Kristie Dunkin
November 18, 2024
Foundations of Programming Python
Assignment 6
https://github.com/mulberrysilk/IntroToProg-Python-Mod06/tree/master/_Module06/_Module06/Assignment

Assignment 6

Contents

Introduction

For our Assignment 6 this week we will incorporate new tools. With a focus on functions, we have started to become more proficient at programming with Python. This assignment also includes use of error handling, dictionaries, use of the JSON library, and use of other supporting tools outside Python, like Git (versioning) and GitHub (repository). The programming assignment continues to build on our work accomplished over the previous weeks. We will create a list of dictionaries and open, read, and write to a JSON library. Assignment 6 feels like we are in view of the summit.

Getting Started

The goals of this week's efforts are:

- Continuing the learning cycle, this time focusing on functions.
- Start this document,
- Review the assignment,
- Draft the bones of the assignment in Pycharm,
- Continue with using error handling with an emphasis on getting better at this, and
- Use GitHub for a second time.

Here's the Assignment 6 objective copied from the Module 6 text:

> "Create a Python program that demonstrates using constants, variables, and print statements to display a message about a student's registration for a Python course. This program is very similar to Assignment 05, but it adds on the use of functions, classes and using the Separation of Concern pattern. Start by opening and reviewing the starter file Assignment06-Starter.py."

First impressions of assignment: We are building on our existing structure with the use of Functions! I feel like we may actually be becoming useful! Ha!

I made no headway last week on understanding Git, beyond installing it, so that might be a good thing to do this week, if time allows. Reading through my reference book (remember I bought it, so I'm going to read it!), I was very interested in the discussion of *encapsulation and abstraction* (Dawson 2002). These two interrelated concepts were great to read about in actual words, as I have been feeling these ideas, especially abstraction, but not understanding how to explain what's going on. Note to self to explore these two concepts more when opportunity presents itself. We are working on the Separation of Concern concept this week, which seems like it is related to "scope" in Dawson.

Here's me thinking about what we are doing in the script itself, this planning language is called Pseudocode.

Note that at the end of the exercise, the script does not do anything new! It will do the same thing as before, but with more advanced tools and organization:

- Opening JSON Library with a function call.
- When the program starts the JSON file is automatically read.
- Actually, there are seven required functions! I will not list them here. I have two already and will write 5 more.
    - I need to update the existing function's names to match requirements.
- Add the docstrings (so very impressed with these! So useful!)
- Offer a menu to choose from –
    - I will write a couple of new functions in this IO part of the script.
- Know what someone's menu choice and what results.
    - add more better error handling.
- Select and perform the correct action for that choice:
    - Choice 1 will collect data, for the "register student for a course" action, and the file will be set to append so it can continue to collect data.
    - Choice 2 will show current data using a print statement – like in previous assignments. The data will be in dictionary form and will be formatted into a string by using the Key for the data. Indexes are no longer needed.
    - Choice 3 will save the data – like last week.
    - Choice 4 will exit the program, as before.
- Add two classes and documentation for each class:
    - Class for FileProcessor – this will be the open and close file functions.
    - Class for IO – "IO" meaning input/output – this will be for the menu offerings and data collection, I believe.

This may be the first week with no planned messing with the computer's software. Yeah!!!

Functions

I'me very happy to have reached this point in gaining an understanding of Python. This level of achievement is way cool! And this has me thinking that I could, maybe, write my own program. When I was cleaning up a huge bunch of files on a drive at work…a program should do this for me! Maybe soon!

Reading about functions in Dawson and he showed how the variables in functions are not available outside of the function for general use. This type of "fencing" seems useful and important to understand. There are many real-world analogies in this concept. Functions also seem a bit dangerous – what if it doesn't do what you think is does?

Not sure I have the "return" concept figured out yet… but the "call a function" seems straightforward. Dawson says, "New functions should do one job well". I think they can be useless too---he gave an example of a function that had print() as its second line. So what does that function do? Print! We already have a function that prints, so not sure why we'd want another one. Maybe for illustration, to keep it simple, in the book?

First Draft of Assignment 6 Script

And into the rabbit hole we go! So far, I can see I have to rearrange everything! But I'm uncertain what goes where! I'm confused by where the function calls go in the loop – so lets just sketch it here:

> If choice is 1 then a function should ask questions to collect input data, format to dictionary, and append to students (is this more than one function?),
>
> If choice is 2 then a function should print a nice message using data in students,
>
> If choice is 3 then I need a call to "the save the file function" (script already does this).

Based on this thought, each "if" or "elif" statement must have a function that is called. But see that I currently have at the start of the loop the following statements: "if", then "try", then "while True." Ugh.

The loop goes in a "Main section" because that makes sense, and based on above there are calls to functions on each step in the loop. I think…

Done. The loop is very tidy now – that's nice (Figure 1)! And no longer indented way right. As there are 13 red exclamation points in Pycharm at this point, this will not run.

Figure 1. Main section at the first attempt of scripting of functions.

```
#MAIN
welcome_str_menu() # Provides a greeting and menu for input
if menu_choice == str(1):
    input_menu_choice1() # asks questions and accepts data
    input_student_data(students) # formats and saves data
    print(students)
elif menu_choice == str(2):
    output_student_courses(students)
elif menu_choice == str(3):
     write_data_to_file(students, FILE_NAME)
     output_student_courses(students)
else:
       print(exit_statement)
       quit()
```

Good things here: I think I know what I'm trying to do, just not the details of how to do it at this point. I had to take out all the error handling to "see" how this worked.

I noticed there is code that I no longer need here as had this routine twice in Assignment 5:

```
for student_row in students:
    print(f'{student_row[FIRST_NAME_KEY]} {student_row[LAST_NAME_KEY]} is '
          f'signed up for {student_row[COURSE_NAME_KEY]}')
```

One of them can go, and I use these written lines to create one function and add the two function calls instead (but this is not really any shorter, just sturdier.)

To fix the script I had to understand how parameters worked and how return statements worked. Also some of the lines in each function were not quite right.

More Things about Functions: Parameters and Arguments

- Uses "def" which is a key word to denote the definition.
- There are parentheses (always, dear).
- Inside the parentheses may be nothing or there may be parameters.
- There is a "docstring" after the first line – this triple quoted comment string has information about the function.
- Pycharm is charming about offering options for the docstring.

Parameters are interesting. Note that the words used to name the parameter are unique to the function and also should not be "shadowed" (that was one thing that Pycharm was identifying as a warning in my script). These names are "Local".

According to the Python documentation, Chapter 4.8 "Defining Functions" (REF), this is because,

> "variable references first look in the local symbol table [meaning that for the function], then in the local symbol table of enclosing functions, then in global symbol table, and finally in the table of built-in names".

Thus, these should be names created and used only in the function ("locally").

More information from the documentation indicated that,

> "Arguments are passed using "CALL BY OBJECT REFERENCE" where the *value* is always an object *reference*, not the value of the object".

My understanding (so far) of how this works is shown in Figure 2 using one of my functions and a call to that function (I took out the docstring to make this clearer). Note that when first writing these, you have to keep track, because the names are not the same in the callout and in the function (at least for what we are doing, and I really wanted them to be the same, apparently!). This may be apparent to some, but I really struggled with this, but now I think I have it.

The arguments I used are "required" arguments, and they are "positional", too. For functions in general there are also "optional arguments" and "keyword arguments", (AKA "named parameters"), which I am not using. If I had this last type, they would have to come after the positional arguments in the def line.

Figure 2. The function and how it receives the correct data.

DEFINITION DENOTING THE FUNCTION

```
def write_data_to_file(save_data: list, file_location: str):
    with open(file_location, "w") as file:
    json.dump(save_data, file, indent=2)
```

CALLOUT OF THE FUNCTION

```
write_data_to_file(all_students_list, FILE_NAME)
```

The Parameters inside the function, including in the definition line, have unique names.

At the callout, the first name(argument)identifies the **value to use at Parameter 1.** The value in this case is the list of student's information (these two are noted in lime green).

For Parameter 2, "file_location" calls the value of the argument FILE_NAME. Which is the value, "Enrollments.json"(noted in turquiose).

Error Handling and Class Confusion

Do the try's go with the function definition or with the function call? How do I make an error handling function, when error handling lines are all over the place now? Can functions which I make, call other functions which I make? More research is needed. To make progress ,I took out all the error handling from the draft 06 script.

After research, answers to the questions:

1) Error handling is at the function primarily. But it looks different from last week's design.
2) Yes, functions can, and aught to, call other functions.
3) The errors handling is like before, in that it wraps around the code line that does something that could cause an issue, like file reading and writing ,or inputting or outputting data.

I added back in error handling – and fiddled with it till it worked. I had issues with the return statement – I thought it had to be at the end of the function, but that location didn't work because the function would end first. I had the "while" after the "try", and that didn't work. "While" needed to come first in the function before "try". The most troublesome function, "input_student_data" now has a long list of error handling under a While statement (Figure 3).

```
def input_student_data():
        while True:
            try:
                student_first_name = input("What is your first name? ").strip().title()
                if not student_first_name.isalpha():
                    raise Exception("Sorry, that was not a good first name")
                student_last_name = input("What is your last name? ").strip().title()
                if not student_last_name.isalpha():
                    raise Exception("Sorry, that was not a good last name")
                course_name = input(
                    "Please enter the name of the course: ").strip().title()  # pass
                if not course_name.isprintable():
                    raise Exception("Sorry, that was not a good course name")
                else:
                    break
            except Exception as e:
                 FileProcessor.output_error_messages("Please try again")
        student_data: dict[str, str] = {FIRST_NAME_KEY: student_first_name,
              LAST_NAME_KEY: student_last_name,
              COURSE_NAME_KEY: course_name}
        return student_data
```

Things were going well here, but then I added the classes! And at this point everything went sideways. Then I went to Office Hours and learned that I somehow had missed the need to update the function names with the class name when a function is called. Once I fixed that, the classes seemed to work and the red errors went away.

Now the script works. Table 1 gives a status towards completion at this point.

Table 1. Status toward completing the assignment – first check in.

1. Read module text	Done
2. Watch the module videos.	I have been reading other sources this week
3. Create a program	Program created and does work!
4. Document your knowledge and make a PDF.	In QC – Make a PDF
5. Submit your work to discussion board.	Submit just the URL GitHub - Done
6. Post Files to GitHub	Done
7. Post Link to GitHub Files on Canvas	Done
8. Post per usual to Canvas 9. Make a Peer Review (optional)	Done

Conclusions

Assignment 06 focused on producing a script that uses error handling, JSON files for saving and reading a list of dictionaries. Importantly, this assignment had our first interface with functions and classes. This was an interesting and useful exercise, and it is clear that more practice is needed to gain mastery of these useful Python tools.

References

Dawson, Michael. 2010. Python Programing for the Absolute Beginner. Third Edition. Published by Course Technology PTR a part of Cengage Learning. Boston, MA, USA.