Electronică Digitală : proiect alternativ

Student: Petre Leonard Macamete

Ce face:
Proiectul își propune a converti unități de măsură din următoarele categorii:

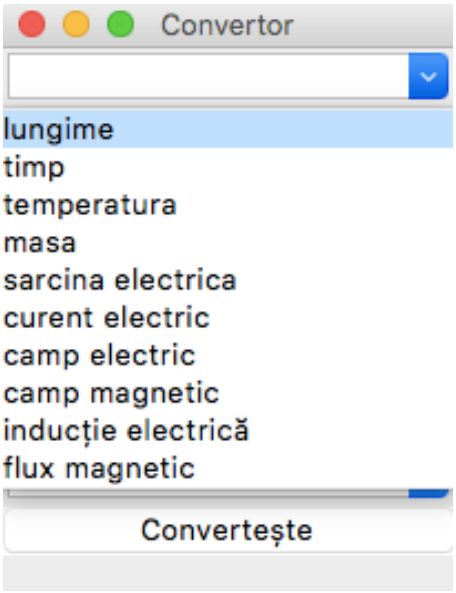
- Lungime
- Timp
- Temperatură
- Masă
- Câmp electric
- Câmp magnetic
- Inducție electrică
- flux magnetic

Structura:

Pentru adăugarea cu ușurință de noi unități, am folosit două dicționare: primul care să transforme categoria într-un set de șiruri corespunzătoare categoriei alese și al doilea este să transforme șirul care se afișează în șirul pe care Pint îl înțelege.
Pentru afișarea opțiunilor, am folosit 5 ComboBox-uri: primul pentru afișarea categoriei, al doilea și al patrulea pentru afișarea sufixelor SI, al treilea și al cincilea pentru afișarea mărimilor din categoria selectată.
Pentru input am folosit un Entry care cere valoarea de convertit.
Pentru output am folosit un Label care afișează valoarea convertită cu marja de eroare de ±0.1%.

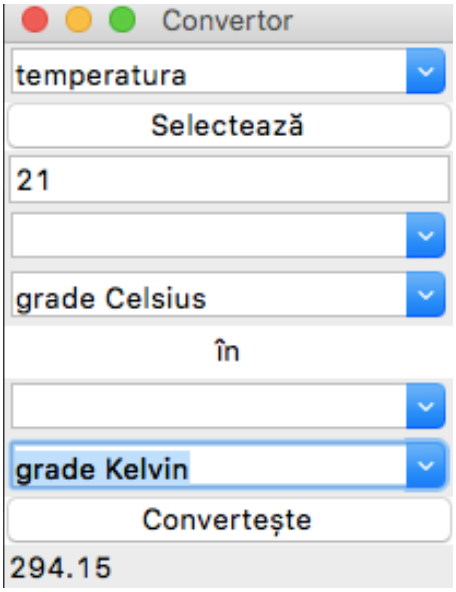
Cum funcționează:

Primul lucru pe care îl facem este să alegem categoria



Alegem una din ele și dăm click pe Selectează:

Punem o valoare, alegem ce unitate convertim și în ce o convertim și ulterior dăm click pe Convertește:



Am folosit:
Python, Tkinter, Pint

Codul sursă:

```
# !/usr/bin/python3
from tkinter import *
from tkinter import ttk

# libraria pint modificată
from pint import UnitRegistry
ureg = UnitRegistry()
Q_ = ureg.Quantity


root = Tk()
root.title("Convertor")
categorie = StringVar()
unitatea_neconvertita = StringVar()
unitatea_convertita = StringVar()
prefixe = ('yocto', 'zepto', 'atto', 'femto', 'pico', 'nano', 'micro', 'mili',
'centi', 'deci', '', 'deca', 'hecto', 'kilo', 'giga', 'tera', 'exa', 'zetta',
'yotta')
prefix_unitate_intrare = StringVar()
prefix_unitate_iesire = StringVar()
unitate_intrare = StringVar()
unitate_iesire = StringVar()
unitati_grupate = {
    'lungime': ('metri', 'mile terestre', 'mile marine', 'țoli', 'lănțișori',
'picioare', 'yarzi'),
    'timp': ('secunde', 'minute', 'ore', 'zile', 'săptămâni', 'luni iuliane', 'ani
iuliani', 'luni gregoriene', 'ani gregorieni'),
```

```
    'temperatura': ('grade Celsius', 'grade Kelvin', 'grade Fahrenheit', 'grade
Rankine'),
    'masa': ('grame'),
    'sarcina electrica': ('columb', 'statcolumb'),
    'curent electric': ('amper', 'statamper'),
    'camp electric': ('volți/metru', 'statvolți/centimetru'),
    'camp magnetic': ('tesla', 'gauss'),
    'inducție electrică': ('columb/metru^2', 'statcolomb/centimetru^2'),
    'flux magnetic':('maxwell','weber')
}

dictionar = {
    'metri':'meter',
    'mile terestre':'mile',
    'mile marine':'nautical_mile',
    'țoli':'inch',
    'lănțișori':'link',
    'picioare':'foot',
    'yarzi':'yard',
    'secunde':'second',
    'minute':'minute',
    'ore':'hour',
    'zile':'day',
    'săptămâni':'week',
    'luni iuliane':'julian_month',
    'ani iuliani':'julian_year',
    'luni gregoriene':'gregorian_month',
    'ani gregorieni':'gregorian_year',
    'grade Celsius':'celsius',
    'grade Kelvin':'kelvin',
    'grade Fahrenheit':'fahrenheit',
    'grade Rankine':'rankine',
    'grame':'gram',
    'moli':'mole',
    'columb':'coulomb',
    'statcolumb':'statC',
    'amper':'ampere',
    'statamper':'statA',
    'volți/metru':'volt/meter',
    'statvolți/centimetru':'statvolt/meter',
    'tesla':'tesla',
    'gauss':'gauss',
    'columb/metru^2':'coulomb / meter**2',
    'statcolumb/centimetru^2':'statC / centimeter**2',
    'maxwell':'maxwell',
    'weber':'weber'
}

combobox_categorie = ttk.Combobox(
    root, textvariable=categorie, state='readonly')
combobox_categorie.grid(row=0)
```

```
combobox_prefix_i = ttk.Combobox(
    root, textvariable=prefix_unitate_intrare, state='readonly')
combobox_prefix_i.grid(row=3)
combobox_i = ttk.Combobox(
    root, textvariable=unitate_intrare, state='readonly')
combobox_i.grid(row=4)

label_to=Label(root, text='în')
label_to.grid(row=5)

combobox_prefix_o = ttk.Combobox(
    root, textvariable=prefix_unitate_iesire, state='readonly')
combobox_prefix_o.grid(row=6)
combobox_o = ttk.Combobox(
    root, textvariable=unitate_iesire, state='readonly')
combobox_o.grid(row=7)

entry_i=ttk.Entry(root)
entry_i.grid(row=2, sticky=W+E+N+S)

label_o=ttk.Label(root, text = "")
label_o.grid(row=9, sticky=W+E+N+S)

def selecteaza_categoria():
    print(categorie.get())
    combobox_i.config(values=unitati_grupate[categorie.get()])
    combobox_o.config(values=unitati_grupate[categorie.get()])
    unitate_intrare.set('')
    unitate_iesire.set('')


b_selectare = Button(root, text="Selectează", command=selecteaza_categoria)
b_selectare.grid(row=1, sticky=W+E+N+S)

def converteste():
    label_o.config(text = Q_(float(entry_i.get()),
prefix_unitate_intrare.get()+dictionar[unitate_intrare.get()]).to(prefix_unitate_ie
sire.get()+dictionar[unitate_iesire.get()]).magnitude)


b_convertire = Button(root, text="Convertește", command= lambda: converteste())
b_convertire.grid(row=8, sticky=W+E+N+S)

combobox_categorie.config(values=tuple(unitati_grupate.keys()))
combobox_prefix_i.config(values=prefixe)
combobox_prefix_o.config(values=prefixe)



root.mainloop()
```

Fișier default_en.txt din librăria pint modificat:

```
# Default Pint units definition file
# Based on the International System of Units
# Language: english
# :copyright: 2013 by Pint Authors, see AUTHORS for more details.

@defaults
   group = international
   system = mks
@end

# decimal prefixes
yocto- = 1e-24 = y-
zepto- = 1e-21 = z-
atto- =  1e-18 = a-
femto- = 1e-15 = f-
pico- =  1e-12 = p-
nano- =  1e-9  = n-
micro- = 1e-6  = u- = µ-
milli- = 1e-3  = m- = mili-
centi- = 1e-2  = c-
deci- =  1e-1  = d-
deca- =  1e+1  = da- = deka
hecto- = 1e2   = h-
kilo- =  1e3   = k-
mega- =  1e6   = M-
giga- =  1e9   = G-
tera- =  1e12  = T-
peta- =  1e15  = P-
exa- =   1e18  = E-
zetta- = 1e21  = Z-
yotta- = 1e24  = Y-

# binary_prefixes
kibi- = 2**10 = Ki-
mebi- = 2**20 = Mi-
gibi- = 2**30 = Gi-
tebi- = 2**40 = Ti-
pebi- = 2**50 = Pi-
exbi- = 2**60 = Ei-
zebi- = 2**70 = Zi-
yobi- = 2**80 = Yi-

# reference
meter = [length] = m = metre =
second = [time] = s = sec
ampere = [current] = A = amp
candela = [luminosity] = cd = candle
gram = [mass] = g
mole = [substance] = mol
kelvin = [temperature]; offset: 0 = K = degK
radian = [] = rad
bit = []
count = []

@import constants_en.txt

# acceleration
[acceleration] = [length] / [time] ** 2
```

```
# Angle
turn = 2 * pi * radian = revolution = cycle = circle
degree = pi / 180 * radian = deg = arcdeg = arcdegree = angular_degree
arcminute = arcdeg / 60 = arcmin = arc_minute = angular_minute
arcsecond = arcmin / 60 =  arcsec = arc_second = angular_second
steradian = radian ** 2 = sr

# Area
[area] = [length] ** 2
are = 100 * m**2
barn = 1e-28 * m ** 2 = b
cmil = 5.067075e-10 * m ** 2 = circular_mils
darcy = 9.869233e-13 * m ** 2
hectare = 100 * are = ha

# EM
esu = C / 2997924580 = statcoulombs = statC = franklin = Fr
esu_per_second = 1 * esu / second = statampere = statA
ampere_turn = 1 * A
gilbert = 10 / (4 * pi ) * ampere_turn
coulomb = ampere * second = C
volt = joule / coulomb = V
farad = coulomb / volt = F
ohm = volt / ampere = Ω
siemens = ampere / volt = S = mho
weber = volt * second = Wb
tesla = weber / meter ** 2 = T
henry = weber / ampere = H
elementary_charge = 1.602176487e-19 * coulomb = e
chemical_faraday = 9.64957e4 * coulomb
physical_faraday = 9.65219e4 * coulomb
faraday =  96485.3399 * coulomb = C12_faraday
gamma = 1e-9 * tesla
gauss = 1e-4 * tesla
maxwell = 1e-8 * weber = mx
oersted = 1000 / (4 * pi) * A / m = Oe
statfarad = 1.112650e-12 * farad = statF = stF
stathenry = 8.987554e11 * henry = statH = stH
statmho = 1.112650e-12 * siemens = statS = stS
statohm = 8.987554e11 * ohm
statvolt = 2.997925e2 * volt = statV = stV
unit_pole = 1.256637e-7 * weber

# Energy
[energy] = [force] * [length]
joule = newton * meter = J
erg = dyne * centimeter
btu = 1.05505585262e3 * joule = Btu = BTU = british_thermal_unit
electron_volt = 1.60217653e-19 * J = eV
quadrillion_btu = 10**15 * btu = quad
thm = 100000 * BTU = therm = EC_therm
calorie = 4.184 * joule = cal = thermochemical_calorie
international_steam_table_calorie = 4.1868 * joule
ton_TNT = 4.184e9 * joule = tTNT
US_therm = 1.054804e8 * joule
watt_hour = watt * hour = Wh = watthour
hartree = 4.35974394e-18 * joule = E_h = hartree_energy
toe = 41.868e9 * joule = tonne_of_oil_equivalent

# Force
[force] = [mass] * [acceleration]
newton = kilogram * meter / second ** 2 = N
```

```
dyne = gram * centimeter / second ** 2 = dyn
force_kilogram = g_0 * kilogram = kgf = kilogram_force = pond
force_gram = g_0 * gram = gf = gram_force
force_ounce = g_0 * ounce = ozf = ounce_force
force_pound = g_0 * lb = lbf = pound_force
force_ton = 2000 * force_pound = ton_force
poundal = lb * feet / second ** 2 = pdl
kip = 1000*lbf

# Frequency
[frequency] = 1 / [time]
hertz = 1 / second = Hz = rps
revolutions_per_minute = revolution / minute = rpm
counts_per_second = count / second = cps

# Heat
#RSI = degK * meter ** 2 / watt
#clo = 0.155 * RSI = clos
#R_value = foot ** 2 * degF * hour / btu

# Information
byte = 8 * bit = B = octet
baud = bit / second = Bd = bps

# Irradiance
peak_sun_hour = 1000 * watt_hour / meter**2 = PSH
langley = thermochemical_calorie / centimeter**2 = Langley

# Length
angstrom = 1e-10 * meter = Å = ångström = Å
parsec = 3.08568025e16 * meter = pc
light_year = speed_of_light * julian_year = ly = lightyear
astronomical_unit = 149597870691 * meter = au

# Mass
carat = 200 * milligram
metric_ton = 1000 * kilogram = t = tonne
atomic_mass_unit = 1.660538782e-27 * kilogram =  u = amu = dalton = Da
bag = 94 * lb

# Textile
denier =  gram / (9000 * meter)
tex = gram / (1000 * meter)
dtex = decitex

# Photometry
lumen = candela * steradian = lm
lux = lumen / meter ** 2 = lx

# Power
[power] = [energy] / [time]
watt = joule / second = W = volt_ampere = VA
horsepower = 33000 * ft * lbf / min = hp = UK_horsepower = British_horsepower
boiler_horsepower = 33475 * btu / hour
metric_horsepower =  75 * force_kilogram * meter / second
electric_horsepower = 746 * watt
hydraulic_horsepower = 550 * feet * lbf / second
refrigeration_ton = 12000 * btu / hour = ton_of_refrigeration

# Pressure
[pressure] = [force] / [area]
Hg = gravity * 13.59510 * gram / centimeter ** 3 = mercury = conventional_mercury
mercury_60F = gravity * 13.5568 * gram / centimeter ** 3
```

H2O = gravity * 1000 * kilogram / meter ** 3 = h2o = water = conventional_water
water_4C = gravity * 999.972 * kilogram / meter ** 3 = water_39F
water_60F = gravity * 999.001 * kilogram / m ** 3
pascal = newton / meter ** 2 = Pa
bar = 100000 * pascal
atmosphere = 101325 * pascal = atm = standard_atmosphere
technical_atmosphere = kilogram * gravity / centimeter ** 2 = at
torr = atm / 760
pound_force_per_square_inch = pound * gravity / inch ** 2 = psi
kip_per_square_inch = kip / inch ** 2 = ksi
barye = 0.1 * newton / meter ** 2 = barie = barad = barrie = baryd = Ba
mm_Hg = millimeter * Hg = mmHg = millimeter_Hg = millimeter_Hg_0C
cm_Hg = centimeter * Hg = cmHg = centimeter_Hg
in_Hg = inch * Hg = inHg = inch_Hg = inch_Hg_32F
inch_Hg_60F = inch * mercury_60F
inch_H2O_39F = inch * water_39F
inch_H2O_60F = inch * water_60F
footH2O = ft * water
cmH2O = centimeter * water
foot_H2O = ft * water = ftH2O
standard_liter_per_minute = 1.68875 * Pa * m ** 3 / s = slpm = slm

# Radiation
Bq = Hz = becquerel
curie = 3.7e10 * Bq = Ci
rutherford = 1e6*Bq = Rd
Gy = joule / kilogram = gray = Sv = sievert
rem = 1e-2 * sievert
rads = 1e-2 * gray
roentgen = 2.58e-4 * coulomb / kilogram

# Temperature
degC = kelvin; offset: 273.15 = celsius
degR = 5 / 9 * kelvin; offset: 0 = rankine
degF = 5 / 9 * kelvin; offset: 255.372222 = fahrenheit

# Time
minute = 60 * second = min
hour = 60 * minute = hr
day = 24 * hour
week = 7 * day
fortnight = 2 * week
year = 31556925.9747 * second
month = year / 12
shake = 1e-8 * second
sidereal_day = day / 1.00273790935079524
sidereal_hour = sidereal_day / 24
sidereal_minute = sidereal_hour / 60
sidereal_second = sidereal_minute / 60
sidereal_year = 366.25636042 * sidereal_day
sidereal_month = 27.321661 * sidereal_day
tropical_month = 27.321661 * day
synodic_month = 29.530589 * day = lunar_month
common_year = 365 * day
leap_year = 366 * day
julian_year = 365.25 * day
julian_month = julian_year/12
gregorian_year = 365.2425 * day
gregorian_month = gregorian_year/12
millenium = 1000 * year = millenia = milenia = milenium
eon = 1e9 * year
work_year = 2056 * hour
work_month = work_year / 12

```
# Velocity
[speed] = [length] / [time]
nautical_mile = 1852 m = nmi # exact
knot = nautical_mile / hour = kt = knot_international = international_knot = nautical_miles_per_hour
mph = mile / hour = MPH
kph = kilometer / hour = KPH

# Viscosity
[viscosity] = [pressure] * [time]
poise = 1e-1 * Pa * second = P
stokes = 1e-4 * meter ** 2 / second = St
rhe = 10 / (Pa * s)

# Volume
[volume] = [length] ** 3
liter = 1e-3 * m ** 3 = l = L = litre
cc = centimeter ** 3 = cubic_centimeter
stere = meter ** 3


@context(n=1) spectroscopy = sp
  # n index of refraction of the medium.
  [length] <-> [frequency]: speed_of_light / n / value
  [frequency] -> [energy]: planck_constant * value
  [energy] -> [frequency]: value / planck_constant
  # allow wavenumber / kayser
  1 / [length] <-> [length]: 1 / value
@end

@context boltzmann
  [temperature] -> [energy]: boltzmann_constant * value
  [energy] -> [temperature]: value / boltzmann_constant
@end

@context(mw=0,volume=0,solvent_mass=0) chemistry = chem
  # mw is the molecular weight of the species
  # volume is the volume of the solution
  # solvent_mass is the mass of solvent in the solution

  # moles -> mass require the molecular weight
  [substance] -> [mass]: value * mw
  [mass] -> [substance]: value / mw

  # moles/volume -> mass/volume and moles/mass -> mass / mass
  # require the  molecular weight
  [substance] / [volume] -> [mass] / [volume]: value * mw
  [mass] / [volume] -> [substance] / [volume]: value / mw
  [substance] / [mass] -> [mass] / [mass]: value * mw
  [mass] / [mass] -> [substance] / [mass]: value / mw

  # moles/volume -> moles requires the solution volume
  [substance] / [volume] -> [substance]: value * volume
  [substance] -> [substance] / [volume]: value / volume

  # moles/mass -> moles requires the solvent (usually water) mass
  [substance] / [mass] -> [substance]: value * solvent_mass
  [substance] -> [substance] / [mass]: value / solvent_mass

  # moles/mass -> moles/volume require the solvent mass and the volume
  [substance] / [mass] -> [substance]/[volume]: value * solvent_mass / volume
  [substance] / [volume] -> [substance] / [mass]: value / solvent_mass * volume
```

```
@end

# Most of the definitions that follows are derived from:
# See http://www.nist.gov/pml/wmd/pubs/hb44.cfm
@group USCSLengthInternational
  inch = yard / 36 = in = international_inch = inches = international_inches
  foot = yard / 3 = ft = international_foot = feet = international_feet
  yard = 0.9144 metres = yd = international_yard
  mile = 1760 yard = mi = international_mile

  square_inch = 1 inch ** 2 = sq_in = square_inches
  square_foot = 1 foot ** 2 = sq_ft = square_feet
  square_yard = 1 yard ** 2 = sq_yd
  square_mile = 1 mile ** 2 = sq_mi

  cubic_inch = 1 in ** 3 = cu_in
  cubic_foot = 1 ft ** 3 = cu_ft = cubic_feet
  cubic_yard = 1 yd ** 3 = cu_yd

  acre_foot = acre * foot = acre_feet
@end

@group USCSLengthSurvey
  link = 0.66 survey_foot = li = survey_link
  survey_foot = foot / 0.999998 = sft
  rod = 16.5 survey_foot = rd = pole = perch
  chain = 66 survey_foot
  survey_mile = 5280 survey_foot

  acre = 43560 survey_foot ** 2
  square_rod = 1 rod ** 2 = sq_rod = sq_pole = sq_perch

  fathom = 6 survey_foot
  us_statute_mile = 5280 survey_foot
  league = 3 us_statute_mile
  furlong = us_statute_mile / 8
@end

@group USCSDryVolume
  dry_pint = 33.6003125 cubic_inch  = dpi = US_dry_pint
  dry_quart = 2 dry_pint = dqt = US_dry_quart
  dry_gallon = 8 dry_pint = dgal = US_dry_gallon
  peck = 16 dry_pint = pk
  bushel = 64 dry_pint = bu
  dry_barrel = 7065 cubic_inch = US_dry_barrel
@end

@group USCSLiquidVolume
  minim = liquid_pint / 7680
  fluid_dram = liquid_pint / 128 = fldr = fluidram = US_fluid_dram
  fluid_ounce = liquid_pint / 16 = floz = US_fluid_ounce = US_liquid_ounce
  gill = liquid_pint / 4 = gi = liquid_gill = US_liquid_gill

  pint = 28.875 cubic_inch = pt = liquid_pint = US_pint

  quart = 2 liquid_pint = qt = liquid_quart = US_liquid_quart
  gallon = 8 liquid_pint = gal = liquid_gallon = US_liquid_gallon
@end

@group USCSVolumeOther
  teaspoon = tablespoon / 3 = tsp
  tablespoon = floz / 2 = tbsp = Tbsp = Tblsp = tblsp = tbs = Tbl
  shot = 3 * tablespoon = jig = US_shot
```

```
  cup = 8 fluid_ounce = cp = liquid_cup = US_liquid_cup
  barrel = 31.5 * gallon = bbl
  oil_barrel = 42 * gallon = oil_bbl
  beer_barrel = 31 * gallon = beer_bbl
  hogshead = 63 * gallon
@end

@group Avoirdupois
  grain = avdp_pound / 7000 = gr
  drachm = pound / 256 = dr = avoirdupois_dram = avdp_dram = dram
  ounce = pound / 16 = oz = avoirdupois_ounce = avdp_ounce
  pound = 453.59237 gram = lb = avoirdupois_pound = avdp_pound

  short_hunderdweight = 100 avoirdupois_pound = ch_cwt
  long_hunderweight = 112 avoirdupois_pound = lg_cwt
  short_ton = 2000 avoirdupois_pound
  long_ton = 2240 avoirdupois_pound
@end

@group Troy
  pennyweight = 24 grain = dwt
  troy_ounce = 480 grain = toz
  troy_pound = 12 troy_ounce = tlb
@end

@group Apothecary
  scruple = 20 grain
  apothecary_dram = 3 scruple = ap_dr
  apothecary_ounce = 8 apothecary_dram = ap_oz
  apothecary_pound = 12 apothecary_ounce = ap_lb
@end

@group AvoirdupoisUK using Avoirdupois
  stone = 14 pound
  quarter = 28 stone
  UK_hundredweight = long_hunderweight = UK_cwt
  UK_ton = long_ton
@end

@group AvoirdupoisUS using Avoirdupois
  US_hundredweight = short_hunderdweight = US_cwt
  US_ton = short_ton = ton
@end

@group Printer
  # Length
  pixel = [printing_unit] = dot = px = pel = picture_element
  pixels_per_centimeter = pixel / cm = PPCM
  pixels_per_inch = pixel / inch = dots_per_inch = PPI = ppi = DPI = printers_dpi
  bits_per_pixel = bit / pixel = bpp

  point = yard / 216 / 12 = pp = printers_point
  thou = yard / 36000 = th = mil
  pica = yard / 216 = P⁄= printers_pica
@end

@group ImperialVolume
  imperial_fluid_ounce = imperial_pint / 20 = imperial_floz = UK_fluid_ounce
  imperial_fluid_drachm = imperial_fluid_ounce / 8 = imperial_fluid_dram
  imperial_gill = imperial_pint / 4 = imperial_gi = UK_gill
  imperial_cup = imperial_pint / 2 = imperial_cp = UK_cup
  imperial_pint = 568.26125 * milliliter = imperial_pt = UK_pint
  imperial_quart = 2 * imperial_pint = imperial_qt = UK_quart
```

```
    imperial_gallon = 8 * imperial_pint = imperial_gal = UK_gallon
    imperial_peck = 16 * imperial_pint = imperial_pk = UK_pk
    imperial_bushel = 64 * imperial_pint = imperial_bu = UK_bushel
    imperial_barrel = 288 * imperial_pint = imperial_bbl = UK_bbl
@end

@system mks using international
    meter
    kilogram
    second
@end

@system cgs using international
    centimeter
    gram
    second
@end

@system imperial using ImperialVolume, USCSLengthInternational, AvoirdupoisUK
    yard
    pound
@end

@system US using USCSLiquidVolume, USCSDryVolume, USCSVolumeOther, USCSLengthInternational, USCSLengthSurvey,
AvoirdupoisUS
    yard
    pound
@end
```