Yearly Precipitation in Portland with Exception Handling
Lab/Project for 5001

In the beginning of your academic career or when transitioning from one career to another, you should explore a wide variety of topics to see what interests you. Thus, it seems appropriate to me to offer a variety of lab assignments.

One skill I have found useful in multiple ways is that of taking large amounts of data and extracting useful information. I've downloaded weather data from NOAA for the local area here in Portland, ME in the form of a CSV file.

Input file: https://www.dropbox.com/s/tonebbitmrmv2d9/2769828.csv?dl=0

You might have opened CSV(comma separated value) files in a spreadsheet program, but the truth is that it is just a text file with data separated by commas. This means we can easily read it in and manipulate it in Python.

I want you to take in this file and extract some useful information from it. This is based on the GradeChart demo we did in class. The first thing to do would be to study the demo code and understand how it works and then tackle this assignment.

Your overall goal is to extract and then calculate the average precipitation for each year for the city of Portland, then print that information out into another file. (See sample output below.)

Below you'll find a walkthrough of how I solved this assignment. You are welcome to do it somewhat differently, but make sure you still accomplish the rubric items listed below.

First, take a minute to familiarize yourself with the data file. You can open this file up in a spreadsheet program or just open it up in Notepad++. A CSV file is just a text file.

Now build your frame. I'd like to see at least three functions: main(); get_avg(list_num); and print_avgs(data, city, starting_year, out_file_name)

def main():

Protect main with a try/except block that will throw the appropriate errors. You might start by adding just an "except:" This will catch any exception, but be wary. Once you do this, you won't get the stack trace from the OS when a runtime error occurs. After you start adding specific exceptions, it might be a good idea to remove this general exception. Get this working as a Hello World program before moving on.

Accept from the user the name of a file to process. Raise a ValueError if the filename does not end with csv. Make sure to use an appropriate message. You'll need to add an except line for ValueErrors to print this message. Test this before moving forward and consider removing that general except line you added at first.

Tip: work smarter not harder. There's a string function that will let you check the end of a string.

Add an except to catch the exception if the file is not found and print the appropriate message here as well. This will look a little different than your except for ValueError or TypeError as the OS will check this for you. You just want to catch the error if it happens so your program doesn't terminate early.

Go ahead and set up a list to hold yearly precipitation data by year from 2010 to 2020. You don't have to put anything in it yet just get it ready:

```
portland_data = [[],[],[],[],[],[],[],[],[],[],[]]
```

Now set up your loop to read in each line of the file one at a time to extract the information you want. Don't try to read the entire file at once. That would waste too much memory. Consider testing this functionality before going forward. You want to make sure you are reading what you think you are.

Read in each line into a temporary list by splitting the data up by comma. Now you'll be able to access each field of data, but you'll have to do some experimentation to figure out what index to use for what data. Add some comments in your code to help you remember what is what. For example, temp[1] you'll discover has the city data. Try printing out some aspects of the data to make sure you get what you want.

Now within this line by line loop we need to start capturing the data into our portland_data list of lists. First only capture the data if the city name starts with "PORTLAND," but make sure you get data from any line that starts this way. That is, you want to capture data for Portland and Portland Jetport.

You can't be certain the data will be sorted by year. So if the city name starts with PORTLAND, check to see what year the data is from and then place it at the right index. That means 2010 data should be appended to portland_data[0] and 2011 data should be appended to portland_data[1] and so on. You could do this with a lot of if statements, but there's an easier way. Think about it, 2010 - ? = 0 and 2011 - ? = 1.

You'll also have to extract the year from the date string. Just look at the date format and decide how you need to split it and what the index is for the year.

There's a catch with this though. The program will throw an exception if you try to get precipitation data from an empty field. You'll likely see this as:

```
"ValueError: could not convert string to float: ''
```

This will shut down your main function. We want to avoid that. So, you'll want to place the section that gets the precipitation data in its own try/except block. You can catch the exception as a general except or as an except ValueError. You don't have to print anything, we just want to make sure the program will continue after this error occurs. (Another use for a try except block). Go ahead and keep a count of how many times this error occurs and print this information out to the user. They might want to know how many lines of data were skipped.

You might consider printing out the final result in portland_data to make sure there is data there before moving forward.

Ok now send this list of lists to print_avgs along with the name of the city, the starting year, and the name of an output file.

get_avgs(list_num)

This function should look very familiar. Accept a list and return a double value that is the average of all values in the list. Make sure the correct type is sent in and if the length of the list is 0 just return 0. This function should not have its own try/except block. It will depend on main to print an appropriate error message based on what error occurred.

```
"Type error occurred in get_avg"
```

Tip: don't try to print different error messages in main. Set the appropriate error message when you raise the exception. See the course demonstration.

print_avgs(data, city, starting_year, out_file)

This function accepts a list, a string, an int, and a string. Check each type and raise a TypeError specific to each possible invalid input. For example, if the first argument is not a list then the message thrown by main should be something like "First argument in print_avgs is not a list." If the second argument is not a string, the message printed should be "Second argument is not a string."

Tip: don't try to print different error messages in main. Set the appropriate error message when you raise the exception. See the course demonstration.

This function will open a file for output, write a header based on the sent city name. (See output sample below.)

Then for each list within the sent data it will calculate the average for that list using get_avg(list_name) and print out the information assuming each list is a year and the year starts with the sent starting year. Round each value to three decimal points.

My output looked like this:

```
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
```

Remember you can get up to 26/30 points for finishing the lab, but for 30 points you have to go above and beyond. You don't have to use any of the below extensions. They are just examples. Explore on your own and come up with something fun.

Extension ideas:

1. Do multiple cities instead of just Portland
2. Chart the data in some way using turtle
3. Add more functionality by allowing user to select more filtering aspects like choosing a specific month during the time frame
4. Add more exceptions
5. Extract show fall as well and add that information

Report:

Reflection:

What was the easiest and hardest part of this assignment?

What did you learn?

What grade would you give yourself?

Output:

Include a copy of your output

Extension:

What extension did you add to the assignment?

Rubric:

Total Points: 30

Exception Handling added as requested: (12 pts)

- Main protected with a try/except block
- Exception thrown with specific error if file doesn't end in csv
- Exception thrown if the file doesn't exist
- Try/Except block used to continue program when a data field is empty
- Get_avg throws a specific TypeError if sent something other than a list
- Print_avgs throws specific errors for incorrect type for each argument

Functionality: (4 pts)

- Code is designed, more or less, as requested
- Extracts only lines that start with "Portland"
- Calculations are correct
- Extracts data by year into a list of lists

Output is as requested: (3 pts)

- Correct header
- Three decimal places
- Values make sense

Extensions - 4 points

Code Quality - 4 points

Report - 3 points