Guzman Assignment 1

Gu Wu UNI: gw2406

February 10, 2021

1. Power Calendar Function

   For this particular project, I unitized Objected-Oriented programming to facilitate the entire development process because there are a lot of the information we need to use repetitively and it is also easier to debug if we cannot program a perfect script at once; Once we find something wrong, we can divide our code piece by piece and to evaluate them separately.

   Before I start to program, it is critical to acknowledge that the flat hours are just all hours during a time period, onpeak hours are weekdays (except holiday) multiply by 16, and offpeak hours are flat hours minus onpeak hours, 2x16H is number of offpeak days multiply by 16 and 7X8 is the time period multiply by 8. Of course the daylight saving should be considered here. The code below essentially illustrate what I have just said

```
    def hours(self):

        if self.peak_type == 'flat':
            return self.flat_hours
        elif self.peak_type == 'onpeak':
            return 16 * (self.duration - self.number_of_offpeak_days)
        elif self.peak_type == 'offpeak':
            return self.flat_hours - 16 * (self.duration - self.number_of_offpeak_days)
        elif self.peak_type == '7x8':
            return self.duration * 8 + self.daylight_saving
        else:
            return self.number_of_offpeak_days * 16
```

   Another thing we also need to consider is the number of holiday and when they occur. For example, new year time is different year by year, if January 1 is Sunday, then new year will be defined on January 2. Christmas day is also different year by year. If December 25 is Sunday, then Christmas day will be defined on December 26 and if Decenber 25 is Saturday, then Christmas day will be defined on 24. Similarly, Independent day cannot be on Sunday as well, therefore, we also need to shift by one. The code below captures the cases I am talking about.

```
    def holidays(self, period):
        days = []
        #New year
        if datetime.date(int(period[:4]),1,1).weekday()== 6:
            days.append(datetime.date(int(period[:4]),1,2))
        else:
            days.append(datetime.date(int(period[:4]),1,1))
        # Memorial
        days.append(calendar.Calendar(0).monthdatescalendar(int(period[:4]), 5)[-1][0])
        # Independent
        if datetime.date(int(period[:4]),7,4).weekday() == 6:
            days.append(datetime.date(int(period[:4]),7,5))
        elif datetime.date(int(period[:4]),7,4).weekday() == 5:
            days.append(datetime.date(int(period[:4]),7,3))
        else:
            days.append(datetime.date(int(period[:4]),7,4))
        # Labor
        if calendar.Calendar(0).monthdatescalendar(int(period[:4]), 9)[0][0].month ==
    8:
            days.append(calendar.Calendar(0).monthdatescalendar(int(period[:4]), 9)
    [1][0])
```

```
        else:
            days.append(calendar.Calendar(0).monthdatescalendar(int(period[:4]), 9)
    [0][0])
        # Thanksgiving
        if calendar.Calendar(0).monthdatescalendar(int(period[:4]), 11)[4][3].month ==
    12:
            days.append(calendar.Calendar(0).monthdatescalendar(int(period[:4]), 11)
    [3][3])
        else:
            days.append(calendar.Calendar(0).monthdatescalendar(int(period[:4]), 11)
    [4][3])
        #Chrismas
        if datetime.date(int(period[:4]),12,25).weekday() == 5:
            days.append(datetime.date(int(period[:4]),12,24))
        elif datetime.date(int(period[:4]),12,25).weekday() == 6:
            days.append(datetime.date(int(period[:4]),12,26))
        else:
            days.append(datetime.date(int(period[:4]),12,25))
        return days
```

Another important piece we need to be aware of is how to consider daylight saving smartly. I basically created an attribute call daylight_saving which is a value takes -1,0 and 1. When the iso is MISO or we are considering the entire year, or we are in the middle two quarters, then the value takes 0. If we are considering the month of March/Nov, or we are talking about the first/last quarter, we need to set this variable to be -1 or 1 accordingly. Once we have the value set up, we can add or subtract this value during our calculation to make our life easier.

```
    def get_daylight_saving(self,period):

        Mar, Nov = self.daylight_saving_dates(period)

        if period[-1] == 'A' or self.iso == 'MISO' or period[-2:] == 'Q2' or period
    [-2:] == 'Q3':
            return 0
        elif period[-3:] == 'Mar' or  period[-2:] == 'Q1' or self.start_date == Mar:
            return -1
        elif period[-3:] == 'Nov' or  period[-2:] == 'Q4' or self.start_date == Nov:
            return 1
        else:
            return 0
```

Once we pay attention to these features and put everything together, we are good to go.