JD Learning Journal

Exercise 1.1: Getting Started with Python

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?***

   Frontend, also known as client-side, is what the user sees. When developing the frontend of an application you'd be focusing on any functions that a user will interact with. This includes the basic webstack of HTML, CSS, and JS. Backend, on the other hand, can also be referred to as server-side development. In this role developers are focused on data handling and integrating the many pieces of an application.

   In a backend role one would be handling operations pertaining to database interactions and API creation/management.

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?***

   Both JavaScript and Python are very popular languages. As such they both offer a wide range of online communities and are constantly evolving to provide better features and modules.

   For this project, I'd propose Python over JavaScript for a couple of reasons. Python is user friendly and comes with a minimal learning curve. While JavaScript has many modules that can be installed, Python comes with a wide range of operations that are offered out of the box. Additionally, Python was built to be readable, which helps to make the language simple to pick up. This is opposed to JavaScript, which has a far steeper learning curve and requires a bit more skill when picking up on existing code bases.

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?***

    1. I want to learn how to take an idea and put it into practice with Python. There are often ideas for automation that I have and am currently unable to cleanly and concisely turn those ideas into a script.
    2. The main thing I want to get out of this Achievement is how I can use Python to automate tasks that would otherwise be manual and time consuming. This can be applied both to pet projects and in my professional projects.
    3. I see myself implementing my new Python skills in my current position as a Security Engineer. With these skills I'd be able to cut down on the time to complete tasks significantly, and improve current processes as well.

Exercise 1.2: Data Types in Python

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?***

   Python's default shell is great when it comes to actually executing python. However, iPython is far more user friendly. The iPython shell uses coloring to distinguish between data types. Additionally, iPythonclearly defines input and output, making it far easier to keep track of what's going on than with Python's default terminal.

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.***

Data type	Definition	Scalar or Non-Scalar?
Float	An integer	Scalar
Integer	A number	Scalar
List	Mutable, ordered sequence	Non-Scalar
Dictionary	Unordered list that stores key-value pairs	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 ordered structures, as such, they can both be indexed. However, they do differ in the fact that tuples are immutable while lists are mutable. While you can append a value onto a tuple, it doesn't quite work the same as with a list. When appending to a tuple, there is a new data structure created in storage that contains a copy of the original tuple plus the appended value(s). With a list, one can make changes to the original list by adding, deleting, and even modifying existing values.

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 were to continue developing the language-learning app beyond vocabulary memorization.

In this situation I'd use an inner and outer structure. The inner structure would contain the details of the words (vocabulary word, definition(s), part of speech, etc.). I would structure this using a dictionary. The details of each word is going to be mapped to a specific key, allowing flexibility when returning specific information to a user down the line. The outer structure would be a list. By adding each dictionary to a list, the user would be able to search through the list based on a specific key.

Exercise 1.3: Functions and Other Operations in Python

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."***

```
travel_location = input("Where would you like to travel? ")

locations = [
  "Dublin",
  "Thailand",
  "London"
]

if travel_location in locations:
  print("Enjoy your stay in " + travel_location + "!")
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.***

    In Python logical operators are used to determine if certain conditions are being met.
    - *and* - evaluates to true if all conditions have been met, otherwise it evaluates to false
    - *or* - evaluates to true if one or all conditions are met
    - *not* - results in the opposite logical evaluation
        - For example, the statement $True$ would resolve itself as **True**, while $not\ True$ would resolve as **False.**

3. ***What are functions in Python? When and why are they useful?***
    A function is essentially a set of instructions that execute lines of code in the order in which they're defined. Functions can be reused throughout your code and are, in general, a foundation of coding. Python gives developers access to a wide range of predefined functions, but also allows developers to define their own functions. They are particularly useful when it comes to repetition.

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.***
   A major goal I have for myself in this Achievement is to be able to transform an idea into code. Being able to think through logic and create a function out of those thoughts has been extremely helpful in beginning to achieve this goal. Creating a function from a prompt has helped me get closer to this goal and I've already been able to write short scripts to automate basic tasks in my day-to-day.

Exercise 1.4: File Handling in Python

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

   File storage is important in Python because apps require data to be persistent. With the recipe app for example, when a user enters a recipe it is expected that they will have the opportunity to view that recipe again in the future. Without file storage the recipes would be deleted from memory as soon as the application finished running. This would require the user to enter their recipes again and again each time they open the app, which defeats the very purpose of the application.

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?***

   A pickle is just a stream of bytes. The pickle package is used to transform Python objects into bytes, which is machine-readable. Pickle can also do the opposite, by transforming those byte streams back into human-readable objects. Pickling or unpickling data allows applications to ensure data persists.

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?***

   In Python one could use the **os** module to manage and traverse directories. $os.getcwd()$ can be used to get the current working directory. $os.chdir()$ can be used to change the directory on is currently working in.

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?***

   The de facto approach is to use a **try-except** block. This type of block is specifically designed for moments where a developer may think an error might occur in the code. The initial block (try) will do exactly that; it will try the code that follows. If an error is found it will be taken care of by the except block, which notifies the user of some sort of error without terminating the entire script. This allows for errors to be remediated rather than termination once the error has been found, which is far more practical for modern day applications.

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.***

   So far I'm pretty excited that things are clicking. When working through exercises I have a clear starting point and am able to start building out my scripts from there. The thing I'm struggling with is having confidence in my own knowledge base. I tend to second guess myself when it feels "too easy" and will turn to discussion boards or StackOverflow for confirmation rather than just trusting that my code is correct.

Exercise 1.5: Object-Oriented Programming in Python

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

   A major appeal to OOP is the ability to implement concepts of inheritance. Being able to create classes that inherit methods from parent classes makes code easier to read, reusable, and easier to maintain. OOP also keeps code maintained in smaller chunks, which makes building projects easier to manage and quicker to troubleshoot.

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

   A class can be looked at as a template. It defines the attributes and methods that can apply to objects created using the class. The object on the other hand, is what's created using that template. It can have it's own unique attributes, but at it's core it is working from it's designated class.

   A real world example could be something like a book at a bookstore. The Book would be the class and each book can have unique attributes like their name, author, genre, or date published. You can go one step further and consider some of the actions that can be performed on each book. Methods like *get_name(), get_author(),* or *get_date_published()* can all be performed on specific books.

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

Method	Description
Inheritance	Inheritance is an OOP concept that many find to be attractive. This allows one class (a child) to inherit properties from it's parent. Inheritance is the backbone of another concept in Pythong, hierarchical relationships. These concepts help developers avoid redundancy, making code cleaner, easier to read, and more manageable. Another key feature to this concept is the ability to override or extend inherited methods.
Polymorphism	Polymorphism gives classes the ability to use different implementations of the same method. Depending on where the method is defined, it may perform different actions specific to the class it's assigned to.
Operator Overloading	Operator Overloading provides the ability to define custom behaviors for built-in operators. This is applicable when defining your own Classes. This concept allows developers to elevate operators to be more beneficial when used on a custom Class. Python also makes it simple to understand what each custom operators by offering methods like __*add*__ or __eq__

Operator overloading, also known as **operator ad-hoc polymorphism**, is a feature in Python that allows you to define custom behavior for built-in operators (such as `+`, `-`, `*`, `/`, etc.) when they are used with objects of user-defined classes. This makes it possible to use operators with objects in a way that is meaningful and intuitive for the specific class, similar to how operators work with built-in data types like integers and strings.

In Python, operators are not inherently tied to specific objects but are instead implemented as **special methods**. These methods, often referred to as **magic methods** or **dunder methods** (because they are surrounded by double underscores, e.g., `__add__`, `__sub__`), enable you to define custom behavior for operators. By overloading these methods, you can enable objects of your class to interact with standard operators.