
{{convert/sandbox|1|l/km|mpgus|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|mpgUS|lk=on}} 1 [[mile]] per [[United
{{convert/sandbox|1|l/km|mpgUS|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|l/km|mpgus|mpgimp|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|l/km|mpgUS|mpgimp|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|U.S.gal/mi|lk=on}} 1 [[United States custon
{{convert/sandbox|1|l/km|U.S.gal/mi|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|l/km|U.S.gal/mi|impgal/mi|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|USgal/mi|lk=on}} 1 [[United States custon
{{convert/sandbox|1|l/km|USgal/mi|lk=on}} 1 [[litre]] per [[kilomé
{{convert/sandbox|1|l/km|USgal/mi|impgal/mi|lk=on}} 1 [[litre]] per [[kilomé

-- gradient
{{convert/sandbox|1|cm/km|lk=on}} 1 [[Grade (slope)|centin
{{convert/sandbox|1|m/km|cm/km|lk=on}} 1 [[Grade (slope)|metre
{{convert/sandbox|1|ft/mi|lk=on}} 1 [[Grade (slope)|foot p
{{convert/sandbox|1|m/km|ft/mi|lk=on}} 1 [[Grade (slope)|metre
{{convert/sandbox|1|ft/nmi|lk=on}} 1 [[Grade (slope)|foot p
{{convert/sandbox|1|m/km|ft/nmi|lk=on}} 1 [[Grade (slope)|metre
{{convert/sandbox|1|in/ft|lk=on}} 1 [[Grade (slope)|inch p
{{convert/sandbox|1|m/km|in/ft|lk=on}} 1 [[Grade (slope)|metre
{{convert/sandbox|1|in/mi|lk=on}} 1 [[Grade (slope)|inch p
{{convert/sandbox|1|m/km|in/mi|lk=on}} 1 [[Grade (slope)|metre
{{convert/sandbox|1|m/km|lk=on}} 1 [[Grade (slope)|metre
{{convert/sandbox|1|mm/km|lk=on}} 1 [[Grade (slope)|millin
{{convert/sandbox|1|m/km|mm/km|lk=on}} 1 [[Grade (slope)|metre
{{convert/sandbox|1|mm/m|lk=on}} 1 [[Grade (slope)|millin
{{convert/sandbox|1|m/km|mm/m|lk=on}} 1 [[Grade (slope)|metre

-- length
{{convert/sandbox|1|µm|lk=on}} 1 [[micrometre]] (3.9<sp
{{convert/sandbox|1|m|µm|lk=on}} 1 [[metre]] (1,000,000&r
{{convert/sandbox|1|Å|lk=on}} 1 [[Angstrom|ångström]]
{{convert/sandbox|1|m|Å|lk=on}} 1 [[metre]] (1.0<span st

```




{{convert/sandbox 1 m km nmi lk=on}}	1	[[metre]] (0.0010
{{convert/sandbox 1 kpc lk=on}}	1	[[Parsec#Parsecs and k
{{convert/sandbox 1 m kpc lk=on}}	1	[[metre]] (3.2<span st
{{convert/sandbox 1 LD lk=on}}	1	[[Lunar distance (ast
{{convert/sandbox 1 m LD lk=on}}	1	[[metre]] (2.6<span st
{{convert/sandbox 1 league lk=on}}	1	[[League (unit) league
{{convert/sandbox 1 m league lk=on}}	1	[[metre]] (0.00021
{{convert/sandbox 1 ly lk=on}}	1	[[light-year]] (63,000
{{convert/sandbox 1 m ly lk=on}}	1	[[metre]] (1.1<span st
{{convert/sandbox 1 m lk=on}}	1	[[metre]] (3
{{convert/sandbox 1 m m foot lk=on}}	1	[[metre]] (1.0
{{convert/sandbox 1 m m ft lk=on}}	1	[[metre]] (1.0
{{convert/sandbox 1 m m yd lk=on}}	1	[[metre]] (1.0
{{convert/sandbox 1 mi lk=on}}	1	[[mile]] (1.6
{{convert/sandbox 1 m mi lk=on}}	1	[[metre]] (0.00062
{{convert/sandbox 1 m mi ft lk=on}}	1	[[metre]] (0.00062
{{convert/sandbox 1 m mi km lk=on}}	1	[[metre]] (0.00062
{{convert/sandbox 1 m mi nmi lk=on}}	1	[[metre]] (0.00062
{{convert/sandbox 1 micrometre lk=on}}	1	[[micrometre]] (3.9<sp
{{convert/sandbox 1 m micrometre lk=on}}	1	[[metre]] (1,000,000&
{{convert/sandbox 1 mil lk=on}}	1	[[Thou (unit of length
{{convert/sandbox 1 m mil lk=on}}	1	[[metre]] (39,000 [[Th
{{convert/sandbox 1 miles lk=on}}	1	[[mile]] (1.6
{{convert/sandbox 1 m miles lk=on}}	1	[[metre]] (0.00062
{{convert/sandbox 1 Mly lk=on}}	1	[[Light-year#Distances
{{convert/sandbox 1 m Mly lk=on}}	1	[[metre]] (1.1<span st
{{convert/sandbox 1 mm lk=on}}	1	[[millimetre]] (0.0390
{{convert/sandbox 1 m mm lk=on}}	1	[[metre]] (1,000
{{convert/sandbox 1 Mm lk=on}}	1	[[megametre]] (620
{{convert/sandbox 1 m Mm lk=on}}	1	[[metre]] (1.0<span st
{{convert/sandbox 1 m mm in lk=on}}	1	[[metre]] (1,000
{{convert/sandbox 1 Mpc lk=on}}	1	[[Parsec#Megaparsecs a
{{convert/sandbox 1 m Mpc lk=on}}	1	[[metre]] (3.2<span st
{{convert/sandbox 1 nm lk=on}}	1	[[nanometre]] (3.9<spa
{{convert/sandbox 1 m nm lk=on}}	1	[[metre]] (1.0<span st
{{convert/sandbox 1 NM lk=on}}	1	[[nautical mile]] (1.9
{{convert/sandbox 1 m NM lk=on}}	1	[[metre]] (0.00054
{{convert/sandbox 1 nmi lk=on}}	1	[[nautical mile]] (1.9
{{convert/sandbox 1 m nmi lk=on}}	1	[[metre]] (0.00054
{{convert/sandbox 1 m nmi km lk=on}}	1	[[metre]] (0.00054
{{convert/sandbox 1 m nmi mi lk=on}}	1	[[metre]] (0.00054
{{convert/sandbox 1 m nmi mi ft lk=on}}	1	[[metre]] (0.00054
{{convert/sandbox 1 oldUKnmi lk=on}}	1	[[nautical mile]] (1.9
{{convert/sandbox 1 m oldUKnmi lk=on}}	1	[[metre]] (0.00054
{{convert/sandbox 1 oldUSnmi lk=on}}	1	[[nautical mile]] (1.9
{{convert/sandbox 1 m oldUSnmi lk=on}}	1	[[metre]] (0.00054
{{convert/sandbox 1 parsec lk=on}}	1	[[parsec]] (3.3
{{convert/sandbox 1 m parsec lk=on}}	1	[[metre]] (3.2<span st
{{convert/sandbox 1 pc lk=on}}	1	[[parsec]] (3.3
{{convert/sandbox 1 m pc lk=on}}	1	[[metre]] (3.2<span st
{{convert/sandbox 1 perch lk=on}}	1	[[Rod (unit) perch]] (
{{convert/sandbox 1 m perch lk=on}}	1	[[metre]] (0.20 [[Rod
{{convert/sandbox 1 Pm lk=on}}	1	[[petametre]] (6.2<spe
{{convert/sandbox 1 m Pm lk=on}}	1	[[metre]] (1.0<span st
{{convert/sandbox 1 pm lk=on}}	1	[[picometre]] (3.9<spa
{{convert/sandbox 1 m pm lk=on}}	1	[[metre]] (1.0<span st
{{convert/sandbox 1 pole lk=on}}	1	[[Rod (unit) pole]] (1
{{convert/sandbox 1 m pole lk=on}}	1	[[metre]] (0.20 [[Rod
{{convert/sandbox 1 pre1954U.S.nmi lk=on}}	1	[[Nautical mile (pre-1
{{convert/sandbox 1 m pre1954U.S.nmi lk=on}}	1	[[metre]] (0.00054
{{convert/sandbox 1 pre1954USnmi lk=on}}	1	[[Nautical mile (pre-1
{{convert/sandbox 1 m pre1954USnmi lk=on}}	1	[[metre]] (0.00054
{{convert/sandbox 1 rd lk=on}}	1	[[Rod (unit) rod]] (1)
{{convert/sandbox 1 m rd lk=on}}	1	[[metre]] (0.20



```

{{convert/sandbox|1|rod|lk=on}} 1 [[Rod (unit)|rod]] (1.08)
{{convert/sandbox|1|m|rod|lk=on}} 1 [[metre]] (0.20&nbsp;[m])
{{convert/sandbox|1|m|royal cubit|lk=on}} 1 [[metre]] (1.9&nbsp;[m])
{{convert/sandbox|1|rtkm|lk=on}} 1 [[Kilometre|route kilometre]] (1.0)
{{convert/sandbox|1|m|rtkm|lk=on}} 1 [[metre]] (0.0010&nbsp;[m])
{{convert/sandbox|1|rtmi|lk=on}} 1 [[Mile|route mile]] (1.6)
{{convert/sandbox|1|m|rtmi|lk=on}} 1 [[metre]] (0.00062&nbsp;[m])

-- length2
{{convert/sandbox|1|shaku|lk=on}} 1 [[Japanese units of measurement|shaku]] (3.0)
{{convert/sandbox|1|m|shaku|lk=on}} 1 [[metre]] (3.3 [[Japanese units of measurement|shaku]])
{{convert/sandbox|1|sm|lk=on}} 1 [[Smoot (unit)|smoot]] (1.0)
{{convert/sandbox|1|m|sm|lk=on}} 1 [[metre]] (0.59&nbsp;[m])
{{convert/sandbox|1|smi|lk=on}} 1 [[statute mile]] (1.6)
{{convert/sandbox|1|m|smi|lk=on}} 1 [[metre]] (0.00062&nbsp;[m])
{{convert/sandbox|1|smoot|lk=on}} 1 [[Smoot (unit)|smoot]] (1.0)
{{convert/sandbox|1|m|smoot|lk=on}} 1 [[metre]] (0.59&nbsp;[m])
{{convert/sandbox|1|m|statmi km|lk=on}} 1 [[metre]] (0.00062&nbsp;[m])
{{convert/sandbox|1|sun|lk=on}} 1 [[Japanese units of measurement|sun]] (3.0)
{{convert/sandbox|1|m|sun|lk=on}} 1 [[metre]] (33 [[Japanese units of measurement|sun]])
{{convert/sandbox|1|thou|lk=on}} 1 [[Thou (unit of length)|thou]] (1.0)
{{convert/sandbox|1|m|thou|lk=on}} 1 [[metre]] (39,000 [[Thou (unit of length)|thou]])
{{convert/sandbox|1|Tm|lk=on}} 1 [[terametre]] (620,000)
{{convert/sandbox|1|m|Tm|lk=on}} 1 [[metre]] (1.0<span style="font-size: 0.00062em;">[Tm])
{{convert/sandbox|1|uin|lk=on}} 1 [[SI prefix#Non-SI unit|uin]] (1.0)
{{convert/sandbox|1|m|uin|lk=on}} 1 [[metre]] (39,000,000)
{{convert/sandbox|1|um|lk=on}} 1 [[micrometre]] (3.9<span style="font-size: 0.00062em;">[um])
{{convert/sandbox|1|m|um|lk=on}} 1 [[metre]] (1,000,000)
{{convert/sandbox|1|verst|lk=on}} 1 [[verst]] (1.1&nbsp;[m])
{{convert/sandbox|1|m|verst|lk=on}} 1 [[metre]] (0.00094 [[verst]])
{{convert/sandbox|1|yard|lk=on}} 1 [[yard]] (0.91&nbsp;[m])
{{convert/sandbox|1|m|yard|lk=on}} 1 [[metre]] (1.1&nbsp;[m])
{{convert/sandbox|1|yards|lk=on}} 1 [[yard]] (0.91&nbsp;[m])
{{convert/sandbox|1|m|yards|lk=on}} 1 [[metre]] (1.1&nbsp;[m])
{{convert/sandbox|1|yd|lk=on}} 1 [[yard]] (0.91&nbsp;[m])
{{convert/sandbox|1|m|yd|lk=on}} 1 [[metre]] (1.1&nbsp;[m])
{{convert/sandbox|1|m|yd m|lk=on}} 1 [[metre]] (1.1&nbsp;[m])
{{convert/sandbox|1|Ym|lk=on}} 1 [[yottametre]] (6.2<span style="font-size: 0.00062em;">[Ym])
{{convert/sandbox|1|m|Ym|lk=on}} 1 [[metre]] (1.0<span style="font-size: 0.00062em;">[Ym])
{{convert/sandbox|1|Zm|lk=on}} 1 [[zettametre]] (6.2<span style="font-size: 0.00062em;">[Zm])
{{convert/sandbox|1|m|Zm|lk=on}} 1 [[metre]] (1.0<span style="font-size: 0.00062em;">[Zm])
{{convert/sandbox|1|µin|lk=on}} 1 [[SI prefix#Non-SI unit|µin]] (1.0)
{{convert/sandbox|1|m|µin|lk=on}} 1 [[metre]] (39,000,000)
{{convert/sandbox|1|µm|lk=on}} 1 [[micrometre]] (3.9<span style="font-size: 0.00062em;">[µm])
{{convert/sandbox|1|m|µm|lk=on}} 1 [[metre]] (1,000,000)

-- lineardensity
{{convert/sandbox|1|kg/cm|lk=on}} 1 [[Linear density|kilogram per centimetre]] (10)
{{convert/sandbox|1|kg/m|kg/cm|lk=on}} 1 [[Linear density|kilogram per centimetre]] (10)
{{convert/sandbox|1|kg/m|lk=on}} 1 [[Linear density|kilogram per metre]] (1)
{{convert/sandbox|1|lb/ft|lk=on}} 1 [[Linear density|pound per foot]] (1)
{{convert/sandbox|1|kg/m|lb/ft|lk=on}} 1 [[Linear density|kilogram per metre]] (1)
{{convert/sandbox|1|lb/yd|lk=on}} 1 [[Linear density|pound per yard]] (1)
{{convert/sandbox|1|kg/m|lb/yd|lk=on}} 1 [[Linear density|kilogram per metre]] (1)

-- mass
{{convert/sandbox|1|µg|lk=on}} 1 [[microgram]] (1.5<span style="font-size: 0.00062em;">[µg])
{{convert/sandbox|1|g|µg|lk=on}} 1 [[gram]] (1,000,000)
{{convert/sandbox|1|g|billion tonne|lk=on}} 1 [[gram]] (1.0<span style="font-size: 0.00062em;">[g])
{{convert/sandbox|1|carat|lk=on}} 1 [[Carat (mass)|carat]] (0.2)
{{convert/sandbox|1|g|carat|lk=on}} 1 [[gram]] (5.0 [[Carat (mass)|carat]])
{{convert/sandbox|1|drachm|lk=on}} 1 [[Dram (unit)|drachm]] (1)
{{convert/sandbox|1|g|drachm|lk=on}} 1 [[gram]] (0.56 [[Dram (unit)|drachm]])
{{convert/sandbox|1|dram|lk=on}} 1 [[Dram (unit)|drachm]] (1)

```



{{convert/sandbox 1 g dram lk=on}}	1	[[gram]] (0.56 [[Dram
{{convert/sandbox 1 dwt lk=on}}	1	[[pennyweight]] (0.055
{{convert/sandbox 1 g dwt lk=on}}	1	[[gram]] (0.64 [[
{{convert/sandbox 1 DWton lk=on}}	1	[[Tonnage deadweight t
{{convert/sandbox 1 g DWton lk=on}}	1	[[gram]] (9.8<span sty
{{convert/sandbox 1 DWtonne lk=on}}	1	[[Tonnage deadweight t
{{convert/sandbox 1 g DWtonne lk=on}}	1	[[gram]] (1.0<span sty
{{convert/sandbox 1 g lk=on}}	1	[[gram]] (0.035 [[
{{convert/sandbox 1 g g gr lk=on}}	1	[[gram]] (1.0 [[[G
{{convert/sandbox 1 g g oz lk=on}}	1	[[gram]] (1.0 [[[O
{{convert/sandbox 1 gr lk=on}}	1	[[Grain (unit) grain]]
{{convert/sandbox 1 g gr lk=on}}	1	[[gram]] (15 [[[G
{{convert/sandbox 1 g gr mg lk=on}}	1	[[gram]] (15 [[[G
{{convert/sandbox 1 Gt lk=on}}	1	[[Tonne gigatonne]] (9
{{convert/sandbox 1 g Gt lk=on}}	1	[[gram]] (1.0<span sty
{{convert/sandbox 1 kg lk=on}}	1	[[kilogram]] (2.2 [[
{{convert/sandbox 1 g kg lk=on}}	1	[[gram]] (0.0010 [[
{{convert/sandbox 1 g kg lb lk=on}}	1	[[gram]] (0.0010 [[
{{convert/sandbox 1 g kg lb st lk=on}}	1	[[gram]] (0.0010 [[
{{convert/sandbox 1 g kg Scwt lk=on}}	1	[[gram]] (0.0010 [[
{{convert/sandbox 1 g kg st lk=on}}	1	[[gram]] (0.0010 [[
{{convert/sandbox 1 g kg st lb lk=on}}	1	[[gram]] (0.0010 [[
{{convert/sandbox 1 g kg stlb lk=on}}	1	[[gram]] (0.0010 [[
{{convert/sandbox 1 kilotonne lk=on}}	1	[[Tonne kilotonne]] (9
{{convert/sandbox 1 g kilotonne lk=on}}	1	[[gram]] (1.0<span sty
{{convert/sandbox 1 lb lk=on}}	1	[[Pound (mass) pound]]
{{convert/sandbox 1 g lb lk=on}}	1	[[gram]] (0.0022 [[
{{convert/sandbox 1 g lb kg lk=on}}	1	[[gram]] (0.0022 [[
{{convert/sandbox 1 g lb kg st lk=on}}	1	[[gram]] (0.0022 [[
{{convert/sandbox 1 g lb ozt lk=on}}	1	[[gram]] (0.0022 [[
{{convert/sandbox 1 g lb st lk=on}}	1	[[gram]] (0.0022 [[
{{convert/sandbox 1 g lb st kg lk=on}}	1	[[gram]] (0.0022 [[
{{convert/sandbox 1 g lb stlb lk=on}}	1	[[gram]] (0.0022 [[
{{convert/sandbox 1 g lboz lk=on}}	1	[[gram]] (0.035 [[
{{convert/sandbox 1 lbs lk=on}}	1	[[Pound (mass) pound]]
{{convert/sandbox 1 g lbs lk=on}}	1	[[gram]] (0.0022 [[
{{convert/sandbox 1 lbt lk=on}}	1	[[Troy weight troy po
{{convert/sandbox 1 g lbt lk=on}}	1	[[gram]] (0.0027 [[[Tro
{{convert/sandbox 1 Lcwt lk=on}}	1	[[Hundredweight long h
{{convert/sandbox 1 g Lcwt lk=on}}	1	[[gram]] (2.0<span sty
{{convert/sandbox 1 lcwt lk=on}}	1	[[Hundredweight long h
{{convert/sandbox 1 g lcwt lk=on}}	1	[[gram]] (2.0<span sty
{{convert/sandbox 1 g long cwt lk=on}}	1	[[gram]] (2.0<span sty
{{convert/sandbox 1 g long qtr lk=on}}	1	[[gram]] (7.9<span sty
{{convert/sandbox 1 g long ton lk=on}}	1	[[gram]] (9.8<span sty
{{convert/sandbox 1 LT lk=on}}	1	[[long ton]] (1.0 [[
{{convert/sandbox 1 g LT lk=on}}	1	[[gram]] (9.8<span sty
{{convert/sandbox 1 lt lk=on}}	1	[[long ton]] (1.0 [[
{{convert/sandbox 1 g lt lk=on}}	1	[[gram]] (9.8<span sty
{{convert/sandbox 1 g LT ST lk=on}}	1	[[gram]] (9.8<span sty
{{convert/sandbox 1 g LT ST t lk=on}}	1	[[gram]] (9.8<span sty
{{convert/sandbox 1 g LT t lk=on}}	1	[[gram]] (9.8<span sty
{{convert/sandbox 1 g LT t ST lk=on}}	1	[[gram]] (9.8<span sty
{{convert/sandbox 1 g metric ton lk=on}}	1	[[gram]] (1.0<span sty
{{convert/sandbox 1 Mg lk=on}}	1	[[Tonne megagram]] (0
{{convert/sandbox 1 g Mg lk=on}}	1	[[gram]] (1.0<span sty
{{convert/sandbox 1 mg lk=on}}	1	[[milligram]] (0.015&
{{convert/sandbox 1 g mg lk=on}}	1	[[gram]] (1,000 [[
{{convert/sandbox 1 g mg gr lk=on}}	1	[[gram]] (1,000 [[
{{convert/sandbox 1 g million tonne lk=on}}	1	[[gram]] (1.0<span sty
{{convert/sandbox 1 MT lk=on}}	1	[[Tonne metric ton]] (
{{convert/sandbox 1 g MT lk=on}}	1	[[gram]] (1.0<span sty
{{convert/sandbox 1 Mt lk=on}}	1	[[Tonne megatonne]] (9
{{convert/sandbox 1 g Mt lk=on}}	1	[[gram]] (1.0<span sty



{{convert/sandbox 1 MTON lk=on}}	1	[[measurement ton]] (1
{{convert/sandbox 1 ng lk=on}}	1	[[Gram nanogram]] (1.5
{{convert/sandbox 1 g ng lk=on}}	1	[[gram]] (1.0<span sty
{{convert/sandbox 1 oz lk=on}}	1	[[ounce]] (28 [[@
{{convert/sandbox 1 g oz lk=on}}	1	[[gram]] (0.035 [[@
{{convert/sandbox 1 g oz g lk=on}}	1	[[gram]] (0.035 [[@
{{convert/sandbox 1 g oz ozt lk=on}}	1	[[gram]] (0.035 [[@
{{convert/sandbox 1 ozt lk=on}}	1	[[troy ounce]] (1.1&nt
{{convert/sandbox 1 g ozt lk=on}}	1	[[gram]] (0.032 [[@
{{convert/sandbox 1 g ozt g lk=on}}	1	[[gram]] (0.032 [[@
{{convert/sandbox 1 g ozt oz lk=on}}	1	[[gram]] (0.032 [[@
{{convert/sandbox 1 pdr lk=on}}	1	[[Pound (mass) pounde
{{convert/sandbox 1 g pdr lk=on}}	1	[[gram]] (0.0022 [[@
{{convert/sandbox 1 scwt lk=on}}	1	[[short hundredweight]
{{convert/sandbox 1 g scwt lk=on}}	1	[[gram]] (2.2<span sty
{{convert/sandbox 1 Scwt lk=on}}	1	[[short hundredweight]
{{convert/sandbox 1 g Scwt lk=on}}	1	[[gram]] (2.2<span sty
{{convert/sandbox 1 g short cwt lk=on}}	1	[[gram]] (2.2<span sty
{{convert/sandbox 1 g short qtr lk=on}}	1	[[gram]] (8.8<span sty
{{convert/sandbox 1 g short ton lk=on}}	1	[[gram]] (1.1<span sty
{{convert/sandbox 1 shtn lk=on}}	1	[[short ton]] (0.91&nt
{{convert/sandbox 1 g shtn lk=on}}	1	[[gram]] (1.1<span sty
{{convert/sandbox 1 shton lk=on}}	1	[[ton]] (0.91 [[[t
{{convert/sandbox 1 g shton lk=on}}	1	[[gram]] (1.1<span sty
{{convert/sandbox 1 g solar mass lk=on}}	1	[[gram]] (5.0<span sty
{{convert/sandbox 1 st lk=on}}	1	[[Stone (unit) stone]]
{{convert/sandbox 1 g st lk=on}}	1	[[gram]] (0.00016 [[@
{{convert/sandbox 1 ST lk=on}}	1	[[short ton]] (0.91&nt
{{convert/sandbox 1 g ST lk=on}}	1	[[gram]] (1.1<span sty
{{convert/sandbox 1 g st and lb lk=on}}	1	[[gram]] (0.0022 [[@
{{convert/sandbox 1 g st kg lk=on}}	1	[[gram]] (0.00016 [[@
{{convert/sandbox 1 g st kg lb lk=on}}	1	[[gram]] (0.00016 [[@
{{convert/sandbox 1 g st lb lk=on}}	1	[[gram]] (0.00016 [[@
{{convert/sandbox 1 g st lb kg lk=on}}	1	[[gram]] (0.00016 [[@
{{convert/sandbox 1 g ST LT lk=on}}	1	[[gram]] (1.1<span sty
{{convert/sandbox 1 g ST LT t lk=on}}	1	[[gram]] (1.1<span sty
{{convert/sandbox 1 g ST t lk=on}}	1	[[gram]] (1.1<span sty
{{convert/sandbox 1 g ST t LT lk=on}}	1	[[gram]] (1.1<span sty
{{convert/sandbox 1 g stlb lk=on}}	1	[[gram]] (0.0022 [[@
{{convert/sandbox 1 stone lk=on}}	1	[[Stone (unit) stone]]
{{convert/sandbox 1 g stone lk=on}}	1	[[gram]] (0.00016 [[@
{{convert/sandbox 1 t lk=on}}	1	[[tonne]] (0.98 [[[lon
{{convert/sandbox 1 g t lk=on}}	1	[[gram]] (1.0<span sty
{{convert/sandbox 1 g t LT lk=on}}	1	[[gram]] (1.0<span sty
{{convert/sandbox 1 g t LT ST lk=on}}	1	[[gram]] (1.0<span sty
{{convert/sandbox 1 g t Scwt lk=on}}	1	[[gram]] (1.0<span sty
{{convert/sandbox 1 g t ST lk=on}}	1	[[gram]] (1.0<span sty
{{convert/sandbox 1 g t ST LT lk=on}}	1	[[gram]] (1.0<span sty
{{convert/sandbox 1 g thousand tonne lk=on}}	1	[[gram]] (1.0<span sty
{{convert/sandbox 1 g ton lk=on}}	1	[[gram]] (9.8<span sty
{{convert/sandbox 1 tonne lk=on}}	1	[[tonne]] (1.1 [[[ton]]
{{convert/sandbox 1 g tonne lk=on}}	1	[[gram]] (1.0<span sty
{{convert/sandbox 1 tonnes lk=on}}	1	[[tonne]] (0.98 [[[lon
{{convert/sandbox 1 g tonnes lk=on}}	1	[[gram]] (1.0<span sty
{{convert/sandbox 1 g troy pound lk=on}}	1	[[gram]] (0.0027 [[[Tre
{{convert/sandbox 1 ug lk=on}}	1	[[microgram]] (1.5<spa
{{convert/sandbox 1 g ug lk=on}}	1	[[gram]] (1,000,000&nt
{{convert/sandbox 1 viss lk=on}}	1	[[Burmese units of mea
{{convert/sandbox 1 g viss lk=on}}	1	[[gram]] (0.00061 [[@
{{convert/sandbox 1 μg lk=on}}	1	[[microgram]] (1.5<spa
{{convert/sandbox 1 g μg lk=on}}	1	[[gram]] (1,000,000&nt
-- mole		
{{convert/sandbox 1 gmol lk=on}}	1	[[Mole (unit) gram-mol



```

{{convert/sandbox|1|g-mol|lk=on}} 1 [[Mole (unit)|gram-mole]]
{{convert/sandbox|1|kmol|lk=on}} 1 [[Mole (unit)|kilomole]]
{{convert/sandbox|1|lbmol|lk=on}} 1 [[Mole (unit)#Other unit]]
{{convert/sandbox|1|lb-mol|lk=on}} 1 [[Mole (unit)#Other unit]]
{{convert/sandbox|1|mol|lk=on}} 1 [[Mole (unit)|mole]]
{{convert/sandbox|1|gmol/d|lk=on}} 1 [[Mole (unit)|gram-mole]]
{{convert/sandbox|1|g-mol/d|lk=on}} 1 [[Mole (unit)|gram-mole]]
{{convert/sandbox|1|gmol/h|lk=on}} 1 [[Mole (unit)|gram-mole]]
{{convert/sandbox|1|g-mol/h|lk=on}} 1 [[Mole (unit)|gram-mole]]
{{convert/sandbox|1|gmol/min|lk=on}} 1 [[Mole (unit)|gram-mole]]
{{convert/sandbox|1|g-mol/min|lk=on}} 1 [[Mole (unit)|gram-mole]]
{{convert/sandbox|1|gmol/s|lk=on}} 1 [[Mole (unit)|gram-mole]]
{{convert/sandbox|1|g-mol/s|lk=on}} 1 [[Mole (unit)|gram-mole]]
{{convert/sandbox|1|kmol/d|lk=on}} 1 [[Mole (unit)|kilomole]]
{{convert/sandbox|1|kmol/h|lk=on}} 1 [[Mole (unit)|kilomole]]
{{convert/sandbox|1|kmol/min|lk=on}} 1 [[Kilomole (unit)|kilomole]]
{{convert/sandbox|1|kmol/s|lk=on}} 1 [[Mole (unit)|kilomole]]
{{convert/sandbox|1|lbmol/d|lk=on}} 1 [[Mole (unit)#Other unit]]
{{convert/sandbox|1|lb-mol/d|lk=on}} 1 [[Mole (unit)#Other unit]]
{{convert/sandbox|1|lbmol/h|lk=on}} 1 [[Mole (unit)#Other unit]]
{{convert/sandbox|1|lb-mol/h|lk=on}} 1 [[Mole (unit)#Other unit]]
{{convert/sandbox|1|lbmol/min|lk=on}} 1 [[Mole (unit)#Other unit]]
{{convert/sandbox|1|lb-mol/min|lk=on}} 1 [[Mole (unit)#Other unit]]
{{convert/sandbox|1|lbmol/s|lk=on}} 1 [[Mole (unit)#Other unit]]
{{convert/sandbox|1|lb-mol/s|lk=on}} 1 [[Mole (unit)#Other unit]]
{{convert/sandbox|1|mol/d|lk=on}} 1 [[Mole (unit)|mole per]]
{{convert/sandbox|1|mol/h|lk=on}} 1 [[Mole (unit)|mole per]]
{{convert/sandbox|1|mol/min|lk=on}} 1 [[Mole (unit)|mole per]]
{{convert/sandbox|1|mol/s|lk=on}} 1 [[Mole (unit)|mole per]]
{{convert/sandbox|1|umol/s|lk=on}} 1 [[Mole (unit)|micromole]]
{{convert/sandbox|1|umol/s|lk=on}} 1 [[Mole (unit)|micromole]]

-- per
{{convert/sandbox|1|/acre|lk=on}} 1 [[Acre|per acre]] (2.5)
{{convert/sandbox|1|/ha|lk=on}} 1 [[Hectare|per hectare]]
{{convert/sandbox|1|/km2|lk=on}} 1 [[Square kilometre|per]]
{{convert/sandbox|1|/sqkm|lk=on}} 1 [[Square kilometre|per]]
{{convert/sandbox|1|/sqmi|lk=on}} 1 [[Square mile|per square]]
{{convert/sandbox|1|/l|lk=on}} 1 [[Litre|per litre]] (3)
{{convert/sandbox|1|/usgal|lk=on}} 1 per [[United States cu]]
{{convert/sandbox|1|/USgal|lk=on}} 1 per [[United States cu]]
{{convert/sandbox|1|£/acre|lk=on}} £1 per [[acre]] (£2.5/[[
{{convert/sandbox|1|£/ha|lk=on}} £1 per [[hectare]] (£0.4
{{convert/sandbox|1|$/acre|lk=on}} $1 per [[acre]] ($2.5/[[
{{convert/sandbox|1|$/ha|lk=on}} $1 per [[hectare]] ($0.4
{{convert/sandbox|1|$/m2|lk=on}} $1 per [[square metre]]
{{convert/sandbox|1|$/sqft|lk=on}} $1 per [[square foot]]
{{convert/sandbox|1|$/kg|lk=on}} $1 per [[kilogram]] ($0
{{convert/sandbox|1|$/lb|lk=on}} $1 per [[Pound (mass)|p

-- populationdensity
{{convert/sandbox|1|pd/acre|lk=on}} 1 [[Acre|inhabitants per]]
{{convert/sandbox|1|PD/km2|pd/acre|lk=on}} 1 [[Square kilometre|inf]]
{{convert/sandbox|1|PD/acre|lk=on}} 1 [[Acre|inhabitants per]]
{{convert/sandbox|1|PD/km2|PD/acre|lk=on}} 1 [[Square kilometre|inf]]
{{convert/sandbox|1|pd/ha|lk=on}} 1 [[Hectare|inhabitants]]
{{convert/sandbox|1|PD/km2|pd/ha|lk=on}} 1 [[Square kilometre|inf]]
{{convert/sandbox|1|PD/ha|lk=on}} 1 [[Hectare|inhabitants]]
{{convert/sandbox|1|PD/km2|PD/ha|lk=on}} 1 [[Square kilometre|inf]]
{{convert/sandbox|1|PD/km2|lk=on}} 1 [[Square kilometre|inf]]
{{convert/sandbox|1|pd/km2|lk=on}} 1 [[Square kilometre|inf]]
{{convert/sandbox|1|PD/km²|lk=on}} 1 [[Square kilometre|inf]]
{{convert/sandbox|1|pd/sqkm|lk=on}} 1 [[Square kilometre|inf]]
{{convert/sandbox|1|PD/sqkm|lk=on}} 1 [[Square kilometre|inf]]

```



```

{{convert/sandbox|1|PD/sqmi|lk=on}} 1 [[Square mile|inhabite
{{convert/sandbox|1|PD/km2|PD/sqmi|lk=on}} 1 [[Square kilometre|inf
{{convert/sandbox|1|pd/sqmi|lk=on}} 1 [[Square mile|inhabite
{{convert/sandbox|1|PD/km2|pd/sqmi|lk=on}} 1 [[Square kilometre|inf

-- power
{{convert/sandbox|1|µW|lk=on}} 1 [[microwatt]] (1.3<spa
{{convert/sandbox|1|W|µW|lk=on}} 1 [[watt]] (1,000,000&nt
{{convert/sandbox|1|aW|lk=on}} 1 [[attowatt]] (1.3<spa
{{convert/sandbox|1|W|aW|lk=on}} 1 [[watt]] (1.0<span sty
{{convert/sandbox|1|BHP|lk=on}} 1 [[Horsepower#Brake hor
{{convert/sandbox|1|W|BHP|lk=on}} 1 [[watt]] (0.0013&nbsp;
{{convert/sandbox|1|bhp|lk=on}} 1 [[Horsepower#Brake hor
{{convert/sandbox|1|W|bhp|lk=on}} 1 [[watt]] (0.0013&nbsp;
{{convert/sandbox|1|BTU/h|lk=on}} 1 [[British thermal unit
{{convert/sandbox|1|W|BTU/h|lk=on}} 1 [[watt]] (3.4&nbsp;[[E
{{convert/sandbox|1|Btu/h|lk=on}} 1 [[British thermal unit
{{convert/sandbox|1|W|Btu/h|lk=on}} 1 [[watt]] (3.4&nbsp;[[E
{{convert/sandbox|1|btu/h|lk=on}} 1 [[British thermal unit
{{convert/sandbox|1|W|btu/h|lk=on}} 1 [[watt]] (3.4&nbsp;[[E
{{convert/sandbox|1|Cal/d|lk=on}} 1 [[Calorie|large calor]
{{convert/sandbox|1|W|Cal/d|lk=on}} 1 [[watt]] (21&nbsp;[[Ca
{{convert/sandbox|1|cal/h|lk=on}} 1 [[Calorie|calorie per
{{convert/sandbox|1|W|cal/h|lk=on}} 1 [[watt]] (860&nbsp;[[C
{{convert/sandbox|1|Cal/h|lk=on}} 1 [[Calorie|calorie per
{{convert/sandbox|1|W|Cal/h|lk=on}} 1 [[watt]] (0.86&nbsp;[[
{{convert/sandbox|1|cW|lk=on}} 1 [[Watt|centiwatt]] (1
{{convert/sandbox|1|W|cW|lk=on}} 1 [[watt]] (100&nbsp;[[W
{{convert/sandbox|1|daW|lk=on}} 1 [[Watt|decawatt]] (0.0
{{convert/sandbox|1|W|daW|lk=on}} 1 [[watt]] (0.10&nbsp;[[
{{convert/sandbox|1|dW|lk=on}} 1 [[Watt|deciwatt]] (0.0
{{convert/sandbox|1|W|dW|lk=on}} 1 [[watt]] (10&nbsp;[[W
{{convert/sandbox|1|EW|lk=on}} 1 [[exawatt]] (1.3<span
{{convert/sandbox|1|W|EW|lk=on}} 1 [[watt]] (1.0<span sty
{{convert/sandbox|1|fW|lk=on}} 1 [[femtowatt]] (1.3<spa
{{convert/sandbox|1|W|fW|lk=on}} 1 [[watt]] (1.0<span sty
{{convert/sandbox|1|GW|lk=on}} 1 [[gigawatt]] (1,300,00
{{convert/sandbox|1|W|GW|lk=on}} 1 [[watt]] (1.0<span sty
{{convert/sandbox|1|hk|lk=on}} 1 [[metric horsepower]]
{{convert/sandbox|1|W|hk|lk=on}} 1 [[watt]] (0.0014&nbsp;
{{convert/sandbox|1|Hp|lk=on}} 1 [[horsepower]] (0.75&
{{convert/sandbox|1|W|Hp|lk=on}} 1 [[watt]] (0.0013&nbsp;
{{convert/sandbox|1|hp|lk=on}} 1 [[horsepower]] (0.75&
{{convert/sandbox|1|W|hp|lk=on}} 1 [[watt]] (0.0013&nbsp;
{{convert/sandbox|1|HP|lk=on}} 1 [[horsepower]] (0.75&
{{convert/sandbox|1|W|HP|lk=on}} 1 [[watt]] (0.0013&nbsp;
{{convert/sandbox|1|hp-electric|lk=on}} 1 [[electric horsepower]]
{{convert/sandbox|1|W|hp-electric|lk=on}} 1 [[watt]] (0.0013&nbsp;
{{convert/sandbox|1|hp-electrical|lk=on}} 1 [[electrical horsepowe
{{convert/sandbox|1|W|hp-electrical|lk=on}} 1 [[watt]] (0.0013&nbsp;
{{convert/sandbox|1|hp-mechanical|lk=on}} 1 [[horsepower]] (0.75&
{{convert/sandbox|1|W|hp-mechanical|lk=on}} 1 [[watt]] (0.0013&nbsp;
{{convert/sandbox|1|hp-metric|lk=on}} 1 [[metric horsepower]]
{{convert/sandbox|1|W|hp-metric|lk=on}} 1 [[watt]] (0.0014&nbsp;
{{convert/sandbox|1|hW|lk=on}} 1 [[Watt|hectowatt]] (0
{{convert/sandbox|1|W|hW|lk=on}} 1 [[watt]] (0.010&nbsp;[[
{{convert/sandbox|1|IHP|lk=on}} 1 [[Horsepower#Indicate
{{convert/sandbox|1|W|IHP|lk=on}} 1 [[watt]] (0.0013&nbsp;
{{convert/sandbox|1|ihp|lk=on}} 1 [[Horsepower#Indicate
{{convert/sandbox|1|W|ihp|lk=on}} 1 [[watt]] (0.0013&nbsp;
{{convert/sandbox|1|kcal/h|lk=on}} 1 [[Calorie|kilocalorie
{{convert/sandbox|1|W|kcal/h|lk=on}} 1 [[watt]] (0.86&nbsp;[[
{{convert/sandbox|1|kJ/d|lk=on}} 1 [[Kilojoule|kilojoule
{{convert/sandbox|1|W|kJ/d|lk=on}} 1 [[watt]] (86&nbsp;[[K]

```



```

{{convert/sandbox|1|kJ/h|lk=on}} 1 [[Kilojoule|kilojoule]] (3.6&nbsp;[[watt]])
{{convert/sandbox|1|W|kJ/h|lk=on}} 1 [[kilowatt]] (1.3&nbsp;[[watt]])
{{convert/sandbox|1|kW|lk=on}} 1 [[watt]] (0.0010&nbsp;[[watt]])
{{convert/sandbox|1|W|kW|lk=on}} 1 [[watt]] (0.0010&nbsp;[[watt]])
{{convert/sandbox|1|W|kW bhp|lk=on}} 1 [[watt]] (0.0010&nbsp;[[watt]])
{{convert/sandbox|1|W|kW hp|lk=on}} 1 [[watt]] (0.0010&nbsp;[[watt]])
{{convert/sandbox|1|W|kW PS|lk=on}} 1 [[watt]] (0.0010&nbsp;[[watt]])
{{convert/sandbox|1|mW|lk=on}} 1 [[milliwatt]] (1.3<span style="font-size: 0.5em;">[[watt]])
{{convert/sandbox|1|W|mW|lk=on}} 1 [[watt]] (1,000&nbsp;[[watt]])
{{convert/sandbox|1|MW|lk=on}} 1 [[megawatt]] (1,300&nbsp;[[watt]])
{{convert/sandbox|1|W|MW|lk=on}} 1 [[watt]] (1.0<span style="font-size: 0.5em;">[[watt]])
{{convert/sandbox|1|nW|lk=on}} 1 [[nanowatt]] (1.3<span style="font-size: 0.5em;">[[watt]])
{{convert/sandbox|1|W|nW|lk=on}} 1 [[watt]] (1.0<span style="font-size: 0.5em;">[[watt]])
{{convert/sandbox|1|PS|lk=on}} 1 [[metric horsepower]]
{{convert/sandbox|1|W|PS|lk=on}} 1 [[watt]] (0.0014&nbsp;[[watt]])
{{convert/sandbox|1|W|PS bhp|lk=on}} 1 [[watt]] (0.0014&nbsp;[[watt]])
{{convert/sandbox|1|W|PS hp|lk=on}} 1 [[watt]] (0.0014&nbsp;[[watt]])
{{convert/sandbox|1|pW|lk=on}} 1 [[picowatt]] (1.3<span style="font-size: 0.5em;">[[watt]])
{{convert/sandbox|1|W|pW|lk=on}} 1 [[watt]] (1.0<span style="font-size: 0.5em;">[[watt]])
{{convert/sandbox|1|PW|lk=on}} 1 [[petawatt]] (1.3<span style="font-size: 0.5em;">[[watt]])
{{convert/sandbox|1|W|PW|lk=on}} 1 [[watt]] (1.0<span style="font-size: 0.5em;">[[watt]])
{{convert/sandbox|1|shp|lk=on}} 1 [[Horsepower#Shaft horsepower|horsepower]]
{{convert/sandbox|1|W|shp|lk=on}} 1 [[watt]] (0.0013&nbsp;[[watt]])
{{convert/sandbox|1|SHP|lk=on}} 1 [[Horsepower#Shaft horsepower|horsepower]]
{{convert/sandbox|1|W|SHP|lk=on}} 1 [[watt]] (0.0013&nbsp;[[watt]])
{{convert/sandbox|1|TW|lk=on}} 1 [[terawatt]] (1.3<span style="font-size: 0.5em;">[[watt]])
{{convert/sandbox|1|W|TW|lk=on}} 1 [[watt]] (1.0<span style="font-size: 0.5em;">[[watt]])
{{convert/sandbox|1|uW|lk=on}} 1 [[microwatt]] (1.3<span style="font-size: 0.5em;">[[watt]])
{{convert/sandbox|1|W|uW|lk=on}} 1 [[watt]] (1,000,000&nbsp;[[watt]])
{{convert/sandbox|1|W|lk=on}} 1 [[watt]] (0.0013&nbsp;[[horsepower]])
{{convert/sandbox|1|whp|lk=on}} 1 [[watt]] (0.0013&nbsp;[[horsepower]])
{{convert/sandbox|1|YW|lk=on}} 1 [[yottawatt]] (1.3<span style="font-size: 0.5em;">[[watt]])
{{convert/sandbox|1|W|YW|lk=on}} 1 [[watt]] (1.0<span style="font-size: 0.5em;">[[watt]])
{{convert/sandbox|1|yW|lk=on}} 1 [[yoctowatt]] (1.3<span style="font-size: 0.5em;">[[watt]])
{{convert/sandbox|1|W|yW|lk=on}} 1 [[watt]] (1.0<span style="font-size: 0.5em;">[[watt]])
{{convert/sandbox|1|ZW|lk=on}} 1 [[zettawatt]] (1.3<span style="font-size: 0.5em;">[[watt]])
{{convert/sandbox|1|W|ZW|lk=on}} 1 [[watt]] (1.0<span style="font-size: 0.5em;">[[watt]])
{{convert/sandbox|1|zW|lk=on}} 1 [[zeptowatt]] (1.3<span style="font-size: 0.5em;">[[watt]])
{{convert/sandbox|1|W|zW|lk=on}} 1 [[watt]] (1.0<span style="font-size: 0.5em;">[[watt]])
{{convert/sandbox|1|µW|lk=on}} 1 [[microwatt]] (1.3<span style="font-size: 0.5em;">[[watt]])
{{convert/sandbox|1|W|µW|lk=on}} 1 [[watt]] (1,000,000&nbsp;[[watt]])

-- pressure
{{convert/sandbox|1|atm|lk=on}} 1 [[Atmosphere (unit)|atmosphere]]
{{convert/sandbox|1|Pa|atm|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|Ba|lk=on}} 1 [[barye]] (0.10&nbsp;[[Pascal (unit)|pascal]])
{{convert/sandbox|1|Pa|Ba|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|bar|lk=on}} 1 [[Bar (unit)|bar]] (10&nbsp;[[Pascal (unit)|pascal]])
{{convert/sandbox|1|Pa|bar|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|Pa|bar kPa|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|dbar|lk=on}} 1 [[Bar (unit)|decibar]]
{{convert/sandbox|1|Pa|dbar|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|GPa|lk=on}} 1 [[Pascal (unit)|gigapascal]]
{{convert/sandbox|1|Pa|GPa|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|hPa|lk=on}} 1 [[Pascal (unit)|hectopascal]]
{{convert/sandbox|1|Pa|hPa|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|Pa|hPa inHg|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|inHg|lk=on}} 1 [[inch of mercury]] (3.386&nbsp;[[Pascal (unit)|pascal]])
{{convert/sandbox|1|Pa|inHg|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|Pa|inHg psi|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|kBa|lk=on}} 1 [[Barye|kilobarye]] (10&nbsp;[[Pascal (unit)|pascal]])
{{convert/sandbox|1|Pa|kBa|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|kgf/cm2|lk=on}} 1 [[Kilogram-force|kilogram-force]] (9.80665&nbsp;[[Pascal (unit)|pascal]])

```



```

{{convert/sandbox|1|Pa|kgf/cm2|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|kg-f/cm2|lk=on}} 1 [[Kilogram-force|kilogram-force]]
{{convert/sandbox|1|Pa|kg-f/cm2|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|kgf/psqcm|lk=on}} 1 [[Kilogram-force|kilogram-force]]
{{convert/sandbox|1|Pa|kgf/psqcm|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|kN/m2|lk=on}} 1 [[Pascal (unit)|kilopascal]]
{{convert/sandbox|1|Pa|kN/m2|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|kPa|lk=on}} 1 [[Pascal (unit)|kilopascal]]
{{convert/sandbox|1|Pa|kPa|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|Pa|kPa inHg|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|Pa|kPa kg/cm2|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|Pa|kPa kgf/cm2|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|Pa|kPa kg-f/cm2|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|Pa|kPa lb/in2|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|Pa|kPa mmHg|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|Pa|kPa psi|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|Pa|kPa Torr|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|ksi|lk=on}} 1 [[Pounds per square inch|kilopascal]]
{{convert/sandbox|1|Pa|ksi|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|lbf/in2|lk=on}} 1 [[Pounds-force per square inch|pascal]]
{{convert/sandbox|1|Pa|lbf/in2|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|mb|lk=on}} 1 [[Bar (unit)|millibar]]
{{convert/sandbox|1|Pa|mb|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|mbar|lk=on}} 1 [[Bar (unit)|millibar]]
{{convert/sandbox|1|Pa|mbar|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|mmHg|lk=on}} 1 [[Millimeter of mercury|pascal]]
{{convert/sandbox|1|Pa|mmHg|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|Pa|mmHg psi|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|MPa|lk=on}} 1 [[Pascal (unit)|megapascal]]
{{convert/sandbox|1|Pa|MPa|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|mPa|lk=on}} 1 [[Pascal (unit)|millipascal]]
{{convert/sandbox|1|Pa|mPa|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|Pa|MPa kgf/cm2|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|Pa|MPa kg-f/cm2|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|Pa|MPa ksi|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|Pa|MPa psi|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|N/cm2|lk=on}} 1 [[Newton (unit)|newton]]
{{convert/sandbox|1|Pa|N/cm2|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|N/m2|lk=on}} 1 [[Newton (unit)|newton]]
{{convert/sandbox|1|Pa|N/m2|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|Pa|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|psf|lk=on}} 1 [[Pounds per square foot|pascal]]
{{convert/sandbox|1|Pa|psf|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|psi|lk=on}} 1 [[Pounds per square inch|pascal]]
{{convert/sandbox|1|Pa|psi|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|Torr|lk=on}} 1 [[torr]] (0.13&nbsp;[[Pascal (unit)|pascal]])
{{convert/sandbox|1|Pa|Torr|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|torr|lk=on}} 1 [[torr]] (0.13&nbsp;[[Pascal (unit)|pascal]])
{{convert/sandbox|1|Pa|torr|lk=on}} 1 [[Pascal (unit)|pascal]]
{{convert/sandbox|1|Pa|Torr psi|lk=on}} 1 [[Pascal (unit)|pascal]]

-- radioactivity
{{convert/sandbox|1|µCi|lk=on}} 1 [[Curie|microcurie]]
{{convert/sandbox|1|Ci|µCi|lk=on}} 1 [[curie]] (1,000,000&nbsp;[[Becquerel]])
{{convert/sandbox|1|Bq|lk=on}} 1 [[becquerel]] (27&nbsp;[[Curie]])
{{convert/sandbox|1|Ci|Bq|lk=on}} 1 [[curie]] (3.7<span style="font-size:smaller">[[Becquerel]])
{{convert/sandbox|1|Ci|lk=on}} 1 [[curie]] (37&nbsp;[[Becquerel]])
{{convert/sandbox|1|EBq|lk=on}} 1 [[Becquerel|exabecquerel]]
{{convert/sandbox|1|Ci|EBq|lk=on}} 1 [[curie]] (3.7<span style="font-size:smaller">[[Becquerel]])
{{convert/sandbox|1|fCi|lk=on}} 1 [[Curie|femtocurie]]
{{convert/sandbox|1|Ci|fCi|lk=on}} 1 [[curie]] (1.0<span style="font-size:smaller">[[Becquerel]])
{{convert/sandbox|1|GBq|lk=on}} 1 [[Becquerel|gigabecquerel]]
{{convert/sandbox|1|Ci|GBq|lk=on}} 1 [[curie]] (37&nbsp;[[Becquerel]])
{{convert/sandbox|1|kBq|lk=on}} 1 [[Becquerel|kilobecquerel]]

```



```

{{convert/sandbox|1|Ci|kBq|lk=on}} 1 [[curie]] (37,000,000&
{{convert/sandbox|1|kCi|lk=on}} 1 [[Curie|kilocurie]] (3
{{convert/sandbox|1|Ci|kCi|lk=on}} 1 [[curie]] (0.0010&nbs
{{convert/sandbox|1|MBq|lk=on}} 1 [[Becquerel|megabecque
{{convert/sandbox|1|Ci|MBq|lk=on}} 1 [[curie]] (37,000&nbs
{{convert/sandbox|1|mBq|lk=on}} 1 [[Becquerel|millibecqu
{{convert/sandbox|1|Ci|mBq|lk=on}} 1 [[curie]] (3.7<span st
{{convert/sandbox|1|MCi|lk=on}} 1 [[Curie|megacurie]] (3
{{convert/sandbox|1|Ci|MCi|lk=on}} 1 [[curie]] (1.0<span st
{{convert/sandbox|1|mCi|lk=on}} 1 [[Curie|millicurie]] (
{{convert/sandbox|1|Ci|mCi|lk=on}} 1 [[curie]] (1,000&nbs
{{convert/sandbox|1|nCi|lk=on}} 1 [[Curie|nanocurie]] (3
{{convert/sandbox|1|Ci|nCi|lk=on}} 1 [[curie]] (1.0<span st
{{convert/sandbox|1|PBq|lk=on}} 1 [[Becquerel|petabecque
{{convert/sandbox|1|Ci|PBq|lk=on}} 1 [[curie]] (3.7<span st
{{convert/sandbox|1|pCi|lk=on}} 1 [[Curie|picocurie]] (0
{{convert/sandbox|1|Ci|pCi|lk=on}} 1 [[curie]] (1.0<span st
{{convert/sandbox|1|TBq|lk=on}} 1 [[Becquerel|terabecque
{{convert/sandbox|1|Ci|TBq|lk=on}} 1 [[curie]] (0.037&nbs
{{convert/sandbox|1|uCi|lk=on}} 1 [[Curie|microcurie]] (
{{convert/sandbox|1|Ci|uCi|lk=on}} 1 [[curie]] (1,000,000&

-- speed
{{convert/sandbox|1|cm/s|lk=on}} 1 [[Metre per second|cer
{{convert/sandbox|1|m/s|cm/s|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|cm/y|lk=on}} 1 [[Orders of magnitude
{{convert/sandbox|1|m/s|cm/y|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|cm/year|lk=on}} 1 [[Orders of magnitude
{{convert/sandbox|1|m/s|cm/year|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|cm/yr|lk=on}} 1 [[Orders of magnitude
{{convert/sandbox|1|m/s|cm/yr|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|foot/s|lk=on}} 1 [[Feet per second|foot
{{convert/sandbox|1|m/s|foot/s|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|m/s|foot/s m/s|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|ft/min|lk=on}} 1 [[Feet per second|foot
{{convert/sandbox|1|m/s|ft/min|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|ft/s|lk=on}} 1 [[Feet per second|foot
{{convert/sandbox|1|m/s|ft/s|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|m/s|ft/s m/s|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|m/s|furlong per fortnight|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|in/s|lk=on}} 1 [[Inch|inch per second]] (
{{convert/sandbox|1|m/s|in/s|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|in/y|lk=on}} 1 [[Orders of magnitude
{{convert/sandbox|1|m/s|in/y|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|in/year|lk=on}} 1 [[Orders of magnitude
{{convert/sandbox|1|m/s|in/year|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|in/yr|lk=on}} 1 [[Orders of magnitude
{{convert/sandbox|1|m/s|in/yr|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|km/h|lk=on}} 1 [[Kilometres per hour|
{{convert/sandbox|1|m/s|km/h|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|m/s|km/h kn|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|m/s|km/h mph|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|km/s|lk=on}} 1 [[Metre per second|ki
{{convert/sandbox|1|m/s|km/s|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|kn|lk=on}} 1 [[Knot (unit)|knot]] (
{{convert/sandbox|1|m/s|kn|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|m/s|kn km/h|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|m/s|kn m/s|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|m/s|kn mph|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|knot|lk=on}} 1 [[Knot (unit)|knot]] (
{{convert/sandbox|1|m/s|knot|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|knots|lk=on}} 1 [[Knot (unit)|knot]] (
{{convert/sandbox|1|m/s|knots|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|kph|lk=on}} 1 [[Kilometres per hour|

```



```

{{convert/sandbox|1|m/s|kph|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|m/min|lk=on}} 1 [[Metre per second|metre per second]] (
{{convert/sandbox|1|m/s|m/min|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|m/s|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|m/s|m/s foot/s|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|m/s|m/s ft/s|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|m/s|m/s kn km/h|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|m/s|m/s mph|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|Mach|lk=on}} 1 [[Mach number|Mach]]&nbsp; (
{{convert/sandbox|1|m/s|Mach|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|mi/h|lk=on}} 1 [[Miles per hour|mile per hour]] (
{{convert/sandbox|1|m/s|mi/h|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|mi/s|lk=on}} 1 [[Mile|mile per second]] (
{{convert/sandbox|1|m/s|mi/s|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|mph|lk=on}} 1 [[Miles per hour|mile per hour]] (
{{convert/sandbox|1|m/s|mph|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|m/s|mph km/h|lk=on}} 1 [[metre per second]] (
{{convert/sandbox|1|m/s|mph kn|lk=on}} 1 [[metre per second]] (

-- temperature
{{convert/sandbox|1|°C|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|°C °F|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|°C °F °R|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|°C °F K|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|°C °R|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|°C °R °F|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|°C °R K|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|°C K|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|°C K °F|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|°C K °R|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|°F|lk=on}} 1&nbsp; [[Fahrenheit|°F]] (3
{{convert/sandbox|1|C|°F|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|°F °C|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|°F °C °R|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|°F °C K|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|°F °R|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|°F °R °C|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|°F °R K|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|°F K|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|°F K °C|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|°F K °R|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|°F R|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|°R|lk=on}} 1&nbsp; [[Rankine scale|°R]] (3
{{convert/sandbox|1|C|°R|lk=on}} 1&nbsp; [[Celsius|°C]] (4
{{convert/sandbox|1|C|°R °C|lk=on}} 1&nbsp; [[Celsius|°C]] (4
{{convert/sandbox|1|C|°R °C °F|lk=on}} 1&nbsp; [[Celsius|°C]] (4
{{convert/sandbox|1|C|°R °C K|lk=on}} 1&nbsp; [[Celsius|°C]] (4
{{convert/sandbox|1|C|°R °F|lk=on}} 1&nbsp; [[Celsius|°C]] (4
{{convert/sandbox|1|C|°R °F °C|lk=on}} 1&nbsp; [[Celsius|°C]] (4
{{convert/sandbox|1|C|°R °F K|lk=on}} 1&nbsp; [[Celsius|°C]] (4
{{convert/sandbox|1|C|°R K|lk=on}} 1&nbsp; [[Celsius|°C]] (4
{{convert/sandbox|1|C|°R K °C|lk=on}} 1&nbsp; [[Celsius|°C]] (4
{{convert/sandbox|1|C|°R K °F|lk=on}} 1&nbsp; [[Celsius|°C]] (4
{{convert/sandbox|1|C|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|C F|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|C F K|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|C F R|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|C K|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|C K F|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|C K R|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|C R|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|C R F|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|C|C R K|lk=on}} 1&nbsp; [[Celsius|°C]] (3
{{convert/sandbox|1|Celsius|lk=on}} 1&nbsp; [[Celsius|°C]] (3

```



```

{{convert/sandbox|1|C|Celsius|lk=on}} 1&nbsp;  [[Celsius|°C]] (1
{{convert/sandbox|1|F|lk=on}} 1&nbsp;  [[Fahrenheit|°F]]
{{convert/sandbox|1|C|F|lk=on}} 1&nbsp;  [[Celsius|°C]] (3
{{convert/sandbox|1|C|F|C|lk=on}} 1&nbsp;  [[Celsius|°C]] (3
{{convert/sandbox|1|C|F|C|K|lk=on}} 1&nbsp;  [[Celsius|°C]] (3
{{convert/sandbox|1|C|F|C|R|lk=on}} 1&nbsp;  [[Celsius|°C]] (3
{{convert/sandbox|1|C|F|K|lk=on}} 1&nbsp;  [[Celsius|°C]] (3
{{convert/sandbox|1|C|F|K|C|lk=on}} 1&nbsp;  [[Celsius|°C]] (3
{{convert/sandbox|1|C|F|K|R|lk=on}} 1&nbsp;  [[Celsius|°C]] (3
{{convert/sandbox|1|C|F|R|lk=on}} 1&nbsp;  [[Celsius|°C]] (3
{{convert/sandbox|1|C|F|R|C|lk=on}} 1&nbsp;  [[Celsius|°C]] (3
{{convert/sandbox|1|C|F|R|K|lk=on}} 1&nbsp;  [[Celsius|°C]] (3
{{convert/sandbox|1|K|lk=on}} 1&nbsp;  [[kelvin|K]] (-27
{{convert/sandbox|1|C|K|lk=on}} 1&nbsp;  [[Celsius|°C]] (2
{{convert/sandbox|1|C|K|°C|lk=on}} 1&nbsp;  [[Celsius|°C]] (2
{{convert/sandbox|1|C|K|°C|°F|lk=on}} 1&nbsp;  [[Celsius|°C]] (2
{{convert/sandbox|1|C|K|°C|°R|lk=on}} 1&nbsp;  [[Celsius|°C]] (2
{{convert/sandbox|1|C|K|°F|lk=on}} 1&nbsp;  [[Celsius|°C]] (2
{{convert/sandbox|1|C|K|°F|°C|lk=on}} 1&nbsp;  [[Celsius|°C]] (2
{{convert/sandbox|1|C|K|°F|°R|lk=on}} 1&nbsp;  [[Celsius|°C]] (2
{{convert/sandbox|1|C|K|°R|lk=on}} 1&nbsp;  [[Celsius|°C]] (2
{{convert/sandbox|1|C|K|°R|°C|lk=on}} 1&nbsp;  [[Celsius|°C]] (2
{{convert/sandbox|1|C|K|°R|°F|lk=on}} 1&nbsp;  [[Celsius|°C]] (2
{{convert/sandbox|1|C|K|C|lk=on}} 1&nbsp;  [[Celsius|°C]] (2
{{convert/sandbox|1|C|K|C|F|lk=on}} 1&nbsp;  [[Celsius|°C]] (2
{{convert/sandbox|1|C|K|C|R|lk=on}} 1&nbsp;  [[Celsius|°C]] (2
{{convert/sandbox|1|C|K|F|lk=on}} 1&nbsp;  [[Celsius|°C]] (2
{{convert/sandbox|1|C|K|F|C|lk=on}} 1&nbsp;  [[Celsius|°C]] (2
{{convert/sandbox|1|C|K|F|R|lk=on}} 1&nbsp;  [[Celsius|°C]] (2
{{convert/sandbox|1|C|K|R|lk=on}} 1&nbsp;  [[Celsius|°C]] (2
{{convert/sandbox|1|C|K|R|C|lk=on}} 1&nbsp;  [[Celsius|°C]] (2
{{convert/sandbox|1|C|K|R|F|lk=on}} 1&nbsp;  [[Celsius|°C]] (2
{{convert/sandbox|1|MK|lk=on}} 1 [[Kelvin|megakelvin]]
{{convert/sandbox|1|C|MK|lk=on}} 1&nbsp;  [[Celsius|°C]] (6
{{convert/sandbox|1|R|lk=on}} 1&nbsp;  [[Rankine scale|°R]]
{{convert/sandbox|1|C|R|lk=on}} 1&nbsp;  [[Celsius|°C]] (4
{{convert/sandbox|1|C|R|C|lk=on}} 1&nbsp;  [[Celsius|°C]] (4
{{convert/sandbox|1|C|R|C|F|lk=on}} 1&nbsp;  [[Celsius|°C]] (4
{{convert/sandbox|1|C|R|C|K|lk=on}} 1&nbsp;  [[Celsius|°C]] (4
{{convert/sandbox|1|C|R|F|lk=on}} 1&nbsp;  [[Celsius|°C]] (4
{{convert/sandbox|1|C|R|F|C|lk=on}} 1&nbsp;  [[Celsius|°C]] (4
{{convert/sandbox|1|C|R|F|K|lk=on}} 1&nbsp;  [[Celsius|°C]] (4
{{convert/sandbox|1|C|R|K|lk=on}} 1&nbsp;  [[Celsius|°C]] (4
{{convert/sandbox|1|C|R|K|C|lk=on}} 1&nbsp;  [[Celsius|°C]] (4
{{convert/sandbox|1|C|R|K|F|lk=on}} 1&nbsp;  [[Celsius|°C]] (4

-- temperature-change
{{convert/sandbox|1|C-change|lk=on}} 1&nbsp;  [[Celsius|°C]] (1
{{convert/sandbox|1|F-change|lk=on}} 1&nbsp;  [[Fahrenheit|°F]]
{{convert/sandbox|1|C-change|F-change|lk=on}} 1&nbsp;  [[Celsius|°C]] (1
{{convert/sandbox|1|K-change|lk=on}} 1&nbsp;  [[Kelvin|K]] (1.8
{{convert/sandbox|1|C-change|K-change|lk=on}} 1&nbsp;  [[Celsius|°C]] (1

-- time
{{convert/sandbox|1|µs|lk=on}} 1 [[microsecond]] (0.001
{{convert/sandbox|1|s|µs|lk=on}} 1 [[second]] (1,000,000&
{{convert/sandbox|1|byr|lk=on}} 1 [[Annum|billion years]]
{{convert/sandbox|1|s|byr|lk=on}} 1 [[second]] (3.2<span s
{{convert/sandbox|1|century|lk=on}} 1 [[century]] (3.2&nbsp;&
{{convert/sandbox|1|s|century|lk=on}} 1 [[second]] (3.2<span s
{{convert/sandbox|1|d|lk=on}} 1 [[day]] (86&nbsp;&[[Ki
{{convert/sandbox|1|s|d|lk=on}} 1 [[second]] (1.2<span s
{{convert/sandbox|1|day|lk=on}} 1 [[day]] (86&nbsp;&[[Ki
{{convert/sandbox|1|s|day|lk=on}} 1 [[second]] (1.2<span s

```



{{convert/sandbox 1 days lk=on}}	1	[[day]] (86 [[Ki
{{convert/sandbox 1 s days lk=on}}	1	[[second]] (1.2<span \$
{{convert/sandbox 1 decade lk=on}}	1	[[decade]] (320 [[K
{{convert/sandbox 1 s decade lk=on}}	1	[[second]] (3.2<span \$
{{convert/sandbox 1 dog year lk=on}}	1	[[dog year]] (7.0 [[K
{{convert/sandbox 1 s dog year lk=on}}	1	[[second]] (4.5<span \$
{{convert/sandbox 1 dog yr lk=on}}	1	[[dog year]] (7.0 [[K
{{convert/sandbox 1 s dog yr lk=on}}	1	[[second]] (4.5<span \$
{{convert/sandbox 1 Es lk=on}}	1	[[exasecond]] (32 [[K
{{convert/sandbox 1 s Es lk=on}}	1	[[second]] (1.0<span \$
{{convert/sandbox 1 fortnight lk=on}}	1	[[fortnight]] (2.0 [[v
{{convert/sandbox 1 s fortnight lk=on}}	1	[[second]] (8.3<span \$
{{convert/sandbox 1 Gs lk=on}}	1	[[gigasecond]] (3.2&nb
{{convert/sandbox 1 s Gs lk=on}}	1	[[second]] (1.0<span \$
{{convert/sandbox 1 Gyr lk=on}}	1	[[Annum thousand milli
{{convert/sandbox 1 s Gyr lk=on}}	1	[[second]] (3.2<span \$
{{convert/sandbox 1 h lk=on}}	1	[[hour]] (3.6 [[K
{{convert/sandbox 1 s h lk=on}}	1	[[second]] (0.00028&nb
{{convert/sandbox 1 hour lk=on}}	1	[[hour]] (3.6 [[K
{{convert/sandbox 1 s hour lk=on}}	1	[[second]] (0.00028&nb
{{convert/sandbox 1 hours lk=on}}	1	[[hour]] (3.6 [[K
{{convert/sandbox 1 s hours lk=on}}	1	[[second]] (0.00028&nb
{{convert/sandbox 1 kmyr lk=on}}	1	[[Annum thousand milli
{{convert/sandbox 1 s kmyr lk=on}}	1	[[second]] (3.2<span \$
{{convert/sandbox 1 kMyr lk=on}}	1	[[Annum thousand milli
{{convert/sandbox 1 s kMyr lk=on}}	1	[[second]] (3.2<span \$
{{convert/sandbox 1 ks lk=on}}	1	[[kilosecond]] (0.28&
{{convert/sandbox 1 s ks lk=on}}	1	[[second]] (0.0010&nb
{{convert/sandbox 1 kyr lk=on}}	1	[[millennium]] (32&nb
{{convert/sandbox 1 s kyr lk=on}}	1	[[second]] (3.2<span \$
{{convert/sandbox 1 long billion year lk=on}}	1	[[Annum billion years]]
{{convert/sandbox 1 s long billion year lk=on}}	1	[[second]] (3.2<span \$
{{convert/sandbox 1 long byr lk=on}}	1	[[Annum billion years]]
{{convert/sandbox 1 s long byr lk=on}}	1	[[second]] (3.2<span \$
{{convert/sandbox 1 millennium lk=on}}	1	[[millennium]] (32&nb
{{convert/sandbox 1 s millennium lk=on}}	1	[[second]] (3.2<span \$
{{convert/sandbox 1 milliard year lk=on}}	1	[[Annum milliard years]]
{{convert/sandbox 1 s milliard year lk=on}}	1	[[second]] (3.2<span \$
{{convert/sandbox 1 million year lk=on}}	1	[[Annum million years]]
{{convert/sandbox 1 s million year lk=on}}	1	[[second]] (3.2<span \$
{{convert/sandbox 1 min lk=on}}	1	[[minute]] (60 [[K
{{convert/sandbox 1 s min lk=on}}	1	[[second]] (0.017 [[K
{{convert/sandbox 1 minute lk=on}}	1	[[minute]] (60 [[K
{{convert/sandbox 1 s minute lk=on}}	1	[[second]] (0.017 [[K
{{convert/sandbox 1 minutes lk=on}}	1	[[minute]] (60 [[K
{{convert/sandbox 1 s minutes lk=on}}	1	[[second]] (0.017 [[K
{{convert/sandbox 1 month lk=on}}	1	[[month]] (2.6 [[K
{{convert/sandbox 1 s month lk=on}}	1	[[second]] (3.8<span \$
{{convert/sandbox 1 months lk=on}}	1	[[month]] (0.083 [[K
{{convert/sandbox 1 s months lk=on}}	1	[[second]] (3.8<span \$
{{convert/sandbox 1 ms lk=on}}	1	[[millisecond]] (0.001
{{convert/sandbox 1 s ms lk=on}}	1	[[second]] (1,000 [[K
{{convert/sandbox 1 Ms lk=on}}	1	[[megasecond]] (1.7 [[
{{convert/sandbox 1 s Ms lk=on}}	1	[[second]] (1.0<span \$
{{convert/sandbox 1 mth lk=on}}	1	[[month]] (2.6 [[K
{{convert/sandbox 1 s mth lk=on}}	1	[[second]] (3.8<span \$
{{convert/sandbox 1 Myr lk=on}}	1	[[Annum million years]]
{{convert/sandbox 1 s Myr lk=on}}	1	[[second]] (3.2<span \$
{{convert/sandbox 1 myr lk=on}}	1	[[Annum million years]]
{{convert/sandbox 1 s myr lk=on}}	1	[[second]] (3.2<span \$
{{convert/sandbox 1 ns lk=on}}	1	[[nanosecond]] (0.0010
{{convert/sandbox 1 s ns lk=on}}	1	[[second]] (1.0<span \$
{{convert/sandbox 1 Ps lk=on}}	1	[[petasecond]] (32&nb
{{convert/sandbox 1 s Ps lk=on}}	1	[[second]] (1.0<span \$



```

{{convert/sandbox|1|s|lk=on}} 1 [[second]] (0.017&nbsp;
{{convert/sandbox|1|second|lk=on}} 1 [[second]] (0.017&nbsp;
{{convert/sandbox|1|seconds|lk=on}} 1 [[second]] (0.017&nbsp;
{{convert/sandbox|1|short billion year|lk=on}} 1 [[Annum|billion years]]
{{convert/sandbox|1|s|short billion year|lk=on}} 1 [[second]] (3.2<span $
{{convert/sandbox|1|short trillion year|lk=on}} 1 [[Annum|trillion years]]
{{convert/sandbox|1|s|short trillion year|lk=on}} 1 [[second]] (3.2<span $
{{convert/sandbox|1|thousand million year|lk=on}} 1 [[Annum|thousand milli
{{convert/sandbox|1|s|thousand million year|lk=on}} 1 [[second]] (3.2<span $
{{convert/sandbox|1|tmyr|lk=on}} 1 [[Annum|thousand milli
{{convert/sandbox|1|s|tmyr|lk=on}} 1 [[second]] (3.2<span $
{{convert/sandbox|1|tryr|lk=on}} 1 [[Annum|trillion years]]
{{convert/sandbox|1|s|tryr|lk=on}} 1 [[second]] (3.2<span $
{{convert/sandbox|1|Ts|lk=on}} 1 [[terasecond]] (32&nbsp;
{{convert/sandbox|1|s|Ts|lk=on}} 1 [[second]] (1.0<span $
{{convert/sandbox|1|tyr|lk=on}} 1 [[millennium]] (32&nbsp;
{{convert/sandbox|1|s|tyr|lk=on}} 1 [[second]] (3.2<span $
{{convert/sandbox|1|us|lk=on}} 1 [[microsecond]] (0.001
{{convert/sandbox|1|s|us|lk=on}} 1 [[second]] (1,000,000&
{{convert/sandbox|1|week|lk=on}} 1 [[week]] (0.60&nbsp;
{{convert/sandbox|1|s|week|lk=on}} 1 [[second]] (1.7<span $
{{convert/sandbox|1|weeks|lk=on}} 1 [[week]] (0.60&nbsp;
{{convert/sandbox|1|s|weeks|lk=on}} 1 [[second]] (1.7<span $
{{convert/sandbox|1|wk|lk=on}} 1 [[week]] (0.60&nbsp;
{{convert/sandbox|1|s|wk|lk=on}} 1 [[second]] (1.7<span $
{{convert/sandbox|1|year|lk=on}} 1 [[Annum|year]] (32&nbsp;
{{convert/sandbox|1|s|year|lk=on}} 1 [[second]] (3.2<span $
{{convert/sandbox|1|years|lk=on}} 1 [[Annum|year]] (32&nbsp;
{{convert/sandbox|1|s|years|lk=on}} 1 [[second]] (3.2<span $
{{convert/sandbox|1|yr|lk=on}} 1 [[Annum|year]] (32&nbsp;
{{convert/sandbox|1|s|yr|lk=on}} 1 [[second]] (3.2<span $
{{convert/sandbox|1|µs|lk=on}} 1 [[microsecond]] (0.001
{{convert/sandbox|1|s|µs|lk=on}} 1 [[second]] (1,000,000&

-- torque
{{convert/sandbox|1|in.lbf|lk=on}} 1 [[Foot-pound (energy)|
{{convert/sandbox|1|Nm|in.lbf|lk=on}} 1 [[newton metre]] (8.96
{{convert/sandbox|1|in.lb-f|lk=on}} 1 [[Foot-pound (energy)|
{{convert/sandbox|1|Nm|in.lb-f|lk=on}} 1 [[newton metre]] (8.96
{{convert/sandbox|1|in.ozf|lk=on}} 1 [[Foot-pound (energy)|
{{convert/sandbox|1|Nm|in.ozf|lk=on}} 1 [[newton metre]] (140&
{{convert/sandbox|1|in.oz-f|lk=on}} 1 [[Foot-pound (energy)|
{{convert/sandbox|1|Nm|in.oz-f|lk=on}} 1 [[newton metre]] (140&
{{convert/sandbox|1|in.lbf|lk=on}} 1 [[Foot-pound (energy)|
{{convert/sandbox|1|Nm|in.lbf|lk=on}} 1 [[newton metre]] (8.96
{{convert/sandbox|1|in.lb-f|lk=on}} 1 [[Foot-pound (energy)|
{{convert/sandbox|1|Nm|in.lb-f|lk=on}} 1 [[newton metre]] (8.96
{{convert/sandbox|1|in.ozf|lk=on}} 1 [[Foot-pound (energy)|
{{convert/sandbox|1|Nm|in.ozf|lk=on}} 1 [[newton metre]] (140&
{{convert/sandbox|1|in.oz-f|lk=on}} 1 [[Foot-pound (energy)|
{{convert/sandbox|1|Nm|in.oz-f|lk=on}} 1 [[newton metre]] (140&
{{convert/sandbox|1|kg.m|lk=on}} 1 [[kilogram metre]] (9
{{convert/sandbox|1|Nm|kg.m|lk=on}} 1 [[newton metre]] (0.1&
{{convert/sandbox|1|kgf.m|lk=on}} 1 [[Kilogram metre|kilo
{{convert/sandbox|1|Nm|kgf.m|lk=on}} 1 [[newton metre]] (0.1&
{{convert/sandbox|1|kgm|lk=on}} 1 [[kilogram metre]] (9
{{convert/sandbox|1|Nm|kgm|lk=on}} 1 [[newton metre]] (0.1&
{{convert/sandbox|1|Nm|kgm lbf|lk=on}} 1 [[newton metre]] (0.1&
{{convert/sandbox|1|kN/m|lk=on}} 1 [[Newton (unit)|kilon
{{convert/sandbox|1|Nm|kN/m|lk=on}} 1 [[newton metre]] (0.0&
{{convert/sandbox|1|lb.ft|lk=on}} 1 [[Pound-foot (torque)|
{{convert/sandbox|1|Nm|lb.ft|lk=on}} 1 [[newton metre]] (0.74
{{convert/sandbox|1|lb.in|lk=on}} 1 [[Pound-foot (torque)|
{{convert/sandbox|1|Nm|lb.in|lk=on}} 1 [[newton metre]] (8.96

```



```

{{convert/sandbox|1|lb·ft|lk=on}} 1 [[Pound-foot (torque)|
{{convert/sandbox|1|Nm|lb·ft|lk=on}} 1 [[newton metre]] (0.74
{{convert/sandbox|1|lbf.ft|lk=on}} 1 [[Pound-foot (torque)|
{{convert/sandbox|1|Nm|lbf.ft|lk=on}} 1 [[newton metre]] (0.74
{{convert/sandbox|1|lb-f.ft|lk=on}} 1 [[Pound-foot (torque)|
{{convert/sandbox|1|Nm|lb-f.ft|lk=on}} 1 [[newton metre]] (0.74
{{convert/sandbox|1|lbf/in|lk=on}} 1 [[pound-force]] per [[
{{convert/sandbox|1|Nm|lbf/in|lk=on}} 1 [[newton metre]] (0.06
{{convert/sandbox|1|lbf·ft|lk=on}} 1 [[Pound-foot (torque)|
{{convert/sandbox|1|Nm|lbf·ft|lk=on}} 1 [[newton metre]] (0.74
{{convert/sandbox|1|lb-f·ft|lk=on}} 1 [[Pound-foot (torque)|
{{convert/sandbox|1|Nm|lb-f·ft|lk=on}} 1 [[newton metre]] (0.74
{{convert/sandbox|1|lbf·ft|lk=on}} 1 [[Pound-foot (torque)|
{{convert/sandbox|1|Nm|lbf·ft|lk=on}} 1 [[newton metre]] (0.74
{{convert/sandbox|1|lb-fft|lk=on}} 1 [[Pound-foot (torque)|
{{convert/sandbox|1|Nm|lbfft|lk=on}} 1 [[newton metre]] (0.74
{{convert/sandbox|1|lb-fft|lk=on}} 1 [[Pound-foot (torque)|
{{convert/sandbox|1|Nm|lb-fft|lk=on}} 1 [[newton metre]] (0.74
{{convert/sandbox|1|lbft|lk=on}} 1 [[Pound-foot (torque)|
{{convert/sandbox|1|Nm|lbft|lk=on}} 1 [[newton metre]] (0.74
{{convert/sandbox|1|Nm|lbft kgm|lk=on}} 1 [[newton metre]] (0.74
{{convert/sandbox|1|m.kgf|lk=on}} 1 [[Kilogram metre|metre]]
{{convert/sandbox|1|Nm|m.kgf|lk=on}} 1 [[newton metre]] (0.16
{{convert/sandbox|1|m.kg-f|lk=on}} 1 [[Kilogram metre|metre]]
{{convert/sandbox|1|Nm|m.kg-f|lk=on}} 1 [[newton metre]] (0.16
{{convert/sandbox|1|mkgf|lk=on}} 1 [[Kilogram metre|metre]]
{{convert/sandbox|1|Nm|mkgf|lk=on}} 1 [[newton metre]] (0.16
{{convert/sandbox|1|mkg-f|lk=on}} 1 [[Kilogram metre|metre]]
{{convert/sandbox|1|Nm|mkg-f|lk=on}} 1 [[newton metre]] (0.16
{{convert/sandbox|1|mN.m|lk=on}} 1 [[Newton metre|milline
{{convert/sandbox|1|Nm|mN.m|lk=on}} 1 [[newton metre]] (1,00
{{convert/sandbox|1|N.m|lk=on}} 1 [[newton metre]] (0.74
{{convert/sandbox|1|N·m|lk=on}} 1 [[newton metre]] (0.74
{{convert/sandbox|1|Nm|lk=on}} 1 [[newton metre]] (0.74
{{convert/sandbox|1|Nm|Nm kgm|lk=on}} 1 [[newton metre]] (1.06
{{convert/sandbox|1|Nm|Nm lbfft|lk=on}} 1 [[newton metre]] (1.06
{{convert/sandbox|1|Nm|Nm lb-fft|lk=on}} 1 [[newton metre]] (1.06
{{convert/sandbox|1|Nm|Nm lbft|lk=on}} 1 [[newton metre]] (1.06

-- volume
{{convert/sandbox|1|µL|lk=on}} 1 [[microlitre]] (6.1<sp
{{convert/sandbox|1|m3|µL|lk=on}} 1 [[cubic metre]] (1.0<sp
{{convert/sandbox|1|µl|lk=on}} 1 [[microlitre]] (6.1<sp
{{convert/sandbox|1|m3|µl|lk=on}} 1 [[cubic metre]] (1.0<sp
{{convert/sandbox|1|m3|acre feet|lk=on}} 1 [[cubic metre]] (0.006
{{convert/sandbox|1|m3|acre foot|lk=on}} 1 [[cubic metre]] (0.006
{{convert/sandbox|1|m3|acre ft|lk=on}} 1 [[cubic metre]] (0.006
{{convert/sandbox|1|acre.ft|lk=on}} 1 [[acre foot]] (1,200&
{{convert/sandbox|1|m3|acre.ft|lk=on}} 1 [[cubic metre]] (0.006
{{convert/sandbox|1|acre.ft|lk=on}} 1 [[acre foot]] (1,200&
{{convert/sandbox|1|m3|acre.ft|lk=on}} 1 [[cubic metre]] (0.006
{{convert/sandbox|1|acre·foot|lk=on}} 1 [[acre foot]] (1,200&
{{convert/sandbox|1|m3|acre·foot|lk=on}} 1 [[cubic metre]] (0.006
{{convert/sandbox|1|acre·ft|lk=on}} 1 [[acre foot]] (1,200&
{{convert/sandbox|1|m3|acre·ft|lk=on}} 1 [[cubic metre]] (0.006
{{convert/sandbox|1|acre-feet|lk=on}} 1 [[acre foot]] (1,200&
{{convert/sandbox|1|m3|acre-feet|lk=on}} 1 [[cubic metre]] (0.006
{{convert/sandbox|1|bdft|lk=on}} 1 [[board foot]] (0.0024
{{convert/sandbox|1|m3|bdft|lk=on}} 1 [[cubic metre]] (420&
{{convert/sandbox|1|m3|board feet|lk=on}} 1 [[cubic metre]] (420
{{convert/sandbox|1|m3|board foot|lk=on}} 1 [[cubic metre]] (420
{{convert/sandbox|1|cc|lk=on}} 1 [[cubic centimetre]] (
{{convert/sandbox|1|m3|cc|lk=on}} 1 [[cubic metre]] (1,000
{{convert/sandbox|1|m3|cc L|lk=on}} 1 [[cubic metre]] (1,000
{{convert/sandbox|1|cid|lk=on}} 1 [[Cubic inch#Engine di
{{convert/sandbox|1|m3|cid|lk=on}} 1 [[cubic metre]] (61,06

```




```

{{convert/sandbox|1|m3|hL U.S.gal|lk=on}} 1 [[cubic metre]] (10&nb
{{convert/sandbox|1|m3|hL U.S.gal impgal|lk=on}} 1 [[cubic metre]] (10&nb
{{convert/sandbox|1|m3|hL U.S.gal impgal|lk=on}} 1 [[cubic metre]] (10&nb
{{convert/sandbox|1|m3|hL USgal|lk=on}} 1 [[cubic metre]] (10&nb
{{convert/sandbox|1|m3|hL USgal|lk=on}} 1 [[cubic metre]] (10&nb
{{convert/sandbox|1|m3|hL USgal impgal|lk=on}} 1 [[cubic metre]] (10&nb
{{convert/sandbox|1|m3|hL USgal impgal|lk=on}} 1 [[cubic metre]] (10&nb
{{convert/sandbox|1|hm3|lk=on}} 1 [[Cubic metre|cubic he
{{convert/sandbox|1|m3|hm3|lk=on}} 1 [[cubic metre]] (1.0<9
{{convert/sandbox|1|impbbl|lk=on}} 1 [[Barrel (unit)|imper
{{convert/sandbox|1|m3|impbbl|lk=on}} 1 [[cubic metre]] (6.1&
{{convert/sandbox|1|impbsh|lk=on}} 1 [[imperial bushel]] (3
{{convert/sandbox|1|m3|impbsh|lk=on}} 1 [[cubic metre]] (27&nb
{{convert/sandbox|1|impbu|lk=on}} 1 [[imperial bushel]] (6
{{convert/sandbox|1|m3|impbu|lk=on}} 1 [[cubic metre]] (27&nb
{{convert/sandbox|1|impfloz|lk=on}} 1 [[imperial fluid ounce
{{convert/sandbox|1|m3|impfloz|lk=on}} 1 [[cubic metre]] (35,00
{{convert/sandbox|1|m3|impfloz U.S.floz|lk=on}} 1 [[cubic metre]] (35,00
{{convert/sandbox|1|Impgal|lk=on}} 1 [[imperial gallon]] (4
{{convert/sandbox|1|m3|Impgal|lk=on}} 1 [[cubic metre]] (220&
{{convert/sandbox|1|impgal|lk=on}} 1 [[imperial gallon]] (4
{{convert/sandbox|1|m3|impgal|lk=on}} 1 [[cubic metre]] (220&
{{convert/sandbox|1|m3|impgal cuyd|lk=on}} 1 [[cubic metre]] (220&

-- volume5
{{convert/sandbox|1|USdrypt|lk=on}} 1 [[Pint|US dry pint]] (
{{convert/sandbox|1|m3|USdrypt|lk=on}} 1 [[cubic metre]] (1,800
{{convert/sandbox|1|USdryqt|lk=on}} 1 [[Quart|US dry quart]]
{{convert/sandbox|1|m3|USdryqt|lk=on}} 1 [[cubic metre]] (910&
{{convert/sandbox|1|USfloz|lk=on}} 1 [[US fluid ounce]] (30
{{convert/sandbox|1|m3|USfloz|lk=on}} 1 [[cubic metre]] (34,00
{{convert/sandbox|1|m3|USfloz impfloz|lk=on}} 1 [[cubic metre]] (34,00
{{convert/sandbox|1|USGAL|lk=on}} 1 [[US gallon]] (3.8&nb
{{convert/sandbox|1|m3|USGAL|lk=on}} 1 [[cubic metre]] (260&
{{convert/sandbox|1|USgal|lk=on}} 1 [[US gallon]] (3.8&nb
{{convert/sandbox|1|m3|USgal|lk=on}} 1 [[cubic metre]] (260&
{{convert/sandbox|1|usgal|lk=on}} 1 [[US gallon]] (3.8&nb
{{convert/sandbox|1|m3|usgal|lk=on}} 1 [[cubic metre]] (260&
{{convert/sandbox|1|m3|usgal impgal|lk=on}} 1 [[cubic metre]] (260&
{{convert/sandbox|1|m3|USgal impgal|lk=on}} 1 [[cubic metre]] (260&
{{convert/sandbox|1|m3|USgal impgal L|lk=on}} 1 [[cubic metre]] (260&
{{convert/sandbox|1|m3|USgal impgal l|lk=on}} 1 [[cubic metre]] (260&
{{convert/sandbox|1|m3|USgal L|lk=on}} 1 [[cubic metre]] (260&
{{convert/sandbox|1|m3|USgal l|lk=on}} 1 [[cubic metre]] (260&
{{convert/sandbox|1|m3|USgal L impgal|lk=on}} 1 [[cubic metre]] (260&
{{convert/sandbox|1|m3|USgal l impgal|lk=on}} 1 [[cubic metre]] (260&
{{convert/sandbox|1|m3|USgal m3|lk=on}} 1 [[cubic metre]] (260&
{{convert/sandbox|1|m3|USgal USdrygal|lk=on}} 1 [[cubic metre]] (260&
{{convert/sandbox|1|USgi|lk=on}} 1 [[Gill (unit)|gill]] (
{{convert/sandbox|1|m3|USgi|lk=on}} 1 [[cubic metre]] (8,500
{{convert/sandbox|1|usgi|lk=on}} 1 [[Gill (unit)|gill]] (
{{convert/sandbox|1|m3|usgi|lk=on}} 1 [[cubic metre]] (8,500
{{convert/sandbox|1|USkenning|lk=on}} 1 [[Kenning (unit)|US ke
{{convert/sandbox|1|m3|USkenning|lk=on}} 1 [[cubic metre]] (57&nb
{{convert/sandbox|1|USmin|lk=on}} 1 [[Minim (unit)|US min]
{{convert/sandbox|1|m3|USmin|lk=on}} 1 [[cubic metre]] (16,00
{{convert/sandbox|1|USoz|lk=on}} 1 [[US fluid ounce]] (30
{{convert/sandbox|1|m3|USoz|lk=on}} 1 [[cubic metre]] (34,00
{{convert/sandbox|1|m3|USoz impoz|lk=on}} 1 [[cubic metre]] (34,00
{{convert/sandbox|1|m3|USoz mL|lk=on}} 1 [[cubic metre]] (34,00
{{convert/sandbox|1|m3|USoz ml|lk=on}} 1 [[cubic metre]] (34,00
{{convert/sandbox|1|USpk|lk=on}} 1 [[Peck|US peck]] (8.8&
{{convert/sandbox|1|m3|USpk|lk=on}} 1 [[cubic metre]] (110&
{{convert/sandbox|1|USpt|lk=on}} 1 [[Pint|US pint]] (0.47

```



```

{{convert/sandbox|1|m3|USpt|lk=on}} 1 [[cubic metre]] (2,100
{{convert/sandbox|1|USqt|lk=on}} 1 [[United States custom
{{convert/sandbox|1|m3|USqt|lk=on}} 1 [[cubic metre]] (1,100
{{convert/sandbox|1|USquart|lk=on}} 1 [[Quart|US quart]] (94
{{convert/sandbox|1|m3|USquart|lk=on}} 1 [[cubic metre]] (1,100
{{convert/sandbox|1|winecase|lk=on}} 1 [[Case (goods)|case]]
{{convert/sandbox|1|m3|winecase|lk=on}} 1 [[cubic metre]] (110
{{convert/sandbox|1|yd3|lk=on}} 1 [[cubic yard]] (0.76&
{{convert/sandbox|1|m3|yd3|lk=on}} 1 [[cubic metre]] (1.3&
{{convert/sandbox|1|µL|lk=on}} 1 [[microlitre]] (6.1<sp
{{convert/sandbox|1|m3|µL|lk=on}} 1 [[cubic metre]] (1.0<sp
{{convert/sandbox|1|µl|lk=on}} 1 [[microlitre]] (6.1<sp
{{convert/sandbox|1|m3|µl|lk=on}} 1 [[cubic metre]] (1.0<sp

-- misc
{{convert/sandbox|1|A.h|lk=on}} 1 [[ampere-hour]] (3,600
{{convert/sandbox|1|A·h|lk=on}} 1 [[ampere-hour]] (3,600
{{convert/sandbox|1|coulomb|lk=on}} 1 [[coulomb]] (6.2<span
{{convert/sandbox|1|e|lk=on}} 1 [[elementary charge]]
{{convert/sandbox|1|cuft/sqmi|lk=on}} 1 [[cubic foot]] per [[sq
{{convert/sandbox|1|dtex|lk=on}} 1 [[Units of textile mea
{{convert/sandbox|1|si tsfc|lk=on}} 1 [[Thrust specific fuel
{{convert/sandbox|1|gCO2/km|lk=on}} 1 [[Exhaust gas|gram of
{{convert/sandbox|1|gCO2/mi|lk=on}} 1 [[Exhaust gas|gram of
{{convert/sandbox|1|hp/LT|lk=on}} 1 [[Power-to-weight rati
{{convert/sandbox|1|hp/ST|lk=on}} 1 [[Power-to-weight rati
{{convert/sandbox|1|impgalh2o|lk=on}} 1 [[Imperial gallon|impe
{{convert/sandbox|1|keVT|lk=on}} 1 [[Electronvolt|kiloelé
{{convert/sandbox|1|kg/kW|lk=on}} 1 [[Kilowatt|kilogram pe
{{convert/sandbox|1|kgCO2/km|lk=on}} 1 [[Exhaust gas|kilogram
{{convert/sandbox|1|kgCO2/L|lk=on}} 1 [[Exhaust gas|kilogram
{{convert/sandbox|1|kgpsqm|lk=on}} 1 [[Kilogram-force|kiloc
{{convert/sandbox|1|kPa/m|lk=on}} 1 [[Pascal (unit)|kilopa
{{convert/sandbox|1|kW/t|lk=on}} 1 [[Power-to-weight rati
{{convert/sandbox|1|tsfc|lk=on}} 1 [[Thrust specific fuel
{{convert/sandbox|1|lb/hp|lk=on}} 1 [[Horsepower|pound per
{{convert/sandbox|1|lbCO2/mi|lk=on}} 1 [[Exhaust gas|pound of
{{convert/sandbox|1|lbCO2/USgal|lk=on}} 1 [[Exhaust gas|pound pe
{{convert/sandbox|1|lbm/cuin|lk=on}} 1 [[Density|pound mass p
{{convert/sandbox|1|lbm/in3|lk=on}} 1 [[Density|pound mass p
{{convert/sandbox|1|LT/acre|lk=on}} 1 [[long ton]] per [[ac
{{convert/sandbox|1|m2/ha|lk=on}} 1 [[Basal area|square mé
{{convert/sandbox|1|m3/ha|lk=on}} 1 [[Hectare|cubic metre]
{{convert/sandbox|1|m3/km2|lk=on}} 1 [[cubic metre]] per [[
{{convert/sandbox|1|ozCO2/mi|lk=on}} 1 [[Exhaust gas|ounce of
{{convert/sandbox|1|PS/t|lk=on}} 1 [[Power-to-weight rati
{{convert/sandbox|1|psi/ft|lk=on}} 1 [[Pounds per square in
{{convert/sandbox|1|sqft/acre|lk=on}} 1 [[square foot per acre
{{convert/sandbox|1|ST/acre|lk=on}} 1 [[short ton]] per [[ac
{{convert/sandbox|1|t/ha|lk=on}} 1 [[tonne]] per [[hectare
{{convert/sandbox|1|U.S.gal/acre|lk=on}} 1 [[U.S. gallon]] per [[
{{convert/sandbox|1|USbu/acre|lk=on}} 1 [[Bushel|US bushel per
{{convert/sandbox|1|USgal/acre|lk=on}} 1 [[US gallon]] per [[ac
{{convert/sandbox|1|usgalh2o|lk=on}} 1 [[United States custom

-- Options.
{{convert/sandbox|10|mi|km|4}} 10 miles (16.0934&nbsp;k
{{convert/sandbox|10|mi|km|sigfig=4}} 10 miles (16.09&nbsp;km)
{{convert/sandbox|100|in|cm|abbr=off}} 100 inches (250 centimet
{{convert/sandbox|100|in|cm|abbr=on}} 100&nbsp;in (250&nbsp;cm)
{{convert/sandbox|100|in|cm|abbr=in}} 100&nbsp;in (250 centimé
{{convert/sandbox|100|in|cm|abbr=out}} 100 inches (250&nbsp;cm)
{{convert/sandbox|100|in|cm|abbr=values}} 100 (250)
{{convert/sandbox|100|in|cm|disp=flip}} 250 centimetres (100&nbsp;

```



{{convert/sandbox 100 in cm abbr=off disp=flip}}	250 centimetres (100 in
{{convert/sandbox 100 in cm abbr=on disp=flip}}	250 cm (100 in
{{convert/sandbox 100 in cm abbr=in disp=flip}}	250 cm (100 inches)
{{convert/sandbox 100 in cm abbr=out disp=flip}}	250 centimetres (100
{{convert/sandbox 100 in cm abbr=values disp=flip}}	250 (100)
{{convert/sandbox 1 cuyd disp=u2}}	m³
{{convert/sandbox 1 cuyd disp=unit}}	cubic yard
{{convert/sandbox 2 cuyd disp=unit}}	cubic yards
{{convert/sandbox 2 cuyd adj=off disp=unit}}	cubic yards
{{convert/sandbox 2 cuyd adj=on disp=unit}}	cubic-yard
{{convert/sandbox 2 cuyd m3 adj=mid extra text}}	2-cubic-yard extra text
{{convert/sandbox 2 cuyd adj=on}}	2-cubic-yard (1.5 n
{{convert/sandbox 190 ft m adj=mid bridge}}	190-foot bridge (58
{{convert/sandbox 190 ft m adj=mid -long}}	190-foot-long (58 n
{{convert/sandbox 2 cuyd m3 adj=mid xyz}}	2-cubic-yard xyz (1.5&n
{{convert/sandbox 2 cuyd m3 adj=mid -xyz}}	2-cubic-yard-xyz (1.5&n
{{convert/sandbox 5 to 10 kg lb 2}}	5 to 10 kilograms (11.0
{{convert/sandbox -12 kg lb disp=5}}	-12 kilograms (-25
{{convert/sandbox 123.45 m dam}}	123.45 metres (12.345&n
{{convert/sandbox 123.45 m dam sp=us}}	123.45 meters (12.345&n
{{convert/sandbox 12.345 dam m sp=us}}	12.345 dekameters (123.4
{{convert/sandbox 123.45 m dam sp=us disp=flip}}	12.345 dekameters (123.4
{{convert/sandbox 12 ft in abbr=off adj=on}}	12-foot (140-inch)
{{convert/sandbox 12 ft m abbr=off adj=on}}	12-foot (3.7-metre)
{{convert/sandbox 1 ft in disp=unit}}	foot
{{convert/sandbox 1 foot in disp=unit}}	foot
{{convert/sandbox 6 ft in disp=unit}}	feet
{{convert/sandbox 6 foot in disp=unit}}	foot
{{convert/sandbox 6 foot in disp=unit abbr=on}}	ft
{{convert/sandbox 6 ft in disp=number}}	72
{{convert/sandbox 6 ft in disp=output number only}}	72
{{convert/sandbox 6 ft in disp=output}}	72 in
{{convert/sandbox 6 ft in disp=output only}}	72 in
{{convert/sandbox 6 ft in disp=output only adj=flip}}	72 inches
{{convert/sandbox 6 ft in disp=output only abbr=off}}	72 inches
{{convert/sandbox 6 ft in disp=output only abbr=off adj=on}}	72-inch
{{convert/sandbox 6 ft in disp=output only abbr=off adj=mid -long}}	72-inch
{{convert/sandbox 3.21 ft cm lk=off}}	3.21 feet (98 cm)
{{convert/sandbox 3.21 ft cm lk=on}}	3.21 [[Foot (unit) feet]
{{convert/sandbox 3.21 ft cm lk=in}}	3.21 [[Foot (unit) feet]
{{convert/sandbox 3.21 ft cm lk=out}}	3.21 feet (98 [[Cer
{{convert/sandbox 3.21 ft cm disp=flip}}	98 centimetres (3.21&nb
{{convert/sandbox 3.21 ft cm lk=off disp=flip}}	98 centimetres (3.21&nb
{{convert/sandbox 3.21 ft cm lk=on disp=flip}}	98 [[centimetre]]s (3.21
{{convert/sandbox 3.21 ft cm lk=in disp=flip}}	98 [[centimetre]]s (3.21
{{convert/sandbox 3.21 ft cm lk=out disp=flip}}	98 centimetres (3.21&nb
{{convert/sandbox 3.21 ft cm lk=off abbr=on}}	3.21 ft (98 cm)
{{convert/sandbox 3.21 ft cm lk=on abbr=on}}	3.21 [[Foot (unit)
{{convert/sandbox 3.21 ft cm lk=in abbr=on}}	3.21 [[Foot (unit)
{{convert/sandbox 3.21 ft cm lk=out abbr=on}}	3.21 ft (98 [[
{{convert/sandbox 3.21 ft cm lk=off abbr=off}}	3.21 feet (98 centimet
{{convert/sandbox 3.21 ft cm lk=on abbr=off}}	3.21 [[Foot (unit) feet]
{{convert/sandbox 3.21 ft cm lk=in abbr=off}}	3.21 [[Foot (unit) feet]
{{convert/sandbox 3.21 ft cm lk=out abbr=off}}	3.21 feet (98 [[centime
{{convert/sandbox 3.21 ft cm lk=off abbr=in}}	3.21 ft (98 centime
{{convert/sandbox 3.21 ft cm lk=on abbr=in}}	3.21 [[Foot (unit)
{{convert/sandbox 3.21 ft cm lk=in abbr=in}}	3.21 [[Foot (unit)
{{convert/sandbox 3.21 ft cm lk=out abbr=in}}	3.21 ft (98 [[centi
{{convert/sandbox 3.21 ft cm lk=off abbr=out}}	3.21 feet (98 cm)
{{convert/sandbox 3.21 ft cm lk=on abbr=out}}	3.21 [[Foot (unit) feet]
{{convert/sandbox 3.21 ft cm lk=in abbr=out}}	3.21 [[Foot (unit) feet]
{{convert/sandbox 3.21 ft cm lk=out abbr=out}}	3.21 feet (98 [[Cer
{{convert/sandbox 3.21 ft cm lk=out abbr=out adj=flip}}	98 centimetres (3.21&nb
{{convert/sandbox 3.21 ft cm lk=out abbr=out disp=flip}}	98 centimetres (3.21&nb



{{convert/sandbox 3.21 ft cm lk=out abbr=out adj=on}}	3.21-foot (98 [[Centimetre]]s)
{{convert/sandbox 3.21 ft cm lk=out abbr=out adj=mid bridge}}	3.21-foot bridge
{{convert/sandbox 3.21 ft cm disp=x (begin end)}}	3.21 feet (begin 98 [[Centimetre]]s)
{{convert/sandbox 3.21 ft cm disp=x or}}	3.21 feet or98 [[Centimetre]]s
{{convert/sandbox 3.21 ft cm disp=x}}	3.21 feet [[98 [[Centimetre]]s]]
{{convert/sandbox 3.21 ft cm disp=comma}}	3.21 feet, 98 [[Centimetre]]s
{{convert/sandbox 3.21 ft cm disp=sqbr}}	3.21 feet [98 [[Centimetre]]s]
{{convert/sandbox 3.21 ft cm disp=or}}	3.21 feet or 98 centimetres
{{convert/sandbox 3.21 ft cm disp=slash}}	3.21 feet / 98 centimetres
{{convert/sandbox 3.21 ft cm disp=s}}	3.21 feet / 98 centimetres
{{convert/sandbox 3.21 ft cm disp=/}}	3.21 feet / 98 centimetres
{{convert/sandbox 3.21 ft cm disp=2}}	98 [[Centimetre]]s;cm
{{convert/sandbox 3.21 ft cm disp=b}}	3.21 feet (98 [[Centimetre]]s)
{{convert/sandbox 3.21 ft cm disp=br}}	3.21 feet 98 centimetres
{{convert/sandbox 9 kPa abbr=~}}	9 kilopascals (kPa, 1.30 [[Pascal]]s)
{{convert/sandbox 1.25 km ftin abbr=~}}	1.25 kilometres (km, 4,1 [[Kilometre]]s)
{{convert/sandbox 33 e9m3 e9cuft adj=on abbr=off}}	33-billion-cubic-metre (33,000,000,000 [[Cubic metre]]s)
{{convert/sandbox 400 LT adj=on abbr=off}}	400-long-ton (410-tonne)
{{convert/sandbox 1.5 mi km adj=1}}	1.5 miles (2.4 [[Kilometre]]s;km)
{{convert/sandbox 1 mi km adj=1}}	1 mile (1.6 [[Kilometre]]s;km)
{{convert/sandbox 0.5 mi km adj=1}}	0.5 mile (0.80 [[Kilometre]]s;km)
{{convert/sandbox 0 mi km adj=1}}	0 miles (0 [[Kilometre]]s;km)
{{convert/sandbox 75 ft m abbr=off adj=j}}	75 feet (23 [[Metre]]s;metres)
{{convert/sandbox 4543.5 m ft disp=flip adj=j}}	14,906 feet (4,543.5 [[Metre]]s)
{{convert/sandbox 75 ft m disp=flip5 adj=j}}	25 metres (75 feet)
{{convert/sandbox 3.21 ft cm abbr=in disp=flip}}	98 [[Centimetre]]s (3.21 feet)
{{convert/sandbox 3.21 ft cm abbr=out disp=flip}}	98 centimetres (3.21 feet)
{{convert/sandbox 3.21 ft cm lk=in disp=flip}}	98 [[centimetre]]s (3.21 feet)
{{convert/sandbox 3.21 ft cm lk=out disp=flip}}	98 centimetres (3.21 feet)
{{convert/sandbox 1 m ft adj=pre FIRST}}	1 FIRST metre (3.3 feet)
{{convert/sandbox 1 m ft adj=pre FIRST disp=flip}}	3.3 FIRST feet (1 metre)
{{convert/sandbox 1 m ft disp=preunit FIRST SECOND}}	1 FIRST metre (3.3 SECONDS)
{{convert/sandbox 1 m ft disp=preunit FIRST SECOND abbr=off}}	1 FIRST metre (3.3 SECONDS)
{{convert/sandbox 1 m ft disp=preunit +}}	1+metre (3.3+ft)
{{convert/sandbox 4 m ft disp=preunit + SECOND}}	4+ metres (13 SECONDSft)
{{convert/sandbox 4 m ft disp=preunit linear}}	4 linearmetres (13 linearmetres)
{{convert/sandbox 1780 kg lb -1 abbr=on}}	1,780 [[Kilogram]]s (3,920 [[Pound]]s)
{{convert/sandbox 1234 tonocomma 5678 kg oz}}	1234 to 5678 kilograms (27,300 to 125,000 [[Pound]]s)
{{convert/sandbox 1234 -nocomma 5678 kg oz}}	1234–5678 kilograms (27,300–125,000 [[Pound]]s)
{{convert/sandbox 233435993 in m sp=us disp=or}}	233,435,993 inches or 5,913.6 [[Metre]]s
{{convert/sandbox 0.12 km yd sing=1}}	0.12 kilometre (130 [[Yard]]s)
{{convert/sandbox 12 km yd sing=1}}	12 kilometres (13,000 [[Yard]]s)
{{convert/sandbox 12 km yd sing=flip}}	13,000 yards (12 [[Kilometre]]s;km)
{{convert/sandbox 12 km yd sing=j}}	12 [[Kilometre]]s (13,000 [[Yard]]s)
{{convert/sandbox 12 km yd FIRST sing=mid}}	12-kilometre FIRST (13,000 [[Yard]]s)
{{convert/sandbox 12 km yd FIRST sing=pre}}	12 FIRST kilometres (13,000 [[Yard]]s)
{{convert/sandbox 12 km yd sing=off}}	12 kilometres (13,000 [[Yard]]s)
{{convert/sandbox 12 km yd sing=on}}	12-kilometre (13,000 [[Yard]]s)
{{convert/sandbox 123.4456 km yd sing=ri1}}	123.4 kilometres (135,000 [[Yard]]s)
{{convert/sandbox 123.4456 km yd sing=ri2}}	123.45 kilometres (135,000 [[Yard]]s)
{{convert/sandbox 123.4456 km yd sing=ri3}}	123.446 kilometres (135,000 [[Yard]]s)
{{convert/sandbox 0.98 AU e9km lk=on}}	0.98 [[astronomical unit]] (146,800,000 [[Kilometre]]s)
{{convert/sandbox 1.23 e12km lk=on}}	1.23 [[1000000000000]] kilometres
{{convert/sandbox 1.23 e12km abbr=off lk=on}}	1.23 [[1000000000000]] kilometres
{{convert/sandbox 1.23 e3m3}}	1.23 thousand cubic metres
{{convert/sandbox 1.23 e3m3 abbr=on}}	1.23thousand cubic metres
{{convert/sandbox 1.23 e3m3 adj=on}}	1.23-thousand-cubic-metres
{{convert/sandbox 1.23 e3m3 lk=in}}	1.23 thousand [[cubic metres]]
{{convert/sandbox 1.23 e6m3 lk=in}}	1.23 million [[cubic metres]]
{{convert/sandbox 1.23 e9m3 lk=in}}	1.23 [[1000000000]] [[cubic metres]]
{{convert/sandbox 1.23 e12m3 lk=in}}	1.23 [[1000000000000]] [[cubic metres]]
{{convert/sandbox 1.23 e15m3 lk=in}}	1.23 [[10000000000000]] [[cubic metres]]
{{convert/sandbox 1.23 e15m3 lk=on}}	1.23 [[10000000000000]] kilometres
{{convert/sandbox 1.23 e15m3 abbr=on lk=on}}	1.23thousand cubic metres



```

{{convert/sandbox|4500|acre ft|e9USgal e6m3}} 4,500 acre feet (1.5<spa
{{convert/sandbox|4500|acre ft|e9USgal e6m3|abbr=off}} 4,500 acre feet (1.5&nb

-- Ranges.
{{convert/sandbox|-3|-|+3|m|ft}} -3+3 metres (-9.8-9.8&
{{convert/sandbox|-3|-|3|m|ft}} -3-3 metres (-9.8-9.8&nt
{{convert/sandbox|-8|-|-3|F|C}} -8&nbsp;-3&nbsp;°F (-2
{{convert/sandbox|-8|-|-3|m|ft}} -8&nbsp;-3
{{convert/sandbox|4|-|9|L|USgal|abbr=off|sp=us|lk=out}} 4-9 liters (1.1-2.4 [[US
{{convert/sandbox|4|-|9|L|USgal|abbr=off}} 4-9 litres (1.1-2.4 US g
{{convert/sandbox|5|-|10|kg|lb|2}} 5-10 kilograms (11.02-22
{{convert/sandbox|5|-|12|kg|abbr=in}} 5-12&nbsp;kg (11-26 pour
{{convert/sandbox|5|-|12|kg|abbr=off}} 5-12 kilograms (11-26 pe
{{convert/sandbox|5|-|12|kg|abbr=out}} 5-12 kilograms (11-26&nt
{{convert/sandbox|5|-|12|kg}} 5-12 kilograms (11-26&nt
{{convert/sandbox|41|-|50|F|K}} 41-50&nbsp;°F (278-283&
{{convert/sandbox|4500|-|4900|acre ft|e9USgal e6m3|abbr=off|lk=on}} 4,500-4,900
{{convert/sandbox|4500|-|4900|acre ft|e9USgal e6m3|abbr=off}} 4,500-4,900 a
{{convert/sandbox|4500|-|4900|acre ft|e9USgal e6m3|lk=on}} 4,500-4,900 |
{{convert/sandbox|4500|-|4900|acre ft|e9USgal e6m3}} 4,500-4,900 acre feet (1
{{convert/sandbox|3|-|4|in|mm}} 3-4 inches (
{{convert/sandbox|12.5|+/-|2.3|kg|lb|abbr=off}} 12.5&nbsp;±
{{convert/sandbox|12.5|+/-|2.3|kg|lb}} 12.5&nbsp;±
{{convert/sandbox|12.5|±|2.3|kg}} 12.5&nbsp;±
{{convert/sandbox|3|&|4|in|mm}} 3 and 4 inch
{{convert/sandbox|3|and(-)|4|in|mm}} 3 and 4 inch
{{convert/sandbox|3|±|4|in|mm}} 3&nbsp;±&nb
{{convert/sandbox|45|+/-|8|mm}} 45&nbsp;±&nt
{{convert/sandbox|5|&|12|kg}} 5 and 12 kilograms (11 a
{{convert/sandbox|5|and|12|kg|lb|abbr=off|adj=on}} 5-and-12-kilogram (11-ar
{{convert/sandbox|5|and|12|kg|lb|abbr=off}} 5 and 12 kilograms (11 a
{{convert/sandbox|5|and|12|kg}} 5 and 12 kilograms (11 a
{{convert/sandbox|8|or|10|ft|m|abbr=off|adj=on}} 8-or-10-foot (2.4-or-3.0
{{convert/sandbox|8|or|10|ft|m|abbr=off}} 8 or 10 feet (2.4 or 3.0
{{convert/sandbox|8|or|10|ft|m|adj=on}} 8-or-10-foot (2.4 or 3.0
{{convert/sandbox|8|or|10|ft|m}} 8 or 10 feet (2.4 or 3.0
{{convert/sandbox|375|to about|500|g|lb|sp=us}} 375 to about 500 grams (
{{convert/sandbox|-3|to(-)|-3|m|ft}} -3&nbsp;to -3 metres (-9
{{convert/sandbox|5|to(-)|12|kg}} 5&nbsp;to 12 kilograms (
{{convert/sandbox|5|to-|12|kg|abbr=on}} 5-12&nbsp;kg (11-26&nb
{{convert/sandbox|5|to-|12|kg}} 5&nbsp;to 12 kilograms (
{{convert/sandbox|41|to|50|F|C}} 41 to 50&nbsp;°F (5 to 2
{{convert/sandbox|5|to|12|kg|abbr=on}} 5 to 12&nbsp;kg (11 to 2
{{convert/sandbox|5|to|12|kg|lb|abbr=off}} 5 to 12 kilograms (11 to
{{convert/sandbox|5|to|12|kg}} 5 to 12 kilograms (11 to
{{convert/sandbox|5|to|7|L|USgal|abbr=mos}} 5 litres to 7 litres (1
{{convert/sandbox|60|by|120|m|ft|abbr=in}} 60 by 120&nbsp;m (200 by
{{convert/sandbox|60|by|120|m|ft|abbr=off}} 60 by 120 metres (200 by
{{convert/sandbox|60|by|120|m|ft|abbr=on}} 60 by 120&nbsp;m (200 by
{{convert/sandbox|60|by|120|m|ft|abbr=out}} 60 by 120 metres (200 by
{{convert/sandbox|60|by|120|m|ft}} 60 by 120 metres (200 by
{{convert/sandbox|3|x|4|in|mm}} 3 by 4 inch
{{convert/sandbox|1|x|1|m|ft|sp=us}} 1 by 1 meters (3.3&nbsp;
{{convert/sandbox|60|x|120|m|ft|abbr=in}} 60&nbsp;m x&nbsp;120&nb
{{convert/sandbox|60|x|120|m|ft|abbr=off}} 60 by 120 metres (200 by
{{convert/sandbox|60|x|120|m|ft|abbr=on}} 60&nbsp;m x&nbsp;120&nb
{{convert/sandbox|60|x|120|m|ft|abbr=out}} 60 by 120 metres (200&nt
{{convert/sandbox|60|x|120|m|ft}} 60 by 120 metres (200&nt
{{convert/sandbox|8|xx|10|ft|m|abbr=off}} 8&nbsp;x&nbsp;10 feet (2
{{convert/sandbox|8|xx|10|ft|m|abbr=on}} 8&nbsp;x&nbsp;10&nbsp;ft
{{convert/sandbox|8|xx|10|ft|m|adj=on}} 8-x-10-foot (2.4&nbsp;x&
{{convert/sandbox|8|xx|10|ft|m}} 8&nbsp;x&nbsp;10 feet (2

```

-- Engineering notation.



```

{{convert/sandbox|1.23|e12U.S.gal}} 1.23&nbsp;trillion U.S.
{{convert/sandbox|1.23|e12cuft}} 1.23&nbsp;trillion cubic
{{convert/sandbox|1.23|e12m3/a}} 1.23&nbsp;trillion cubic
{{convert/sandbox|1.23|e3BTU}} 1.23&nbsp;thousand Briti
{{convert/sandbox|1.23|e3acre}} 1.23&nbsp;thousand acre
{{convert/sandbox|1.23|e3m2}} 1.23&nbsp;thousand squa
{{convert/sandbox|1.23|e3m3/a}} 1.23&nbsp;thousand cubic
{{convert/sandbox|1.23|e6BTU}} 1.23&nbsp;million Britis
{{convert/sandbox|1.23|e6L}} 1.23&nbsp;million litres
{{convert/sandbox|1.23|e6km}} 1.23&nbsp;million kilome
{{convert/sandbox|1.23|e6m3/h}} 1.23&nbsp;million cubic
{{convert/sandbox|1.23|e6mi}} 1.23&nbsp;million miles
{{convert/sandbox|1.23|e9USgal/a}} 1.23&nbsp;billion US gal
{{convert/sandbox|1.23|e9impgal}} 1.23&nbsp;billion imperi
{{convert/sandbox|123.4|e3m2}} 123.4&nbsp;thousand squa
{{convert/sandbox|1.23|e6L}} 1.23&nbsp;million litres
{{convert/sandbox|1.23|e6L|e3usgal|abbr=off|lk=on}} 1.23&nbsp;million [[litr
{{convert/sandbox|1.23|e9impgal}} 1.23&nbsp;billion imperi
{{convert/sandbox|1.23|e9impgal|e3usgal|abbr=off|lk=on}} 1.23&nbsp;[[1000000000
{{convert/sandbox|1.23|e12cuft}} 1.23&nbsp;trillion cubic

-- SI prefixes.
{{convert/sandbox|1234.56789|Ym|in|lk=on}} 1,234.56789 [[yottametre
{{convert/sandbox|1234.56789|Zm|in|lk=on}} 1,234.56789 [[zettametre
{{convert/sandbox|1234.56789|Em|in|lk=on}} 1,234.56789 [[exametre]]
{{convert/sandbox|1234.56789|Pm|in|lk=on}} 1,234.56789 [[petametre]
{{convert/sandbox|1234.56789|Tm|in|lk=on}} 1,234.56789 [[terametre]
{{convert/sandbox|1234.56789|Gm|in|lk=on}} 1,234.56789 [[gigametre]
{{convert/sandbox|1234.56789|Mm|in|lk=on}} 1,234.56789 [[megametre]
{{convert/sandbox|1234.56789|km|in|lk=on}} 1,234.56789 [[kilometre]
{{convert/sandbox|1234.56789|hm|in|lk=on}} 1,234.56789 [[hectometre]
{{convert/sandbox|1234.56789|dam|in|lk=on}} 1,234.56789 [[decametre]
{{convert/sandbox|1234.56789|dm|in|lk=on}} 1,234.56789 [[decimetre]
{{convert/sandbox|1234.56789|cm|in|lk=on}} 1,234.56789 [[centimetre]
{{convert/sandbox|1234.56789|mm|in|lk=on}} 1,234.56789 [[millimetre]
{{convert/sandbox|1234.56789|µm|in|lk=on}} 1,234.56789 [[micrometre]
{{convert/sandbox|1234.56789|µm|in|lk=on}} 1,234.56789 [[micrometre]
{{convert/sandbox|1234.56789|um|in|lk=on}} 1,234.56789 [[micrometre]
{{convert/sandbox|1234.56789|nm|in|lk=on}} 1,234.56789 [[nanometre]
{{convert/sandbox|1234.56789|pm|in|lk=on}} 1,234.56789 [[picometre]
{{convert/sandbox|1234.56789|fm|in|lk=on}} 1,234.56789 [[femtometre]
{{convert/sandbox|1234.56789|am|in|lk=on}} 1,234.56789 [[Metre|att
{{convert/sandbox|1234.56789|zm|in|lk=on}} 1,234.56789 [[Metre|zept
{{convert/sandbox|1234.56789|ym|in|lk=on}} 1,234.56789 [[Metre|yoct
{{convert/sandbox|1234.56789|Ym|in|lk=on|abbr=on}} 1,234.56789&nbsp;[[Yotta
{{convert/sandbox|1234.56789|Zm|in|lk=on|abbr=on}} 1,234.56789&nbsp;[[Zetta
{{convert/sandbox|1234.56789|Em|in|lk=on|abbr=on}} 1,234.56789&nbsp;[[Exame
{{convert/sandbox|1234.56789|Pm|in|lk=on|abbr=on}} 1,234.56789&nbsp;[[Petai
{{convert/sandbox|1234.56789|Tm|in|lk=on|abbr=on}} 1,234.56789&nbsp;[[Terai
{{convert/sandbox|1234.56789|Gm|in|lk=on|abbr=on}} 1,234.56789&nbsp;[[Giga
{{convert/sandbox|1234.56789|Mm|in|lk=on|abbr=on}} 1,234.56789&nbsp;[[Mega
{{convert/sandbox|1234.56789|km|in|lk=on|abbr=on}} 1,234.56789&nbsp;[[Kilon
{{convert/sandbox|1234.56789|hm|in|lk=on|abbr=on}} 1,234.56789&nbsp;[[Hecte
{{convert/sandbox|1234.56789|dam|in|lk=on|abbr=on}} 1,234.56789&nbsp;[[Deca
{{convert/sandbox|1234.56789|dm|in|lk=on|abbr=on}} 1,234.56789&nbsp;[[Deci
{{convert/sandbox|1234.56789|cm|in|lk=on|abbr=on}} 1,234.56789&nbsp;[[Centi
{{convert/sandbox|1234.56789|mm|in|lk=on|abbr=on}} 1,234.56789&nbsp;[[Milli
{{convert/sandbox|1234.56789|µm|in|lk=on|abbr=on}} 1,234.56789&nbsp;[[Micro
{{convert/sandbox|1234.56789|µm|in|lk=on|abbr=on}} 1,234.56789&nbsp;[[Micro
{{convert/sandbox|1234.56789|um|in|lk=on|abbr=on}} 1,234.56789&nbsp;[[Micro
{{convert/sandbox|1234.56789|nm|in|lk=on|abbr=on}} 1,234.56789&nbsp;[[Nanon
{{convert/sandbox|1234.56789|pm|in|lk=on|abbr=on}} 1,234.56789&nbsp;[[Picon
{{convert/sandbox|1234.56789|fm|in|lk=on|abbr=on}} 1,234.56789&nbsp;[[Femte
{{convert/sandbox|1234.56789|am|in|lk=on|abbr=on}} 1,234.56789&nbsp;[[Metre

```



```

{{convert/sandbox|1234.56789|zm|in|lk=on|abbr=on}}      1,234.56789&nbsp;[[Metre
{{convert/sandbox|1234.56789|ym|in|lk=on|abbr=on}}      1,234.56789&nbsp;[[Metre

-- Spell.
{{convert/sandbox|1+2/3|in|mm|spell=in}}                one and two-thirds inch
{{convert/sandbox|2/3|mi|km|spell=In}}                 Two-thirds mile (1.1&nbsp;
{{convert/sandbox|1000000|mi|km|spell=in}}              one million miles (1,600
{{convert/sandbox|1/4|m|m|spell=in}}                    one-quarter metre (0.25&
{{convert/sandbox|1+1/2|ft|spell=In}}                   One and a half feet (0.4

-- New units.
{{convert/sandbox|111|$/oilbbl}}                         $111 per barrel ($700/m
{{convert/sandbox|75,000|bushels|L|adj=on}}              75,000-US-bushel (2,600
{{convert/sandbox|39427|acre-ft}}                        39,427 acre feet (48,632
{{convert/sandbox|26|kt}}                                 26 kilotonnes (26,000 lo
{{convert/sandbox|100|m|yds}}                             100 metres (110&nbsp;y
{{convert/sandbox|123|km/s2}}                             123 kilometres per secon
{{convert/sandbox|123|km/s2|5|sp=us|abbr=on|lk=on}}     123&nbsp;[[Acceleration

]==]

local p = require('Module:Convert/tester')
p.tests = tests
return p

```