Python Learning Journal

Exercise 1.1

Learning Goals;

- Summarize the uses and benefits of Python for web development
- Prepare your developer environment for programming with Python

Reflection Questions:

1) **In your own words, what is the difference between frontend and backend web development? If you were hired to work on backend programming for a web application, what kinds of operations would you be working on?**

   Back-end development focuses more on server-side programming and infrastructure, while a front end developer is more involved in the UI and user facing elements of the website/app. Operations that a back end developer might be working on include; database management, user information handling, building and maintaining servers, working to improve website security and monitoring and improving performance and functionality.

2) **Imagine you're working as a full-stack developer in the near future. Your team is asking for your advice on whether to use JavaScript or Python for a project, and you think Python would be the better choice. How would you explain the similarities and differences between the two languages to your team? Drawing from what you learned in this Exercise, what reasons would you give to convince your team that Python is the better option? (Hint: refer to the Exercise section "The Benefits of Developing with Python")**

   If I was recommending Python for a project, I would look to highlight the ease of use, built in package manager, simplicity and support. Python was designed to be easy to learn and the standardized syntax of the language enables anyone with some coding experience to understand the code at a glance. This simplicity extends to the built in package manager - PIP. The base install of Python comes with many useful packages and it is very easy to install further packages through the console. Python is also well supported by an active online community that continually improve the language.

3) **Now that you've had an introduction to Python, write down 3 goals you have for yourself and your learning during this Achievement. You can reflect on the following questions if it helps you. What do you want to learn about Python? What do you want to get out of this Achievement? Where or what do you see yourself working on after you complete this Achievement?**

a) My first goal is to make notes throughout the project on how I think the things learnt would be useful in other situations or projects.
b) I want to understand the language to the point where I am seeing solutions to tasks and issues without needing to rely solely on the learning material.
c) Ultimately, I want to be confident about putting Python on my CV/portfolio and comfortable using the language and answering questions about it.

Exercise 1.2

Learning Goals;

● Explain variables and data types in Python
● Summarize the use of objects in Python
● Create a data structure for your Recipe app

Reflection Questions
1. Imagine you're having a conversation with a future colleague about whether to use the iPython Shell instead of Python's default shell. What reasons would you give to explain the benefits of using the iPython Shell over the default one?

I would make note that the ipython shell is considerably more user friendly and practical. The colour coding within the ipython shell makes it easier to read, and the ipython shell contains automatic indenting and syntax highlighting.

2. Python has a host of different data types that allow you to store and organize information. List 4 examples of data types that Python recognizes, briefly define them, and indicate whether they are scalar or non-scalar.

Integer - whole numbers - Scalar
Bool - Boolean, returns either True or False - Scalar
Tuples - linear arrays to store multiple types of data - Non-scalar
Lists - mutable arrays of data - Non-scalar

3. A frequent question at job interviews for Python developers is: what is the difference between lists and tuples in Python? Write down how you would respond.

Both Lists and Tuples are used to store collections of data/items but the key difference is that Lists are mutable, meaning that they can be modified once the list is created. Tuples are immutable, which means that once created, their elements cannot be changed. Visually in code, lists are defined using square brackets, while tuples use parentheses.

4. In the task for this Exercise, you decided what you thought was the most suitable data structure for storing all the information for a recipe. Now, imagine you're creating a

language-learning app that helps users memorize vocabulary through flashcards. Users can input vocabulary words, definitions, and their category (noun, verb, etc.) into the flashcards. They can then quiz themselves by flipping through the flashcards. Think about the necessary data types and what would be the most suitable data structure for this language-learning app. Between tuples, lists, and dictionaries, which would you choose? Think about their respective advantages and limitations, and where flexibility might be useful if you

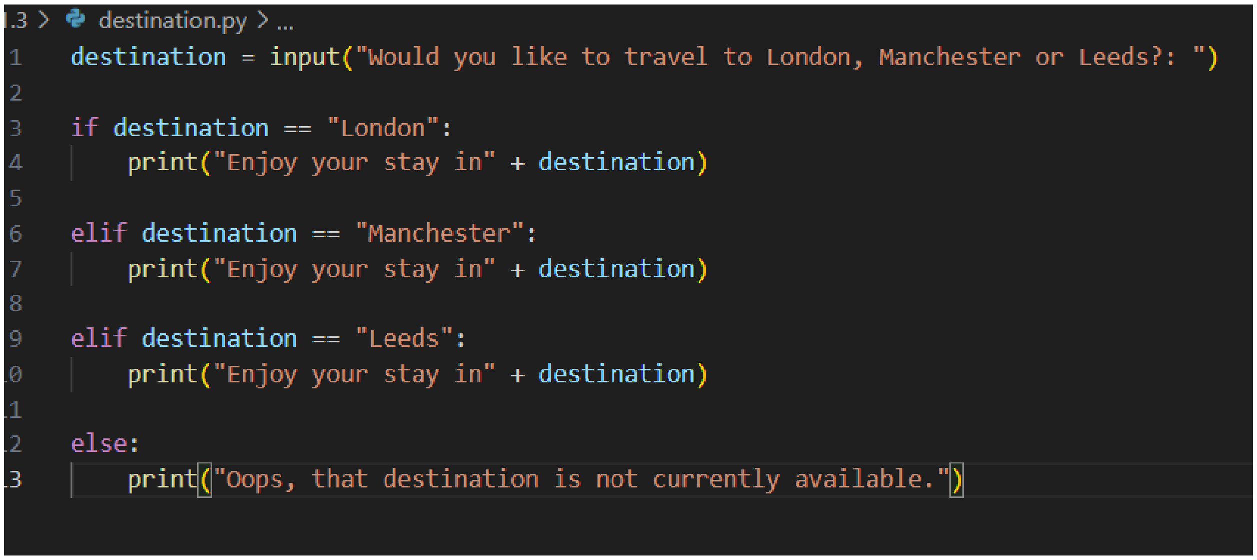
For the language-learning app, I would choose dictionaries as the most suitable data structure. Dictionaries can store the vocabulary words as keys and the corresponding definitions and categories as values. This allows for efficient lookup and easy organization of data. Additionally, dictionaries' flexibility allows users to update or add new words and definitions, enhancing the learning experience.

Exercise 1.3:

Learning Goals
● Implement conditional statements in Python to determine program flow
● Use loops to reduce time and effort in Python programming
● Write functions to organize Python code

Reflection Questions
1. In this Exercise, you learned how to use if-elif-else statements to run different tasks based on conditions that you define. Now practice that skill by writing a script for a simple travel app using an if-elif-else statement for the following situation:

● The script should ask the user where they want to travel.
● The user's input should be checked for 3 different travel destinations that you define.
● If the user's input is one of those 3 destinations, the following statement should be printed: "Enjoy your stay in ______!"
● If the user's input is something other than the defined destinations, the following statement should be printed: "Oops, that destination is not currently available."


.3 > destination.py > ...

```python
destination = input("Would you like to travel to London, Manchester or Leeds?: ")

if destination == "London":
    print("Enjoy your stay in" + destination)

elif destination == "Manchester":
    print("Enjoy your stay in" + destination)

elif destination == "Leeds":
    print("Enjoy your stay in" + destination)

else:
    print("Oops, that destination is not currently available.")
```

2. Imagine you're at a job interview for a Python developer role. The interviewer says "Explain logical operators in Python". Draft how you would respond.

Logical operators allow you to perform operations on boolean values, checking that multiple conditions are satisfied. Python supports three main logical operators - And, Or, Not.
The And operator returns True if both operands are True, otherwise, it returns False. Alternatively, the Or operator returns True if at least one of the operands is True. If both operands are False, it returns False. Finally, the Not operator negates the boolean value of its operation and returns True is the operation is False and vice versa.

3. What are functions in Python? When and why are they useful?

Similar to in other programming languages, functions within Python are reusable blocks of code, comprising sets of instructions designed to process and manipulate code, enabling you to achieve specific tasks or functionalities. You will find yourself creating functions for repetitive tasks, where repeatedly typing the code would be time consuming and they facilitate better code maintenance and debugging by breaking complex tasks into smaller, manageable units. .

4. In the section for Exercise 1 in this Learning Journal, you were asked in question 3 to set some goals for yourself while you complete this course. In preparation for your next mentor call, make some notes on how you've progressed towards your goals so far.

Out of the three goals shared above, I think I am understanding the principles behind the language and seeing solutions to tasks and issues without external assistance. Additionally, I have had a few absent thoughts on Python's application to other projects.

Exercise 1.4: File Handling in Python

Learning Goals

● Use files to store and retrieve data in Python

Reflection Questions

1. Why is file storage important when you're using Python? What would happen if you didn't store local files?

Local data storage is important when using Python, as it can help to prevent data loss, which would inturn, prevent your script from running. Data storage is an essential part of programming for modern, real-life applications.

2. In this Exercise you learned about the pickling process with the pickle.dump() method. What are pickles? In which situations would you choose to use pickles and why?

Pickles are binary files, containing complex data that has been turned into a stream of bytes that a computer can easily read. They are particularly useful in Python for serializing and deserializing Python objects, allowing data to be saved to disk and later retrieved in its original form, facilitating data storage and sharing across different Python programs and platforms.

3. In Python, what function do you use to find out which directory you're currently in? What if you wanted to change your current working directory?

To find out the current working directory, you can use the os.getcwd() function. To change the current working directory, you can use the os.chdir() function.

4. Imagine you're working on a Python script and are worried there may be an error in a block of code. How would you approach the situation to prevent the entire script from terminating due to an error?

When you want to prevent the entire Python script from terminating due to an error in a specific block of code, you can use a try-except block to catch and handle the error.

5. You're now more than halfway through Achievement 1! Take a moment to reflect on your learning in the course so far. How is it going? What's something you're proud of so far? Is there something you're struggling with? What do you need more practice with? Feel free to use these notes to guide your next mentor call.

I feel fine about my current progress on this course. I do not think there has been too much personal achievement to be proud of at this stage but I am enjoying learning Python.

Exercise 1.5: Object-Oriented Programming in Python

Learning Goals
● Apply object-oriented programming concepts to your Recipe app

Reflection Questions
1. In your own words, what is object-oriented programming? What are the benefits of OOP?

In Python, object-oriented programming structures code around objects, which contain data and behaviors. Classes define object blueprints. OOP models real-world ideas, promoting organized, reusable, and readable code. It simplifies complex tasks and enhances collaboration by representing actual concepts as objects.

2. What are objects and classes in Python? Come up with a real-world example to illustrate how objects and classes work.

Objects can be defined as a series of data and methods used to interact with the data, while classes act as a blueprint for creating objects.

If you considered different tiers of a football league to be classes, then each team within each league would be an object.

3. In your own words, write brief explanations of the following OOP concepts; 100 to 200 words per method is fine

Inheritance:
Inheritance enables the creation of new classes, based on existing classes and taking attributes from these classes. Inheritance only works downwards, with the child class inheriting from the parent class. This enables you to create a hierarchy, which imparts structure on your classes and promotes code reusability.
Polymorphism:
Polymorphism enables treating different class objects as instances of a shared superclass. By method overriding, it allows subclasses to provide unique method implementations, meaning objects respond differently to the same method call.
Operator Overloading:
Required for operators to work within your class, it lets operators have context-dependent meanings in user-defined classes. This also allows you to be flexible, as it allows operators to have different meanings and behaviors depending on the context of their usage.