Tops Technology

Module 15) Advance Python Programming

Presented By :

Nandni Vala

Class and Object (OOP Concepts)

- 1.Understanding the concepts of classes, objects, attributes, and methods in Python.
- **Classes**
- A **class** is a blueprint or template for creating objects.
- Defines the attributes (data) and methods (functions) that its objects will have.
- Example:
- class Car:
-     # Class definition
-     pass
- **Objects**
- An **object** is an instance of a class.
- Represents a specific realization of the class blueprint.
- Example:
- my_car = Car() # Object of the Car class

➢ **Attributes**

➢ **Attributes** are variables associated with a class or its objects.

➢ They represent the properties of an object (e.g., color, model, speed).

➢ Attributes can be:

➢ **Instance Attributes**: Unique to each object.

➢ **Class Attributes**: Shared among all objects of the class.

➢ Example:

```
➢ class Car:
➢     wheels = 4  # Class attribute (common to all cars)
➢     def __init__(self, color, model):
➢         self.color = color  # Instance attribute
➢         self.model = model  # Instance attribute
➢ # Creating objects
➢ car1 = Car("Red", "Sedan")
➢ car2 = Car("Blue", "SUV")
➢ print(car1.color)  # Output: Red (unique to car1)
➢ print(Car.wheels)  # Output: 4 (common for all cars)
```

Methods

- **Methods** are functions defined inside a class that operate on the attributes of the object.
- Common types of methods:
- **Instance Methods**: Operate on instance attributes.
- **class Methods**: Operate on class attributes.
- **Static Methods**: General utility functions that do not operate on class or instance attributes.
- Example:

```
class Car:
    def __init__(self, color, model):
        self.color = color
        self.model = model
    def display_info(self):  # Instance method
        print(f"Color: {self.color}, Model: {self.model}")
# Creating an object and calling a method
car1 = Car("Red", "Sedan")
car1.display_info()  # Output: Color: Red, Model: Sedan
```

2. Difference between local and global variables.

Feature	Local Variable	Global Variable
Defined in	Inside a function or block	Outside all functions or at the top level
Scope	Accessible only within the defining function	Accessible throughout the program
Lifetime	Created and destroyed within a function call	Exists for the entire program execution
Default Behavior	Local unless specified otherwise	Global unless shadowed by a local variable