Chase Richardson

May 29, 2023

Foundations of Programming: Python

Assignment 07

https://github.com/jcrich13/IntroToProg-Python-Mod07

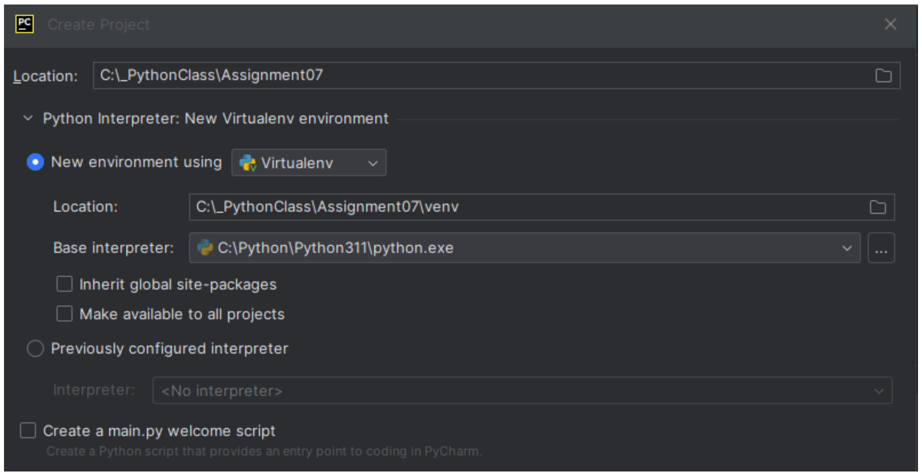
Demo Python Script for Pickling/Error Handling

Introduction

The primary purpose of Assignment 07 is to experiment with Pickling and Error Handing within Python. I decided to continue to use the To Do List as the primary use case for deploying Pickling and Error Handling techniques. In order to successfully complete this task, I utilized the lessons learned from Assignment 06 pertaining to functions and dictionaries. Furthermore, I reviewed the course material by modules, book, and other internet sources to learn more about Pickling and Error Handing. The information provided below outlines the steps I completed in order to create Assignment 07.

Getting Started

To create the Python script in Assignment 07, I opened the PyCharm Integrated Development Environment (IDE) and created a brand-new project. When creating the Project, I assigned the location to be underneath the _PythonClass and Assignment07 subfolders within the C:\, created a new virtual environment, and unchecked the "Create a main.py welcome script" to avoid redundancy in the assignment folder (Figure 1).


Create Project
Python Interpreter: New Virtualenv environment
Previously configured interpreter
Create a Python script that provides an entry point to coding in PyCharm.

Location: C:_PythonClass\Assignment07

New environment using: Virtualenv

Location: C:_PythonClass\Assignment07\venv

Base interpreter: C:\Python\Python311\python.exe

☐ Inherit global site-packages

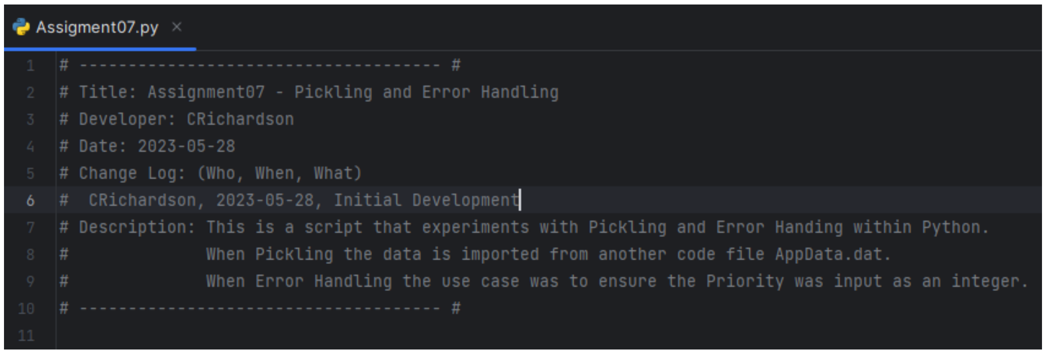
☐ Make available to all projects

Interpreter: <No interpreter>

☐ Create a main.py welcome script

Figure 1: Creating a New Project within PyCharm for Assignment07

Since there was no starter script available this week I created a brand-new python file called "Assigment07.py" to complete the subsequent steps. Once Assigment07 was created, I filled out the Developer Notes/Change Log section in PyCharm for documentation purposes (Figure 2).

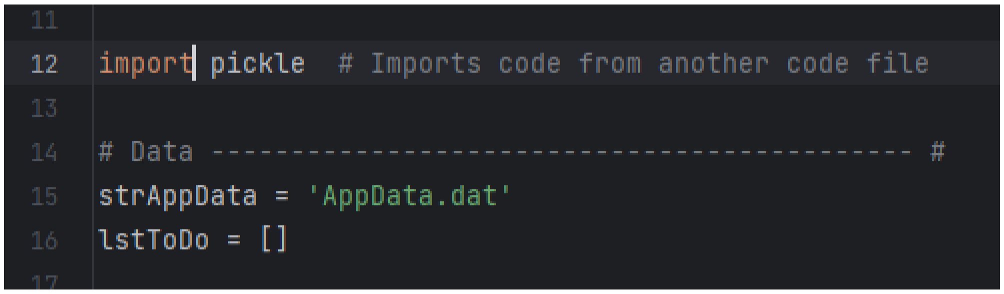


```
# ------------------------------------ #
# Title: Assignment07 - Pickling and Error Handling
# Developer: CRichardson
# Date: 2023-05-28
# Change Log: (Who, When, What)
#  CRichardson, 2023-05-28, Initial Development
# Description: This is a script that experiments with Pickling and Error Handing within Python.
#              When Pickling the data is imported from another code file AppData.dat.
#              When Error Handling the use case was to ensure the Priority was input as an integer.
# ------------------------------------ #
```

Figure 2: Developer Notes and Change Log

Introducing Pickling and Data Section

Working with text files have their limitations, so the concept pickling is available within Python. Our course textbook states, "The pickle module allows you to pickle and store complex data in a file" (Dawson, 200). Therefore, to utilize pickling within Python/PyCharm I first had to "import pickle" in order to import code from another code file. More information on the next steps with pickling will be covered in the later sections. Lastly, I declared the key variables that will be referenced throughout the code for the AppData.dat file and To Do list as my use case. The entire section is captured below (Figure 3).

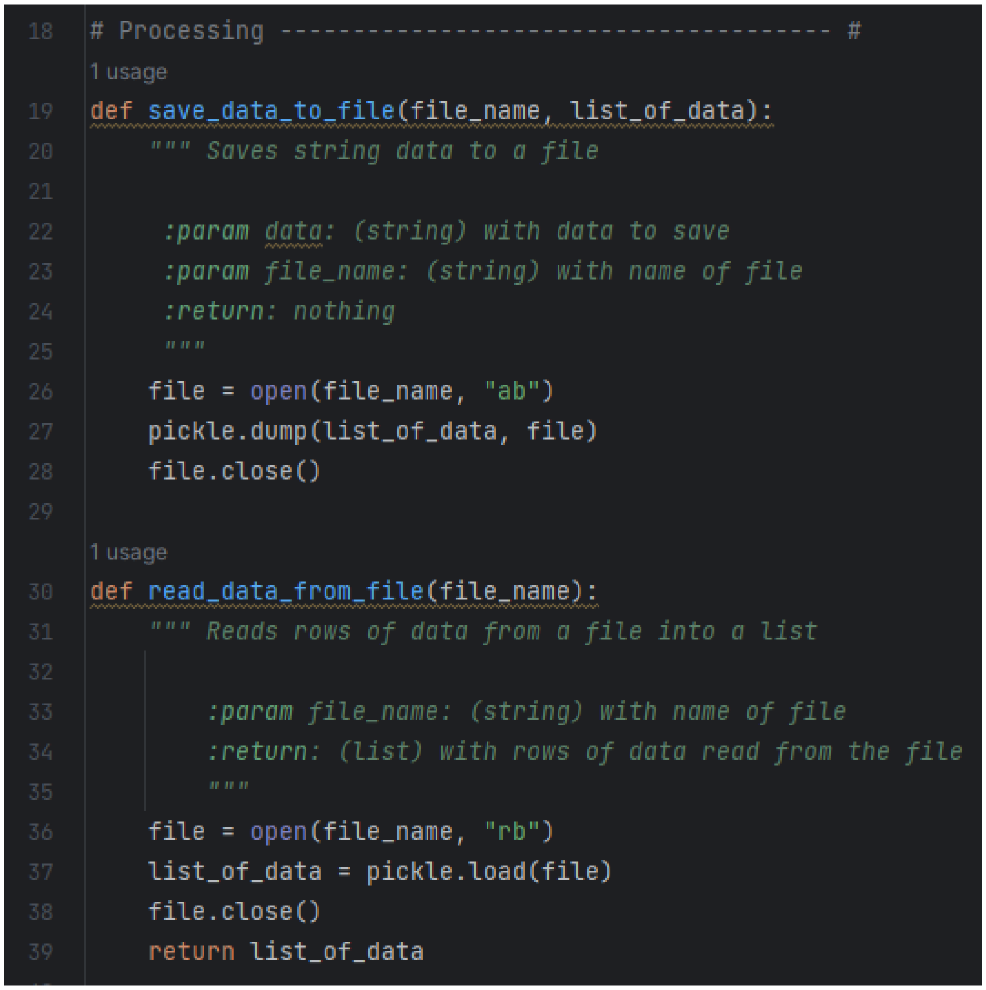


```
import pickle  # Imports code from another code file

# Data ---------------------------------------------- #
strAppData = 'AppData.dat'
lstToDo = []
```

Figure 3: Add and Remove Processing the Data Section

Processing the Data Section

Within the Processing the Data section, I defined two functions "save_data_to_file" and "read_data_from_file". With pickling the steps in performing the function are slightly different than what was shared in earlier assignments. First, when opening the file instead of using just write, read, or append, I had to state "ab" for append to binary for save and "rb" for read to binary for read. This is because, "Pickled objects must be stored in a binary file-they can't be stored in a text file" (Dawson, 201). Second, rather than just using a "for" loop or append for list or dictionaries, pickle functions must be used to correctly relay the data. For save data, I used pickle.dump() and for read data, I used pickle.load(). The entire Processing the Data section is captured below (Figure 4).

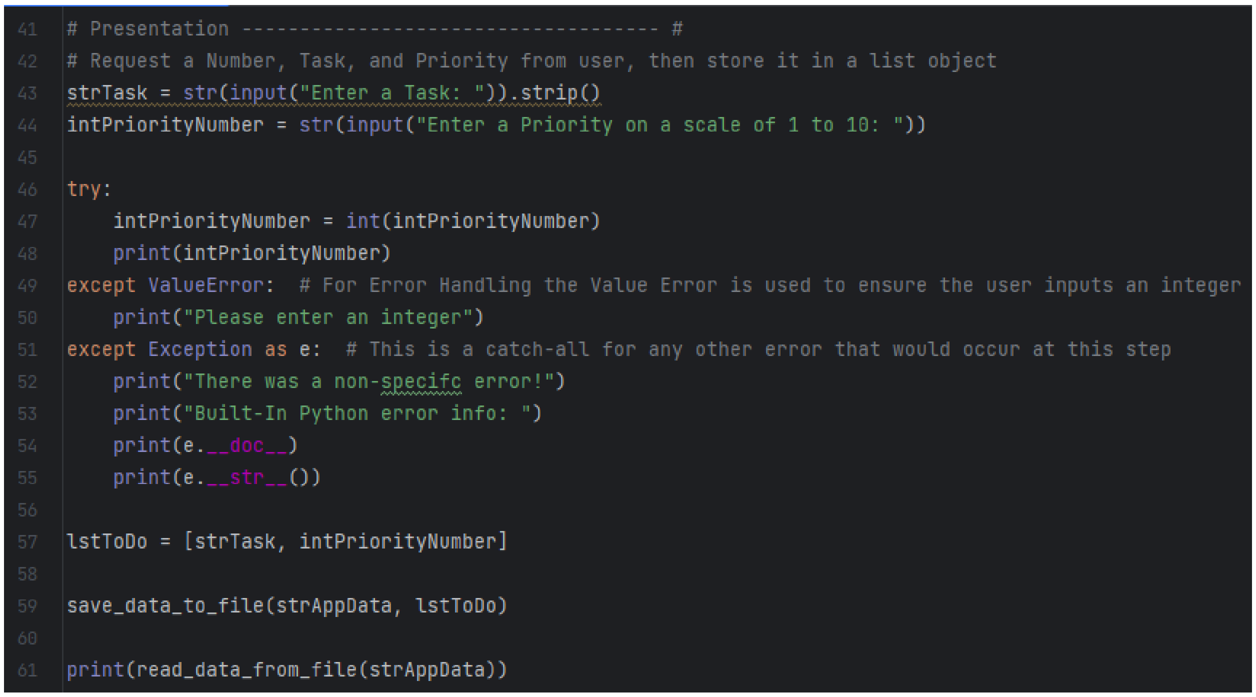


```python
# Processing -------------------------------------- #
1 usage
def save_data_to_file(file_name, list_of_data):
    """ Saves string data to a file

     :param data: (string) with data to save
     :param file_name: (string) with name of file
     :return: nothing
     """
    file = open(file_name, "ab")
    pickle.dump(list_of_data, file)
    file.close()

1 usage
def read_data_from_file(file_name):
    """ Reads rows of data from a file into a list

        :param file_name: (string) with name of file
        :return: (list) with rows of data read from the file
        """
    file = open(file_name, "rb")
    list_of_data = pickle.load(file)
    file.close()
    return list_of_data
```

Figure 4: Processing the Data Section

Presentation of the Data with Error Handling

For the first part of the Presentation of the Data section, the user will be prompted to input the task and priority in string format. However, for error handling experimentation purposes, I really want the user to input the priority using integers from 1-10. For demonstration purposes, the user will be able to input any value as a string, but the error handling will remind them to change their behavior accordingly. This is great, because when, "Using Python's exception handling functionality, you can intercept and handle exceptions so that your program doesn't abruptly end" (Dawson, 205). I utilized the try/except clause, for a ValueError exception type and regular Exception type as a catch-all if another error message were to be generated. Lastly, to complete this section, I saved the Task and Priority to the List and used the functions declared earlier to save and read the data to and from the file accordingly. The complete code is listed directly below (Figure 5):

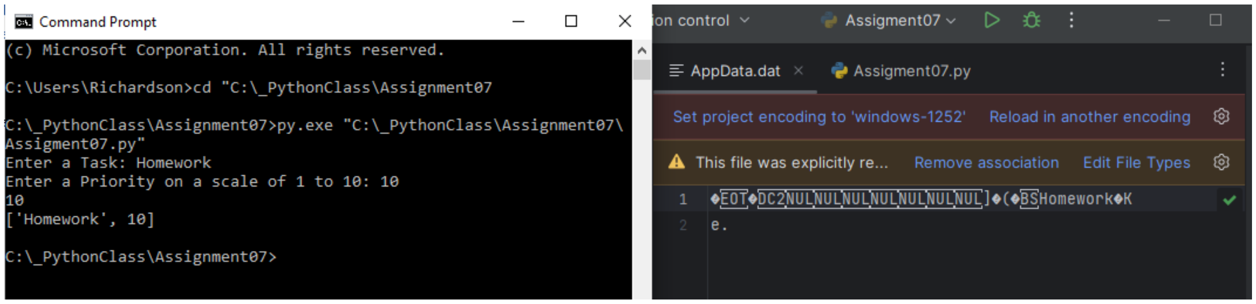


```python
# Presentation ------------------------------------ #
# Request a Number, Task, and Priority from user, then store it in a list object
strTask = str(input("Enter a Task: ")).strip()
intPriorityNumber = str(input("Enter a Priority on a scale of 1 to 10: "))

try:
    intPriorityNumber = int(intPriorityNumber)
    print(intPriorityNumber)
except ValueError:  # For Error Handling the Value Error is used to ensure the user inputs an integer
    print("Please enter an integer")
except Exception as e:  # This is a catch-all for any other error that would occur at this step
    print("There was a non-specifc error!")
    print("Built-In Python error info: ")
    print(e.__doc__)
    print(e.__str__())

lstToDo = [strTask, intPriorityNumber]

save_data_to_file(strAppData, lstToDo)

print(read_data_from_file(strAppData))
```

Figure 5: Presenting the Data Section

Running the Program and Error Handling Examples

Figure 6 displays the final solution in Command prompt. I made sure to change the directory first to the location where the python script and text file relative path is located to ensure the script wrote to the file correctly.



```
(c) Microsoft Corporation. All rights reserved.

C:\Users\Richardson>cd "C:\_PythonClass\Assignment07

C:\_PythonClass\Assignment07>py.exe "C:\_PythonClass\Assignment07\
Assigment07.py"
Enter a Task: Homework
Enter a Priority on a scale of 1 to 10: 10
10
['Homework', 10]

C:\_PythonClass\Assignment07>
```

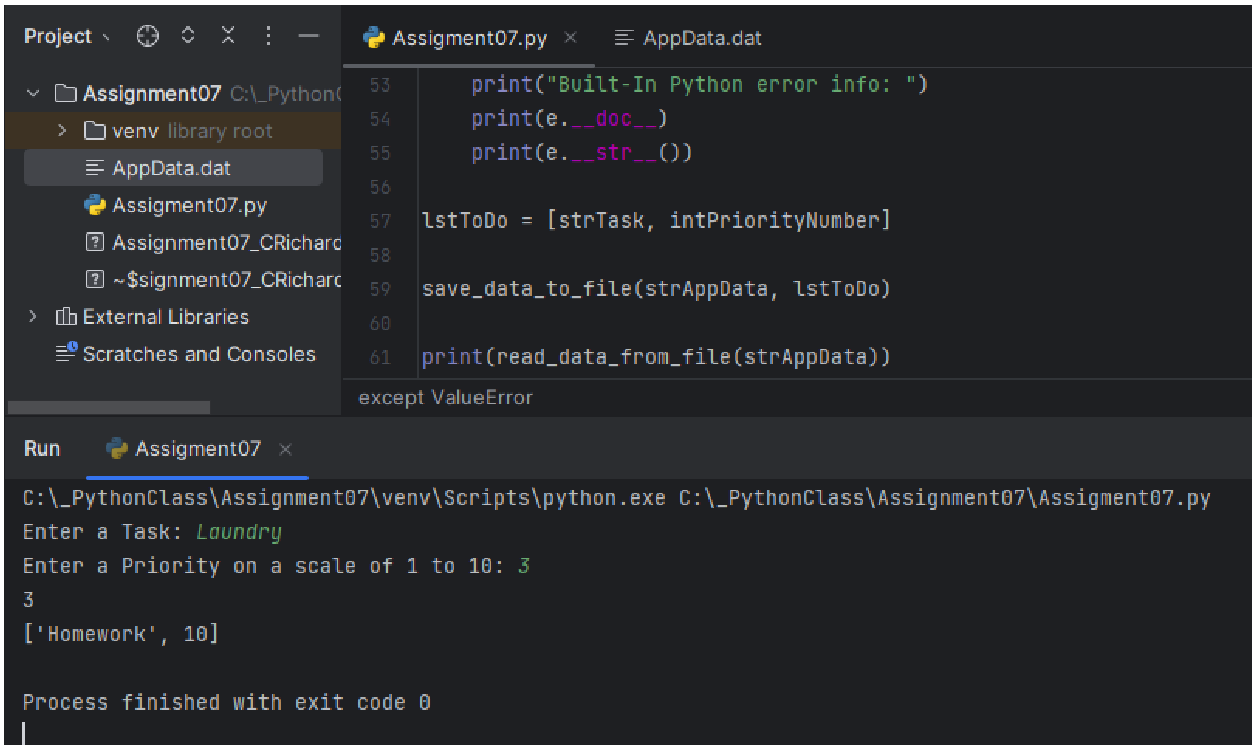
Set project encoding to 'windows-1252' Reload in another encoding

This file was explicitly re... Remove association Edit File Types

```
◆EOT◆DC2NULNULNULNULNULNULNUL]◆(◆BSHomework◆K
e.
```

Figure 6: Pickling from To Do List in Command Prompt

Figure 7 displays the last part of the solution using the Run Feature within PyCharm Community Edition.

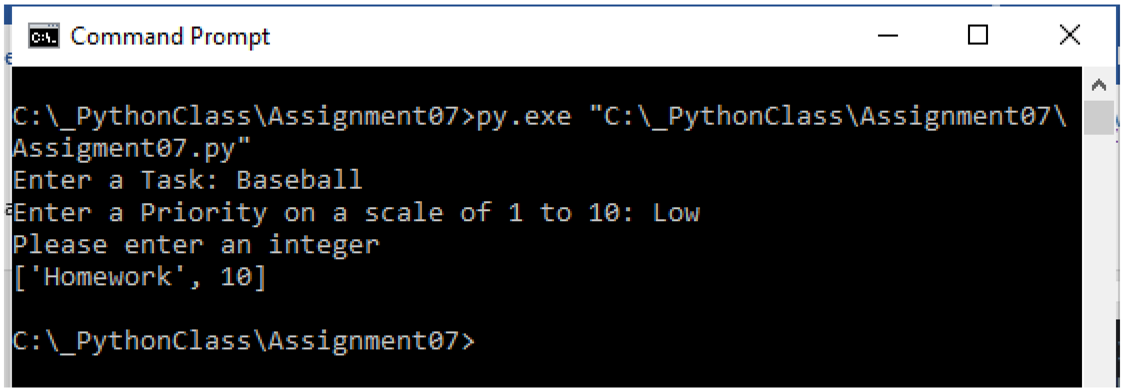


```
        print("Built-In Python error info: ")
        print(e.__doc__)
        print(e.__str__())

lstToDo = [strTask, intPriorityNumber]

save_data_to_file(strAppData, lstToDo)

print(read_data_from_file(strAppData))
```

```
C:\_PythonClass\Assignment07\venv\Scripts\python.exe C:\_PythonClass\Assignment07\Assigment07.py
Enter a Task: Laundry
Enter a Priority on a scale of 1 to 10: 3
3
['Homework', 10]

Process finished with exit code 0
```

Figure 7: Pickling from To Do List in Run mode of PyCharm Community Edition

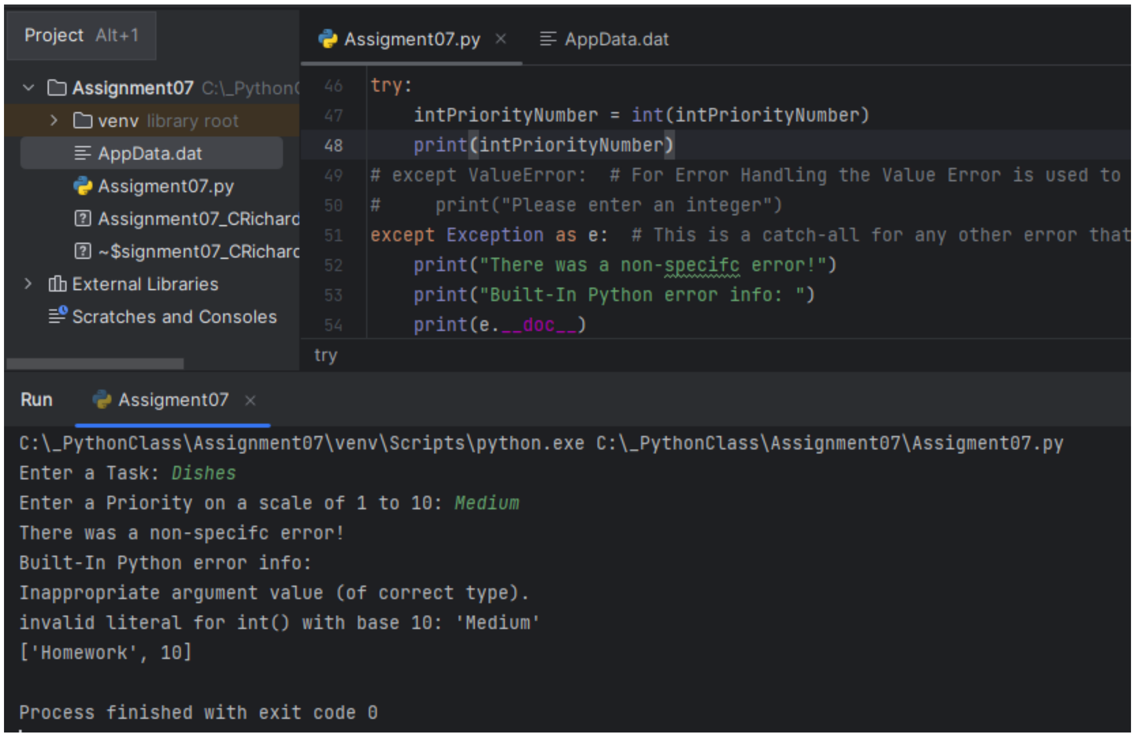
Figure 8 displays the Error Handling solution in Command prompt if the user selects the priority as a string and not an integer. Again, I made sure to change the directory first to the location where the python script and text file relative path is located to ensure the script wrote to the file correctly.



```
C:\_PythonClass\Assignment07>py.exe "C:\_PythonClass\Assignment07\
Assigment07.py"
Enter a Task: Baseball
Enter a Priority on a scale of 1 to 10: Low
Please enter an integer
['Homework', 10]

C:\_PythonClass\Assignment07>
```

Figure 8: Error Handling from To Do List in Command Prompt

Figure 9 displays the Error Handling solution in PyCharm Community Edition if the user selects the priority as a string and not an integer. However, I commented out the ValueError section, to simulate the different behavior if the user were to fall into the "catch-all" bucket. This provides a bit more information and clearer than the generic python error message.



```
try:
    intPriorityNumber = int(intPriorityNumber)
    print(intPriorityNumber)
# except ValueError:  # For Error Handling the Value Error is used to
#     print("Please enter an integer")
except Exception as e:  # This is a catch-all for any other error that
    print("There was a non-specifc error!")
    print("Built-In Python error info: ")
    print(e.__doc__)
```

```
C:\_PythonClass\Assignment07\venv\Scripts\python.exe C:\_PythonClass\Assignment07\Assigment07.py
Enter a Task: Dishes
Enter a Priority on a scale of 1 to 10: Medium
There was a non-specifc error!
Built-In Python error info:
Inappropriate argument value (of correct type).
invalid literal for int() with base 10: 'Medium'
['Homework', 10]

Process finished with exit code 0
```

Figure 9: Error Handling from To Do List in Run mode of PyCharm Community Edition

Summary

In conclusion Assignment 07, provided a refresher once again on again on advanced functions, and earlier concepts. New concepts like pickling and error handling were introduced. The end result was more of a test script to experiment within the previous created to do list resource that allows the user to work with binary files and the try/except clauses. This was a drastically different type of assignment given the free form nature that was provided.

Bibliography

Dawson, Michael. *Python Programming for the Absolute Beginner*. Boston, Course Technology, Cop, 2008.