Chris Perry

November 20, 2019

Foundations of Programming: Python Assignment 07

Pickle Module and Try/Except Python Scripts

Introduction

In this assignment, I will go over how to use the pickle module with python to serialize data. Also, I will show how to capture and trap errors with a try/except loop/block. I will be creating the script using Python3, PyCharm CE and macOS.

Coming up with the Code

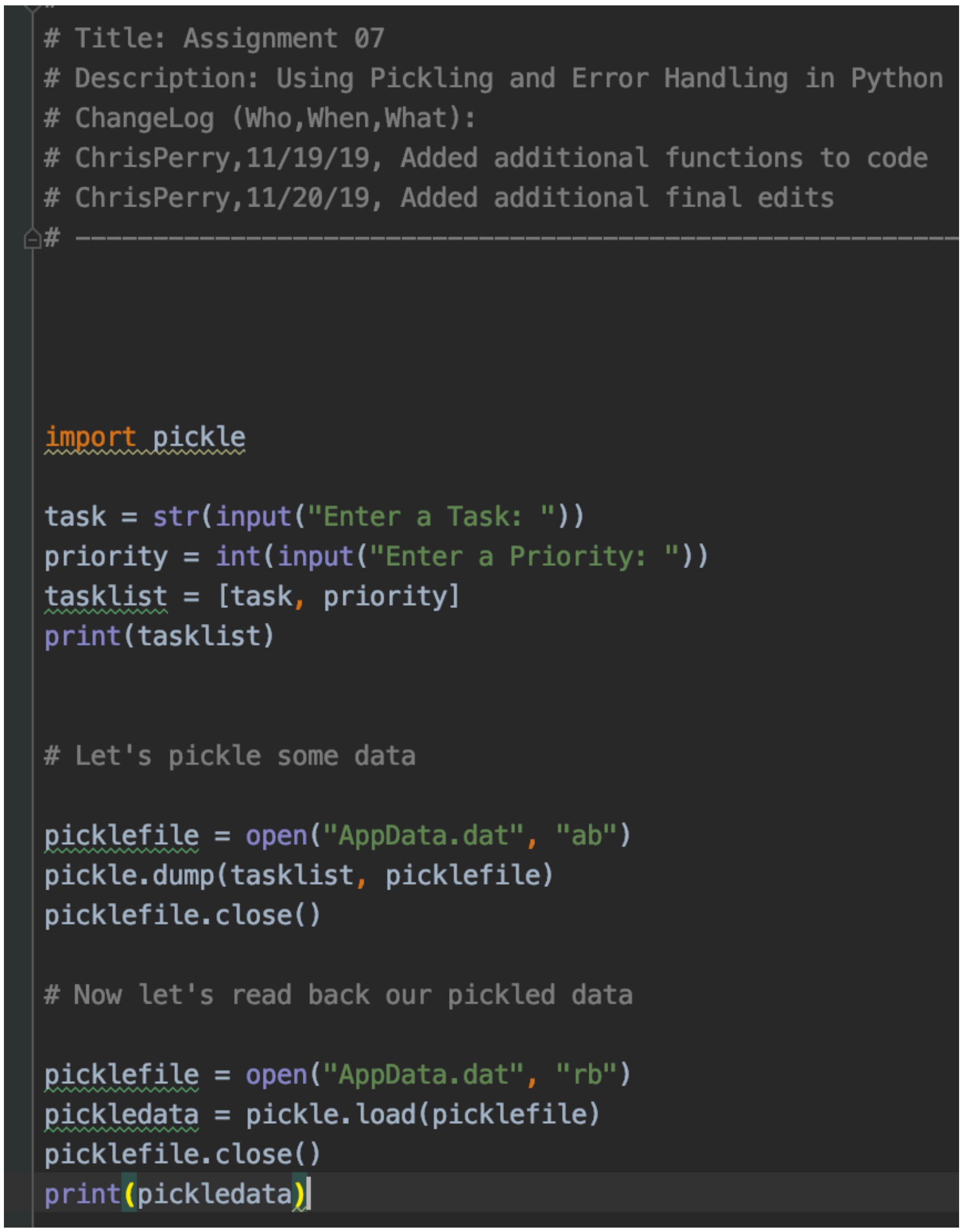
To write my code, I used PyCharm CE - a popular python programming editor/application.

I decided to demonstrate both new concepts for this assignment in 2 different scripts. First, with the Pickle module, pickling is used for serializing and de-serializing a Python object structure. Any object in python can be pickled so that it can be saved on disk. With Pickling, an object can be serialized and then de-serialized to be transferred to another Python script and/or written and then read back from a file.

With Try/Except, we first use try to execute any code we want - in this assignment I use Try/Except as a loop to check for errors. We can use the except part of the loop to trap any error that may happen with our try code. I have used Try/Except in a few assignments already this year and like a good programmer, I wanted to re-use some code that was already written.

How the Code Works

First up, let's look at our Pickle example. I kept the code as simple as possible to just be able to spell out quickly how the Pickle Module works and functions. Below is a screenshot of my Pickle code:



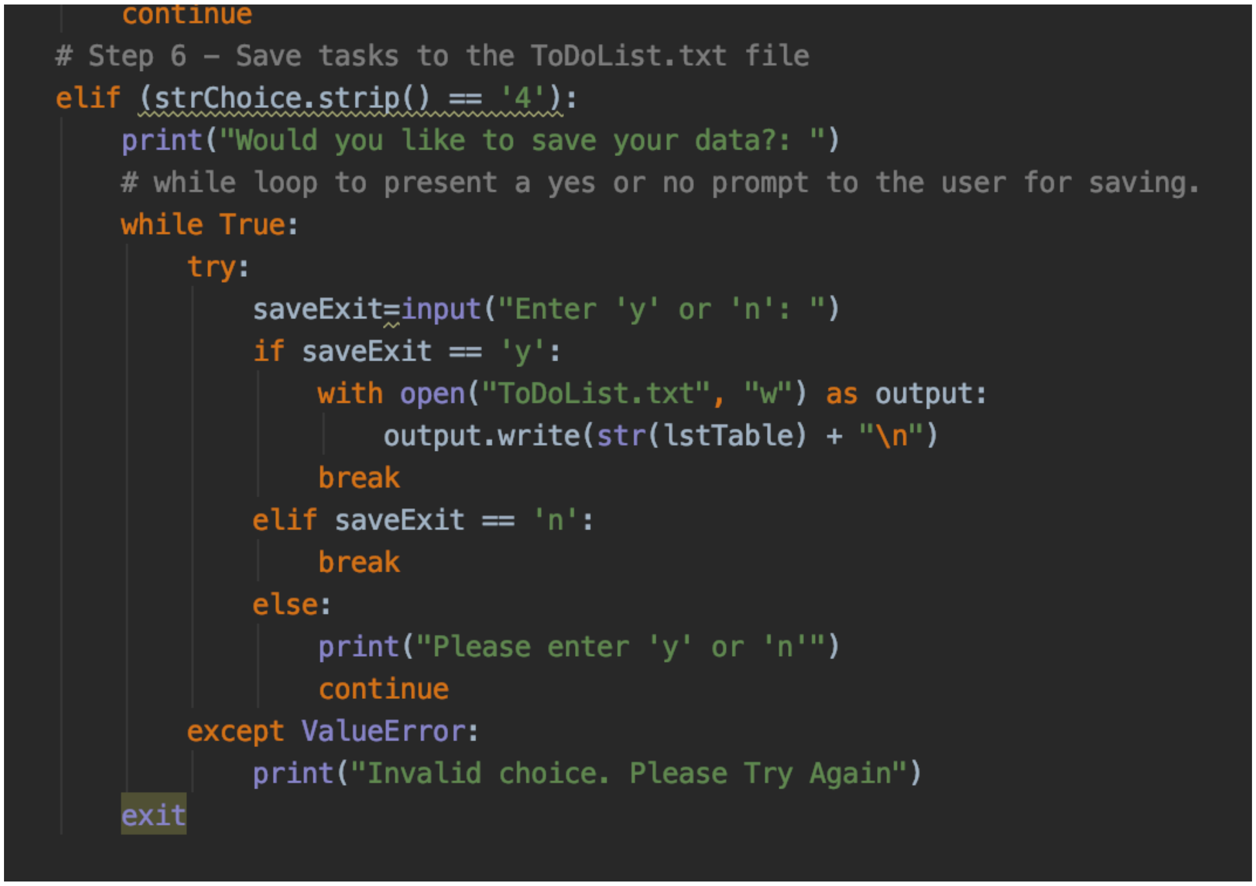
```
# Title: Assignment 07
# Description: Using Pickling and Error Handling in Python
# ChangeLog (Who,When,What):
# ChrisPerry,11/19/19, Added additional functions to code
# ChrisPerry,11/20/19, Added additional final edits
# -----------------------------------------------------------




import pickle

task = str(input("Enter a Task: "))
priority = int(input("Enter a Priority: "))
tasklist = [task, priority]
print(tasklist)


# Let's pickle some data

picklefile = open("AppData.dat", "ab")
pickle.dump(tasklist, picklefile)
picklefile.close()

# Now let's read back our pickled data

picklefile = open("AppData.dat", "rb")
pickledata = pickle.load(picklefile)
picklefile.close()
print(pickledata)
```

The first thing you must do if you want to use Pickle is import the module into your script. You can do that as pictured above by using “import pickle” with the import syntax.

Next, I just came up with some simple input boxes to collect some data and write it to a list. Our next section of code is where the fun happens. I give a simple name to our new file and call it “picklefile” to match the theme of what this script is explaining. Using the open function, the script creates an app data file called “AppData.dat” and uses the “ab” syntax to append binary. It’s very important that we use “ab” and not just “a” like in other assignments. Since pickling the data serializes it, we need to write binary to a file and not string text. The dump function from the pickle module is then used with pickle.dump with our list variable (tasklist) passed to our file name (picklefile). Once this is done, we need to close the file we opened, or else the script will go in an infinite loop and/or error out.

Lastly, we want to read the data back from our binary file so we need to de-serialize it with pickle. We use the open function with our file (picklefile) but this time we use “rb” because we only want to read. Once again it is important to use “rb” and not just “r” because we need to read a binary file and binary data. We then set a new variable called “pickledata” and use the load function from the pickle module to load our de-serialized data. We make sure to close out our file once we’re done and then use print the data back with a simple print function.

Moving on to our 2nd script, we look at our Try/Except example. I decided to reuse code I had written before for a different assignment since it applied here as well. Below is the code I am using for my try/except example:



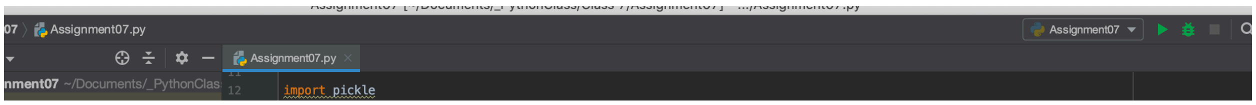
```
        continue
    # Step 6 - Save tasks to the ToDoList.txt file
    elif (strChoice.strip() == '4'):
        print("Would you like to save your data?: ")
        # while loop to present a yes or no prompt to the user for saving.
        while True:
            try:
                saveExit=input("Enter 'y' or 'n': ")
                if saveExit == 'y':
                    with open("ToDoList.txt", "w") as output:
                        output.write(str(lstTable) + "\n")
                    break
                elif saveExit == 'n':
                    break
                else:
                    print("Please enter 'y' or 'n'")
                    continue
            except ValueError:
                print("Invalid choice. Please Try Again")
        exit
```

This code represents a choice from a main menu of a script. In this choice, we are asking the user if they would like to save any of the data they have so far input to the running session. The try/except loop here is nested inside a while loop. The while loop ensures we keep trying until we get the answer we want for our question to the user.

However, just trying over and over again isn't sufficient. We need to be able to account for any input the user may put in. For example, they are presented with a choice of "y" or "n" (yes or no) when asked if they would like to save their data. If I don't trap it, what happens when the user puts in hello or just some random characters instead of y or n? The script will get a value error and fail out unexpectedly. As a result, I used a try/except loop with the except doing the error handling in this example. With my except, I am basically saying if it isn't y or n, don't accept that input and try again. It's also important to note that with the except, I could also print a number of things like a "_doc_" statement for any docstrings I may have for errors or even just print out the exact error itself. I chose to keep it simple so I just focused on the exact error that would occur if the user did not type "y" or "n", which in this case is a ValueError.

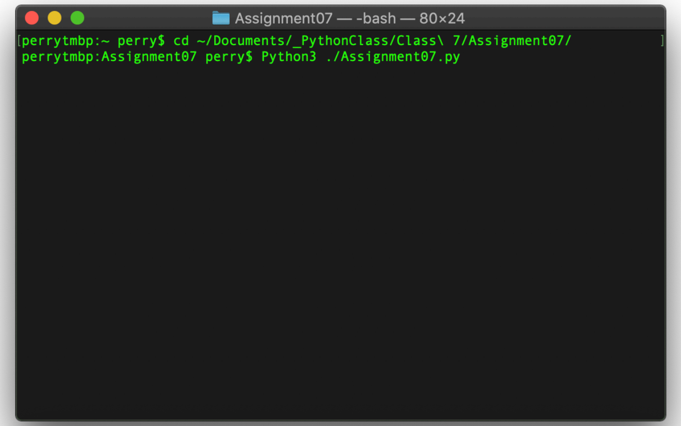
Launching and Saving the Script

We are using PyCharm to run a script for an assignment. You can press the play button in the upper right corner or hit run from the menu bar to run the script. Below is a screenshot of how I run it from PyCharm:

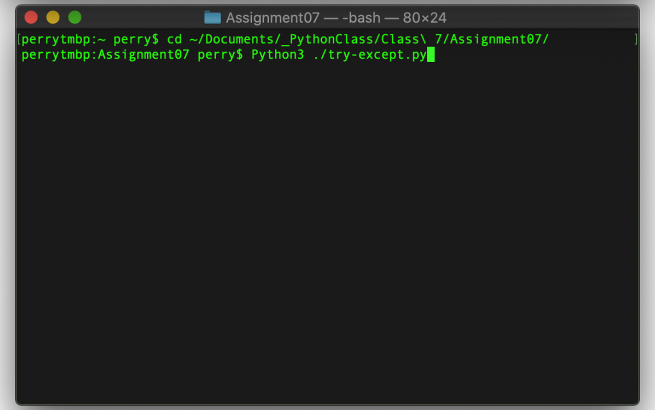


```
import pickle
```

When launching from the shell, since I am using Python3 on my Mac and macOS can have more than one version of Python installed, I have to make sure to designate Python3 when I call the script to run. Below is a screenshot of how I call the script to run from terminal:



```
perrytmbp:~ perry$ cd ~/Documents/_PythonClass/Class\ 7/Assignment07/
perrytmbp:Assignment07 perry$ Python3 ./Assignment07.py
```

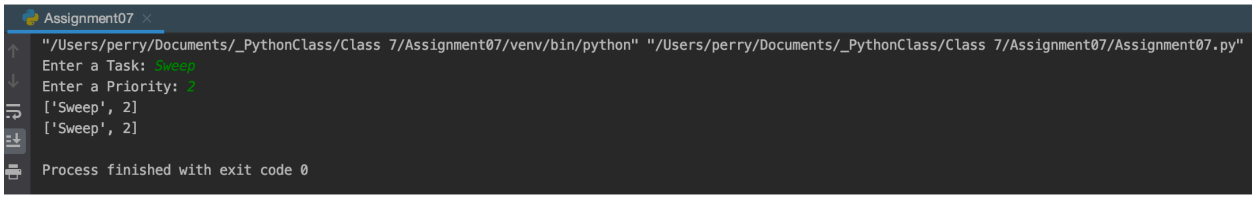


```
perrytmbp:~ perry$ cd ~/Documents/_PythonClass/Class\ 7/Assignment07/
perrytmbp:Assignment07 perry$ Python3 ./try-except.py
```

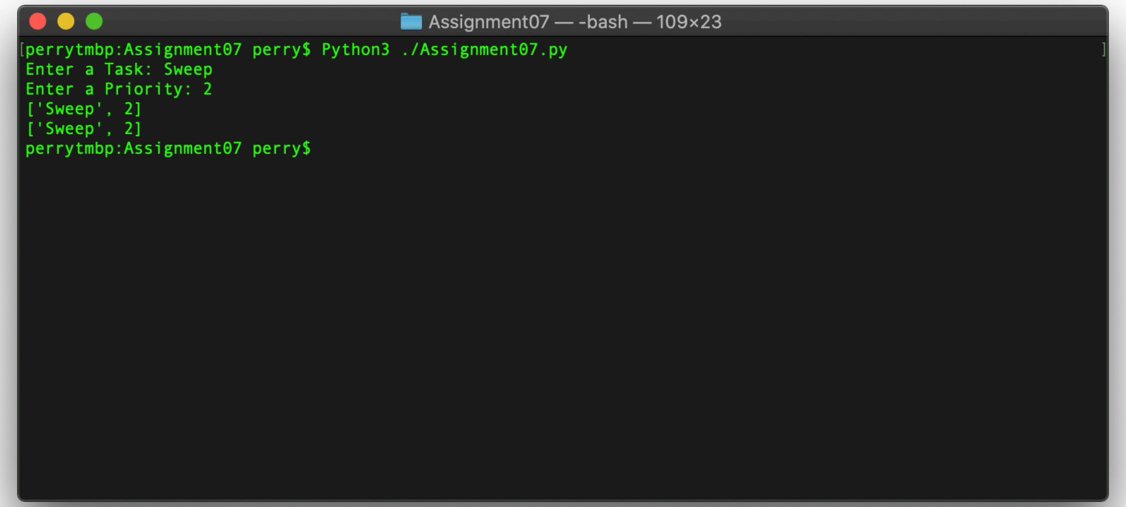
The script is saved as a .py file (for python script) in my class directory: /Users/perry/Documents/_PythonClass/Class 7/Assignment06/Assignment07.py and same directory, but ./try-except.py for the 2nd script.

Output of the Final Script

Please see below for a final output screenshot of pickle Python script. First, the output while running in PyCharm and then from the shell:

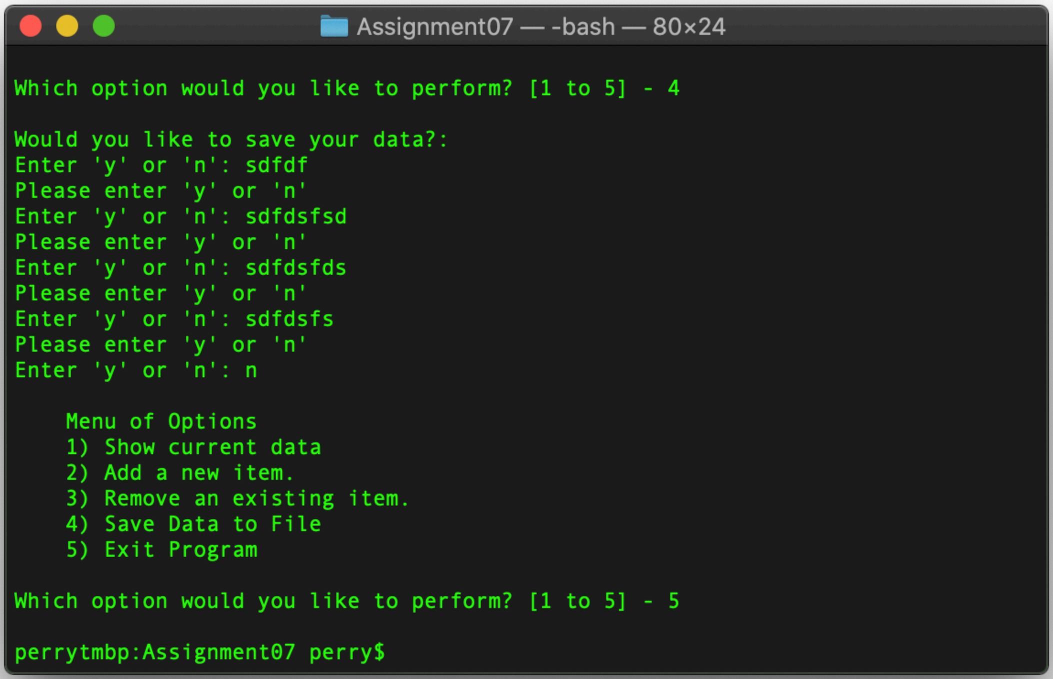


```
"/Users/perry/Documents/_PythonClass/Class 7/Assignment07/venv/bin/python" "/Users/perry/Documents/_PythonClass/Class 7/Assignment07/Assignment07.py"
Enter a Task: Sweep
Enter a Priority: 2
['Sweep', 2]
['Sweep', 2]

Process finished with exit code 0
```



```
perrytmbp:Assignment07 perry$ Python3 ./Assignment07.py
Enter a Task: Sweep
Enter a Priority: 2
['Sweep', 2]
['Sweep', 2]
perrytmbp:Assignment07 perry$
```

Output from running the try-except script from the shell (terminal) in macOS:



```
Which option would you like to perform? [1 to 5] - 4

Would you like to save your data?:
Enter 'y' or 'n': sdfdf
Please enter 'y' or 'n'
Enter 'y' or 'n': sdfdsfsd
Please enter 'y' or 'n'
Enter 'y' or 'n': sdfdsfds
Please enter 'y' or 'n'
Enter 'y' or 'n': sdfdsfs
Please enter 'y' or 'n'
Enter 'y' or 'n': n

    Menu of Options
    1) Show current data
    2) Add a new item.
    3) Remove an existing item.
    4) Save Data to File
    5) Exit Program

Which option would you like to perform? [1 to 5] - 5

perrytmbp:Assignment07 perry$
```

Final Summary

During this seventh assignment, I was able to demonstrate how to use the pickle module to serialize and de-serialize to a local app data binary file. I was also able to show how to read back the data from that file.

I was also able to demonstrate how to use a try/except loop/block to trap an error based on what the try portion of the code was/is using. With the except portion of the loop/block, I could chose how I wanted to display this error to the user - I could just print out the error they would get back like a ValueError if the input doesn't match what the script needs or I could use a custom print function to display any message.