Tops Technology

Module 15) Advance Python Programming

Presented By :

Nandni Vala

Method Overloading and Overriding

1.Method overloading: defining multiple methods with the same name but different parameters.

- **method overloading** (defining multiple methods with the same name but different parameters) is not natively supported as it is in some other languages like Java. However, you can simulate method overloading using:
- **Default Arguments**: Define a method with default values for parameters to allow flexibility in the number of arguments.
- Example:
- def add(a, b=0, c=0):
- return a + b + c
- **Variable-Length Arguments (*args, **kwargs)**: Use *args (for an arbitrary number of positional arguments) and **kwargs (for keyword arguments) to handle different numbers and types of parameters.
- Example:
- def add(*args):
- return sum(args)

- **Conditional Logic**: Use if statements inside the method to handle different parameter types or counts.
- **Example :**
- def operate(a, b=None):
- if b is None:
- return a * a # Square of a
- return a + b # Sum of a and b

2.Method overriding: redefining a parent class method in the child class.

- **Method overriding** in Python occurs when a child class defines a method with the same name and parameters as a method in the parent class, effectively replacing the parent class method with a new version.
- **Key Points:**
- The child class method **replaces** the parent class method when called on an instance of the child class.
- The method signature (name and parameters) in the child class must match the parent class method.

- You can still call the parent class method using super() if you want to extend or reuse its behavior.
- **Example:**
- class Parent:
- def greet(self):
- print("Hello from Parent")

- class Child(Parent):
- def greet(self):
- print("Hello from Child")

- obj = Child()
- obj.greet() # Output: Hello from Child