CS5001 Project 8 - Yearly Precipitation in Portland with Exception Handling - Report

Name: Chun Sheung Ng
NUID: 002911540

1. Reflection

- In this project, the easiest part is setting up the function of calculating average numbers because there is only one parameter to handle, and the set-up is direct calculation. The hardest part is the planning ahead of building up the whole thing because all those functions are highly interrelated. Therefore, before working on the main and exceptions-handling, we need some planning time of thinking through the process and clearing our mind.

- I have learned a lot of things through this project, for example, getting familiar with the repeatedly used split(), strip() and replace() string methods when handling data. I understood more about the features of a try/except block, and how to create and use custom errors. More importantly, I started to know how to approach when handling tons of data in a CSV file. When trying to do extensions, I learned when we would need smaller try/except sub-blocks and their pros and cons.

- I would give myself an A-range because I was willing to challenge myself during the process and try to gain more insights when polishing my codes and adding interesting extensions.

2. Output

lab8.py

- The program asks the user to input a CSV data file for handling in the first place:

```
Please enter a CSV data file : mainedata.csv
```

- After getting the right input and running through the main function, the program asks the user to enter a name for a new output file.

```
Enter name of new file: portland.txt
```

portland.txt

Yearly Precipitation Data for Portland in a new output file

- A new output file is created for containing the calculated precipitation data each year in Portland.

lab8.py

- Below is the try/except block inside the main function to maintain the control flow even encountering errors

.....

```
        try:
            # make sure the file is in CSV
            if file_name.endswith(".csv") == False :
                # raise file format error if file is not ended with CSV
                raise FileFormatError
            elif op.exists(file_name) == False :
                # raise file not found error if file not exists
                raise FileNotFoundError
            # open file and assign it to a variable to read through
            file = open(file_name, "r")
            flag = False
```

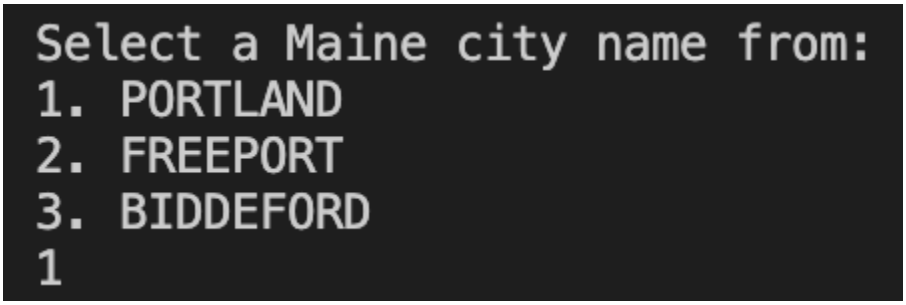
.....

```
180         # set exceptions
181         # print respective error messages
182         # handle custom error - FileNotFoundError
183         except FileNotFoundError as fe :
184             print(fe.message)
185             print("File not exists. Please enter an existing file in folder.")
186         # handle custom error - FileFormatError
187         except FileFormatError as ffe :
188             print(ffe.message)
189             print("Unable to open file. Please enter a CSV file.")
190         # handle TypeError - remind type of particular parameter
191         except TypeError as te :
192             print(te)
193         # handle ValueError - remind the right value of particular parameter
194         except ValueError as ve :
195             print(ve)
196         # handle ZeroDivisionError - remind divisor should not be '0'
197         except ZeroDivisionError as zde:
198             print(zde)
199             print("Please append elements for not having any empty lists.")
```

3. Extension

- For extension, I intend to provide options for users to select cities and do multiple cities instead of only Portland, so:

1. I printed a **menu** allowing user input to select the city (among Portland, Freeport and Biddeford) they would like the program to work on with number 1-3:



2. The program will prompt the user to re-input their option if the input is not a number or not in range of 1-3 so that the program can keep running with incorrect input.

3. I created different output files for each selection with their precipitation data from 2010 to 2020 (Portland output file is already shown above):

Freeport (freeport.txt)

5001_Project_8 > freeport.txt

```
1   FREEPORT Average Rainfall :
2   2010 : 0.161
3   2011 : 0.170
4   2012 : 0.170
5   2013 : 0.150
6   2014 : 0.178
7   2015 : 0.157
8   2016 : 0.139
9   2017 : 0.129
10  2018 : 0.137
11  2019 : 0.158
12  2020 : 0.140
13
```

Biddeford (biddeford.txt)

5001_Project_8 > biddeford.txt

```
1   BIDDEFORD Average Rainfall :
2   2010 : 0.116
3   2011 : 0.134
4   2012 : 0.122
5   2013 : 0.126
6   2014 : 0.148
7   2015 : 0.115
8   2016 : 0.123
9   2017 : 0.139
10  2018 : 0.153
11  2019 : 0.144
12  2020 : 0.124
13
```

4. Acknowledgement

I would like to thank **Professor Gary Cantrell** and his Teaching Assistants, **Bailee Bartash** and **Jeff Cuartas** for their helpful advice during this project.

Also, here is the **list of resources** I have referenced:

Course note: https://docs.google.com/document/d/1-bArrwlgil1F7FyD9OsdRUb6eCVa3VMnNpsZdDLDc0w/edit

W3schools - Python String Methods: https://www.w3schools.com/python/python_ref_string.asp

W3schools - Python List/Array Methods: https://www.w3schools.com/python/python_ref_list.asp

W3schools - Python File Methods: https://www.w3schools.com/python/python_ref_file.asp

Tutorials Point - Python String replace() Method: https://www.tutorialspoint.com/python/string_replace.htm

Python Tutorial - Python Check If File Exists: https://www.pythontutorial.net/python-basics/python-check-if-file-exists/

Tutorials Teacher - Python - Error Types: https://www.tutorialsteacher.com/python/error-types-in-python

Macworld - Master the macOS command line: How to navigate files and folders in Terminal: https://www.macworld.com/article/221277/command-line-navigating-files-folders-mac-terminal.html