Reflection Questions

- 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."

Write your script here. (Hint: remember what you learned about indents!)

"

```
destinations = ["Paris", "Tokyo", "New York"]


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


if destination in destinations:
    print(f"Enjoy your stay in {destination}!")
else:
    print("Oops, that destination is not currently available.")
```

"

- 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 in Python are like puzzle pieces that let us put together different conditions to make decisions. Imagine you have three main types of puzzle pieces: and, or, and not.

      The and piece gives you a True result only if both conditions are true. If one or both conditions are false, it gives you False.

      The or piece gives you a True result if at least one condition is true. It's like saying, "either this or that, or maybe even both!"

      The not piece flips the truth. If something is true, not makes it false, and if it's false, not makes it true.

- What are functions in Python? When and why are they useful?
  - Functions in Python are blocks of reusable code that perform a specific task.
  - Functions allow you to write a piece of code once and reuse it multiple times
  - By breaking your code into smaller, manageable functions, you make it easier to understand, test, and debug. Each function can be developed and tested independently.
  - Well-named functions can serve as documentation, making it clear what a block of code is supposed to do.

- 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.