

Traci McGrew

Randall Root

IT FDN 110 A

Assignment 06 due 5/22/24

<https://github.com/tmcgrew/IntroToProg-Python-Mod06>

Functions

Introduction

In this paper I will go over how I completed creating a Python program like the program in Assignment05, but which also demonstrates using functions, classes, and the separation of concerns pattern. I will also touch on some concepts from the module, and my observations while doing the assignments in the module.

Functions

Functions make code more reusable and portable. Example of a simple function I wrote that takes *name* as a parameter and passes in the value from user's input to *first_name* as an argument is shown in figure 1. In this case, the value is "Traci" as shown in figure 2 running in PyCharm.

```

# -- data -- #
first_name: str = ""

first_name = input("What is your first name? ")

# -- processing and presentation-- #
1 usage
def greeting(name):
    '''Display a simple greeting to the user'''
    print(f"Hello, {name}!")

# call the function
greeting(first_name)

```

Figure 1 Simple function in Pycharm

```

What is your first name? Traci
Hello, Traci!

Process finished with exit code 0
|

```

Figure 2 Simple module calling a function running in PyCharm

Functions with return values make it easier to break code into separations of concern. You can more easily separate code into data, processing, and presentation sections. Figure 3 shows the function reworked to separate the data, processing, and presentation layers and

therefore show better separation of concerns. Figure 4 shows it running in PyCharm and that we get same result that was in figure 2.

```
# -- data -- #
first_name: str = ""

first_name = input("What is your first name? ")

# -- processing -- #
1 usage
def greeting(name):
    '''Return a simple greeting to the user'''
    # print(f"Hello, {name}!")
    return f"Hello, {name}!"

# -- presentation (I/O) -- #

# call the function
print(greeting(first_name))
```

Figure 3 Simple module calling a function running in Pycharm reworked to use better separations of concerns

```
/Users/tracimcgrew/Desktop/UW/IT_FDI
What is your first name? Traci
Hello, Traci!

Process finished with exit code 0
|
```

Figure 4 Simple module calling a function running in PyCharm same result as figure 2

Parameters are positional when you have more than one (like if we took first and last name as an argument) and can be named. You can also put in default values for the arguments so if none provided in the function call it uses the default. Figure 5 shows the function reworked to have a default. Figure 6 shows it running in PyCharm. The first function call is called without anything so it uses the default value. The second call passes in an argument.

```
9      # -- data -- #
10     first_name: str = ""
11
12     first_name = input("What is your first name? ")
13
14
15     # -- processing -- #
16     2 usages
17     def greeting(name = "You"):
18         '''Return a simple greeting to the user'''
19         # print(f"Hello, {name}!")
20         return f"Hello, {name}!"
21
22     # -- presentation (I/O) -- #
23
24     # call the function
25     print(greeting())
26     print(greeting(first_name))
27
```

Figure 5 Simple module with a function reworked to have a default value for its parameter shown in PyCharm

```

/Users/tracimcgreww/Desktop/UW/IT_FDN
What is your first name? Traci
Hello, You!
Hello, Traci!

Process finished with exit code 0

```

Figure 6 Simple module calling a function running in PyCharm showing a default argument call and one with an argument passed in

Global vs. Local Variables

Variables declared outside of a function like *first_name* is declared on line 10 of figure 5, is a global variable. However, variables declared inside a function are local to that function. Global variables can be used both inside and outside the function. Global variables can be reassigned in a function. However, local variables in a function must be returned to have the value used outside the function. You should also be careful naming a local variable same as a global. Python will possibly confuse the two and think you are ‘shadowing’ the global. If using a global variable inside a function and reassigning it, you should put *global* in front of it.

Docstring

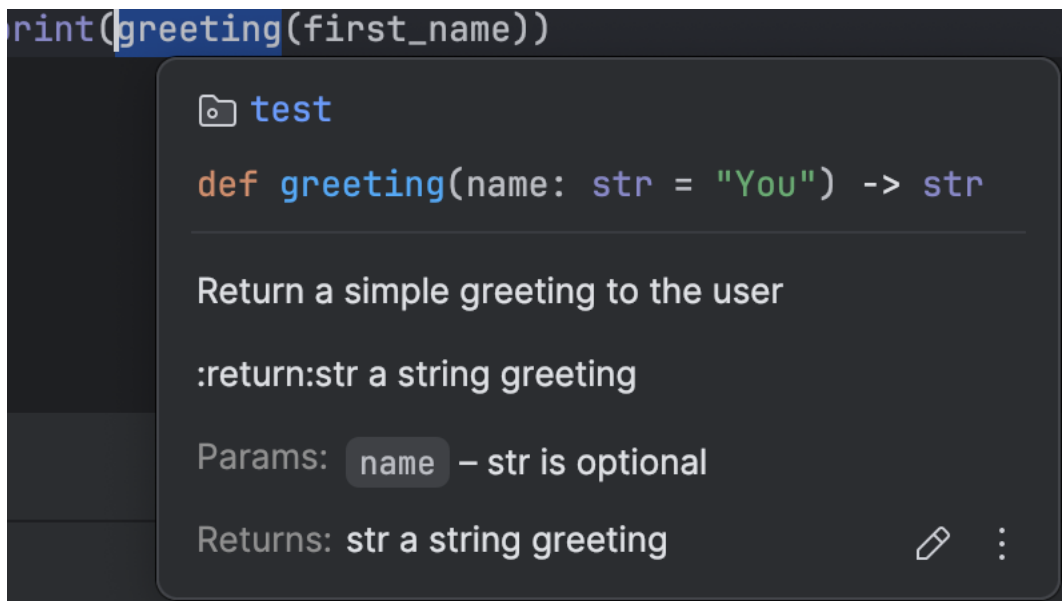
It is helpful to create a function or class header which is a notation to yourself and other developers what the function or class does. It’s called a docstring in Python. It can be seen in figure 7 on a function. Professor Root said in lecture that pressing *control* + *q* would give you hints based on the doctring when you highlight a function. This did not work for me. Tried to ask on ChatGPT and it said to use *command* + *q* since on a Mac in PyCharm but also did not

work. It kept asking me if I wanted to exit PyCharm. However, just highlighting the function and waiting a second it brought it up itself. This can be seen in figure 8.

```
def greeting(name = "You"):
    '''Return a simple greeting to the user
    :param name: str is optional
    :return: str greeting'''

    # print(f"Hello, {name}!")
    return f"Hello, {name}!"
```

Figure 7 Function docstring example



```
print(greeting(first_name))
```

test

def greeting(name: str = "You") -> str

Return a simple greeting to the user

:return: str a string greeting

Params: name – str is optional

Returns: str a string greeting

Figure 8 Docstring code hint in PyCharm on a Mac OS

Classes

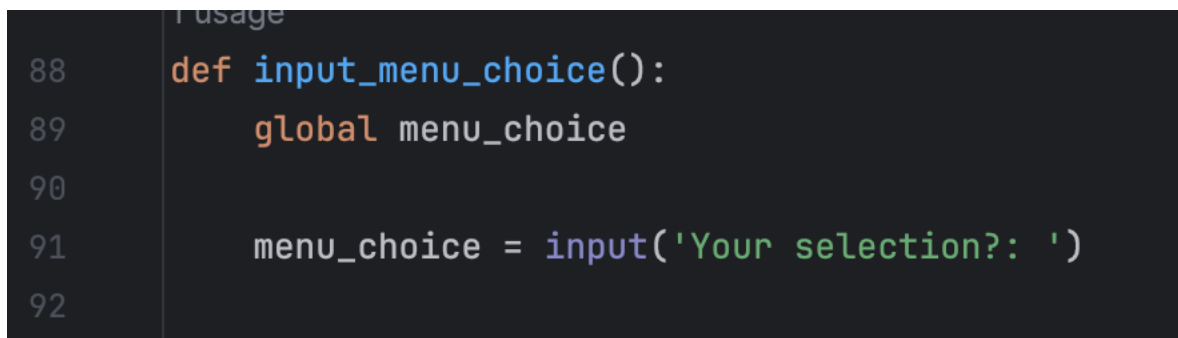
Classes are another way of organizing your code by grouping related functions, variables, and constants. Functions that are part of classes are called methods. If you declare a method with an `@staticmethod` decorator, you can use it without needing an object. Using classes helps

you organize into separation of concerns. Ideally, you should split your code into sections like Data, Processing, and Presentation (or Input-Output).

Program Assignment 06

I created a program that was like the program in Assignment05, but builds on those concepts to also demonstrate using functions, classes and methods, and the separation of concerns pattern.

The program functions the same as in Assignment05. The code was just refactored. I first moved the functionality into functions using mostly global variables.

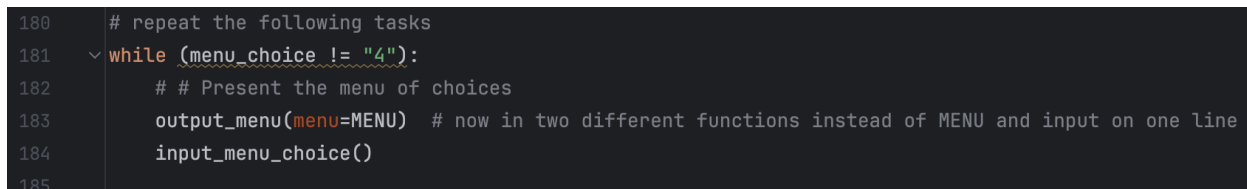


```

88     def input_menu_choice():
89         global menu_choice
90
91         menu_choice = input('Your selection?: ')
92

```

Figure 9 Shows the part where the user inputs a menu choice now in a function using the global variable



```

180     # repeat the following tasks
181     while (menu_choice != "4"):
182         # Present the menu of choices
183         output_menu(menu=MENU) # now in two different functions instead of MENU and input on one line
184         input_menu_choice()
185

```

Figure 10 shows the function now being called in the while loop instead of written here

I then changed the functions to have local variables or have parameters and take arguments. This can also be seen in figure 10 where I call the new output_menu function and the constant MENU gets passed into it. The function itself can be seen in figure 11.


```

83     def output_menu(menu: str):
84         # Present the menu of choices
85         print(menu)
86

```

Figure 11 The new output_menu function

After I got that all working, I made an IO class and a FileProcessing class and moved and reworked the functions to become methods of those classes. I also added docstrings to the classes and the functions.

```

1 usage
45     @staticmethod
46     def output_menu(menu: str):
47         ''' This function displays a menu of choices to the user
48
49         ChangeLog: (Who, When, What)
50         TMcGrew, 04.29.2024, Created function
51
52         :return: None
53         '''
54         # Present the menu of choices
55         print(menu)
56

```

Figure 12 Here is that same output_menu function but now a method of the IO class and with better docstring info

The program still works by taking input from the user both to register a student and save that info to a JSON file. It also displays a message about a student's registration for a Python course and those previously registered already by what was previously saved to the JSON file.

The program displays a menu of choices. It continues by way of a while loop to continually let the user register until they decide to quit. There is a series of if statements that

check to see what option is chosen and act accordingly. However, now the if statements are concise as they call methods of the classes instead of having it all written there.

```

242 while (menu_choice != "4"):
243     # # Present the menu of choices
244     IO.output_menu(menu=MENU) # now in two different functions instead of MENU and input on one line
245     IO.input_menu_choice()
246
247     if (menu_choice == '1'):
248         IO.input_student_data(student_data=students)
249         continue
250
251     elif (menu_choice == '2'):
252         # Present the current data
253         IO.output_student_courses(students)
254         print("\nPlease now select '3' to save the registrations you entered to a file.\n")
255         continue
256
257     elif (menu_choice == '3'):
258         FileProcessor.write_data_to_file(file_name=FILE_NAME, student_data=students)
259         continue
260
261     # Stop the loop
262     elif (menu_choice == '4'):
263         exit()
264         # break? or continue? or exit()?
265
266     else:
267         # spacing
268         print()
269         # They they picked something other than the options given
270         print("Please pick one of the options.")
271         # spacing
272         print()
273         continue

```

Figure 13 While loop and if statements showing same logic but code moved into classes and methods

The program starts by reading from the JSON file that was already saved and making a dictionary with each row of the information it got and putting those rows into a table (list of dictionaries). There is exception handling when the reading of the JSON file is attempted which is now handled by way of a module from the IO class.

If the user chooses menu item 1, the program prompts the user to enter their first and last name and then the course name. It gets these by the use of `input()` and variables. These variables are no longer global but are contained in the `input_student_data` method of the IO class.

```

85     @staticmethod
86     def input_student_data(student_data=list):
87         ''' This function takes a list of dictionaries and gets input from the user, formats
88             it into a dictionary row and appends that list that was passed in
89
90             ChangeLog: (Who, When, What)
91             TMcGrew,04.29.2024, Created function
92
93             :return: None
94             '''
95
96         student_first_name: str = ""
97         student_last_name: str = ""
98         course_name: str = ""
99         student_row: dict[str, str] = {} # one row of student data as a dictionary
100
101         # variables capturing the input asked from the user
102         try:
103             student_first_name = input("Please enter your first name: ")
104             if not student_first_name.isalpha():
105                 raise ValueError("The first name should not contain numbers.")
106
107             student_last_name = input("Please enter your last name: ")
108             if not student_last_name.isalpha():
109                 raise ValueError("The last name should not contain numbers.")
110
111             course_name = input("Please enter the course name: ")

```

Figure 14 Showing that the global variables for student information are now local to the `input_student_data` method in the IO class

There is also exception handling around the first and last name in case the user enters non-alpha numeric input on either of those fields. However, entering the information on this option, doesn't save what they entered to a file, so I prompt the user to then choose option number 3 to complete their registration. Not visible to the user, it also takes what was entered into the variables and makes a dictionary row. It then enters that row it made into a table (list of dictionaries) by appending the row just made to it.

If user chooses item 2, it presents all the collected data both from the JSON file and that has been entered by the user in item 1, by looping through the rows of the table (list of dictionaries) and presenting each row in a f-string.

Choosing menu item 3, the program writes the table (list of dictionaries) to the JSON file. It also displays to the user what is written to the file by looping through the information and presented by way of a formatted f-string. This also has exception handling when it attempts to write to the JSON file.

To quit the program, menu option 4 is chosen. This is necessary or the while loop will keep going.

Figures 15, 16, and 17 show the module running in PyCharm and figures 18 and 19 in Terminal. Figure 20 shows the contents of the JSON file Enrollments.json To view the code of my module, here is a link to my GitHub repository <https://github.com/tmcgrew/IntroToProg-Python-Mod06>

```
--- Course Registration Program ---
Select from the following menu:
1. Register a Student for a Course
2. Show current data
3. Save data to a file
4. Exit the program
-----

Your selection?: 2
Bob Smith is registered for Python 100.
Jenny Jones is registered for Python 100.
Steve Stoneman is registered for Python 100.
Libby Ludtke is registered for Python 100.
Billy Bowler is registered for Python 100.
Gerry George is registered for Python 100.
Lenny Lerman is registered for Python 100.
George Germaine is registered for Python 100.
Robert Robertson is registered for Python 100.
Nevill Newman is registered for Python 100.
Kelly Kooper is registered for Python 100.
Terry Thompson is registered for Python 100.
Danny Denver is registered for Python 100.

Please now select '3' to save the registrations you entered to a file.
```

Figure 15 Module running in PyCharm

```
--- Course Registration Program ---
Select from the following menu:
1. Register a Student for a Course
2. Show current data
3. Save data to a file
4. Exit the program
-----

Your selection?: 1
Please enter your first name: Vic
Please enter your last name: Vu
Please enter the course name: Python 100

Thank you! Please now select '3' to save the registration to a file.

--- Course Registration Program ---
Select from the following menu:
1. Register a Student for a Course
2. Show current data
3. Save data to a file
4. Exit the program
-----
```

Figure 16 Continuation of figure 15

```
Your selection?: 3
Bob Smith is fully registered for Python 100.
Jenny Jones is fully registered for Python 100.
Steve Stoneman is fully registered for Python 100.
Libby Ludtke is fully registered for Python 100.
Billy Bowler is fully registered for Python 100.
Gerry George is fully registered for Python 100.
Lenny Lerman is fully registered for Python 100.
George Germaine is fully registered for Python 100.
Robert Robertson is fully registered for Python 100.
Nevill Newman is fully registered for Python 100.
Kelly Kooper is fully registered for Python 100.
Terry Thompson is fully registered for Python 100.
Danny Denver is fully registered for Python 100.
Vic Vu is fully registered for Python 100.

--- Course Registration Program ---
Select from the following menu:
1. Register a Student for a Course
2. Show current data
3. Save data to a file
4. Exit the program
-----

Your selection?: 4

Process finished with exit code 0
```

Figure 17 Continuation of figure 16

```
--- Course Registration Program ---
Select from the following menu:
1. Register a Student for a Course
2. Show current data
3. Save data to a file
4. Exit the program
-----

Your selection?: 1
Please enter your first name: Kerry
Please enter your last name: Kelly
Please enter the course name: Python 100

Thank you! Please now select '3' to save the registration to a file.

--- Course Registration Program ---
Select from the following menu:
1. Register a Student for a Course
2. Show current data
3. Save data to a file
4. Exit the program
-----

Your selection?: 2
Bob Smith is registered for Python 100.
Jenny Jones is registered for Python 100.
Steve Stoneman is registered for Python 100.
Libby Ludtke is registered for Python 100.
Billy Bowler is registered for Python 100.
Gerry George is registered for Python 100.
Lenny Lerman is registered for Python 100.
George Germaine is registered for Python 100.
Robert Robertson is registered for Python 100.
Nevill Newman is registered for Python 100.
Kelly Kooper is registered for Python 100.
Terry Thompson is registered for Python 100.
Danny Denver is registered for Python 100.
Vic Vu is registered for Python 100.
Kerry Kelly is registered for Python 100.

Please now select '3' to save the registrations you entered to a file.

--- Course Registration Program ---
Select from the following menu:
1. Register a Student for a Course
2. Show current data
3. Save data to a file
4. Exit the program
-----
```

Figure 18 Module running in Terminal


```
Your selection?: 3
Bob Smith is fully registered for Python 100.
Jenny Jones is fully registered for Python 100.
Steve Stoneman is fully registered for Python 100.
Libby Ludtke is fully registered for Python 100.
Billy Bowler is fully registered for Python 100.
Gerry George is fully registered for Python 100.
Lenny Lerman is fully registered for Python 100.
George Germaine is fully registered for Python 100.
Robert Robertson is fully registered for Python 100.
Nevill Newman is fully registered for Python 100.
Kelly Kooper is fully registered for Python 100.
Terry Thompson is fully registered for Python 100.
Danny Denver is fully registered for Python 100.
Vic Vu is fully registered for Python 100.
Kerry Kelly is fully registered for Python 100.

--- Course Registration Program ---
Select from the following menu:
1. Register a Student for a Course
2. Show current data
3. Save data to a file
4. Exit the program
-----

Your selection?: 4
(base) tracimcgreww@Tracis-Air A06 %
```

Figure 19 Continuation of figure 18

A screenshot of a code editor window titled 'Enrollments.json'. The window has a dark background and shows a JSON array of 15 objects. Each object has three keys: 'FirstName', 'LastName', and 'CourseName'. The 'CourseName' for all objects is 'Python 100'. The names are: Bob Smith, Jenny Jones, Steve Stoneman, Libby Ludtke, Billy Bowler, Gerry George, Lenny Lerman, George Germaine, Robert Robertson, Nevill Newman, Kelly Kooper, Terry Thompson, Danny Denver, Vic Vu, and Kerry Kelly. The text is white with some syntax highlighting.

```
[{"FirstName": "Bob", "LastName": "Smith", "CourseName": "Python 100"}, {"FirstName": "Jenny", "LastName": "Jones", "CourseName": "Python 100"}, {"FirstName": "Steve", "LastName": "Stoneman", "CourseName": "Python 100"}, {"FirstName": "Libby", "LastName": "Ludtke", "CourseName": "Python 100"}, {"FirstName": "Billy", "LastName": "Bowler", "CourseName": "Python 100"}, {"FirstName": "Gerry", "LastName": "George", "CourseName": "Python 100"}, {"FirstName": "Lenny", "LastName": "Lerman", "CourseName": "Python 100"}, {"FirstName": "George", "LastName": "Germaine", "CourseName": "Python 100"}, {"FirstName": "Robert", "LastName": "Robertson", "CourseName": "Python 100"}, {"FirstName": "Nevill", "LastName": "Newman", "CourseName": "Python 100"}, {"FirstName": "Kelly", "LastName": "Kooper", "CourseName": "Python 100"}, {"FirstName": "Terry", "LastName": "Thompson", "CourseName": "Python 100"}, {"FirstName": "Danny", "LastName": "Denver", "CourseName": "Python 100"}, {"FirstName": "Vic", "LastName": "Vu", "CourseName": "Python 100"}, {"FirstName": "Kerry", "LastName": "Kelly", "CourseName": "Python 100"}]
```

Figure 20 Showing the contents of the JSON file Enrollments.json

Summary

I went over some core concepts from the module, and then showed a Python program I created that was like the program in Assignment05, but builds on those concepts to also demonstrate using functions, classes and methods, and the separation of concerns pattern.

Works Cited

OpenAI. "ChatGPT." OpenAI, April 25, 2024, <https://chat.openai.com>