Name: Natta Panapornsirikun
Date: 05/20/2023
Course: IT FDN 110 A: Foundation Of Programming Python
Assignment 06
https://github.com/nattpana/NattClassFile https://nattpana.github.io/ITFnd100-Mod06/

Assignment06 - Creating scripts using Functions

Introduction

In this chapter, I'm expected to learn how to use a parameter, which acts like a bucket that holds inputs and passes them to the output. Additionally, I will learn that functions can be reused and help organize the code. Once defined, functions can be reused multiple times from different parts of the code, making it easier to maintain and update the logic. It also helps implement the "Separation of Concerns" pattern by separating different tasks or concerns into modular units, making the code more focused and easier to understand. When I call functions, I will pass arguments to them. By using the return statement, a function provides output or information by returning a value.

A class differs from a function as it serves as a blueprint or template for creating objects. It defines the structure, behavior, and properties of objects, allowing them to hold data and perform operations.

I also expected to learn about the difference between local and global variables. A local variable is declared inside a function and can only be accessed within that function. On the other hand, a global variable is declared outside any function and can be accessed from any part of the code.

In the PyCharm integrated development environment (IDE) for Python programming, there are various debugging tools available. These include breakpoints, stepping through code, variable inspection, call stack viewing, and error message analysis, which assist in identifying and resolving issues in code.

GitHub enables the publishing of static web content (HTML, CSS, JavaScript) and facilitates sharing projects, documentation, personal websites, and more directly from the repository.

Practice Functions

Execute statements by calling functions

After defining the function and when call it, it will execute all processes under that function

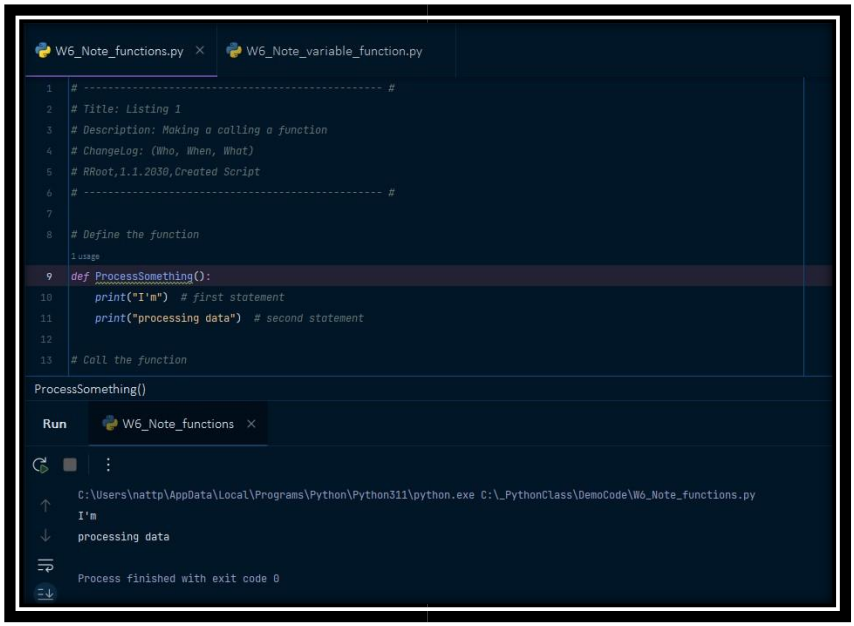


Figure 1: Example defined and called function.

Passing arguments in a variable through the functions.

value1 and *value2* are the variables that use for passing as an argument to a function called *Addvalues*

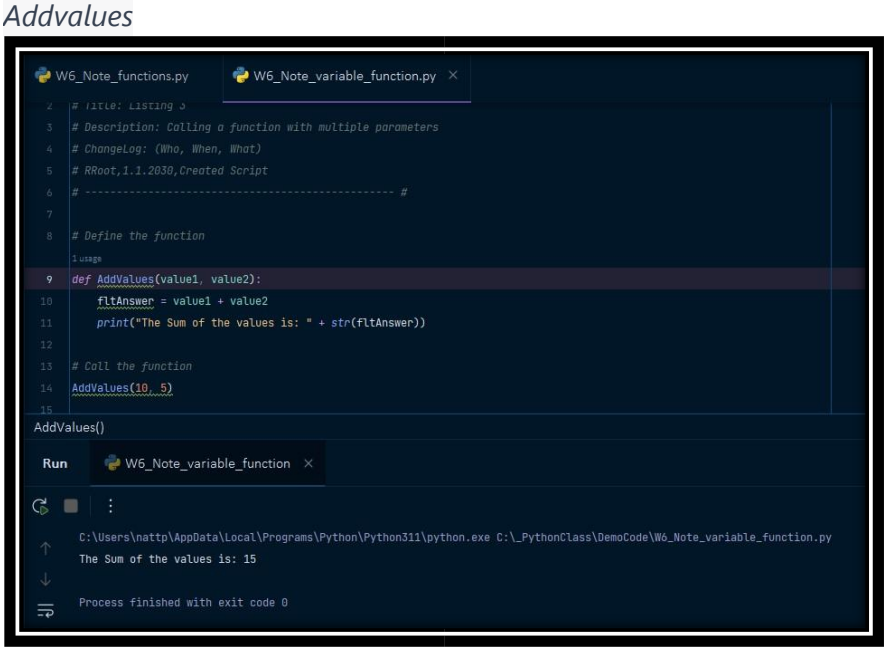


Figure 2: Example of passing variable through arguments to function.

Exercise Lab 6 -1

Create and call 4 functions to do the +, -, *, and / operations.

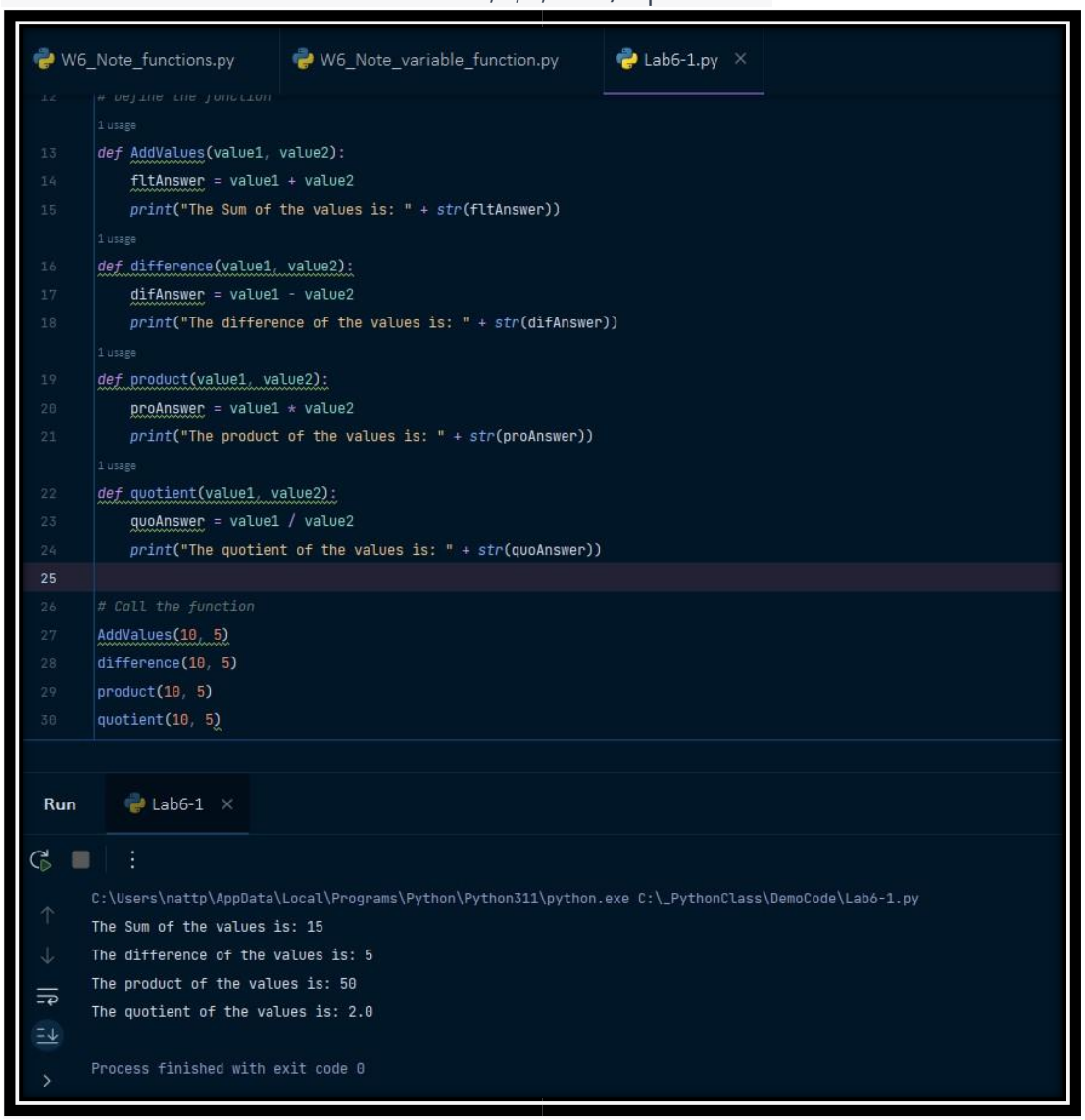


Figure 3: Lab6-1 call function to do operations and results.

Now I 'm working with the exercise that uses variables as Arguments, in this code is using "Separation of Concerns" In This example, this exercise starts from

1) Data Code part: identify the variable to use in the program
2) Processing Code: define the function > process function > print output
3) Presentation part separated in

a. Input from users to value 1 and value2
b. Output from call function Addvalue

Figure 4: Example to get input data to a variable and pass through called function.

When I don't use the return value and try to use the variable inside the function directly, the data show as None but when I use the return, I can use that variable from inside the function at the outside function as well.

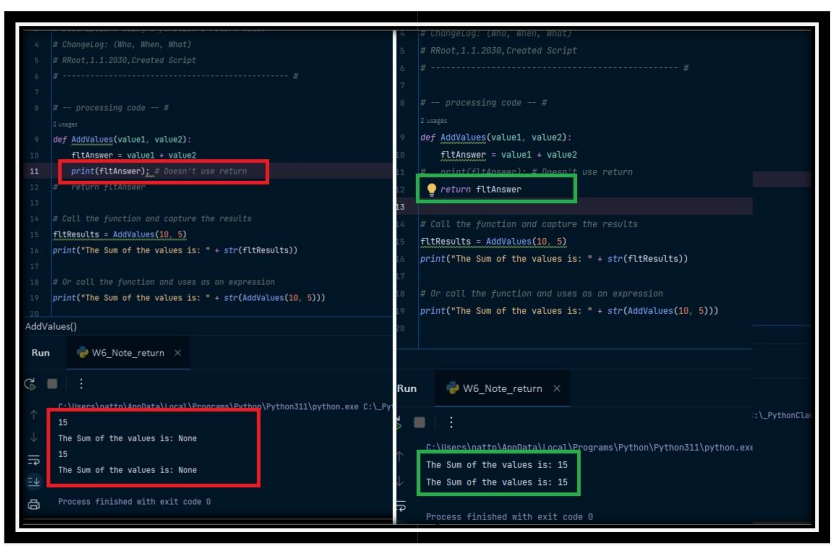


Figure 5. Show the difference between not call returns and call returns.

I learned that return can work with multiple variables in this program it assigns a new variable to the multiple variable that returns from the function Addvalue

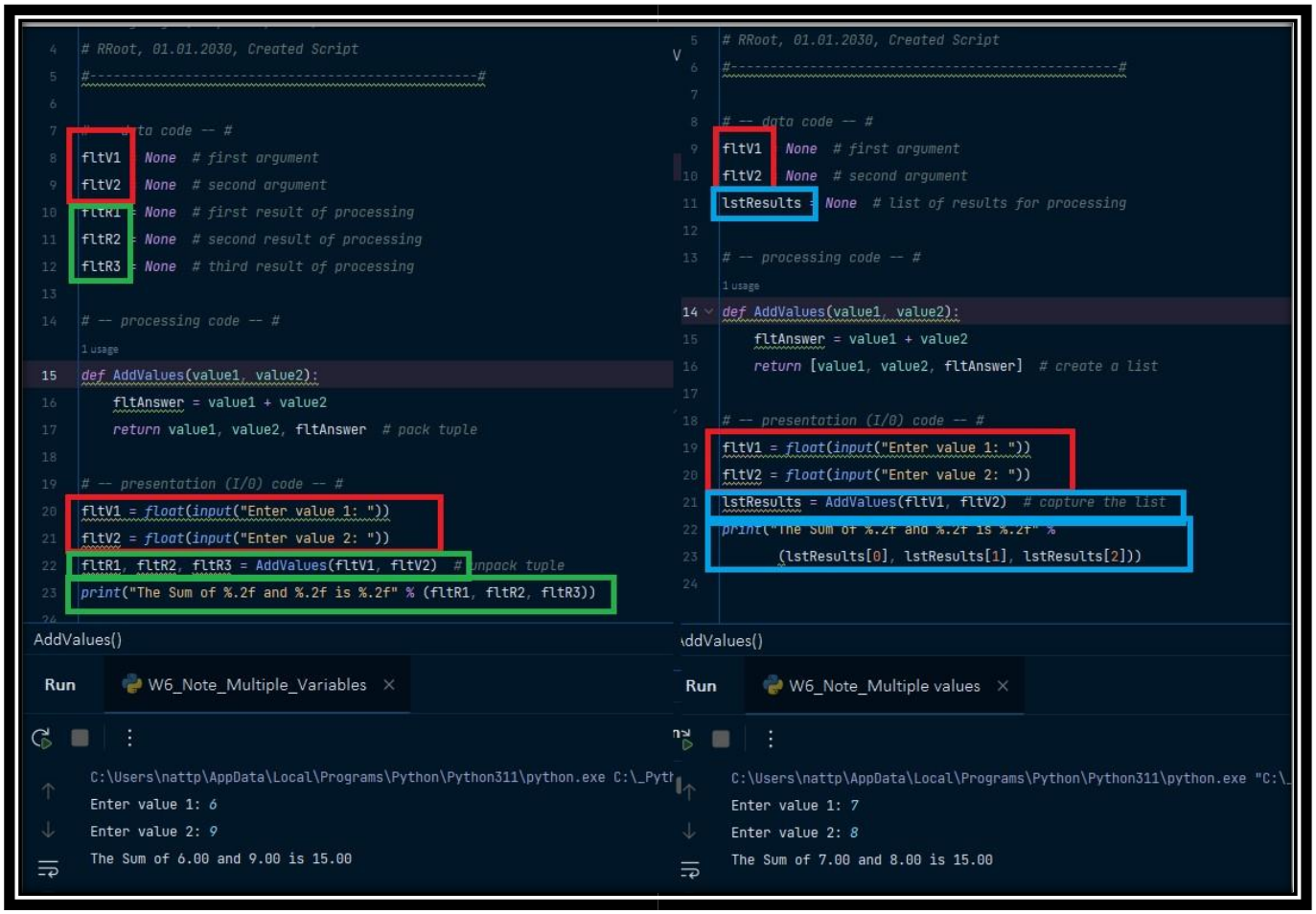


Figure 6. The results of multi-variable returns can use 3 variables to hold the data (on the left) or hold the return value in the list (on the right)

Exercise Lab 6 -2

I did this exercise by using 2 variables to get input data and using 4 functions to process the code

```python
# -- processing code -- #
def sumValues(value1, value2):
    sumAnswer = value1 + value2
    return [value1, value2, sumAnswer]  # create a list
def diffValues(value1, value2):
    diffAnswer = value1 - value2
    return [value1, value2, diffAnswer]  # create a list
def proValues(value1, value2):
    proAnswer = value1 * value2
    return [value1, value2, proAnswer]  # create a list
def quoValues(value1, value2):
    quoAnswer = value1 / value2
    return [value1, value2, quoAnswer]  # create a list

# -- presentation (I/O) code -- #
fltV1 = float(input("Enter value 1: "))
fltV2 = float(input("Enter value 2: "))

sumResults = sumValues(fltV1, fltV2)  # capture the list
diffResults = diffValues(fltV1, fltV2)  # capture the list
proResults = proValues(fltV1, fltV2)  # capture the list
quoResults = quoValues(fltV1, fltV2)  # capture the list
```

Figure 7. Lab 6-2 using the list to store and present data when calling function part 1

```python
sumResults = sumValues(fltV1, fltV2)  # capture the list
diffResults = diffValues(fltV1, fltV2)  # capture the list
proResults = proValues(fltV1, fltV2)  # capture the list
quoResults = quoValues(fltV1, fltV2)  # capture the list

print("The Sum of %.2f and %.2f is %.2f" %
      (sumResults[0], sumResults[1], sumResults[2]))

print("The Diff of %.2f and %.2f is %.2f" %
      (diffResults[0], diffResults[1], diffResults[2]))

print("The Pro of %.2f and %.2f is %.2f" %
      (proResults[0], proResults[1], proResults[2]))

print("The Quo of %.2f and %.2f is %.2f" %
      (quoResults[0], quoResults[1], quoResults[2]))
```

Run Lab 6_2

```
C:\Users\nattp\AppData\Local\Programs\Python\Python311\python.exe "C:\_PythonCla
Enter value 1: 4
Enter value 2: 7
The Sum of 4.00 and 7.00 is 11.00
The Diff of 4.00 and 7.00 is -3.00
The Pro of 4.00 and 7.00 is 28.00
The Quo of 4.00 and 7.00 is 0.57

Process finished with exit code 0
```

Figure 8. Lab 6-2 using the list to store and present data when calling function part 2

Working with Positional and Name Arguments

It is very convenient to name parameters in arguments.

```python
# -- data code -- #
fltV1 = None  # first argument
fltV2 = None  # second argument
fltR1 = None  # first result of processing
fltR2 = None  # second result of processing
fltR3 = None  # third result of processing

# -- processing code -- #
def AddValues(value1, value2):
    fltAnswer = value1 + value2
    return value1, value2, fltAnswer

# -- presentation (I/O) code -- #
fltV1 = float(input("Enter value 1: "))
fltV2 = float(input("Enter value 2: "))
fltR1, fltR2, fltR3 = AddValues(value1 = fltV1, value2 = fltV2)
print("The Sum of %.2f and %.2f is %.2f" % (fltR1, fltR2, fltR3))
```

```
Run    W6_Note_positional_name_arguments
C:\Users\nattp\AppData\Local\Programs\Python\Python311\python.exe C:\_Py
Enter value 1: 4
Enter value 2: 6
The Sum of 4.00 and 6.00 is 10.00

Process finished with exit code 0
```

Figure 9. Name parameters in arguments.

```python
# -- data code -- #
fltV1 = None  # first argument
fltV2 = None  # second argument
fltR1 = None  # first result of processing
fltR2 = None  # second result of processing
fltR3 = None  # third result of processing

# -- processing code -- #
def AddValues(value1 = 0, value2 = 0):
    fltAnswer = value1 + value2
    return value1, value2, fltAnswer

# -- presentation (I/O) code -- #
fltV1 = float(input("Enter value 1: "))
fltV2 = float(input("Enter value 2: "))
fltR1, fltR2, fltR3 = AddValues(value2 = fltV2)
print("The Sum of %.2f and %.2f is %.2f" % (fltR1, fltR2, fltR3))
```

```
w6_Note_Default parameter value
C:\Users\nattp\AppData\Local\Programs\Python\Python311\python.exe "C:\_Py
Enter value 1: 5
Enter value 2: 8
The Sum of 0.00 and 8.00 is 8.00

Process finished with exit code 0
```

Figure 10. Name parameters in arguments.

I found out in the next step that I can use overload which makes my code simpler by using the function with if_elif, I can use two variables and identify the variable with the condition of each operation through the arguments.

```python
# -- processing code -- #
4 usages
def CalcValues(value1 = 0, value2 = 0, operation = '+'):
    if operation.lower() == '+': fltAnswer = value1 + value2
    elif operation.lower() == '-': fltAnswer = abs(value1 - value2)
    elif operation.lower() == '*': fltAnswer = value1 * value2
    elif operation.lower() == '/': fltAnswer = value1 / value2
    else: fltAnswer = "Error"
    return value1, operation, value2, fltAnswer

# -- presentation (I/O) code -- #
print(CalcValues())
print(CalcValues(5,10))
print(CalcValues(5,10,'*'))
print(CalcValues(5,10,'/'))
```

```
Run    W6_Note_overload
C:\Users\nattp\AppData\Local\Programs\Python\Python311\python.exe C:\_
(0, '+', 0, 0)
(5, '+', 10, 15)
(5, '*', 10, 50)
(5, '/', 10, 0.5)

Process finished with exit code 0
```

Figure 11. example of overload

I learned to make the code neater by using the “is None” to prevent any blank input data.

```python
def CalcValues(value1 = None, value2 = None, operation = None):
    if value1 is None or value2 is None or operation is None:
        fltAnswer = "Error: Missing Arguments"
    elif operation.lower() == '+': fltAnswer = value1 + value2
    elif operation.lower() == '-': fltAnswer = abs(value1 - value2)
    elif operation.lower() == '*': fltAnswer = value1 * value2
    elif operation.lower() == '/': fltAnswer = value1 / value2
    else: fltAnswer = "Error"
    return value1, operation, value2, fltAnswer

# -- presentation (I/O) code -- #
print(CalcValues())
print(CalcValues(5, 10))
print(CalcValues(5, 10, '*'))
```

```
Run    W6_Note_None_Key_Word
C:\Users\nattp\AppData\Local\Programs\Python\Python311\python.exe C:\_PythonClass\DemoCode\W6_Note_None_Key_Word.py
(None, None, None, 'Error: Missing Arguments')
(5, None, 10, 'Error: Missing Arguments')
(5, '*', 10, 50)

Process finished with exit code 0
```

Figure 11. example of “is None”

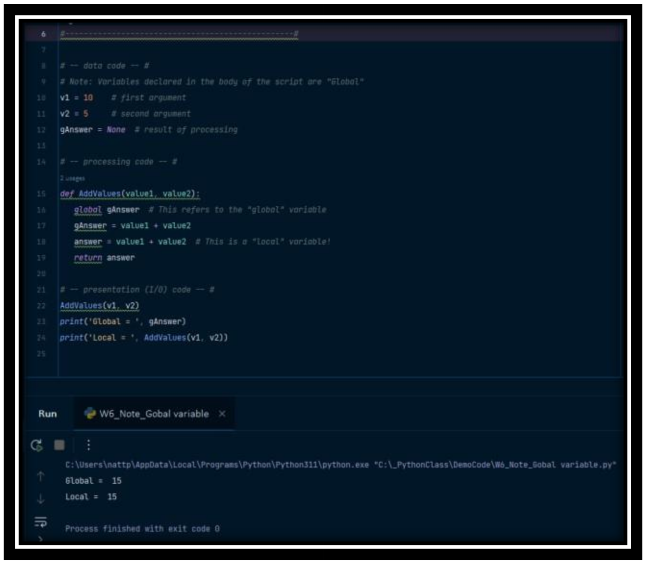
I can call functions through both list and dictionary, for list it will be based on the position of data which starts from 0, for the dictionary, I can use short words to name in the arguments and make it easier to understand and follow.

Figure 12. Compare results calling a function on a list and dictionary

I learned that global parameters should be assigned at the top of the program so that they can be accessed and used throughout the entire program. These global parameters have a broader scope and can be utilized in any part of the program. The global parameter will take priority over all local parameter that has the same name, which calls the shadowing global variable. For better results, we should not use global variation inside the function.

On the other hand, local parameters are assigned within specific functions or blocks of code and are only accessible within that specific function or block. They have a narrower scope and are used exclusively for that internal function. This practice helps ensure proper organization and separation of variables in a program, making it easier to understand and maintain.

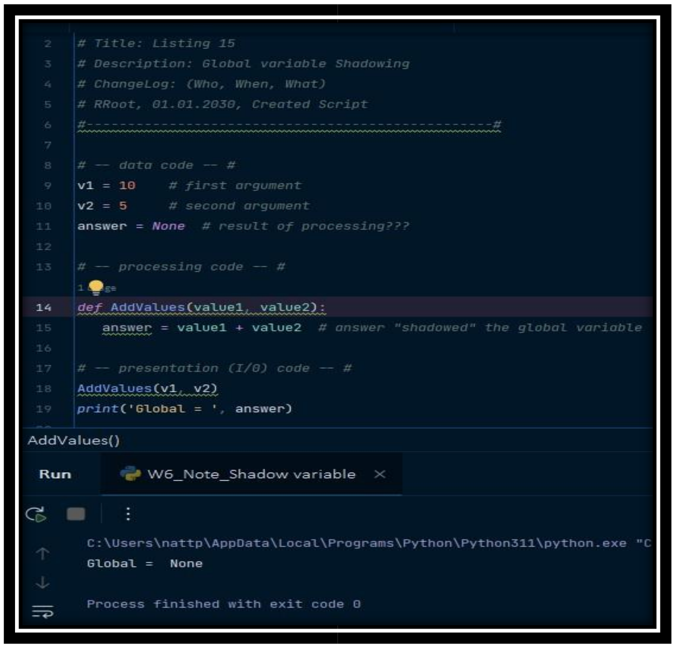
Docstrings, short for documentation strings, are used in programming languages, particularly in Python, to provide a description and documentation for functions, modules, classes, or methods. They are enclosed within triple quotes (""") and are placed at the beginning of the entity they are documenting.



```python
#---------------------------------------------------#

# -- data code -- #
# Note: Variables declared in the body of the script are "Global"
v1 = 10    # first argument
v2 = 5     # second argument
gAnswer = None  # result of processing

# -- processing code -- #
def AddValues(value1, value2):
   global gAnswer  # This refers to the "global" variable
   gAnswer = value1 + value2
   answer = value1 + value2  # This is a "local" variable!
   return answer

# -- presentation (I/O) code -- #
AddValues(v1, v2)
print('Global = ', gAnswer)
print('Local = ', AddValues(v1, v2))
```

```
C:\Users\nattp\AppData\Local\Programs\Python\Python311\python.exe "C:\_PythonClass\DemoCode\W6_Note_Gobal variable.py"
Global =  15
Local =  15

Process finished with exit code 0
```

Figure 13. Compare global parameter and local parameter



```python
# Title: Listing 15
# Description: Global variable Shadowing
# ChangeLog: (Who, When, What)
# RRoot, 01.01.2030, Created Script
#--------------------------------------------------#

# -- data code -- #
v1 = 10    # first argument
v2 = 5     # second argument
answer = None  # result of processing???

# -- processing code -- #
def AddValues(value1, value2):
   answer = value1 + value2  # answer "shadowed" the global variable

# -- presentation (I/O) code -- #
AddValues(v1, v2)
print('Global = ', answer)
```

```
C:\Users\nattp\AppData\Local\Programs\Python\Python311\python.exe "C
Global =  None

Process finished with exit code 0
```

Figure 14. Shadowing global parameter; when it is the same name

The global parameter takes priority over the local parameter

Lab 6_3

Practice using the functions from the Mathprocessor class to perform addition (+), subtraction (-), multiplication (*), and division (/). I noticed that we don't need to add any operations as they are already predefined within the Mathprocessor class. To call the function, you only need to pass value1 and value2

as arguments. The code format will have Mathprocessor in front of the function name and use "." to separate them.

```python
# RRoot, 01.01.2030, Created Script
# ----------------------------------------------Z
#LAB 6_3
#7.Create class called MathProcessor
class MathProcessor:
    # 3. Add a function called AddValues to process the sum of two values and return the result as a float.
    def AddValues(self, value1=0.0, value2=0.0):
        return float(value1 + value2)

    # 4. Add a function called SubtractValues to process the difference of two values and return the result as a floa
    def SubtractValues(self, value1=0.0, value2=0.0):
        return float(value1 - value2)

    # 5. Add a function called MultiplyValues to process the product of two values and return the result as a float.
    def MultiplyValues(self, value1=0.0, value2=0.0):
        return float(value1 * value2)

    # 6. Add a function called DivideValues to process the quotient of two values and return the result as a float.
    def DivideValues(self, value1=0.0, value2=0.0):
        return float(value1 / value2)


# 7. Add code to capture two values from a user.
print("Please choose the operation:")
print("Choice 1: AddValues")
print("Choice 2: SubtractValues")
print("Choice 3: MultiplyValues")
print("Choice 4: DivideValues")
```

Figure 15. Lab 6_3 Lean to use Mathprocessor class part 1

```python
choice = int(input("Enter your choice: "))
value1 = float(input("Please enter the first value: "))
value2 = float(input("Please enter the second value: "))

math_processor = MathProcessor()

if choice == 1:
    result = math_processor.AddValues(value1, value2)
elif choice == 2:
    result = math_processor.SubtractValues(value1, value2)
elif choice == 3:
    result = math_processor.MultiplyValues(value1, value2)
elif choice == 4:
    result = math_processor.DivideValues(value1, value2)
else:
    result = "Invalid choice"

print("Result:", result)
```

```
Run    Lab 6_3
C:\Users\nattp\AppData\Local\Programs\Python\Python311\python.exe "C:\_PythonClass\DemoCode\Lab 6_3.py"
Please choose the operation:
Choice 1: AddValues
Choice 2: SubtractValues
Choice 3: MultiplyValues
Choice 4: DivideValues
Enter your choice: 1
Please enter the first value: 4
Please enter the second value: 5
Result: 9.0
```

Figure 16. Lab 6_3 Lean to use Mathprocessor class part 2

Debug

When I click to assign the breakpoints in front of the line, when I ran until that break line it will stop and show the details parameter at that step.

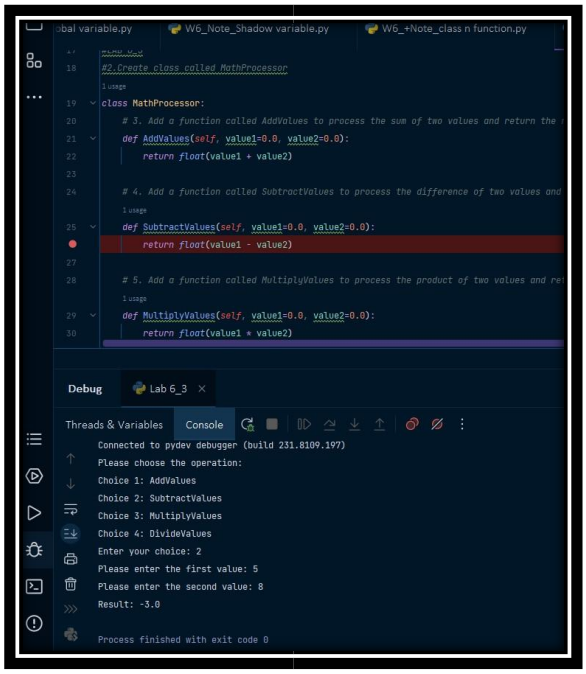


Figure 17. Example of Debug Part 1

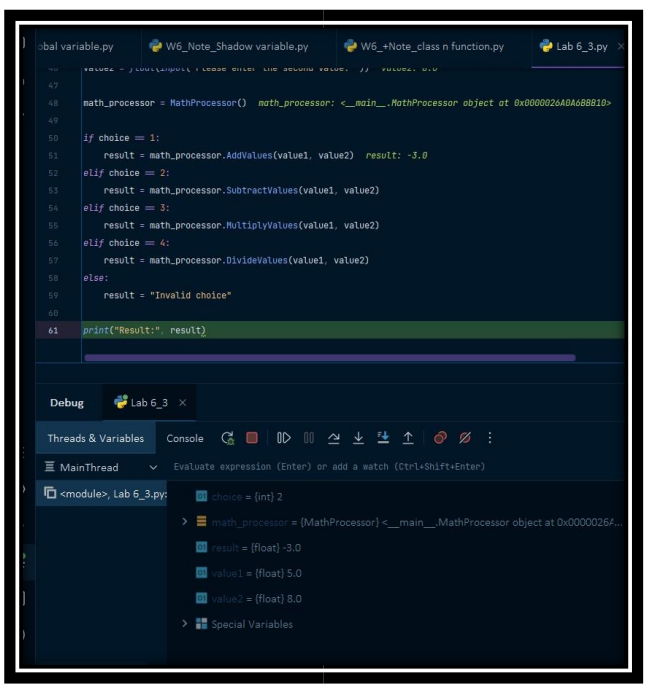


Figure 18. Example of Debug Part 2

GitHub

I have created a folder and a new file, I have an issue when trying to do the configuration, it's not as same as in the exercise, and I'm not able to find the step of drop-down source https://nattpana.github.io/ITFnd100-Mod06/

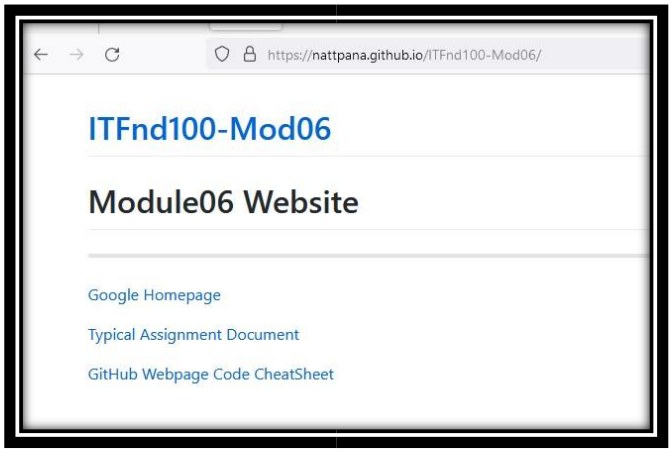


Figure 19. Example of building Github

Assignment 06

This assignment has the precoding and the focus is to add some coding to complete the pre-coding. Assignment06 wants me to be able to use the choice to be able to 1) Add a new Task 2)Remove an existing task 3) save Data in the file (name ToDoFile.txt) into the Python dictionary.

```
# ------------------------------------------------------------------------------- #
# Title: Assignment 06_Natta
# Description: Working with functions in a class,
#              When the program starts, load each "row" of data
#              in "ToDoFile.txt" into a python Dictionary.
#              Add each dictionary "row" to a python list "table"
# ChangeLog (Who,When,What):(Natta Panapornsirikun , 05/24/2023, Filling the missing code)
# RRoot,1.1.2030,Created started script
# <Natta Panapornsirikun>,<05/24/2023>,Modified code to complete assignment 06
# ------------------------------------------------------------------------------- #

# Data ------------------------------------------------------------------------- #
# Declare variables and constants
ToDoFile_str = "ToDoFile.txt"  # The name of the data 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


# 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
        """

```

Figure20. Assignment 06 code PART1

```python
        :param list_of_rows: (list) you want filled with file data:
        :return: (list) of dictionary rows
        """

        list_of_rows.clear()  # clear current data
        try:
            file = open(file_name, "r")  # load each "row" of data
            for line in file:
                task, priority = line.strip().split(",")
                row = {"Task": task, "Priority": priority}
                list_of_rows.append(row)
            file.close()
        except FileNotFoundError:
            print("File not found. Creating a new file.")
        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()}
        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

```

Figure21. Assignment 06 code PART2

```python

        :param task: (string) with name of task:
        :param list_of_rows: (list) you want filled with file data:
        :return: (list) of dictionary rows
        """
        list_of_rows = [row for row in list_of_rows if row["Task"].lower() != task.lower()]
        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")  # load each "row" of data
        for row in list_of_rows:
            task = row["Task"]
            priority = row["Priority"]
            file.write(f"{task},{priority}\n")
        file.close()
        return list_of_rows


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

```

Figure22. Assignment 06 code PART3

```python

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) Please add a new Task
        2) Please remove an existing Task
        3) Please 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):
```

Figure23. Assignment 06 code PART4

```python
    @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 in list_of_rows:
            print(row["Task"] + " (" + row["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
        """
        task = str(input("Enter Task? - ")).strip()
        priority = str(input("Enter priority? - ")).strip()
        return task, priority

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

        :return: (string) with task
        """
        task = str(input("Enter Task you want to delete: - ")).strip()
        return task

```

Figure24. Assignment 06 code PART5

```python
# Main Body of Script  ---------------------------------------------------- #

# Step 1 - When the program starts, Load data from ToDoFile_str.
Processor.read_data_from_file(ToDoFile_str, 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=ToDoFile_str, list_of_rows=table_lst)
        print("Data Saved!")
        continue  # to show the menu

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

Figure25. Assignment 06 code PART6

```
C:\Users\nattp\AppData\Local\Programs\Python\Python311\python.exe
File not found. Creating a new file.
******* The current tasks ToDo are: *******
*******************************************


        Menu of Options
        1) Please add a new Task
        2) Please remove an existing Task
        3) Please Save Data to File
        4) Exit Program


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

Enter Task? - swim
Enter priority? - 2
******* The current tasks ToDo are: *******
swim (2)
*******************************************


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

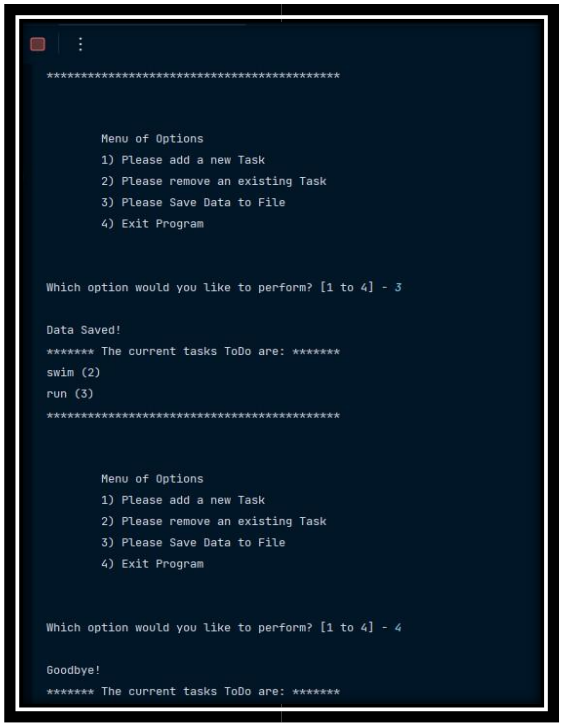
Figure26. Assignment 06 Results PART1

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

Enter Task? - read
Enter priority? - 1
******* The current tasks ToDo are: *******
swim (2)
read (1)
*****************************************


        Menu of Options
        1) Please add a new Task
        2) Please remove an existing Task
        3) Please Save Data to File
        4) Exit Program


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

Enter Task? - run
Enter priority? - 3
******* The current tasks ToDo are: *******
swim (2)
read (1)
run (3)
*****************************************
```

Figure27. Assignment 06 Results PART2

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


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

Enter Task you want to delete: - 2
******* The current tasks ToDo are: *******
swim (2)
read (1)
run (3)
*****************************************


        Menu of Options
        1) Please add a new Task
        2) Please remove an existing Task
        3) Please Save Data to File
        4) Exit Program


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

Enter Task you want to delete: - read
******* The current tasks ToDo are: *******
swim (2)
run (3)
```

Figure28. Assignment 06 Results PART3



```
*******************************************


        Menu of Options
        1) Please add a new Task
        2) Please remove an existing Task
        3) Please 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: *******
swim (2)
run (3)
*******************************************


        Menu of Options
        1) Please add a new Task
        2) Please remove an existing Task
        3) Please Save Data to File
        4) Exit Program


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

Goodbye!
******* The current tasks ToDo are: *******
```

Figure29. Assignment 06 Results PART4

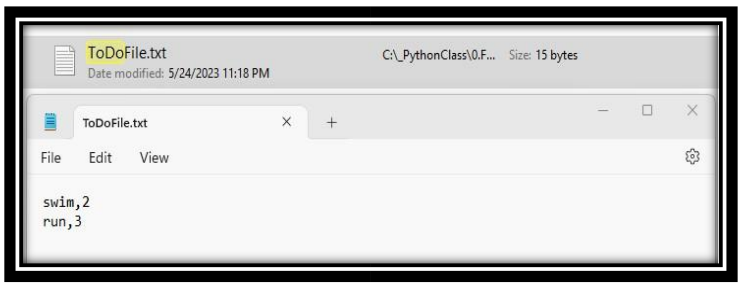


Figure30. Assignment 06 Results PART5 in To do list text file

```
C:\WINDOWS\system32\cmd.
Natta.py"
******* The current tasks ToDo are: *******
swim (2)
run (3)
*****************************************

        Menu of Options
        1) Please add a new Task
        2) Please remove an existing Task
        3) Please Save Data to File
        4) Exit Program

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

Enter Task? - Read
Enter priority? - 1
******* The current tasks ToDo are: *******
swim (2)
run (3)
Read (1)
*****************************************

        Menu of Options
        1) Please add a new Task
        2) Please remove an existing Task
        3) Please Save Data to File
        4) Exit Program

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

Enter Task? - SWIM
Enter priority? - 2
******* The current tasks ToDo are: *******
swim (2)
run (3)
Read (1)
SWIM (2)
*****************************************

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

Figure31. Assignment 06 Results in bat file

Summary

I have learned about functions, parameters, return statements, classes, local and global variables, debugging tools, and GitHub. These concepts will help me organize my code, promote reusability, handle inputs and outputs, define object structures, debug my code, and facilitate project management and sharing.