Adam Packer
April 1, 2025
UW: IT FDN 110 A Wi 25: Foundations Of Programming: Python
Module 6 Assignment
GitHub URL: https://github.com/adp-pdx/IntroToProg-Python-Mod06

Assignment 6: Bringing together the Python tools I need to move beyond basic scripts

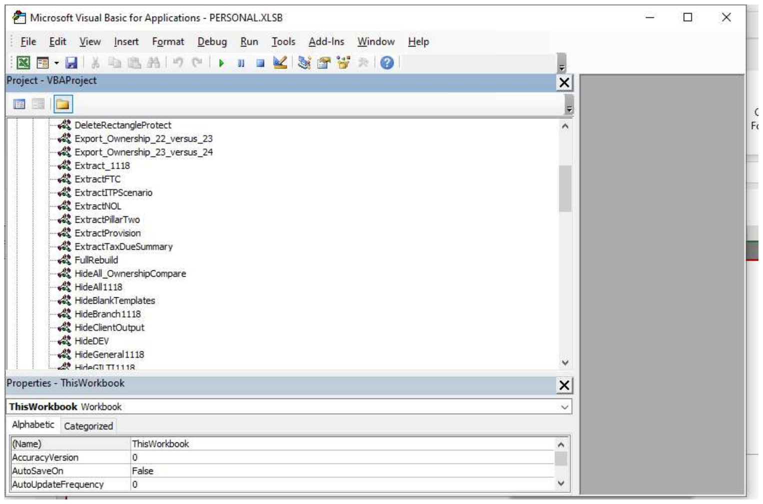
Introduction

More and more as I progress through this course I find that each subsequent module is more satisfying for me. I think the earlier modules were so basic that I struggled a little bit to see how it's all coming together. I understand why the course is set up the way that it is, and I like how it is set up. But I like being further along because it's more satisfying to put more puzzle pieces together. It amplifies the learning process because I remember more when I see it all linked together. The topics from the this module really came together for me in the end, piece by piece.

Studying how to Organize Code

So in the AI-enabled prompt-based coding that I have been doing in Excel VBA, one of my main concerns was that I needed to develop a more sophisticated method for organizing code. Right now I am on the only person on my team, so from one perspective I think, "well maybe it doesn't matter how well my code is organized because I am the only person working in here, and probably will be for a while." That said, my method isn't really a great way to do things. Also, I read online that early stage developers like me, especially those using AI, get a bad reputation for not organizing the code well. That definitely could apply to the work that I have done, so I appreciated this section of the course, helping me learning this.

Figure 1: Adam's 'Bachelor Pad' VBA Editor – "No one else lives here!"



DeleteRectangleProtect
Export_Ownership_22_versus_23
Export_Ownership_23_versus_24
Extract_1118
ExtractFTC
ExtractITPScenario
ExtractNOL
ExtractPillarTwo
ExtractProvision
ExtractTaxDueSummary
FullRebuild
HideAll_OwnershipCompare
HideAll1118
HideBlankTemplates
HideBranch1118
HideClientOutput
HideDEV
HideGeneral1118
HideGILTI1118

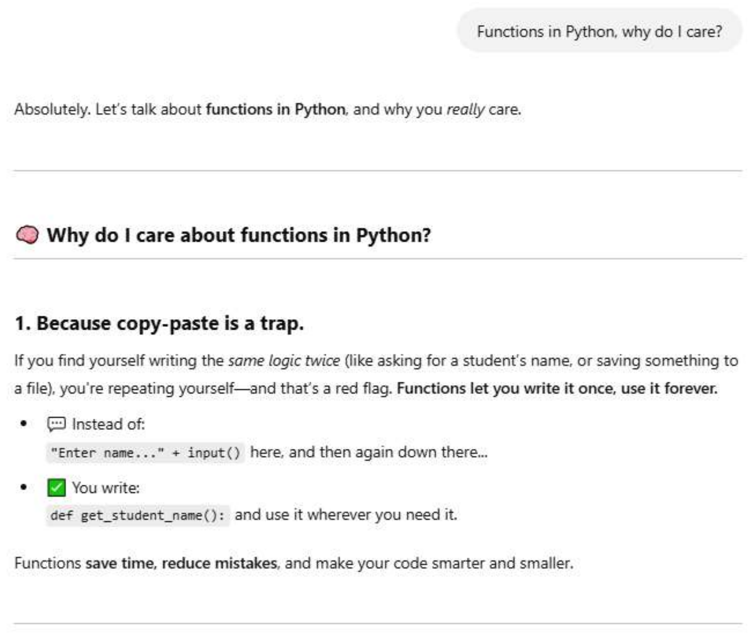
(Name)	ThisWorkbook
AccuracyVersion	0
AutoSaveOn	False
AutoUpdateFrequency	0

Functions

This topic was super interesting to me. I am well aware that much of what I write here is focused on how I am basically linking my Excel-coding brain into my Python-coding brain, and as Reader, you're stuck hearing that story again and again. Excel built in functions form a very important part of the foundation of the work that I do in Excel. Earlier in my career, learning a new Excel formula each day was key, learning how the syntax (parameters) were written, was essential to my career development and candidly, I really liked it. But the idea of being restricted to only the built in functions of Excel is sometimes limiting. There are plenty of great Excel functions that are built in, and recent developments make it possible to draft User defined functions in Excel, which further blurs the line between Excel formula drafting and actual coding. But Python completely opens up the ability to draft customized functions where I don't have to rely on built in functions from Microsoft – I can draft them myself. There is a huge freedom to that.

The course notes on this section were really good, but I also really enjoy discussing the topic with ChatGPT. As I have noted before, it helps me to play a game with ChatGPT that I call, "Why do I care?", where I state a topic, and then ask the question "Why do I care?" This Module resulted in lots of great moments of targeted learning in the "Why do I care" format that help me see the full context of a topic that I am learning.

Figure 2: Playing "Why do I care?" with ChatGPT, starting with Functions



Functions in Python, why do I care?

Absolutely. Let's talk about **functions in Python**, and why you *really* care.

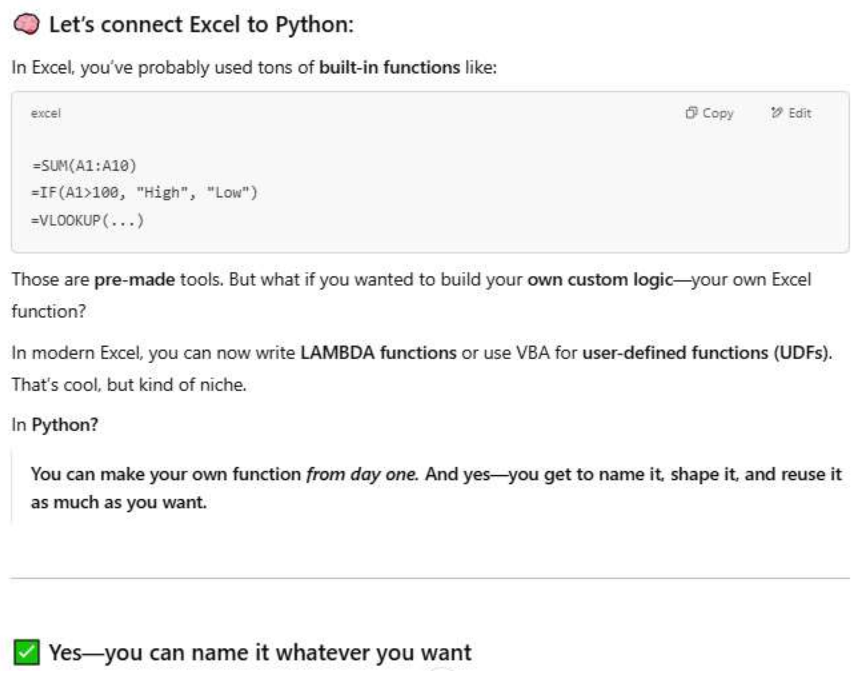
🧠 **Why do I care about functions in Python?**

1. Because copy-paste is a trap.

If you find yourself writing the *same logic twice* (like asking for a student's name, or saving something to a file), you're repeating yourself—and that's a red flag. **Functions let you write it once, use it forever.**

- 💬 Instead of:
  `"Enter name..." + input()` here, and then again down there...
- ✅ You write:
  `def get_student_name():` and use it wherever you need it.

Functions **save time, reduce mistakes**, and make your code smarter and smaller.



🧠 Let's connect Excel to Python:

In Excel, you've probably used tons of **built-in functions** like:

```excel
=SUM(A1:A10)
=IF(A1>100, "High", "Low")
=VLOOKUP(...)
```

Those are **pre-made** tools. But what if you wanted to build your **own custom logic**—your own Excel function?

In modern Excel, you can now write **LAMBDA functions** or use VBA for **user-defined functions (UDFs)**. That's cool, but kind of niche.

In **Python?**

> **You can make your own function *from day one.* And yes—you get to name it, shape it, and reuse it as much as you want.**

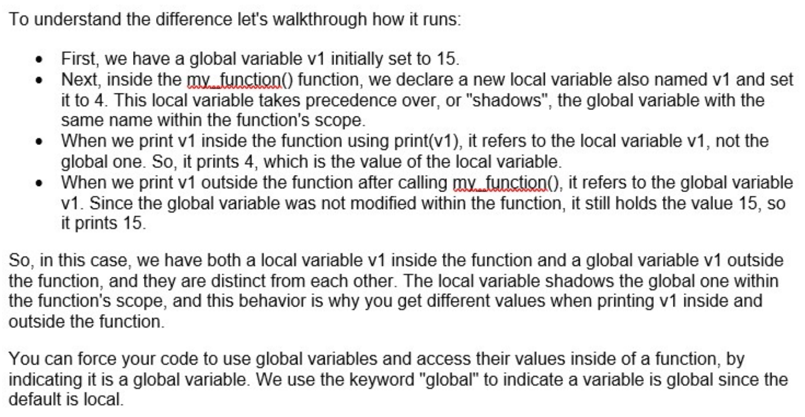
✅ Yes—you can name it whatever you want

Global versus Local Variables

This topic really helped me to focus on the broader view of how I will build programs as I am moving forward. I think on some level I was aware of this topic generally because you can do this in VBA, but I am not very good at it. I tend to draft really specific functions with specific variables in a discrete VBA module to do one targeted thing. It works great. But as a developer, to draft something more meaningful I need to know how to do this.

The course notes helped identify the impact of the defining a variable as global versus local in a way that helped me:

Figure 3: Course text that helped me see the difference:



To understand the difference let's walkthrough how it runs:

- First, we have a global variable v1 initially set to 15.
- Next, inside the my_function() function, we declare a new local variable also named v1 and set it to 4. This local variable takes precedence over, or "shadows", the global variable with the same name within the function's scope.
- When we print v1 inside the function using print(v1), it refers to the local variable v1, not the global one. So, it prints 4, which is the value of the local variable.
- When we print v1 outside the function after calling my_function(), it refers to the global variable v1. Since the global variable was not modified within the function, it still holds the value 15, so it prints 15.

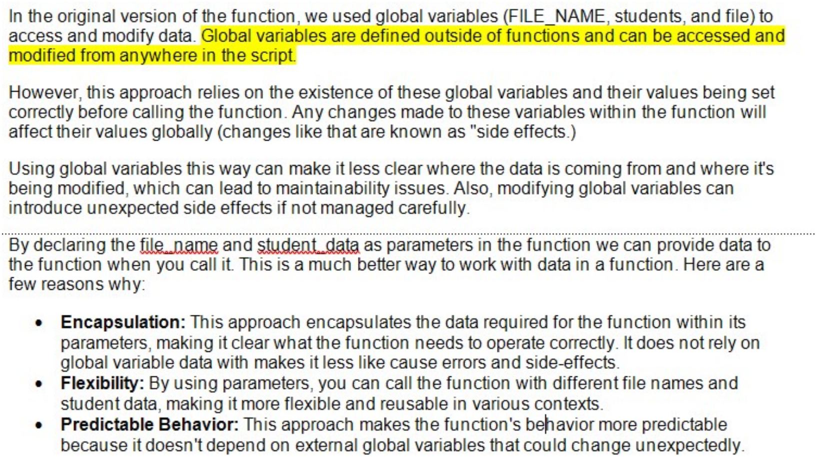
So, in this case, we have both a local variable v1 inside the function and a global variable v1 outside the function, and they are distinct from each other. The local variable shadows the global one within the function's scope, and this behavior is why you get different values when printing v1 inside and outside the function.

You can force your code to use global variables and access their values inside of a function, by indicating it is a global variable. We use the keyword "global" to indicate a variable is global since the default is local.

Parameters & Arguments, including Default Parameter values

When I started studying parameters, it seemed to me to bear a strong resemblance to the pre-defined syntax of Excel-based formulas. As I studied the course materials more, I started to see why parameters evoked that comparison to me.

Figure 4: Course text on Parameters that involved Global vs. Local Variables.



In the original version of the function, we used global variables (FILE_NAME, students, and file) to access and modify data. Global variables are defined outside of functions and can be accessed and modified from anywhere in the script.

However, this approach relies on the existence of these global variables and their values being set correctly before calling the function. Any changes made to these variables within the function will affect their values globally (changes like that are known as "side effects.)

Using global variables this way can make it less clear where the data is coming from and where it's being modified, which can lead to maintainability issues. Also, modifying global variables can introduce unexpected side effects if not managed carefully.

By declaring the file_name and student_data as parameters in the function we can provide data to the function when you call it. This is a much better way to work with data in a function. Here are a few reasons why:

- **Encapsulation:** This approach encapsulates the data required for the function within its parameters, making it clear what the function needs to operate correctly. It does not rely on global variable data with makes it less like cause errors and side-effects.
- **Flexibility:** By using parameters, you can call the function with different file names and student data, making it more flexible and reusable in various contexts.
- **Predictable Behavior:** This approach makes the function's behavior more predictable because it doesn't depend on external global variables that could change unexpectedly.

So for my Excel-based brain, the idea of a global variable actually kind of bothers a little bit, which is immature for sure, but the question is why? Well, I think it comes down to this point about encapsulation. When I am drafting an Excel formula, I have a really targeted thing to do, and I want the formula to specifically reference and operate on what I am working on. The idea that I would draft something that would affect the whole workbook globally sounds like a nightmare. Now, I am sure it won't be as I learn how to use a tool like global variables in the correct context. That said, I appreciate this approach of declaring the parameters specifically in the function to create this predicable way of the function doing one thing and affecting one data set. At the same time, it still is really flexible, it's just more limited in scope which I think gives me the type of control that I need in how I design things.

Classes

As we've gone through this course, I've spent time thinking about the whole process of introducing a student to a new complex system of knowledge. Don't worry – this will relate to classes, just give me a second. There's this process when you sit down with a group after a dinner party and someone suggests that the group play a new awesome board game. Then for 20 minutes the group talks about teaching everyone the rules. Usually one person has played the game, and naturally that person takes the lead in deciding how to try and explain the game to the others. There's different approaches, but a lot of times the group doesn't actually pull out the printed game rules from the box, they just listen to the person that has started trying to explain the game. After a period of time where one or two people are talking about the rules and premises of the game, eventually everyone says let's play a few sample rounds and we'll learn as we play. There there is this moment during the first few rounds of the game where it starts to click, and each player starts to see how the different points the 'teaching' players were making earlier start to come together. "Oh, I get it – this is what we're trying to do." That group learning process is sort of reflected in a course like this. For me, learning about classes was the moment where it all started to come together. In some ways, the earlier moments in the course (I know this sounds weird) are a little too simple, and the points felt harder to connect. Reading the course materials on classes in this module (and also the next Module 7 – spoiler alert) are really what started to make this get for me, in combination with separation of concerns, to really understand object oriented programming.

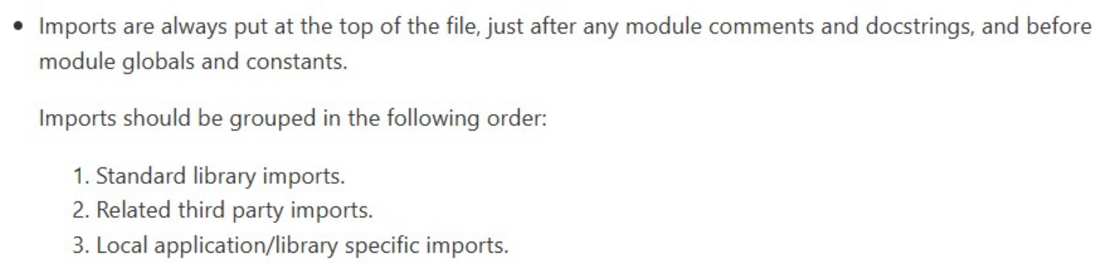
I also talked with ChatGPT about classes, and played why do I care? The analogies from ChatGPT, and also from the course text, help to explain. Classes are a self contained cookie cutter, and I want to make different objects with different attributes using that cookie cutter in a reusable way.

There are several things about the purpose of classes that help me understand how this all comes together, and mainly it's that each class is structured to do one main thing, organized around a series of functions (methods) within the class.

While I was drafting the assignment, the organization of all came together.

Flow:

1. Headers (obviously)
2. Imports
    a. start by telling Python what pre-existing modules the code will need. I looked up PEP 8 at https://peps.python.org/pep-0008/, and noted that the imports are always at the top, as a matter of convention, and that makes sense.



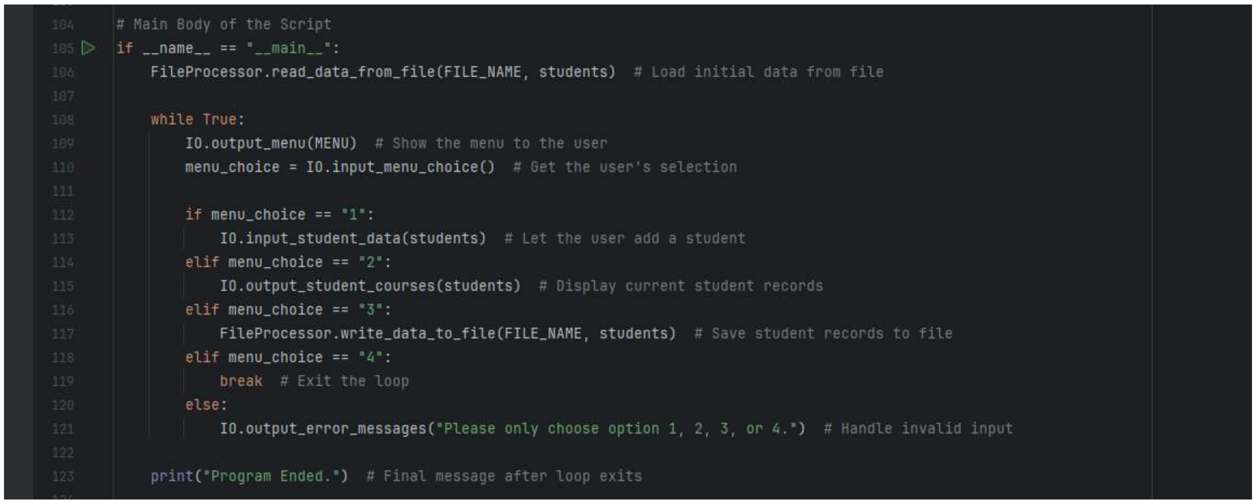
- Imports are always put at the top of the file, just after any module comments and docstrings, and before module globals and constants.

  Imports should be grouped in the following order:

  1. Standard library imports.
  2. Related third party imports.
  3. Local application/library specific imports.

    b.
3. Constants
4. Variables
5. Class definitions
    a. IO
    b. DataProcessing
    c. Depending on the program there would be lots of class, lots of different tool kits designed to iteratively create objects and then work on those objects individually.
6. NOW we get to the main body of the script, where the logic all comes together in the 'brain' of the script. It happens at the end, and now is much simpler because each piece of the script was drafted earlier in the flow. It's like the climax of the show where all the plot elements come together.

Figure 5: Much simpler / clearer logic at the end of the script, because the classes up above did a lot of the work!



```python
# Main Body of the Script
if __name__ == "__main__":
    FileProcessor.read_data_from_file(FILE_NAME, students)  # Load initial data from file

    while True:
        IO.output_menu(MENU)  # Show the menu to the user
        menu_choice = IO.input_menu_choice()  # Get the user's selection

        if menu_choice == "1":
            IO.input_student_data(students)  # Let the user add a student
        elif menu_choice == "2":
            IO.output_student_courses(students)  # Display current student records
        elif menu_choice == "3":
            FileProcessor.write_data_to_file(FILE_NAME, students)  # Save student records to file
        elif menu_choice == "4":
            break  # Exit the loop
        else:
            IO.output_error_messages("Please only choose option 1, 2, 3, or 4.")  # Handle invalid input

    print("Program Ended.")  # Final message after loop exits
```

In drafting the lowest section of the code, the while loop combined with the if statements is really clear on a step by step basis, if this then run that, else run that, else run that, then end. Really easy to follow!

Separation of Concerns Pattern

Reading the course materials on this made a lot of sense. We want one piece of code to do one thing, and do it well. This makes it much easier to collaborate with another developer by segmenting modules of code, where each developer can share part of the work. It makes it easier to debug, and improve section by section. I imagine this spaghetti code, maybe created by AI, that is super complex and hard to follow, and maybe does the work that it is supposed to do, but is so tangled that it's super hard for anyone else to work on. Instead, I like the way we are breaking apart our code into discrete sections that do one thing and do it well. I know this sounds silly, but structuring the code really well reminds me of well structured poetry.

Summary

This module gave me a new level of understanding of software code development by helping me see how layers of structure lead to well-drafted code. Classes contain functions (methods) that contain parameters and arguments to create objects, while each section is doing one thing and doing it well. After I wrote the program and re-read it several times just because I enjoyed seeing how all the parts were coming together – super proud of how this is going.