```
# This is timeutil.py.
# This is version 1.5, dated June 11, 2020
# The original version was developed on August 10, 2018.
#
# Copyright 2020 Gary R. Evans
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
# Maintained by Professor Evans
#
# timeutil.py is a master time utility file designed to be used with
# finutil.py, the financial utility file.These utilities draw upon the
# various python time and date libraries to format output specifically
# for the PalmIslandTrader financial models.
#
import datetime
from datetime import date
#
# Debugger to make sure we are using the right version of timutil
#
def which_timeutil():
    return('Version 1.5 of timeutil.py.')
#
# Method right_now() checks for the exact time now using
# datetime.datetime.now() and converts it to an information string
# formatted "Friday, August 10, 2018, 03:02:13 PM local and 12334 ms"
# It returns a tuple [0] is the string and [1] is the standard datetime
# format in case the program has to operate on one of the time components.
# NOTE: In version 1.4 the variable "right_now" was changed to "now".
# The old names are left in comments in case a bug appears.
#
def right_now():
    now = datetime.datetime.now()
    weekday_code = datetime.datetime.isoweekday(now)
    weekday = dayname_full(weekday_code - 1)
    month_now = monthname_full(now.month)
    am_or_pm = "AM"
    hour12check = now.hour

    if hour12check >= 12:
        am_or_pm = "PM"
        if hour12check > 12:
            hour12check -= 12
    if hour12check < 10:
        hour12_str = "0" + str(hour12check)
    else:
        hour12_str = str(hour12check)
    if now.minute < 10:
        rn_minute_str = "0" + str(now.minute)
```

```
    else:
        rn_minute_str = str(now.minute)
    if now.second < 10:
        rn_second_str = "0" + str(now.second)
    else:
        rn_second_str = str(now.second)

    right_now_exact = (weekday+', '+month_now+' '+str(now.day)+', '
        +str(now.year)+', '+hour12_str+':'+rn_minute_str+':'+rn_second_str
        +' '+am_or_pm+' local') # and '+str(right_now.microsecond)+' ms.')

    return [right_now_exact, now]
#
# SECTION FOR NAME LABEL STRINGS
#
def dayname(day: int) -> str:
    daylabel = ['Mon','Tue','Wed','Thu','Fri','Sat','Sun']
    return daylabel[day]
#
def dayname_full(day: int) -> str:
    daylabel_full = ['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday']
    return daylabel_full[day]
#
def monthname(mo: int) -> str:
    mo -= 1
    monthlabel = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug','Sep',
        'Oct', 'Nov', 'Dec']
    return monthlabel[mo]
#
def monthname_caps(mo:int) -> str:
    mo -= 1
    monthlabel = ['JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL', 'AUG','SEP',
        'OCT', 'NOV', 'DEC']
    return monthlabel[mo]
#
def monthname_full(mo:int) -> str:
    mo -= 1
    monthlabel = ['January', 'February', 'March', 'April', 'May', 'June', 'July',
        'August','September', 'October', 'November', 'December']
    return monthlabel[mo]
#
# date_str puts out a standard date string like Jun 8, 2020 and datestring full results
# in a date string June 8, 2020. date_str_noyear is used in the mapping programs.
#
def date_str(eyear: int, emonth: int, eday: int) -> str:
    month_str = monthname(emonth)
    day_str = str(eday)
    year_str = str(eyear)
    return (month_str + ' ' + day_str + ', ' + year_str)
#
def date_str_noyr(emonth: int, eday: int) -> str:
    month_str = monthname(emonth)
    day_str = str(eday)
    return (month_str + ' ' + day_str)
#
def date_str_full(eyear: int, emonth: int, eday: int) -> str:
    month_str = monthname_full(emonth)
    day_str = str(eday)
    year_str = str(eyear)
```

```
    return (month_str + ' ' + day_str + ', ' + year_str)
#
# SECTION FOR TIME AND CALENDAR COUNTS
#
# daysto calculates the number of days between now and some event, such as days2expiry
# as an integer float.
#
def daysto(eyear: int, emonth: int, eday: int) ->  float:
    tnow = datetime.date.today()
    expiry = datetime.date(eyear, emonth, eday)
    days2expiry = abs(expiry - tnow)
    return float(days2expiry.days)
#
# iso_daysto accepts integers for year, day, month and converts to an iso date
# format, passing out the iso-format string, labeled here as expiry because that
# is what it usually is in our models. First used in ib_inpos_strangle.
#
def iso_daysto(eyear: int, emonth: int, eday: int) -> str:
    edate = datetime.date(eyear,emonth,eday)
    if emonth < 10:
        e_mo_str = "0" + str(edate.month)
    else:
        e_mo_str = str(edate.month)
    if eday < 10:
        e_day_str = "0" + str(edate.day)
    else:
        e_day_str = str(edate.day)
    expiry = str(edate.year) + str(e_mo_str) + str(e_day_str)
    return str(expiry)
#
# iso_daysto_days is identical to iso_daysto except that it also passes out the number
# of days to expiry. iso_daysto was not modified to this for the fear of breaking an
# older program.
#
def iso_daysto_days(eyear: int, emonth: int, eday: int) -> list:
    edate = datetime.date(eyear,emonth,eday)
    if emonth < 10:
        e_mo_str = "0" + str(edate.month)
    else:
        e_mo_str = str(edate.month)
    if eday < 10:
        e_day_str = "0" + str(edate.day)
    else:
        e_day_str = str(edate.day)
    expiry = str(edate.year) + str(e_mo_str) + str(e_day_str)
    days = daysto(eyear,emonth,eday)
    return [expiry,days]
#

if __name__ == "__main__":
    x = right_now()
    print ("Test of right_now: {}, {} ".format(x[0],x[1]))
    # The date provided below must be in the future when this is tested.
    exyear = int(2020)
    exmonth = int(9)
    exday = int(19)
    print ("Test of date_str: ", date_str(exyear,exmonth,exday))
    print ("Test of date_str_noyr: ", date_str_noyr(exmonth,exday))
    print ("Test of date_str_full: ", date_str_full(exyear,exmonth,exday))
```

```
    days = daysto(exyear,exmonth,exday)
    print ("Test of daysto: {}".format(days))
    expiry = iso_daysto(exyear,exmonth,exday)
    print ("Test of iso_daysto: {}".format(expiry))
    expiry_days = iso_daysto_days(exyear,exmonth,exday)
    print ("Test of iso_daysto_days: {}".format(expiry_days))
    this_month = monthname(exmonth)
    print ("Test of monthname: {}".format(this_month))
    this_month = monthname_caps(exmonth)
    print ("Test of monthname_caps: {}".format(this_month))
    this_month = monthname_full(exmonth)
    print ("Test of monthname_full: {}".format(this_month))
```