Lecture slides - Week 7

Object Oriented Programming using Python - The Basics

Dr. Aamir Akbar

Director of both AWKUM AI Lab and AWKUM Robotics, Final Year Projects (FYPs) coordinator, and lecturer at the department of Computer Science
Abdul Wali Khan University, Mardan (AWKUM)

Contents

Introduction

Overview of Programming Paradigms

Procedural Programming: This paradigm focuses on writing procedures or functions that are executed sequentially. It is based on the concept of procedures and data structures. *C* is an example of a language that primarily follows procedural programming.

Functional Programming: Functional programming emphasizes treating computation as the evaluation of mathematical functions and avoids changing state and mutable data. Languages like *Haskell, Lisp, and Erlang* are examples of functional programming languages.

Object-Oriented Programming (OOP): OOP is a programming paradigm that organizes data and behavior into objects. Python, Java, C# are all object-oriented languages.

Object Oriented Programming (OOP)

Object-Oriented Programming (OOP)

Data: In OOP, data refers to the attributes or properties of an object. These attributes represent the state of the object and are stored as variables within the object. Data can include information like the object's characteristics, properties, and values. For example, in a class representing a "Car," data attributes could include "make," "model," and "year."

Behavior: Behavior in OOP is defined by the methods or functions associated with an object. Methods represent the actions that an object can perform or the operations it can undergo. These methods operate on the object's data, and they encapsulate the object's functionality. Using the "Car" class example, behavior can be defined by methods like "start_engine," "accelerate," and "brake."

Why OOP?

- **Modularity**: OOP promotes modularity, allowing you to break down complex systems into smaller, manageable parts called objects. Each object represents a self-contained module that can be developed and tested independently. This makes code more organized and easier to maintain.
- **Collaboration**: OOP is well-suited for modeling real-world systems and relationships. By representing entities as objects and their interactions through methods, OOP aligns closely with how we think about and describe systems, making it an intuitive approach for many software development tasks.

Four Pillars of OOP? i

The four pillars of Object-Oriented Programming (OOP) are a set of fundamental principles that guide the design and implementation of object-oriented systems. They are also often referred to as the four core concepts of OOP. These pillars are:

- **Encapsulation**: OOP provides encapsulation, which means bundling data (attributes) and methods (functions) together within an object. This protects the integrity of data by preventing unauthorized access and modification. Encapsulation enhances data security and maintainability.
- **Inheritance**: Inheritance is a fundamental OOP concept that allows you to create new classes based on existing ones. This promotes code reuse and helps you establish hierarchies of classes, making it easier to model real-world relationships and concepts.

Four Pillars of OOP? ii

- **Polymorphism**: Polymorphism means "many shapes" and is a feature that allows objects of different classes to be treated as objects of a common base class. This flexibility enables you to write code that can work with a variety of objects without needing to know their specific types.
- **Abstraction**: OOP allows you to abstract complex systems into simpler, more understandable representations. By creating classes that hide the underlying complexity, developers can work with high-level concepts and interfaces, making it easier to design and understand code.

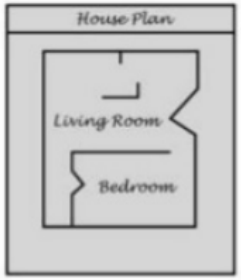
OOP in Real Life

Class Vs. Object

Class

3 objects / instances / individuals

Blueprint that describes a house



Instances of the house described by the blueprint

OOP with a Taxi Example

- To learn OOP, we will use an example of a Taxi.

Taxi Object - Data and Behavior

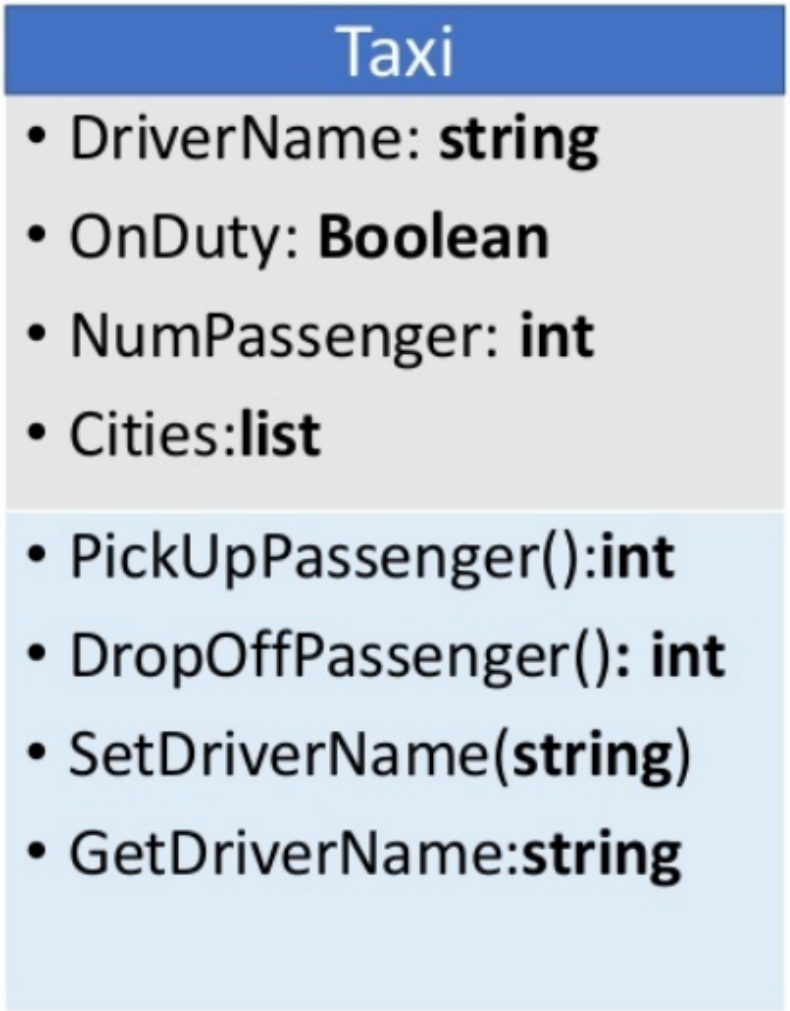
Example Object - Taxi

Every object has two main components:

- Data (the attributes about it)
- Behavior (the methods)

DATA	BEHAVIOR
• DriverName	• PickUpPassenger
• OnDuty	• DropOffPassenger
• NumPassenger	• SetDriverName
• Cities	• GetDriverName

Taxi Class Diagram (UML)



Person Class