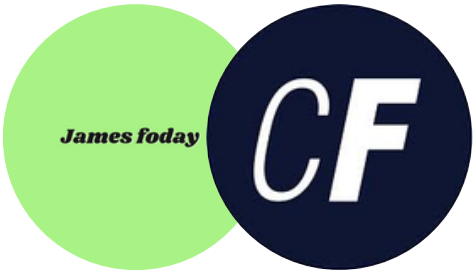
JAMES FODAY




# REFLECTION QUESTION

# Learning Journal - Exercise 1.3 Reflection Questions

### 1. Write a simple travel app script using if-elif-else statements:

The program asks the user where they want to travel and compares the input against three specific cities I chose. If the input matches any of these destinations, it prints a welcoming message. If not, it informs the user that the destination isn't available.

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

destination = input("Where do you want to travel? ")



if destination == destinations[0]:

    print("Enjoy your stay in Paris!")

elif destination == destinations[1]:

    print("Enjoy your stay in Tokyo!")

elif destination == destinations[2]:

    print("Enjoy your stay in New York!")

else:

    print("Sorry, we don't offer trips to that destination right now.")
```

### 2. Explain logical operators in Python as if in an interview:

Logical operators help combine multiple conditions and control the flow of decisions in Python programs.

- The **and** operator returns True only if *both* conditions are true. For example, x > 10 and x < 20 checks if x is between 10 and 20.
- The **or** operator returns True if *at least one* condition is true, like x < 5 or x > 15.
- The **not** operator reverses a condition, so not(x == 5) is True if x is not 5.

These operators are essential for complex condition checks within if statements and loops.

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

Functions are reusable blocks of code that perform a specific task. They allow programmers to avoid repeating code by writing it once and calling it whenever needed.

Functions improve readability and maintainability by isolating logic into named blocks. For example:

```
def add_numbers(a, b):
    return a + b
```

Using functions helps organize programs into smaller, testable parts, which is important for larger projects and collaborative development.

### 4. Progress towards my course goals so far:

I have strengthened my understanding of Python basics such as control flow and data handling. I'm getting comfortable with writing clean, modular code and applying concepts like conditionals and functions effectively. My next steps include learning about error handling and object-oriented programming in more depth, which will prepare me for building more complex applications like web apps with Django.

# Learning Journal - Exercise 1.4 Reflection Questions

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

File storage is essential because it allows programs to save data permanently, beyond the runtime of the program. Without storing data locally, all information processed or generated by a Python script would be lost as soon as the script ends. This would make it impossible to preserve user inputs, program states, or any results that need to persist over multiple program executions.

### 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 a way to serialize Python objects into a binary format that can be saved to a file and later reloaded to restore the original objects. This process is called pickling.

Pickles are useful when you want to save complex data structures like lists, dictionaries, or custom objects without manually converting them to text formats like JSON or CSV. They allow quick and easy persistence of Python objects, making them ideal for caching, saving program state, or transferring data between programs that use Python.

### 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 use:

```
import os
```

```
os.getcwd()
```

To change the current working directory, you use:

```
os.chdir('/path/to/new/directory')
```

This allows your script to read from or write to files relative to the new directory path.

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?

I would use a try-except block to catch exceptions that may occur in that code block. By handling errors explicitly, the program can respond gracefully (e.g., by logging the error, providing a user-friendly message, or attempting a fallback) without crashing.

```
try:
    # risky code here
except SomeError as e:
    print(f"An error occurred: {e}")
```

This approach ensures the rest of the program can continue running even if a specific part fails.

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, the course has been very engaging, and I'm pleased with how much I've learned about Python's fundamentals, including data types, control flow, functions, and basic file handling. I'm proud that I can now write scripts that collect user input, process data, and save it persistently.

I find exception handling and debugging to be areas where I still need more practice, as these are critical for building robust applications. Additionally, I want to improve my understanding of Python's object-oriented concepts.

Moving forward, I plan to focus more on error handling and testing, and I look forward to applying these skills in more complex projects like web applications with Django.

# Learning Journal - Exercise 1.5 Reflection Questions

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

Object-oriented programming (OOP) is a programming paradigm based on the concept of “objects”, which are instances of classes. Objects can contain both data (attributes) and functions (methods) that operate on that data. OOP helps organize code into modular, reusable components that model real-world entities more naturally.

The benefits of OOP include:

- **Encapsulation:** Bundling data and methods together to hide internal details and expose only necessary parts, which increases code security and clarity.
- **Reusability:** Classes and objects can be reused across programs, reducing duplication and speeding up development.
- **Inheritance:** New classes can inherit properties and behaviors from existing classes, allowing for easy extension and maintenance.
- **Polymorphism:** Objects of different classes can be treated through the same interface, enabling flexibility and integration.
- **Improved maintainability:** OOP makes complex programs easier to understand and modify by organizing code logically.

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

In Python, a **class** is a blueprint or template for creating objects. It defines attributes (data) and methods (functions) that characterize the objects. An **object** is an instance of a class, containing actual values for the attributes and able to invoke the methods defined by its class.

**Real-world example:**
Consider a class called Car. This class might have attributes like color, make, and model, and methods like start_engine() and stop_engine(). Each individual car you own or see on the road is an object (instance) of the Car class. For example, a red Toyota Corolla is one object, and a blue Ford Mustang is another. They share the same structure (class) but have different attribute values.

**3. In your own words, write brief explanations of the following OOP concepts: Inheritance, Polymorphism, Operator Overloading.**

| Method | Description |
| --- | --- |
| Inheritance | Inheritance allows a new class (called a subclass or child class) to acquire properties and methods from an existing class (called the superclass or parent class). This promotes code reuse and allows specialized behavior by extending or overriding the base class. For example, a Truck class can inherit from a general Vehicle class but add features unique to trucks. |
| Polymorphism | Polymorphism means “many forms”. It allows objects of different classes to be treated through the same interface, typically by implementing methods with the same name but different behaviors. This enables writing flexible and extensible code. For instance, both Cat and Dog classes might have a speak() method that behaves differently, but can be called in the same way on any animal object. |
| Operator Overloading | Operator overloading allows developers to define custom behavior for built-in operators (like +, -, *, etc.) when applied to objects of user-defined classes. This lets objects interact using natural syntax. For example, a Vector class might overload the + operator to perform vector addition instead of concatenation or numeric addition. |

# Learning Journal - Exercise 1.6 Reflection Questions

### 1. What are databases and what are the advantages of using them?

Databases are organized collections of data stored electronically, allowing efficient access, management, and updating of information. They provide a structured way to store vast amounts of data that can be quickly retrieved and manipulated using query languages like SQL.

Advantages of using databases include:

- **Data Integrity:** Ensures accuracy and consistency of data through constraints and relationships.
- **Efficient Data Management:** Enables fast querying, updating, and reporting.
- **Multi-user Access:** Supports concurrent access by multiple users while maintaining data security.
- **Scalability:** Can handle large volumes of data growing over time.
- **Backup and Recovery:** Offers mechanisms to safeguard data against loss or corruption.

2. List 3 data types that can be used in MySQL and describe them briefly:

| Data type | Definition |
| --- | --- |
| INT | Used to store integer numbers (whole numbers) without decimals. Often used for IDs or counts. |
| VARCHAR(n) | Variable-length string that can store up to n characters. Used for text data such as names or descriptions. |
| DATE | Stores date values in 'YYYY-MM-DD' format, used for birthdates, timestamps, or other date-related data. |

## 3. In what situations would SQLite be a better choice than MySQL?

SQLite is a better choice than MySQL in scenarios where:

- The application is lightweight, embedded, or requires minimal setup.
- There is a need for a self-contained, serverless, zero-configuration database.
- You want to avoid the overhead of managing a separate database server.
- The application does not require high concurrency or complex multi-user access.
- Development or testing environments where simplicity and portability are preferred.

SQLite is ideal for small to medium-sized applications, mobile apps, or desktop software, while MySQL suits larger, multi-user, web-based, or enterprise applications.

## 4. Think back to what you learned in the Immersion course. What do you think about the differences between JavaScript and Python as programming languages?

JavaScript is primarily designed for client-side web development, enabling interactive web pages, while Python is a general-purpose programming language favored for its readability and simplicity.

JavaScript is event-driven and asynchronous by nature, making it suitable for responsive UI, whereas Python excels in backend development, data science, automation, and scripting.

Python's syntax is cleaner and easier for beginners, with extensive libraries for various domains. JavaScript runs in browsers natively, while Python requires an interpreter or runtime environment. Both have strong ecosystems but differ in typical use cases: JavaScript in frontend and full-stack web, Python in backend, scientific computing, and automation.

# Exercise 1.7: Finalizing Your Python Program

### 1. What is an ORM and advantages?

An ORM (Object-Relational Mapper) allows developers to interact with databases using object-oriented code instead of raw SQL. Advantages include:

- Simplified database operations using Python syntax
- Database abstraction making it easier to switch DBMS
- Protection against SQL injection
- Easier maintenance and readability

### 2. How did creating the Recipe app go?

The project helped me consolidate my skills in Python, database interaction, and application logic. I successfully built CRUD features and handled errors. I enjoyed structuring the app clearly.

### 3. How to respond to interview questions on app development experience?

I would explain my experience building a Recipe app using Python and SQLAlchemy, implementing database models, handling user input, and ensuring data persistence. I would emphasize problem-solving and following best practices.

### 4. Reflect on your learning so far

- What went well: Learning database integration and OOP concepts
- Proud of: Building a functional app and handling edge cases
- Challenges: Managing SQLAlchemy sessions and debugging queries
- Expectations met: Yes, I feel more confident in backend development
- Keep in mind: Continue improving testing, documentation, and deployment skills