OBJECT ORIENTED PROGRAMMING

INTRODUCTION

- Object-Oriented Programming (OOP) in Python is a programming paradigm that organizes software design around data, or objects, rather than functions and logic.

- OOP empowers developers to build modular, maintainable, and scalable applications. By understanding the core OOP principles, programmers can leverage Python's OOP capabilities to design elegant and efficient solutions to complex problems.

- OOP improves code flexibility, scalability, and productivity. It allows for better organization of code, making it easier to understand, modify, and extend.

CORE CONCEPTS

Classes

A class is a blueprint for creating objects. It defines the properties (attributes) and behaviors (methods) that an object of that class should have.
Example: A Car class might have attributes like color, model, and year, and methods like start_engine() and stop_engine().

Objects

An object is an instance of a class. It represents a real-world entity with specific attribute values and can perform actions defined by the class's methods.
Example: Creating a my_car object from the Car class would give us a specific car with its own color, model, year, and the ability to start and stop its engine.

CLASS

```python
# Define a class named "Person"
class Person:
     # Constructor method - called when an object is
instantiated
    def __init__(self, name, age):
        self.name = name                # Instance variable
        self.age = age                  # Instance variable

    # Method to display person's information
    def introduce_yourself(self):
        print(f"Hello, my name is {self.name} and I am
{self.age} years old.")

# Create an instance of the Person class
person1 = Person("Alice", 30)
# Call the method to display the person's information
```

INHERITANCE

- ❑ Inheritance is a mechanism in OOP where a new class (derived class or subclass) is created from an existing class (base class or superclass).

- ❑ The derived class inherits the properties and methods of the base class, allowing for code reuse and reducing redundancy.

- ❑ This establishes a hierarchical relationship between classes, enabling more specific behaviors to be built upon more general ones.

- ❑ Inheritance supports various types including single, multiple, hierarchical, multilevel, and hybrid inheritance.

POLYMORPHISM

- Polymorphism refers to the ability of objects to take on many forms.

- Specifically, it allows a single function or method to perform differently based on the object it is called on, enabling objects of different classes to be treated as objects of a common superclass.

- This is achieved through two main types: compile-time (static) polymorphism, typically through method overloading, and runtime (dynamic) polymorphism, usually through method overriding.

- Python only supports runtime

THANK YOU