Exercise 1.3: Functions and Other Operations in Python

Learning Goals

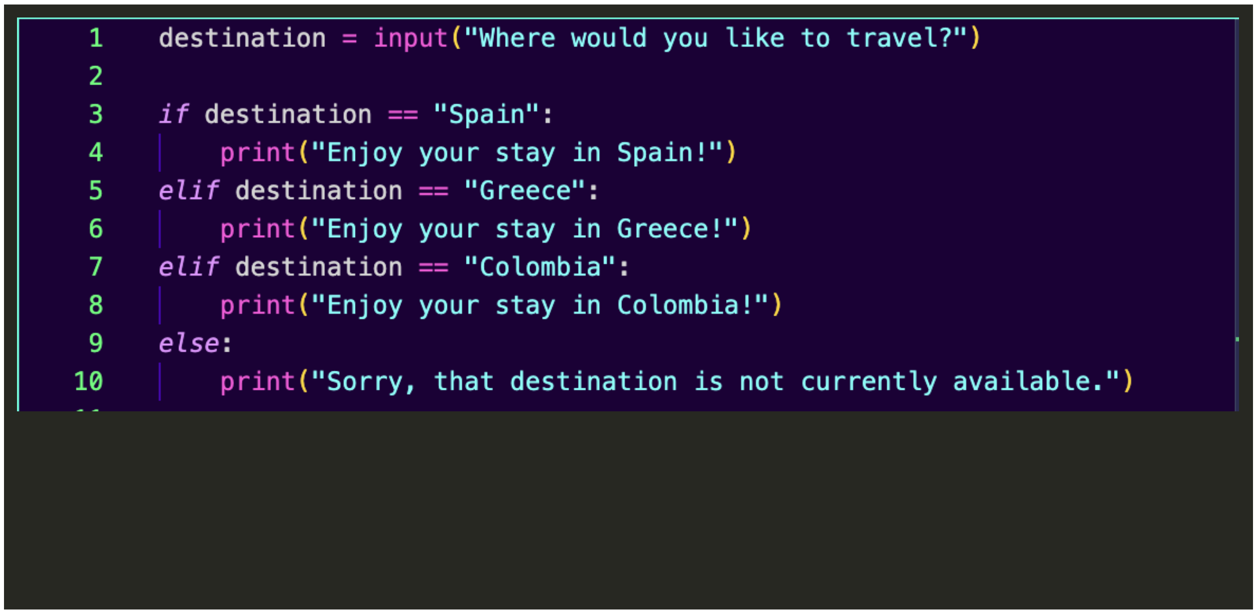
- Implement conditional statements in Python to determine program flow
- Use loops to reduce time and effort in Python programming
- Write functions to organize Python code

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."

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



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

if destination == "Spain":
    print("Enjoy your stay in Spain!")
elif destination == "Greece":
    print("Enjoy your stay in Greece!")
elif destination == "Colombia":
    print("Enjoy your stay in Colombia!")
else:
    print("Sorry, 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 in Python are tools that help us make decisions in code. "and" checks if both conditions are true, "or" checks if at least one is true, and "not" flips the truth value. They're like puzzle pieces for combining conditions to create smart code paths.

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

   Functions in Python are reusable blocks of code that perform specific tasks. They enhance code modularity, readability, and reusability. By encapsulating logic, functions reduce redundancy and promote easier debugging. They're crucial for managing complexity in programs, allowing tasks to be broken down into smaller, manageable chunks. This promotes collaboration, code organization, and maintenance. Functions are especially useful when a task needs to be executed multiple times, improving efficiency and maintainability.

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.

   I've learnt more on python and its syntax, so I'm definitely making progress towards those goals.