**Class vs. Object in Python**

In Python, a **class** is a blueprint or template for creating objects. It defines the structure and behavior that the objects instantiated from it will have. Essentially, a class outlines the attributes (data) and methods (functions) that objects of the class will possess, much like how a blueprint provides the design for a building.

On the other hand, an **object** is an instance of a class. It represents a specific realization of the class with actual data values. Each object can have its own unique set of attribute values but utilizes the structure and behavior defined by its class. To illustrate, think of an object as a house built from the blueprint provided by the class. Each house (object) can have different features, such as color and size, but they all follow the same design outlined by the blueprint (class).

For example, consider a **Dog** class that defines attributes like **Name** and **Breed**, and a method like **Bark**. When you create an object such as **Dog1** or **Dog2** from this class, each object can have its own specific Name and Breed values but will share the same **Bark method**. This distinction emphasizes that while a class provides the structure and behavior, objects are the individual instances that embody these attributes and functionalities.

**EXAMPLE**

class Dog:

def \_\_init\_\_(Saif, Name, Breed):

Saif.Name = Name

Saif.Breed = Breed

def Bark(Saif):

return f"{Saif.Name} says woof!"

Dog1 = Dog("Buddy", "Golden Retriever")

Dog2 = Dog("Max", "Bulldog")

print(Dog1.Bark())

print(Dog2.Bark())

**Constructor Method (\_\_init\_\_) vs. \_\_str\_\_() Function**

In Python, the **\_\_init\_\_ method** initializes an object’s attributes when it is created, setting up its initial state. For example, in a Car class, \_\_init\_\_ sets values for Make, Model, and Year when a new Car object is instantiated.

The **\_\_str\_\_ method** returns a human-readable string representation of the object, useful for display purposes. It is called by functions like print(). For the same Car class, \_\_str\_\_ might return a string like "2020 Toyota Corolla" to present the car’s details in a readable format.

**EXAMPLE**

class Car:

def \_\_init\_\_(Saif, Make, Model, Year):

Saif.Make = Make

Saif.Model = Model

Saif.Year = Year

def \_\_str\_\_(Saif):

return f"{Saif.Year} {Saif.Make} {Saif.Model}"

Car1 = Car("Toyota", "Corolla", 2020)

print(Car1.Make)

print(Car1.Model)

print(Car1.Year)

print(Car1)