Report 08

1. **Reflections**

- What was the easiest and hardest part of this assignment?

  Easiest part:

  The average function is an easy part.

  As long as you lock down the location of useful data in the csv file, it is not hard to find the way to make it a list.

  Hardest part:

  There are too many places that could possibly cause an error, it is hard to find them all, such as what if the user inputs a location or starting year which do not exist. You need to find them all and raise an error and give a message accordingly, protect the main function by try…except syntax.

  A lot of details such as the transform of year and index of the list, skip the first line of the csv, 3 digits remain after dot, treatment of raw data to delete irrelevant part and get the useful information.

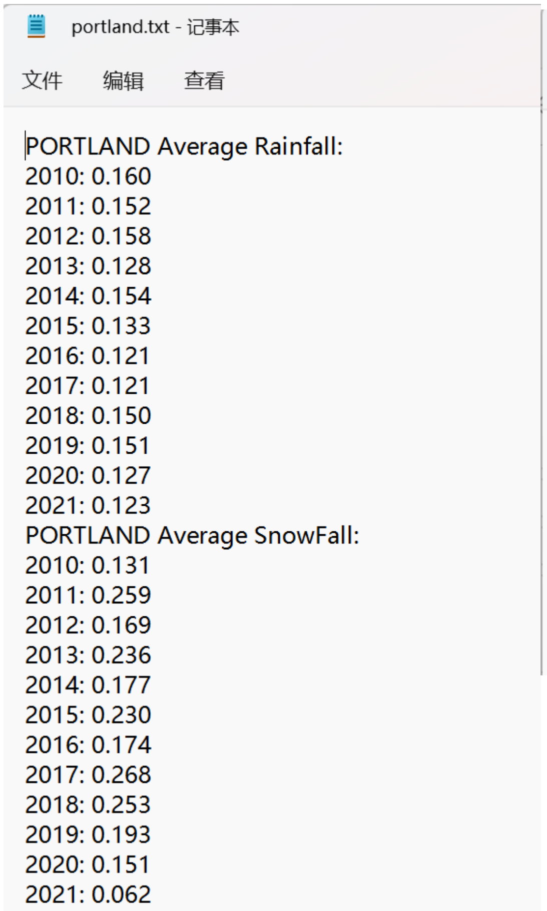
- What did you learn?

  Raise an error in different condition and provide useful information based on the condition, handle errors by try.. except syntax.

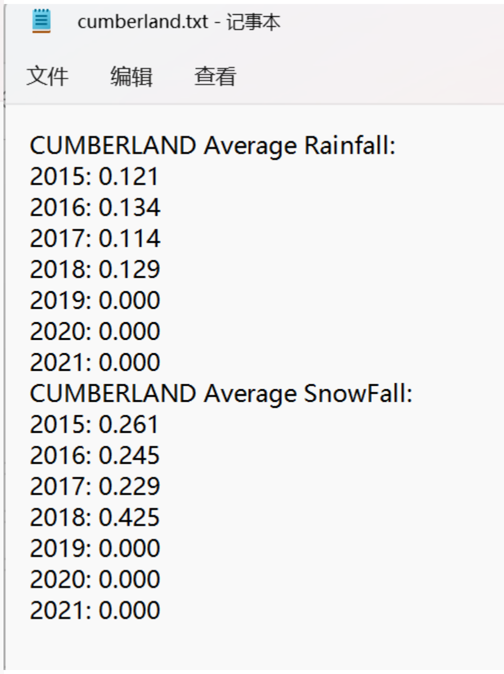
  A lot of practice on string and list methods to get useful information from raw data.

  File I/O, read data in a csv file and write to an out put file.

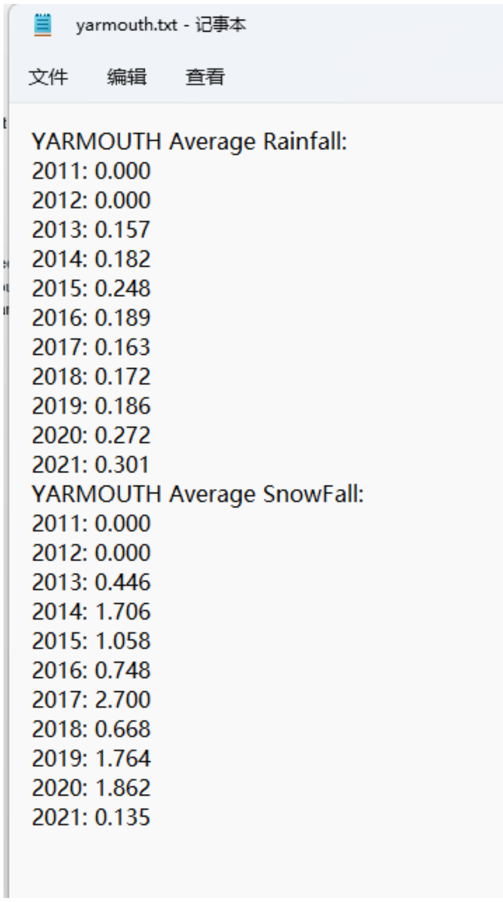
2. **Output**



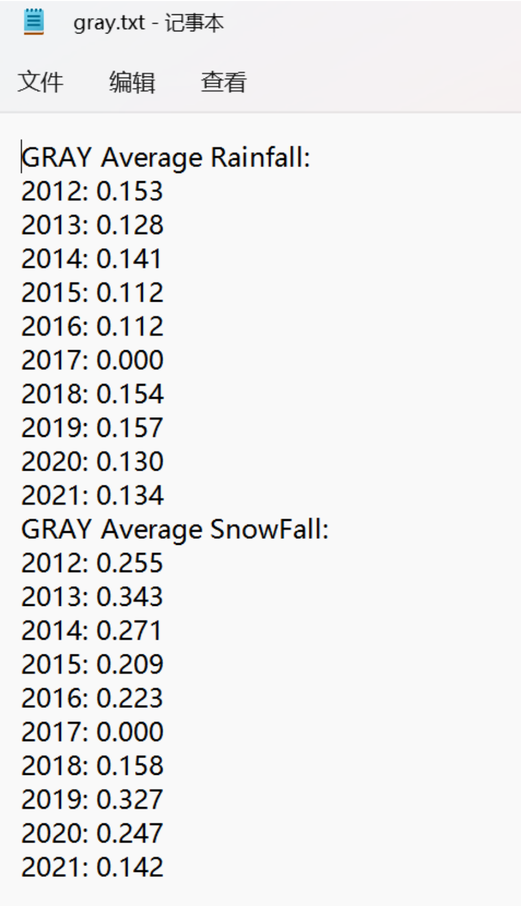
```
PORTLAND Average Rainfall:
2010: 0.160
2011: 0.152
2012: 0.158
2013: 0.128
2014: 0.154
2015: 0.133
2016: 0.121
2017: 0.121
2018: 0.150
2019: 0.151
2020: 0.127
2021: 0.123
PORTLAND Average SnowFall:
2010: 0.131
2011: 0.259
2012: 0.169
2013: 0.236
2014: 0.177
2015: 0.230
2016: 0.174
2017: 0.268
2018: 0.253
2019: 0.193
2020: 0.151
2021: 0.062
```



```
CUMBERLAND Average Rainfall:
2015: 0.121
2016: 0.134
2017: 0.114
2018: 0.129
2019: 0.000
2020: 0.000
2021: 0.000
CUMBERLAND Average SnowFall:
2015: 0.261
2016: 0.245
2017: 0.229
2018: 0.425
2019: 0.000
2020: 0.000
2021: 0.000
```



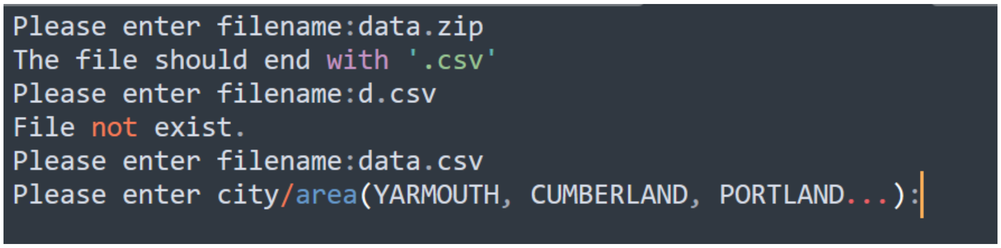
```
YARMOUTH Average Rainfall:
2011: 0.000
2012: 0.000
2013: 0.157
2014: 0.182
2015: 0.248
2016: 0.189
2017: 0.163
2018: 0.172
2019: 0.186
2020: 0.272
2021: 0.301
YARMOUTH Average SnowFall:
2011: 0.000
2012: 0.000
2013: 0.446
2014: 1.706
2015: 1.058
2016: 0.748
2017: 2.700
2018: 0.668
2019: 1.764
2020: 1.862
2021: 0.135
```



```
GRAY Average Rainfall:
2012: 0.153
2013: 0.128
2014: 0.141
2015: 0.112
2016: 0.112
2017: 0.000
2018: 0.154
2019: 0.157
2020: 0.130
2021: 0.134
GRAY Average SnowFall:
2012: 0.255
2013: 0.343
2014: 0.271
2015: 0.209
2016: 0.223
2017: 0.000
2018: 0.158
2019: 0.327
2020: 0.247
2021: 0.142
```

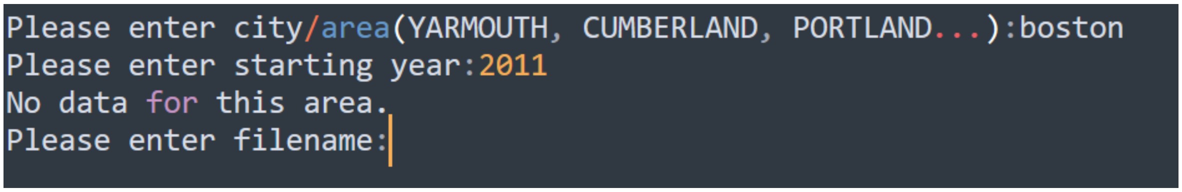
Output(Portland 2010~2021, Cumberland 2015~2021, yarmouth 2011~2021,gray2012~2021)

Filename error handling



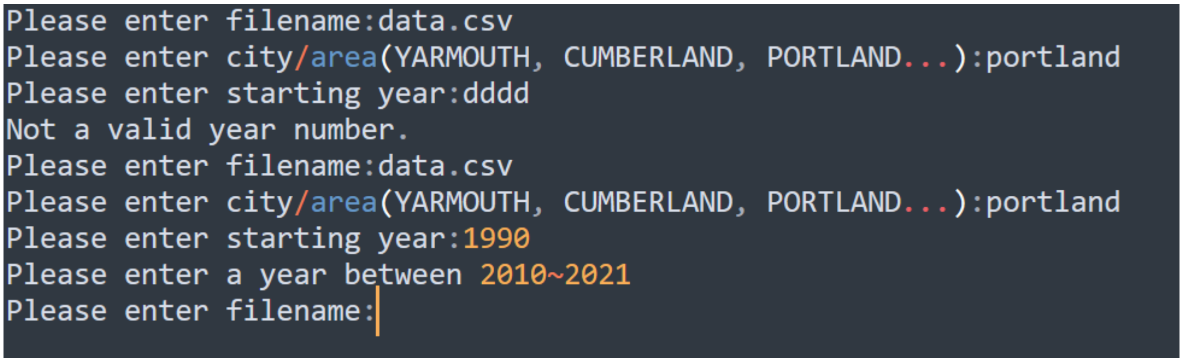
```
Please enter filename:data.zip
The file should end with '.csv'
Please enter filename:d.csv
File not exist.
Please enter filename:data.csv
Please enter city/area(YARMOUTH, CUMBERLAND, PORTLAND...):
```

City name error handling



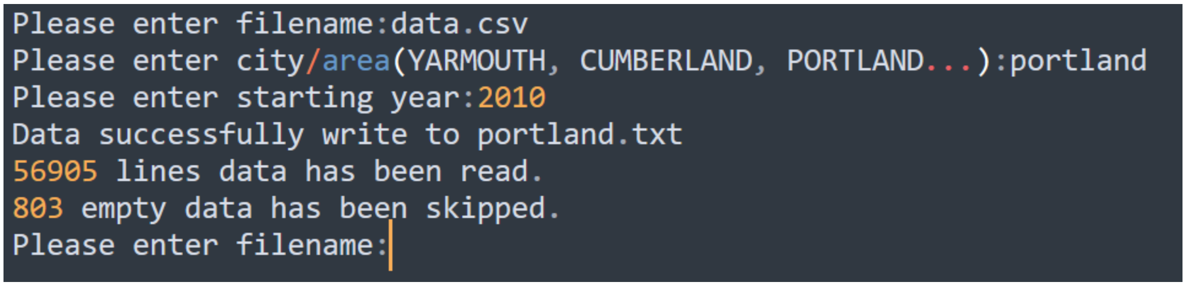
```
Please enter city/area(YARMOUTH, CUMBERLAND, PORTLAND...):boston
Please enter starting year:2011
No data for this area.
Please enter filename:
```

Starting year error handling



```
Please enter filename:data.csv
Please enter city/area(YARMOUTH, CUMBERLAND, PORTLAND...):portland
Please enter starting year:dddd
Not a valid year number.
Please enter filename:data.csv
Please enter city/area(YARMOUTH, CUMBERLAND, PORTLAND...):portland
Please enter starting year:1990
Please enter a year between 2010~2021
Please enter filename:
```

Complete operation of data extract



```
Please enter filename:data.csv
Please enter city/area(YARMOUTH, CUMBERLAND, PORTLAND...):portland
Please enter starting year:2010
Data successfully write to portland.txt
56905 lines data has been read.
803 empty data has been skipped.
Please enter filename:
```

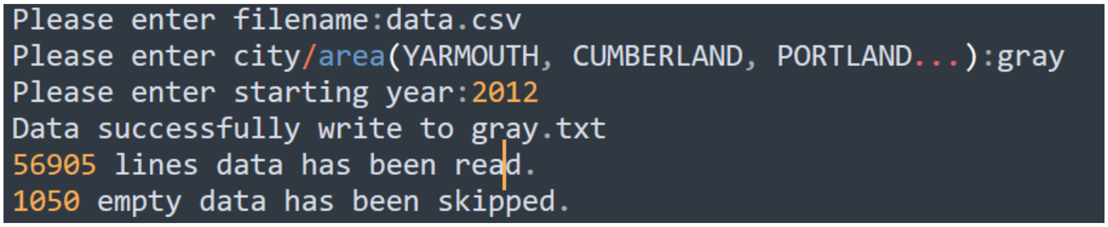
3. Extensions

- **Multiple cities instead of just Portland**

  As can be seen in part 2, multiple cities in the csv file can be selected to get data.

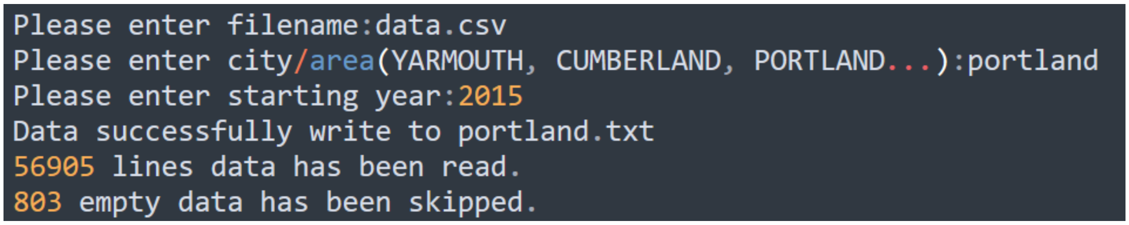
- **Let user select city and starting year, gave a notification if no data for input city, negelect upper or lowercase input.**

  Filter data of certain city and starting year based on the user input, if no data found, the program will notify user "No data for this area"



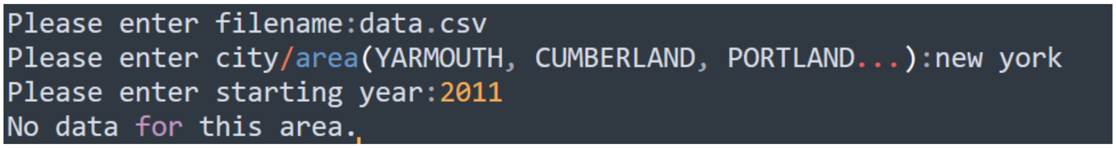
```
Please enter filename:data.csv
Please enter city/area(YARMOUTH, CUMBERLAND, PORTLAND...):gray
Please enter starting year:2012
Data successfully write to gray.txt
56905 lines data has been read.
1050 empty data has been skipped.
```

Get data of gray start from 2021



```
Please enter filename:data.csv
Please enter city/area(YARMOUTH, CUMBERLAND, PORTLAND...):portland
Please enter starting year:2015
Data successfully write to portland.txt
56905 lines data has been read.
803 empty data has been skipped.
```

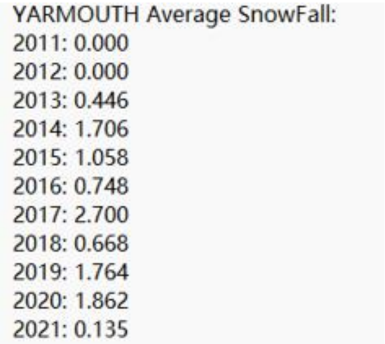
Get data of portland start from 2015



```
Please enter filename:data.csv
Please enter city/area(YARMOUTH, CUMBERLAND, PORTLAND...):new york
Please enter starting year:2011
No data for this area.
```

Data not found in new york

- **Snow data is Extracted.**



```
YARMOUTH Average SnowFall:
2011: 0.000
2012: 0.000
2013: 0.446
2014: 1.706
2015: 1.058
2016: 0.748
2017: 2.700
2018: 0.668
2019: 1.764
2020: 1.862
2021: 0.135
```

- **Add some other exceptions.**

Starting_year value checking, raise a value error if value is not between 2010 and 2021

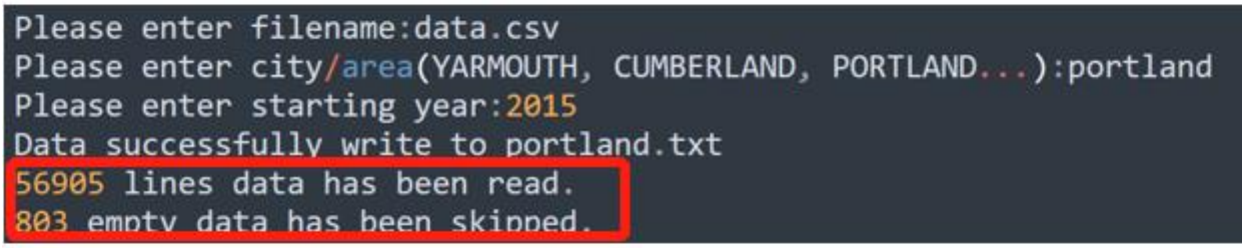
```
if starting_year > 2021 or starting_year < 2010:
    raise ValueError("Please enter a year between 2010~2021")
```

Location checking, if user input location is not found, raise a value error.

```
if location_found:
    print_avgs(rain_data, snow_data, city, starting_year, f"{city}.txt")
    print (f"{line_num} lines data has been read.")
    print (f"{empty_data} empty data has been skipped.")
else:
    raise ValueError("No data for this area.")
```

- **Line read and empty data counting.**

After extracting data, give number of lines read and skipped empty data.



```
Please enter filename:data.csv
Please enter city/area(YARMOUTH, CUMBERLAND, PORTLAND...):portland
Please enter starting year:2015
Data successfully write to portland.txt
56905 lines data has been read.
803 empty data has been skipped.
```