Lecture 2.5

Lecture 5: OOP Concepts, Class and Object

Agenda

- What is Object-Oriented Programming (OOP)?
- The Four Pillars of OOP
- Understanding Classes
- Understanding Objects
- Classes and Objects in Python

Recap: Python Data Structures

- Lists, Tuples, Sets, and Dictionaries
- Arrays in Python using the array module and NumPy
- Differences between Lists and Arrays

What is Object-Oriented Programming?

- A programming paradigm based on the concept of 'objects'.
- Objects can contain data (attributes) and code (methods).
- Focuses on structuring software as reusable pieces of code blueprints.

Why Use OOP?

- Modularity for easier troubleshooting.
- Reuse of code through inheritance.
- Flexibility through polymorphism.
- Effective problem-solving for complex systems.

The Four Pillars of OOP

- Encapsulation
- Abstraction
- Inheritance
- Polymorphism

Pillar 1: Encapsulation

- Bundling of data and methods into a single unit.
- Restricting direct access to some of an object's components.
- Helps prevent accidental interference and misuse.

Pillar 2: Abstraction

- Hiding complex implementation details.
- Showing only the essential features of an object.
- Simplifies interaction for the user.

Pillar 3: Inheritance

- Mechanism where a new class derives from an existing class.
- Promotes code reusability.
- Establishes a hierarchical relationship between classes.

Pillar 4: Polymorphism

- The ability of different objects to respond to the same method call in their own way.
- Allows methods to do different things based on the object it is acting upon.
- Enhances flexibility in code.

What is a Class?

- A user-defined blueprint or prototype.
- Defines a set of attributes that characterize any object of the class.
- It does not allocate memory when created.

Real-world Analogy of a Class

- Class: Car
- Attributes: Color, Brand, Speed
- Methods: Accelerate, Brake, Turn

What is an Object?

- A specific instance of a class.
- It contains real values instead of variables.
- Memory is allocated when an object is created.

Real-world Analogy of an Object

- Object 1: Red Toyota Corolla
- Object 2: Blue Ford Mustang
- Object 3: Black Honda Civic

Defining a Class in Python

- Use the 'class' keyword followed by the class name.
- Class names typically use CamelCase.

Creating Objects in Python

- Instantiate an object by calling the class name like a function.

Adding Attributes and Methods

- Methods are defined as functions inside a class.
- The first parameter of methods is typically 'self'.

Summary

- OOP structures code into classes and objects.
- Four pillars: Encapsulation, Abstraction, Inheritance, Polymorphism.
- Class = Blueprint.
- Object = Specific Instance.

Next Lecture

- Constructors in Python
- The __init__ method
- Types of methods (Instance, Class, Static)