Arun Naidu

February 25, 2023

IT FDN 110 A Wi 23: Foundations of Programming: Python

Assignment 07

https://github.com/arunnaidu2021/IntroToProg-Python-Mod07

Exception Handling and Pickling in Python

Introduction

Exception handling is important in programming to control what happens when something unexpected happens in the flow of the code. This could be an error by the user, or something isn't as planned in the operating environment, i.e. the wrong directory, filename, etc., or anything else unforeseen.

Pickling is the process of converting a Python object into a byte stream[1], forming a binary rather than a readable ASCII text file. Pickling is actually not just for files but for network transfers and other binary transactions. Pickling is also known as serialization or flattening or marshalling[2].

In the example presented here, I have taken a database similar to one I have had to use at work and imported it from a large text file in csv format into appropriate Python variables, displayed some parameters, and created a pickled binary file. I can then compare the size of the files: in text form and in binary form.

Since many potential errors can happen along the way, I use different forms of exception handling to try to inform the user of problems.

[1] "Understanding Python pickling and how to use it securely"
https://www.synopsys.com/blogs/software-security/python-pickling/#:~:text=Pickle%20in%20Python%20is%20primarily,transport%20data%20over%20the%20network.

[2] "Python Pickling"
https://www.tutorialspoint.com/python-pickling

Overview

The database I chose to use was a collection of weather data for Seattle for the month of February 2023. I set up an account and downloaded a csv file from this site:

https://www.visualcrossing.com/weather/

The name of the data file is seattle_wa.csv.

The high level pseudocode for my program is as follows:

1. Data
   Set up variables to hold the filenames, file objects, and the lists of data.

2. Processing
   In future iterations I could put some data analysis here.

3. Input/Output
   Get the filename from the user.
   Check that the file exists and is in the correct format.
   Read in the csv file using text mode.
   Present the variables to the user.
   Allow the user to select the variable of interest.
   Save the file as a binary.

Detailed Walkthrough

The first step is to get the user to enter the available filename in the correct format. I raised a custom exception here if the user entered the full filename:

```python
# -- Input/Output -- #
def get_file_name():
    """  Gets the filename from the user
    :param: none
    :return: filename without suffix
    """
    str_file = input("Please enter csv filename or return to exit: ")
    if '.' in str_file:  # user entered full filename
        raise Exception("Please enter the filename without the suffix.")

    return str_file
```

This was done to demonstrate a custom exception. In a practical program, it would not be very friendly to simply exit with an error message, so I would instead loop the input until I get the format I want.

A standard exception was used here while opening the file once the correct name is entered:

```
def open_file(str_file_csv):
    """ Opens the text file
    :param str_file_csv:
    :return obj_handle:
    """
    try:
        return open(str_file_csv, 'r')
    except FileNotFoundError:
        print("File not found.")
        exit(-1)
```

Then I read in the comma separated text file and split the header into elements like we have in previous assignments:

```
def print_header(obj_file):
    """ Show the items to the user
    :param obj_file:
    :return: list of header items
    """
    lst_header_data = obj_file.readline().split(',')  # Pull the header row from
the dataset
    index = 0
    for i in lst_header_data:
        print(index, '\t', i)
        index += 1
    return lst_header_data   # return the list with the header row
```

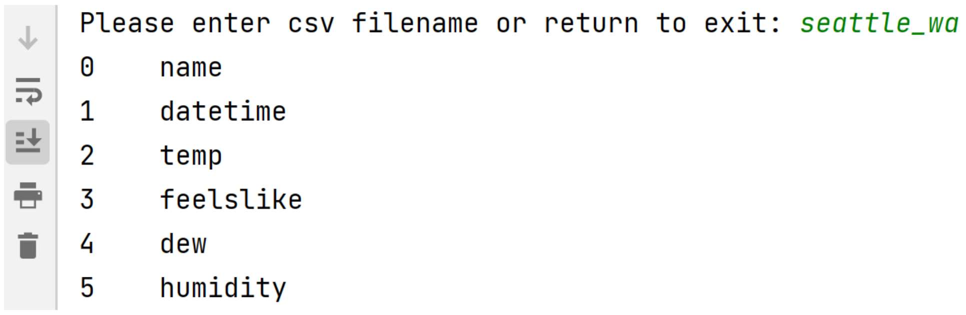
Then there is the pickling function that takes the csv file and pickles it into binary:

```
def compactor(str_file_csv, str_file_bin):
    """ Compress text file into binary

    :param str_file_csv:
    :param str_file_bin:
    :return: none
    """
    lst_table = []
    obj_file_csv = open(str_file_csv, 'r')
    index = 0
    for i in lstHeaderData:  # read in data
        lstRow = obj_file_csv.readline().split()
        lst_table.append(lstRow)   # build table
    obj_file_csv.close()
    obj_file_bin = open(str_file_bin, "wb")
    pickle.dump(lst_table, obj_file_bin)
    obj_file_bin.close()
```

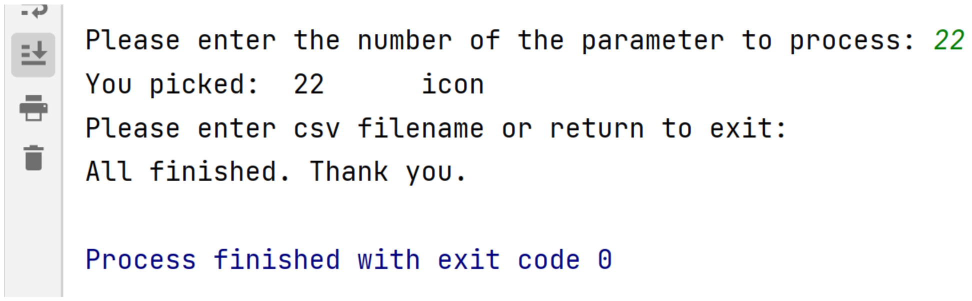
The main section of code allows the user to pick out one of the parameters in the data. In a final version of the program, the user would get some statistical analysis of their selected parameter. I decided to leave that out since it wasn't in the scope of the assignment.

Here are some screenshots of the code running in PyCharm:



```
Please enter csv filename or return to exit: seattle_wa
0     name
1     datetime
2     temp
3     feelslike
4     dew
5     humidity
```

Figure 1.

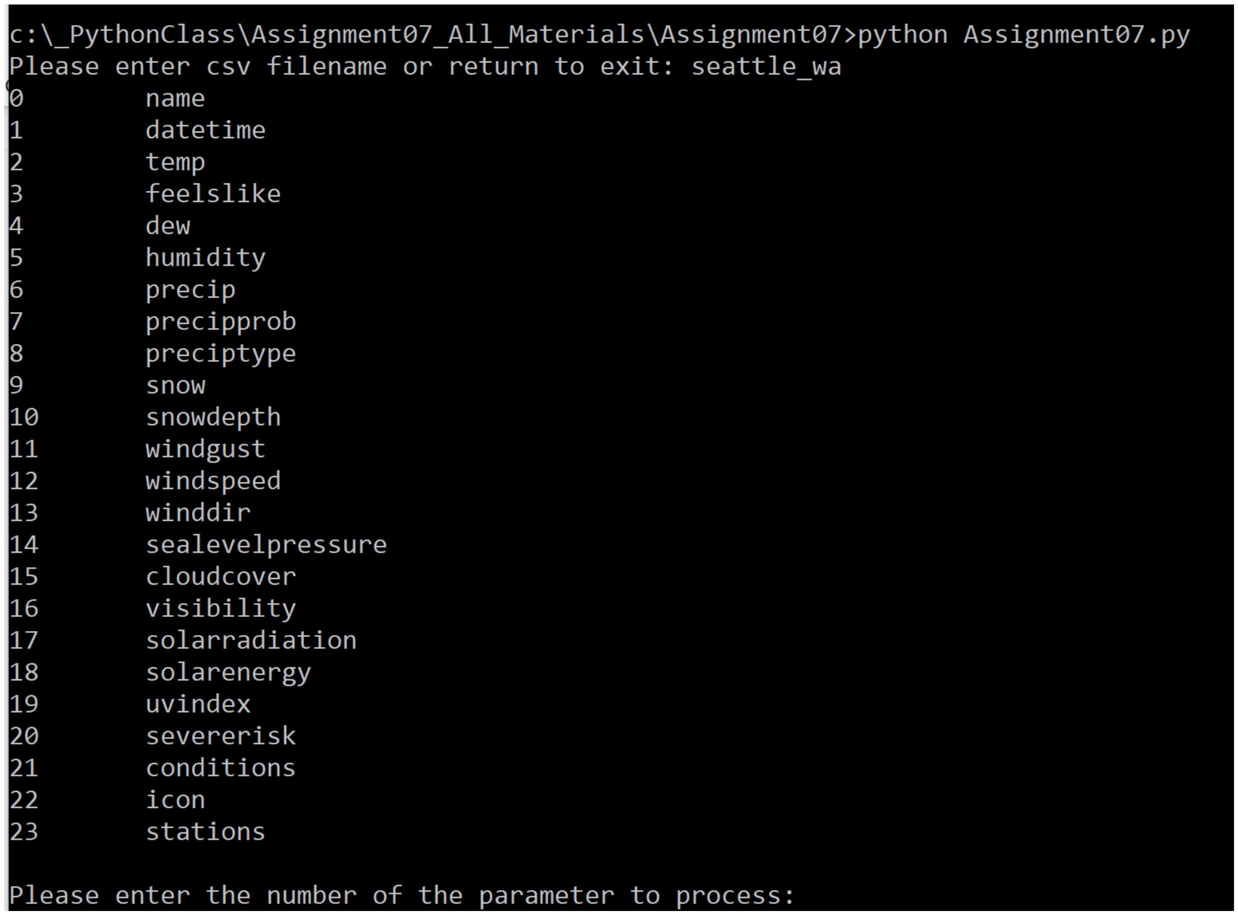


```
Please enter the number of the parameter to process: 22
You picked:  22      icon
Please enter csv filename or return to exit:
All finished. Thank you.

Process finished with exit code 0
```

Figure 2.

And here is it running in the console:

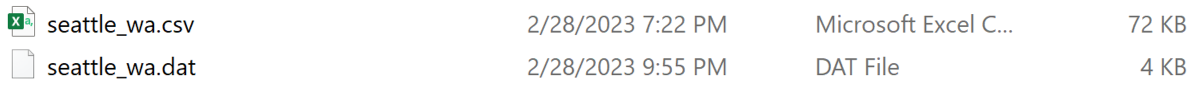


```
c:\_PythonClass\Assignment07_All_Materials\Assignment07>python Assignment07.py
Please enter csv filename or return to exit: seattle_wa
0        name
1        datetime
2        temp
3        feelslike
4        dew
5        humidity
6        precip
7        precipprob
8        preciptype
9        snow
10       snowdepth
11       windgust
12       windspeed
13       winddir
14       sealevelpressure
15       cloudcover
16       visibility
17       solarradiation
18       solarenergy
19       uvindex
20       severerisk
21       conditions
22       icon
23       stations

Please enter the number of the parameter to process:
```

Figure 3.

Finally, here is the comparison of the file size in csv and binary. Pretty impressive compression:



seattle_wa.csv	2/28/2023 7:22 PM	Microsoft Excel C...	72 KB
seattle_wa.dat	2/28/2023 9:55 PM	DAT File	4 KB

Figure 4.

Summary

This assignment was quite different in that there was no existing structure to work from. I had to rely heavily on previous assignments and the class textbook as well as two other textbooks I have: "Automate the Boring Stuff with Python" 2nd edition, by Al Sweigert, and "Python Crash Course" 2nd edition, by Eric Matthes.

But there is potential to do a lot of interesting things with databases that are applicable to my work.