Name: Coco Jones
Date: February 19, 2023
Class: IT FDN 110 A
Module: 06
GitHub: https://github.com/cocopuffnagoya/IntroToProg-Python-Mod06

Python Functions

Introduction

In module 05, I learned Python Functions. This week's assignment was to create a ToDoList using python dictionaries and functions. I am documenting the knowledge by answering the assignment questions and using the scripts that I create for the assignment where applicable.

What is a function?

A function can group multiple statements into one. To use a function, you first need to define it and you also need to write a script to call the function and execute. **Fig 1** below shows an example of function used in the assignment. The input_menu_choice () function was defined from Line 127 – 134. This function contains a statement to prompt the user to input their choice in Line 132 and a statement to print an extra space in Line 133. And in Line 134, this function returns the user choice as the result of the function. **Fig 2** shows the result of this function. The This is how you can use a function to organize your code.

```
127 def input_menu_choice():
128     """ Gets the menu choice from a user
129
130     :return: string
131     """
132     choice = str(input("Which option would you like to perform? [1 to 4] - ")).strip()
133     print()  # Add an extra line for looks
134     return choice
```

Fig 1 – A function used in the Assignment

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


Please enter a task:
```

Fig 2 – The result of the function shown in Terminal

What are parameters?

You can pass values or parameters into a function for processing. Parameters are variables that are passed into a function. A parameter and an argument are not the same thing. Arguments are values that are passed to a function.

Fig 3 below shows an example in which a function used parameters. The three parameters - task, priority and list_of_rows were passed to the add_data_to_list() function. The three parameters were respectively assigned some values in the previous step in this program. And those values were used for the function to process the data.

```
49  def add_data_to_list(task, priority, list_of_rows):
50      """ Adds data to a list of dictionary rows
51
52      :param task: (string) with name of task:
53      :param priority: (string) with name of priority:
54      :param list_of_rows: (list) you want to add more data to:
55      :return: (list) of dictionary rows
56      """
57      row = {"Task": str(task).strip(), "Priority": str(priority).strip()}
58      # TODO: Add Code Here!
59      list_of_rows.append(row)        # add the dictionary row into list_of_rows
60      return list_of_rows
```

Fig 3 – An example in which parameters are used in a function

What are arguments?

As mentioned above, a parameter and an argument are not the same thing. Arguments are values like strings or integers and could be put into Parameters. In **Fig 4** below, parameters x and y are passed to the mathProgram() function. Into the two parameters, arguments 1, 2 are assigned.

```
>>> def mathProgram(x,y):
...     return x + y
...
>>> mathProgram(1,2)
3
```

Fig 4 – x, y are parameters and 1,2 are arguments

In the Fig 3 example, parameters task and priority were used. The arguments are coming from a prior step. (**Fig 5**) Line 154 and 156 prompts the user to enter a task and a priority. The values that the user inputs will be assigned into the parameters task and priority in Fig 3.

```
149  def input_new_task_and_priority():
150      """  Gets task and priority values to be added to the list
151
152      :return: (string, string) with task and priority
153      """
154      # pass  # TODO: Add Code Here!
155      task = input("Please enter a task: ")       # get a user input
156      priority = input("Please enter a priority: ")       # get a user input
157      row = task, priority       # put the user inputs into a row
158      return row
```

Fig 5 – Arguments are input by the user and later will be called by another function

What is the difference between parameters and arguments?

Both parameters and arguments are listed inside the parentheses when a function is defined. But they are not the same. The difference is that the parameters are variables and arguments are values like strings or integers and arguments are called when the function is executed.

What are return values?

When a function is defined, a result needs to be captured. A return value in the function captures the result. And the return value will be used later in the program. **Fig 6** below shows a return value that captures the KeyToRemove variable in Line 168. The user selects the task that he/she would like to remove in the previous line 167. The user's choice is stored is stored into the variable KeyToRemove. And the variable will be used later in the program.

```
161 def input_task_to_remove():
162     """  Gets the task name to be removed from the list
163
164     :return: (string) with task
165     """
166     # pass  # TODO: Add Code Here!
167     KeyToRemove = input("What task would you like to remove?: ").strip()
168     return KeyToRemove
```

Fig 6 – return value captures into KeyToRemove variable

What is the difference between a global and a local variable?

A global variable is defined at the top of the program and can be used everywhere in the same program. The variables shown in **Fig 7** are all global variables. A local variable is defined within a def block where a function is defined and can be used only in the function. If you try to use the local function outside the function, python errors.

```
# Data ------------------------------------------------------------------- #
# Declare variables and constants
file_name_str = "ToDoFile.txt"  # The name of the data file
file_obj = None  # An object that represents a file
row_dic = {}  # A row of data separated into elements of a dictionary {Task,Priority}
table_lst = []  # A list that acts as a 'table' of rows
choice_str = ""  # Captures the user option selection
```

Fig 7 – Global variables are defined at the top of the script and can be used throughout the script

In **Fig 8** below shows local variable examples in Line 72 and 73 – blnItemRemoved and intRowNumber. Those variables are only used within the same function. If you try to use intRowNumber outside the function, python shows an error message saying intRowNumber is not defined. **Fig 9**

```python
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
    """
    # TODO: Add Code Here!
    blnItemRemoved = False      # Create ItemRemoved Indicator and default it to False
    intRowNumber = 0        # Use the counter to identify the row number to be removed from t

    for row in list_of_rows:        # Loop through list_of_rows
        Task, Priority = row.values()       # Unpack each rows into Task and Priority values
        if Task == task:                    # If the user input Value matches the Task value
            del list_of_rows[intRowNumber]      # then delete the row
            blnItemRemoved = True               # ItemRemoved indicator is turned to True
        intRowNumber += 1           # Each row is assigned a row number
    if blnItemRemoved == True:      # If Item removed flag is True
        print("The task was removed.")      # message is shown that the task was removed from
    else:                           # Otherwise, no row was removed.
        print("Sorry. The task was not found.")
    return list_of_rows
```

Fig 8 – local variable examples

```
Traceback (most recent call last):
  File "C:\_PythonClass\Assignment06\Assigment06_Starter.py", line 193, in <module>
    print(intRowNumber)
NameError: name 'intRowNumber' is not defined
```

Fig 9 – When a local variable intRowNumber is used outside the function, python shows an error

How do you use functions to organize your code?

You can use functions to group multiple statements. You can use the functions later on in the script. By creating functions and grouping many statements, your code will look organized and easier to look at and understand. Another benefit is that you can collapse those functions and make it easier to grasp what is going on. **Fig 10**

```python
# Presentation (Input/Output) -------------------------------
class IO:
    """ Performs Input and Output tasks """

    @staticmethod
    def output_menu_tasks():...

    @staticmethod
    def input_menu_choice():...

    @staticmethod
    def output_current_tasks_in_list(list_of_rows):...

    @staticmethod
    def input_new_task_and_priority():...

    @staticmethod
    def input_task_to_remove():...
```

Fig 10 – Functions collapsed

What is the difference between a function and a class?

The difference between a function and a class is that a function groups multiple related statements and a class groups multiple related functions. In Fig 1 above, we see 2 statements were contained in one function. In the assignment this week, we used two classes. One class is for data processing and contains 4 functions: read_data_from_file, add_data_to_list, remove_data_from_list, write_data_to_file. **Fig 11**

```
27  # Processing  ------------------------------------------------------------
28  class Processor:
29      """  Performs Processing tasks """
30
31      @staticmethod
32      def read_data_from_file(file_name, list_of_rows):...
47
48      @staticmethod
49      def add_data_to_list(task, priority, list_of_rows):...
62
63      @staticmethod
64      def remove_data_from_list(task, list_of_rows):...
86
87      @staticmethod
88      def write_data_to_file(file_name, list_of_rows):...
```

Fig 11 – a class groups multiple related functions

How do functions help you program using the "Separation of Concerns" pattern?

A function groups multiple related statements and a class groups multiple related functions. With use of functions and classes, you can group data related items such as declared variables to the Data section, Data processing functions to the Processing section, data input / output items to the Presentation section. That is how your program can be grouped and organized. **Fig 12** shows the example of Separation of concerns used in the assignment. The program is broken out to four different blocks.

```
# Declare variables and constants
file_name_str = "ToDoFile.txt"  # The name of the data file
file_obj = None  # An object that represents a file
row_dic = {}  # A row of data separated into elements of a dictionary {Task,Priority}
table_lst = []  # A list that acts as a 'table' of rows
choice_str = ""  # Captures the user option selection
                                                                  Define Variables

# Preparation  ------------------------------------------------------------ #

f = open(file_name_str, "w")     # Create a blank text file
f.close()        # Close the file                                  Create File

# Processing  ------------------------------------------------------------ #
class Processor:...                                               Process Data

# Presentation (Input/Output)  ------------------------------------------- #

                                                                  Present Data
class IO:...                                                      Input Output

# Main Body of Script  --------------------------------------------------- #


# Step 1 - When the program starts, Load data from ToDoFile.txt.
Processor.read_data_from_file(file_name=file_name_str, list_of_rows=table_lst)  # read file

# Step 2 - Display a menu of choices to the user
while (True):...
```

Fig 12 – Separations of Concerns with use of Functions and Classes – Data, Processing, I/O

How are the debugging tools used in PyCharm?

There are many debugging functions in PyCharm, but the most basic way of using it is: first click on the line number next to the script that you want to take a close look at. Then right click and select Debug. In **Fig 13** the red dot is called Block Point. That is where debugging is currently performed.

```
21  # Preparation  ----------------------------------------------------------
22
23  f = open(file_name_str, "w")     # Create a blank text file   f: <_io.Tex
24  f.close()        # Close the file
25
```

Fig 13 – Block Point is now at Line 23.

You might need to input some data in the Console to proceed, so you need to go between the Console and Debugger tabs. See **Fig 14.** In this example, the program is waiting for the user to input a value. So provide a value. In this example "1", "Taxes" and "High" is entered.

```
Debugger   Console

        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program


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

Please enter a task: >? Taxes
Please enter a priority: >? High

>>>
```

Fig 14 – Need to go between Debugger and Console tabs to proceed. Input values as needed

To proceed to the next step, click on the Step Over Icon. **Fig 15**

Fig 15 -Step Over

And the debugger moves to the next script. The Debugger tab it shows the values that were just entered. See **Fig 16.** The debugger tab shows the values entered and the data types and those data values were now entered into table_lst.

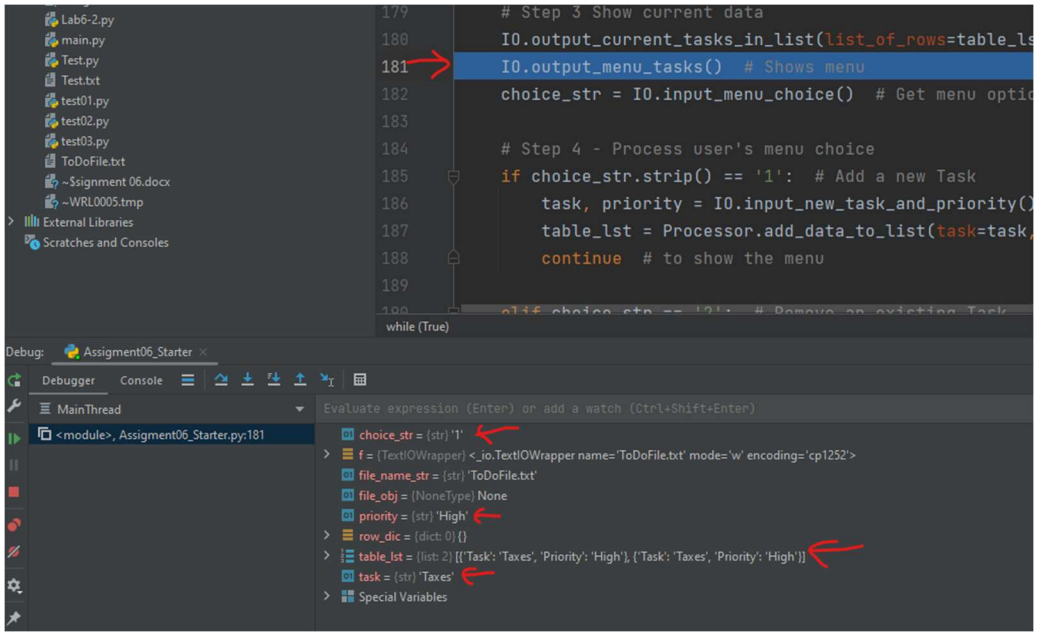


Fig 16 – Debugger tab shows the entered values and other information

There is many more functions than this, but this is how you can execute your program one line by one line to see what is happening in the background in the debugging tool in PyCharm.

What is a GitHub webpage?

A GitHub webpage is a webpage on which you can share your coding work with your team or the world to collaborate. Your webpage can be customized. You can select the theme for your webpage. Guests do not need to login to visit your page. Your contents are published there and accessible.

Summary

In module 06, I learned Python function and class. I documented the knowledge that I acquired this week by answering the assignment questions.

References

https://www.youtube.com/watch?v=P5wOsnPjn6Y Intro to Python Mod06 (Last accessed: 2/20/2023)

Python Programming, Third Edition, Course Technology, Michael Dawson 2019 ISBN-13 978-1-4354-5500-9

Screenshot of Python program:

```
# ------------------------------------------------------------------------------
- #
# Title: Assignment 06
# Description: Working with functions in a class,
#              When the program starts, load each "row" of data
#              in "ToDoToDoList.txt" into a python Dictionary.
#              Add the each dictionary "row" to a python list "table"
# ChangeLog (Who,When,What):
# RRoot,1.1.2030,Created started script
# CJones,2.19.2023,Modified code to complete assignment 06
# ------------------------------------------------------------------------------
- #

# Data ---------------------------------------------------------------------
#
# Declare variables and constants
file_name_str = "ToDoFile.txt"  # The name of the data file
file_obj = None  # An object that represents a file
row_dic = {}  # A row of data separated into elements of a dictionary
{Task,Priority}
table_lst = []  # A list that acts as a 'table' of rows
choice_str = ""  # Captures the user option selection

# Preparation  -------------------------------------------------------------
- #
# Create a blank text file
file_obj = open(file_name_str, "w")     # Create a blank text file
file_obj.close()        # Close the file

# Processing  --------------------------------------------------------------
#
class Processor:
    """  Performs Processing tasks """

    @staticmethod
    def read_data_from_file(file_name, list_of_rows):
        """ Reads data from a file into a list of dictionary rows

        :param file_name: (string) with name of file:
        :param list_of_rows: (list) you want filled with file data:
        :return: (list) of dictionary rows
        """
        list_of_rows.clear()  # clear current data
        file_obj = open(file_name, "r")
        for line in file_obj:
            task, priority = line.split(",")
            row = {"Task": task.strip(), "Priority": priority.strip()}
            list_of_rows.append(row)
        file_obj.close()
        return list_of_rows

    @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()}
        # TODO: Add Code Here!
        list_of_rows.append(row)        # add the dictionary row into
list_of_rows
        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
        """
        # TODO: Add Code Here!
        blnItemRemoved = False      # Create ItemRemoved Indicator and
default it to False
        intRowNumber = 0        # Use the counter to identify the row number
to be removed from the list

        for row_dic in list_of_rows:        # Loop through list_of_rows
            Task, Priority = row_dic.values()       # Unpack each rows into
Task and Priority values
            if Task == task:                    # If the user input Value
matches the Task value
                del list_of_rows[intRowNumber]      # then delete the row
                blnItemRemoved = True               # ItemRemoved indicator
is turned to True
            intRowNumber += 1           # Each row is assigned a row number
        if blnItemRemoved == True:      # If Item removed flag is True
            print("The task was removed.")      # message is shown that the
task was removed from the list
        else:                           # Otherwise, no row was removed.
            print("Sorry. The task was not found.")
        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
        """
        # TODO: Add Code Here!
        file_obj = open(file_name, "w")     # Open the text file in write
mode
        for row_dic in list_of_rows:      # loop through list_of_rows for row
            task, priority = row_dic.values()     # Unpack each row for task
and priority values
            file_obj.write(task+', '+priority+'\n')     # write the values to
the file
```

```
        file_obj.close()        # Close the file
        return list_of_rows

# Presentation (Input/Output)  ------------------------------------------ #
class IO:
    """ Performs Input and Output tasks """

    @staticmethod
    def output_menu_tasks():
        """  Display a menu of choices to the user

        :return: nothing
        """
        print('''
        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program
        ''')
        print()  # Add an extra line for looks

    @staticmethod
    def input_menu_choice():
        """ Gets the menu choice from a user

        :return: string
        """
        choice = str(input("Which option would you like to perform? [1 to 4]
- ")).strip()
        print()  # Add an extra line for looks
        return choice

    @staticmethod
    def output_current_tasks_in_list(list_of_rows):
        """ Shows the current Tasks in the list of dictionaries rows

        :param list_of_rows: (list) of rows you want to display
        :return: nothing
        """
        print("******* The current tasks ToDo are: *******")
        for row_dic in list_of_rows:
            print(row_dic["Task"] + " (" + row_dic["Priority"] + ")")
        print("*****************************************")
        print()  # Add an extra line for looks

    @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  # TODO: Add Code Here!
        task = input("Please enter a task: ")       # get a user input
        priority = input("Please enter a priority: ")       # get a user
input
        row = task, priority       # put the user inputs into a row
```

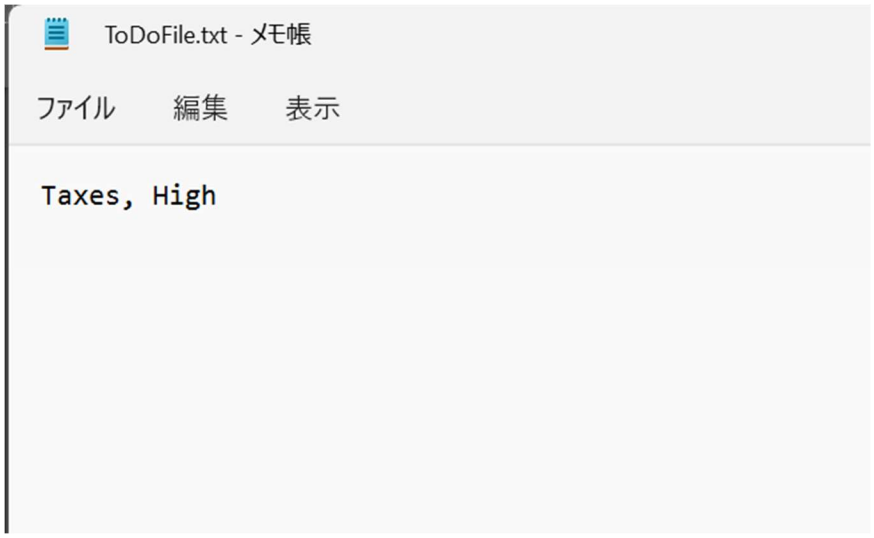
```
        return row

    @staticmethod
    def input_task_to_remove():
        """  Gets the task name to be removed from the list

        :return: (string) with task
        """
        # pass  # TODO: Add Code Here!
        KeyToRemove = input("What task would you like to remove?: ").strip()
# get the task name to be removed
        return KeyToRemove

# Main Body of Script  ---------------------------------------------------------
#
# Step 1 - When the program starts, Load data from ToDoFile.txt.
Processor.read_data_from_file(file_name=file_name_str,
list_of_rows=table_lst)  # read file data

# Step 2 - Display a menu of choices to the user
while (True):
    # Step 3 Show current data
    IO.output_current_tasks_in_list(list_of_rows=table_lst)  # Show current
data in the list/table
    IO.output_menu_tasks()  # Shows menu
    choice_str = IO.input_menu_choice()  # Get menu option

    # Step 4 - Process user's menu choice
    if choice_str.strip() == '1':  # Add a new Task
        task, priority = IO.input_new_task_and_priority()
        table_lst = Processor.add_data_to_list(task=task, priority=priority,
list_of_rows=table_lst)
        continue  # to show the menu

    elif choice_str == '2':  # Remove an existing Task
        task = IO.input_task_to_remove()
        table_lst = Processor.remove_data_from_list(task=task,
list_of_rows=table_lst)
        continue  # to show the menu

    elif choice_str == '3':  # Save Data to File
        table_lst = Processor.write_data_to_file(file_name=file_name_str,
list_of_rows=table_lst)
        print("Data Saved!")
        continue  # to show the menu

    elif choice_str == '4':  # Exit Program
        print("Goodbye!")
        break  # by exiting loop
```

Copy of Terminal:

```
C:\_PythonClass\Assignment06\venv\Scripts\python.exe
C:\_PythonClass\Assignment06\Assigment06.py
******* The current tasks ToDo are: *******
*****************************************


    Menu of Options
    1) Add a new Task
    2) Remove an existing Task
    3) Save Data to File
    4) Exit Program


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

Please enter a task: Taxes
Please enter a priority: High
******* The current tasks ToDo are: *******
Taxes (High)
*****************************************


    Menu of Options
    1) Add a new Task
    2) Remove an existing Task
    3) Save Data to File
    4) Exit Program


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

Please enter a task: Go to Costco
Please enter a priority: Low
******* The current tasks ToDo are: *******
Taxes (High)
Go to Costco (Low)
*****************************************


    Menu of Options
    1) Add a new Task
    2) Remove an existing Task
    3) Save Data to File
    4) Exit Program
```

```
Which option would you like to perform? [1 to 4] - 2

What task would you like to remove?: Go to Costco
The task was removed.
******* The current tasks ToDo are: *******
Taxes (High)
*****************************************


    Menu of Options
    1) Add a new Task
    2) Remove an existing Task
    3) Save Data to File
    4) Exit Program


Which option would you like to perform? [1 to 4] - 3

Data Saved!
******* The current tasks ToDo are: *******
Taxes (High)
*****************************************


    Menu of Options
    1) Add a new Task
    2) Remove an existing Task
    3) Save Data to File
    4) Exit Program


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

Goodbye!

Process finished with exit code 0
```

File Saved:



Command Prompt:

```
Command Prompt

C:\_PythonClass\Assignment06>python "C:\_PythonClass\Assignment06\Assigment06.py"
******* The current tasks ToDo are: *******
*******************************************


        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program


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

Please enter a task: Walk Dog
Please enter a priority: High
******* The current tasks ToDo are: *******
Walk Dog (High)
*******************************************


        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program


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

Please enter a task: Feed Cat
Please enter a priority: Low
******* The current tasks ToDo are: *******
Walk Dog (High)
Feed Cat (Low)
*******************************************


        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program
```

```
Which option would you like to perform? [1 to 4] - 2

What task would you like to remove?: Walk Dog
The task was removed.
******* The current tasks ToDo are: *******
Feed Cat (Low)
*****************************************


        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program


Which option would you like to perform? [1 to 4] - 3

Data Saved!
******* The current tasks ToDo are: *******
Feed Cat (Low)
*****************************************


        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program


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

Goodbye!

C:\_PythonClass\Assignment06>
```

File saved: