Mike Galloway
May 23, 2023
IT FDN 110 A
Assignment 06

ToDoList (Python)

Introduction

This week's module 6 covered functions with parameters, using variables as arguments, assigning classes, doc strings, debugging in Pycharm and creating a GitHub webpage. We learned functions are used to group one or more statements and must be defined before calling the function into use. This can be done by using the *def* function to define a term that executes statements in code. Parameters were introduced to pass values into the function for data processing and are officially called 'arguments' with return values used to make the function act like an expression. We used tuples to pack and unpack multiple values in return and list functions and found return function to be less tedious. The 'none' data type was introduced as a placeholder or omission of a parameter data, which is useful for writing with str or conditional statements. Function document headers (Doc Strings) were introduced as a means for developers to include additional notes in a docstring. Classes are used as a way of grouping functions, variables, and constants, and using them in code later so you do not have to rewrite the functions again. Programming statements are wrapped up into functions which can be wrapped up into classes. We were shown how to create a webpage in GitHub which will be useful for application and web development later in our Python journey.

This week's assignment asked us to revise an existing script by referencing the doc strings and completing the functions required. The existing script had set variables, menu choice and a while loop created. The main body of the script used a class called 'Processor' and 'I0' with defined functions that I would need to complete to make the main body script work. My first code set parameters to add data to a list of directory rows which was done by appending the list of rows. My next code was to remove a row from the directory list in a file using a *for row in* function with conditional *if* statement. I also completed a code to write data from directory list memory to a file on the hard drive. My last code collected input data using a string input function with *strip* method to remove any leading and trailing character spacing.

I validated my script by running the program in PyCharm to ensure the program was writing to the txt file and no other issues were found. Next, I opened Python in the Windows command shell and entered my directory script location. Once I got a clean pass-through without any errors, I saved the script to my Assignment03 subfolder in the _PythonClass folder on my C: drive. I also added my file to GitHub and posted it for the class on the discussion board.

Summary

Overall the topic of classes and functions makes sense but it did take me some time to understand what concept and how it can be used in Python coding. I thought the introduction of GitHub webpages was exciting and can't wait to do more work in that application. I would also really like to start writing to a spreadsheet from a directory list, hopefully that will happen soon. I thought the doc strings were useful in explaining the intent of the class or function definition and can't wait to write a code from scratch.

```python
@staticmethod
def add_data_to_list(task, priority, list_of_rows):
    """ Adds data to a list of dictionary rows

    :param task: (string) with name of task:
    :param priority: (string) with name of priority:
    :param list_of_rows: (list) you want to add more data to:
    :return: (list) of dictionary rows
    """
    row = {"Task": str(task).strip(), "Priority": str(priority).strip()}
    list_of_rows.append(row)
    return list_of_rows

@staticmethod
def remove_data_from_list(task, list_of_rows):
    """ Removes data from a list of dictionary rows

    :param task: (string) with name of task:
    :param list_of_rows: (list) you want filled with file data:
    :return: (list) of dictionary rows
    """

    for row in list_of_rows:
        if row['Task'].lower() == task.lower():
            list_of_rows.remove(row)
    return list_of_rows

@staticmethod
def write_data_to_file(file_name, list_of_rows):
    """ Writes data from a list of dictionary rows to a File

    :param file_name: (string) with name of file:
    :param list_of_rows: (list) you want filled with file data:
    :return: (list) of dictionary rows
    """
    file = open(file_name,"w")
    for row in list_of_rows:
        file.write(row["Task"] + ',' + row['Priority'] + '\n')
    file.close()
    return list_of_rows
```

Figure 1.0 | Completing Class Processor

```python
@staticmethod
def input_new_task_and_priority():
    """  Gets task and priority values to be added to the list

    :return: (string, string) with task and priority
    """
    pass
    task = str(input('Enter Task Name? - ')).strip()
    priority = str(input('Enter Priority? - ')).strip()
    return task, priority #returning function data
```

Figure 2.0 | Completing Class I0