**Python Class and Object:**

A class is a user-defined blueprint or prototype from which objects are created. Classes provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new instances of that type to be made. Each class instance can have attributes attached to it for maintaining its state. Class instances can also have methods (defined by their class) for modifying their state.

To understand the need for creating a class and object in Python let’s consider an example, let’s say you wanted to track the number of dogs that may have different attributes like breed and age. If a list is used, the first element could be the dog’s breed while the second element could represent its age. Let’s suppose there are 100 different dogs, then how would you know which element is supposed to be which? What if you wanted to add other properties to these dogs? This lacks organization and it’s the exact need for classes.

**Syntax:** Class Definition

class ClassName:

# Statement

## **Object of Python Class**

In Python, an object is a core concept of object-oriented programming (OOP). It is a self-contained unit that consists of both data (attributes) and functions (methods) that work on that data. An object is an instance of a class, where a class serves as a blueprint or template to define the structure and behavior of objects.

**Example 1: Bank Account Class**

class BankAccount:

def \_\_init\_\_(self, account\_number, holder\_name, balance=0):

self.account\_number = account\_number

self.holder\_name = holder\_name

self.balance = balance

def deposit(self, amount):

self.balance += amount

print(f"Deposited {amount}. Current balance: {self.balance}")

def withdraw(self, amount):

if amount <= self.balance:

self.balance -= amount

print(f"Withdrew {amount}. Current balance: {self.balance}")

else:

print("Insufficient balance.")

# Creating BankAccount objects

account1 = BankAccount(123456789, "John Doe", 1000)

account2 = BankAccount(987654321, "Jane Smith")

account1.deposit(500) # Output: Deposited 500. Current balance: 1500

account1.withdraw(200) # Output: Withdrew 200. Current balance: 1300

account2.deposit(1000) # Output: Deposited 1000. Current balance: 1000

account2.withdraw(1500) # Output: Insufficient balance.

**Example 2:** Rectangle Class for Area and Perimeter

class Rectangle:

def \_\_init\_\_(self, length, width):

self.length = length

self.width = width

def area(self):

return self.length \* self.width

def perimeter(self):

return 2 \* (self.length + self.width)

# Creating Rectangle objects

rect1 = Rectangle(5, 3)

rect2 = Rectangle(10, 7)

print("Rectangle 1 - Area:", rect1.area()) # Output: Rectangle 1 - Area: 15

print("Rectangle 1 - Perimeter:", rect1.perimeter()) # Output: Rectangle 1 - Perimeter: 16

print("Rectangle 2 - Area:", rect2.area()) # Output: Rectangle 2 - Area: 70

print("Rectangle 2 - Perimeter:", rect2.perimeter()) # Output: Rectangle 2 - Perimeter: 34

**Example 3:** Student Class with Methods for Grading

class Student:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

self.grades = []

def add\_grade(self, grade):

self.grades.append(grade)

def average\_grade(self):

if not self.grades:

return 0

return sum(self.grades) / len(self.grades)

# Creating Student objects

student1 = Student("Alice", 20)

student2 = Student("Bob", 22)

student1.add\_grade(85)

student1.add\_grade(90)

student2.add\_grade(78)

student2.add\_grade(82)

print(f"{student1.name}'s average grade:", student1.average\_grade()) # Output: Alice's average grade: 87.5

print(f"{student2.name}'s average grade:", student2.average\_grade()) # Output: Bob's average grade: 80.0

**Example 1:** List Object

# List object representing a list of fruits

fruits = ['apple', 'banana', 'orange', 'kiwi']

# Accessing elements of the list object

print(fruits[0]) # Output: apple

print(fruits[2]) # Output: orange

# Modifying the list object

fruits.append('grape')

print(fruits) # Output: ['apple', 'banana', 'orange', 'kiwi', 'grape']

# Iterating through the list object

for fruit in fruits:

print(fruit)

# Output:

# apple

# banana

# orange

# kiwi

# grape

**Example 2:** String Object

# String object representing a sentence

sentence = "Hello, how are you?"

# Accessing characters of the string object

print(sentence[0]) # Output: H

print(sentence[7]) # Output: w

# Modifying the string object

new\_sentence = sentence.replace('you', 'they')

print(new\_sentence) # Output: Hello, how are they?

# Checking if a substring is present in the string object

if 'how' in sentence:

print("Substring found.")

else:

print("Substring not found.")

# Output: Substring found.

**Example 3:** Dictionary Object

# Dictionary object representing a person's information

person = {

'name': 'John Doe',

'age': 30,

'occupation': 'Engineer',

'city': 'New York'

}

# Accessing values of the dictionary object

print(person['name']) # Output: John Doe

print(person['occupation']) # Output: Engineer

# Modifying the dictionary object

person['age'] = 31

person['city'] = 'San Francisco'

print(person)

# Output: {'name': 'John Doe', 'age': 31, 'occupation': 'Engineer', 'city': 'San Francisco'}