TOPSTechnologies

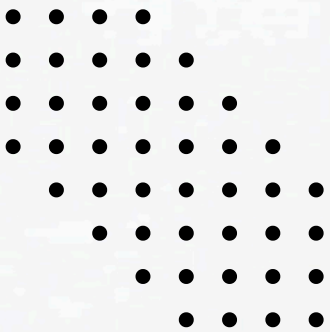
Class and Object (OOP Concepts)

Presented for :

TOPs Technologies

Presented by :

Sumit B Yadav



Advance Python Programming

Que 1

Python is an object-oriented programming (OOP) language, which means it supports the creation and use of classes and objects.

1. Classes

A class is a blueprint or template for creating objects. It defines a set of attributes and methods that an object of that class will have.

basic syntax:

```
class Car:
pass  # An empty class for now
```

2. Objects

An object is an instance of a class. When a class is defined, no memory is allocated until an object is created.

basic syntax:

```
my_car = Car()  # Creating an object of the Car class
print(my_car)  # Output: <__main__.Car object at 0x...>
```

3. Attributes

Attributes are variables that belong to an object. They store data related to the object.

```
class Car:
def __init__(self, brand, model, year):
self.brand = brand  # Attribute
self.model = model  # Attribute
self.year = year    # Attribute
```

- __init__() is a constructor method that initializes the object's attributes when an object is created.
- self refers to the instance of the class (the object).

4. Methods

Methods are functions that belong to a class and define behaviors for the objects.

Adding Methods to a Class:

```
class Car:
def __init__(self, brand, model, year):
self.brand = brand
self.model = model
self.year = year

def start_engine(self):
print(f"The {self.brand} {self.model} engine has started!")

# Creating an object and calling a method
car2 = Car("Honda", "Civic", 2023)
car2.start_engine()  # Output: The Honda Civic engine has started!
```

Que. 2

1. Global Variables

- A global variable is declared outside of a function or class.
- It can be accessed and modified from anywhere in the script.
- To modify a global variable inside a function, the global keyword is required.

basic syntax:

```
x = 10  # Global variable

def display():
print("Global x:", x)  # Accessing global variable

display()  # Output: Global x: 10
```

2. Local Variables

- A local variable is declared inside a function and can only be accessed within that function.
- It cannot be accessed outside the function where it is defined.

```
def my_function():
    z = 30  # Local variable
    print("Inside function:", z)

my_function()
# print(z)  # This will cause an error because 'z' is not defined outside the function.
```