EE551 Project
Power Demand Data Collection

Justin Thompson

I pledge my honor I have abided by the Stevens Honor System

https://github.com/justinbthompson/myEE551projectPowerDemand

Inspiration

- Electrical grid is aging, Smart Grid is way of the future
  - Help reduce wasteful energy usage and help fight climate change
- A Smart Grid is possible with data communication and analysis tools of today.
- Analysis could allow better prediction of power demand trends, permitting decisions on peak demand shaving, energy generation, and energy usage to be made.
- New York Independent System Operator posts analysis of real time consumption, but archives all old data in 5 minute intervals.
- Goal of this project is to analyze that old data.

Tasks

- The program has several main objectives:
  - Receive user input for day to be examined
  - Use that input to find data files on the internet
    - Last ~10 days are posted a CSV files
    - Any older information is posted in monthly zip files
  - Open the data and manipulate it to collect meaningful information
  - Print the results

Functions

def user_input()

- Simple function to collect user input
- “= input()” used to allow the user to type a number
- Several “if” and “try” statements to check if the input is valid.
- Used the lines below to check if the date request exists and is not in the future
  - Requires “datetime” library
- Outputs the data

```
try:
    datetime.datetime(int(inputyear), int(inputmonth), int(inputday))
    if datetime.datetime.now() < datetime.datetime(int(inputyear), int(inputmonth), int(inputday)):
        x = True
```

Functions

def access_data(date)

- Inputs the date from the user input
- Uses that data to create a URL and search for the data
- If the URL exists, “data = pd.read_csv(url)” uses the “pandas” library to open the CSV file
- If the URL does not exist, that day’s CSV file is not posted directly on website but in a ZIP file. The library “zipfile” is needed to read the data.
- The data extracted from the site with pandas is then returned from the function

Real-Time Fuel Mix

CSV Files	Last Updated
05-01-2019	05/01/19 16:20 EDT
04-30-2019	05/01/19 00:05 EDT
04-29-2019	04/30/19 00:05 EDT
04-28-2019	04/29/19 00:05 EDT
04-27-2019	04/28/19 00:05 EDT
04-26-2019	04/27/19 00:05 EDT
04-25-2019	04/26/19 00:05 EDT
04-24-2019	04/25/19 00:05 EDT
04-23-2019	04/24/19 00:05 EDT
04-22-2019	04/23/19 00:05 EDT

Archived Files (zip format)

CSV Files	Last Updated
05-2019	05/01/19 16:20 EDT
04-2019	05/01/19 00:05 EDT
03-2019	04/01/19 00:05 EDT
02-2019	03/01/19 00:05 EST
01-2019	02/01/19 00:05 EST
12-2018	01/01/19 00:05 EST
11-2018	12/01/18 00:05 EST
10-2018	11/01/18 00:05 EDT
09-2018	10/01/18 00:05 EDT

Functions

def analyze_data(data):

- Requires the input of the data
- Runs a loop through all rows of the CSV file, usually over 2000 rows
- Tallies a sum of total data for total energy usage of the day
- Uses a dictionary with keys matching the generation sources to tally total energy usage per each generation source
- Also keeps track of the minute and the hour with the highest energy usage
- Outputs all of these findings

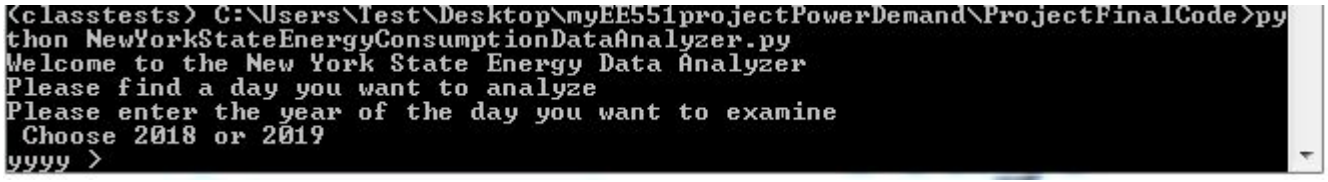
```
while i < data.shape[0]: #data.shape[0] = number of rows

    #Total Daily Energy Consumption
    dt['All Sources'] += data['Gen MW'][i]

    #Daily Energy Consumption Per Generation Source
    for x in dt:
        if data['Fuel Category'][i] == x:
            dt[x] += data['Gen MW'][i]
```

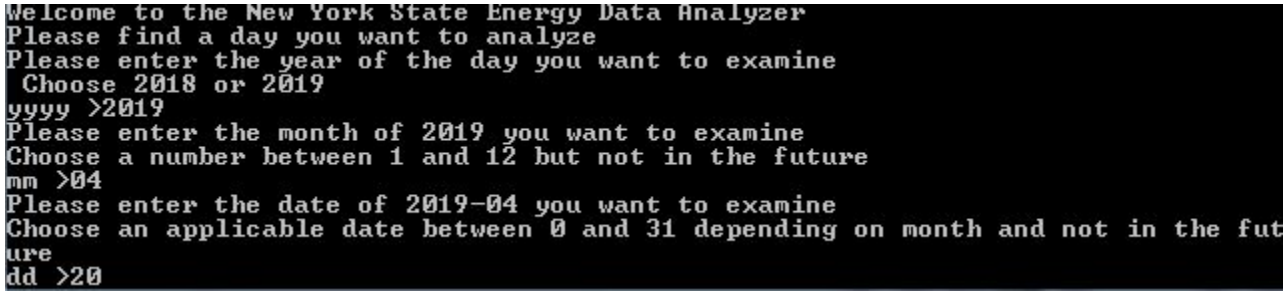
```
dt = {'All Sources':0,'Dual Fuel':0, 'Natural Gas':0, 'Nuclear': 0,
      'Other Fossil Fuels':0, 'Other Renewables':0, 'Wind':0, 'Hydro':0}
```

Main Program

- Imports the functions from their .py files
- Was separated into its own file so that infinite while loop would not run during test code
- Includes a while loop that runs until the user decides to exit, requesting input dates and printing the information

Usage - User Input





Usage - Result