calender

February 16, 2018

In [ ]:
```
import sys
from datetime import *
from time import *

print ("日付を入力してください '2016/07/30'.")
user_input_date = input("your date :")

yobi = ["月","火","水","木","金","土","日"]

while user_input_date != "bye":
    try:
        a = datetime.strptime(user_input_date,'%Y/%m/%d')
        print ("{}は{}曜日です".format(user_input_date,yobi[a.weekday()]))
    except ValueError:
        print ("誤った日付です")
    user_input_date = input("your date :")
else:
    sys.exit(1)
```

In [19]:
```
import datetime
str_date = input('date: ')
date_obj = datetime.datetime.strptime(str_date, '%Y/%m/%d')
week_name = ['月', '火', '水', '木', '金', '土', '日']
'{}は{}曜日です'.format(str_date, week_name[date_obj.weekday()])
```

```
date: 2018/9/1
```

Out[19]:
```
'2018/9/1 は土曜日です'
```

In [133]:
```
import jpholiday
import datetime
#jpholiday.is_holiday(datetime.date(2030, 1, 1))
def calc_working_day(date, convention):
    week_name = ['月', '火', '水', '木', '金', '土', '日']
    date_obj_original = datetime.datetime.strptime(date,  '%Y/%m/%d')
    date_original = date_obj_original.strftime('%Y/%m/%d')
    if (convention=='following'):
```

```
            while (check_holiday(date) or check_Sat(date) or check_Sun(date)):
                date_obj = datetime.datetime.strptime(date,  '%Y/%m/%d')
                date_obj = date_obj + datetime.timedelta(days=1)
                date = date_obj.strftime('%Y/%m/%d')
            return date
        elif (convention=='modified following'):
            date_obj = datetime.datetime.strptime(date,  '%Y/%m/%d')
            while (check_holiday(date) or check_Sat(date) or check_Sun(date)):
                date_obj = date_obj + datetime.timedelta(days=1)
                date = date_obj.strftime('%Y/%m/%d')
            if (date_obj.month == date_obj_original.month):
                return date
            else:
                date_for_mf = date_original
                while (check_holiday(date_for_mf ) or check_Sat(date_for_mf ) or check_Sun
                    date_obj = datetime.datetime.strptime(date_original,  '%Y/%m/%d')
                    date_obj = date_obj - datetime.timedelta(days=1)
                    date_for_mf  = date_obj.strftime('%Y/%m/%d')
                return date_for_mf

    def check_holiday(str_date):
        date_obj = datetime.datetime.strptime(str_date,  '%Y/%m/%d')
        year = date_obj.year
        month = date_obj.month
        day = date_obj.day
        date = datetime.date(year, month, day)
        is_holiday = jpholiday.is_holiday(date)
        return is_holiday

    def calc_date_after_holiday(date):
        while (check_holiday(date)==True):
            date_obj = datetime.datetime.strptime(date,  '%Y/%m/%d')
            date_obj = date_obj + datetime.timedelta(days=1)
            date = date_obj.strftime('%Y/%m/%d')
        return date

    def check_Sat(date):
        date_obj = datetime.datetime.strptime(date, '%Y/%m/%d')
        if (date_obj.weekday() == 5):
            return True
        else:
            return False

    def check_Sun(date):
        date_obj = datetime.datetime.strptime(date, '%Y/%m/%d')
        if (date_obj.weekday() == 6):
            return True
        else:
```

```
            return False


        def cash_flow_generator(start_day, tenor, convention):
            start_obj = datetime.datetime.strptime(start_day, '%Y/%m/%d')
            day = start_day[-2:]
            month = start_obj.month + int(tenor[0])
            year = start_obj.year
            if (month <= 12):
                pass
            elif (month > 12):
                month -= 12
                year += 1
            end_date = str(year) + '/' + str(month) + '/' + str(day)
            working_end_date = calc_working_day(end_date, convention)
            return working_end_date
```

In [36]:
```
import datetime

date = '2018/5/3'
def
while (check_holiday(date)==True):
    date_obj = datetime.datetime.strptime(date,  '%Y/%m/%d')
    date_obj = date_obj + datetime.timedelta(days=1)
    date = date_obj.strftime('%Y/%m/%d')

date
```

Out[36]: '2018/05/06'

In [73]:
```
date = '2018/5/3'
while (check_holiday(date) or check_Sat(date) or check_Sun(date)):
    date_obj = datetime.datetime.strptime(date,  '%Y/%m/%d')
    date_obj = date_obj + datetime.timedelta(days=1)
    date = date_obj.strftime('%Y/%m/%d')

date
```

Out[73]: '2018/05/07'

In [137]:
```
#print(calc_working_day('2018/5/2', 'following'))
print(calc_working_day('2018/7/14', 'modified following'))
```

```
2018/07/17
```

In [7]:
```
cash_flow_generator('2017/12/30', '6M', 'following')
```

Out[7]: '2018/07/02'

```
In [125]: import dateutil
```

```
In [133]: from dateutil.relativedelta import relativedelta
          # 今日の日付を取得
          today = datetime.datetime.today()
          # 翌月を取得
          next_month = today + relativedelta(months=1)
          print(today.strftime('%Y-%m-%d'))
          print(next_month.strftime('%Y-%m-%d'))
```

```
2018-01-31
2018-02-28
```

```
In [31]: import japanese_holiday
         import datetime

         now = datetime.date.today()
         date_from = datetime.date(now.year+2, 1, 1)
         date_to = datetime.date(now.year+2, 12, 31)
         print(now.year-2)
         holidays = japanese_holiday.getholidays(
         "API key",
         japanese_holiday.HOLIDAY_TYPE_OFFICIAL_JA,
         date_from.strftime('%Y-%m-%d'),
         date_to.strftime('%Y-%m-%d')
         )

         for holiday in holidays:
             for key in holiday:
                 print(key ,holiday[key])
```

```
2016
('japanese__ja@holiday.calendar.google.com', 'AIzaSyDdPbr9mW4iUA4aeoWCFLcM5tD_U6vco3w', '2020-01
```

```
In [2]: import japanese_holiday
        import datetime

        now = datetime.date.today()
        date_from = datetime.date(now.year-2, 1, 1)
        date_to = datetime.date(now.year+1, 12, 31)

        holidays = japanese_holiday.getholidays(
        "API key",
        japanese_holiday.HOLIDAY_TYPE_OFFICIAL_JA,
        date_from.strftime('%Y-%m-%d'),
        date_to.strftime('%Y-%m-%d')
        )
```

```
        for holiday in holidays:
            print(holiday['start']['date'],holiday['summary'])
```

```
('japanese__ja@holiday.calendar.google.com', 'AIzaSyDdPbr9mW4iUA4aeoWCFLcM5tD_U6vco3w', '2016-01
2017-01-01 元日
2017-01-02 元日 振替休日
2017-01-09 成人の日
2017-02-11 建国記念の日
2017-03-20 春分の日
2017-04-29 昭和の日
2017-05-03 憲法記念日
2017-05-04 みどりの日
2017-05-05 こどもの日
2017-07-17 海の日
2017-08-11 山の日
2017-09-18 敬老の日
2017-09-23 秋分の日
2017-10-09 体育の日
2017-11-03 文化の日
2017-11-23 勤労感謝の日
2017-12-23 天皇誕生日
2018-01-01 元日
2018-01-08 成人の日
2018-02-11 建国記念の日
2018-02-12 建国記念の日 振替休日
2018-03-21 春分の日
2018-04-29 昭和の日
2018-04-30 昭和の日 振替休日
2018-05-03 憲法記念日
2018-05-04 みどりの日
2018-05-05 こどもの日
2018-07-16 海の日
2018-08-11 山の日
2018-09-17 敬老の日
2018-09-23 秋分の日
2018-09-24 秋分の日 振替休日
2018-10-08 体育の日
2018-11-03 文化の日
2018-11-23 勤労感謝の日
2018-12-23 天皇誕生日
2018-12-24 天皇誕生日 振替休日
2019-01-01 元日
2019-01-14 成人の日
2019-02-11 建国記念の日
2019-03-21 春分の日
2019-04-29 昭和の日
2019-05-03 憲法記念日
```

```
2019-05-04 みどりの日
2019-05-05 こどもの日
2019-05-06 こどもの日 振替休日
2019-07-15 海の日
2019-08-11 山の日
2019-08-12 山の日 振替休日
2019-09-16 敬老の日
2019-09-23 秋分の日
2019-10-14 体育の日
2019-11-03 文化の日
2019-11-04 文化の日 振替休日
2019-11-23 勤労感謝の日
2019-12-23 天皇誕生日
```

```
In [61]: import jpholiday
         import datetime
         jpholiday.is_holiday(datetime.date(2018, 10, 15))

Out[61]: True

In [13]: tnow = datetime.datetime.now()
         type(tnow.year)

Out[13]: int

In [60]: jpholiday.year_holidays(2018)

Out[60]: [(datetime.date(2018, 1, 1), '元日'),
          (datetime.date(2018, 1, 15), '成人の日'),
          (datetime.date(2018, 2, 11), '建国記念の日'),
          (datetime.date(2018, 2, 12), '建国記念の日 振替休日'),
          (datetime.date(2018, 3, 21), '春分の日'),
          (datetime.date(2018, 4, 29), '昭和の日'),
          (datetime.date(2018, 4, 30), '昭和の日 振替休日'),
          (datetime.date(2018, 5, 3), '憲法記念日'),
          (datetime.date(2018, 5, 4), 'みどりの日'),
          (datetime.date(2018, 5, 5), 'こどもの日'),
          (datetime.date(2018, 7, 16), '海の日'),
          (datetime.date(2018, 8, 11), '山の日'),
          (datetime.date(2018, 9, 17), '敬老の日'),
          (datetime.date(2018, 9, 23), '秋分の日'),
          (datetime.date(2018, 9, 24), '秋分の日 振替休日'),
          (datetime.date(2018, 10, 15), '体育の日'),
          (datetime.date(2018, 11, 3), '文化の日'),
          (datetime.date(2018, 11, 23), '勤労感謝の日'),
          (datetime.date(2018, 12, 23), '天皇誕生日'),
          (datetime.date(2018, 12, 24), '天皇誕生日 振替休日')]
```

1 BACKUP

営業日を求める関数の昔のバージョン

```
In [74]: '''
         def calc_working_day(str_date, convention):
             week_name = ['月', '火', '水', '木', '金', '土', '日']
             date_after_holiday = calc_date_after_holiday(str_date)
             date_obj = datetime.datetime.strptime(date_after_holiday, '%Y/%m/%d')
             month = date_obj.month
             one_day_after_obj = date_obj + datetime.timedelta(days = 1)
             two_day_after_obj = date_obj + datetime.timedelta(days = 2)
             one_day_after_month = one_day_after_obj.month
             two_day_after_month = two_day_after_obj.month
             # folloing convention
             # TO DO月曜日が祝日の場合のチェックができていない
             if (convention == 'following'):
                 if (date_obj.weekday() == 5):
                     date_obj_new = date_obj + datetime.timedelta(days = 2)
                     str_new_date = date_obj_new.strftime('%Y/%m/%d')
                     if (check_holiday(str_new_date)==True):
                         str_new_date = calc_date_after_holiday(str_new_date)
                         return str_new_date
                     else:
                         return date_obj_new.strftime('%Y/%m/%d')
                 elif (date_obj.weekday() == 6):
                     date_obj_new = date_obj + datetime.timedelta(days = 1)
                     return date_obj_new.strftime('%Y/%m/%d')
                 else:
                     return date_obj.strftime('%Y/%m/%d')
             # modified following
             elif (convention == 'modified following'):
                 if (month == two_day_after_month):
                     if (date_obj.weekday() == 5):
                         date_obj_new = date_obj + datetime.timedelta(days = 2)
                         return date_obj_new.strftime('%Y/%m/%d')
                     elif (date_obj.weekday() == 6):
                         date_obj_new = date_obj + datetime.timedelta(days = 1)
                         return date_obj_new.strftime('%Y/%m/%d')
                     else:
                         return date_obj.strftime('%Y/%m/%d')
                 if (month != two_day_after_month or month != one_day_after_month ):
                     if (date_obj.weekday() == 5):
                         date_obj_new = date_obj + datetime.timedelta(days = -1)
                         return date_obj_new.strftime('%Y/%m/%d')
                     elif (date_obj.weekday() == 6):
                         date_obj_new = date_obj + datetime.timedelta(days = -2)
                         return date_obj_new.strftime('%Y/%m/%d')
```

```
                else:
                    return date_obj.strftime('%Y/%m/%d')
        '''
```

Out[74]: "\ndef calc_working_day(str_date, convention):\n   week_name = ['月', '火', '水', '木