# Python

Full stack Skills Bootcamp

Step8Up Academy

# Introducing Python Classes

## What is Class?

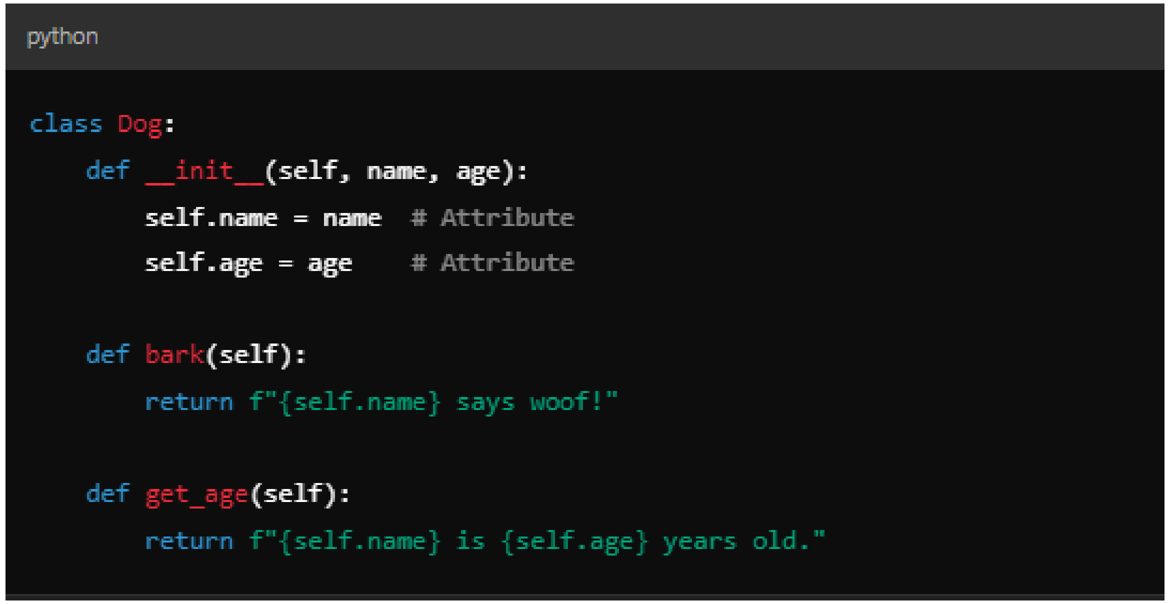
- A class is a fundamental concept in Object-Oriented Programming (OOP). It acts as a template or blueprint for creating objects.
- Attributes represent the data that describes the object.
- Methods are functions defined inside the class that define the object's behaviour.

__init__(self, name, age):

- The constructor method used to initialize the attributes name and age when a new object is created.
- The self-parameter refers to the current instance of the class. It is how the object keeps track of its attributes and methods.

Attributes like name and age define the state of the object.

Methods like bark() and get_age() define what actions the object can perform.


python

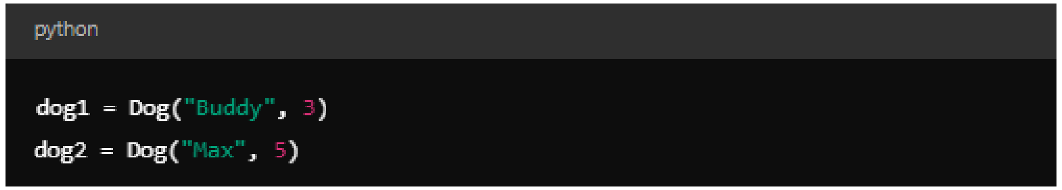
```python
class Dog:
    def __init__(self, name, age):
        self.name = name  # Attribute
        self.age = age    # Attribute

    def bark(self):
        return f"{self.name} says woof!"

    def get_age(self):
        return f"{self.name} is {self.age} years old."
```

Objects can have different values for their attributes.

## Creating Objects

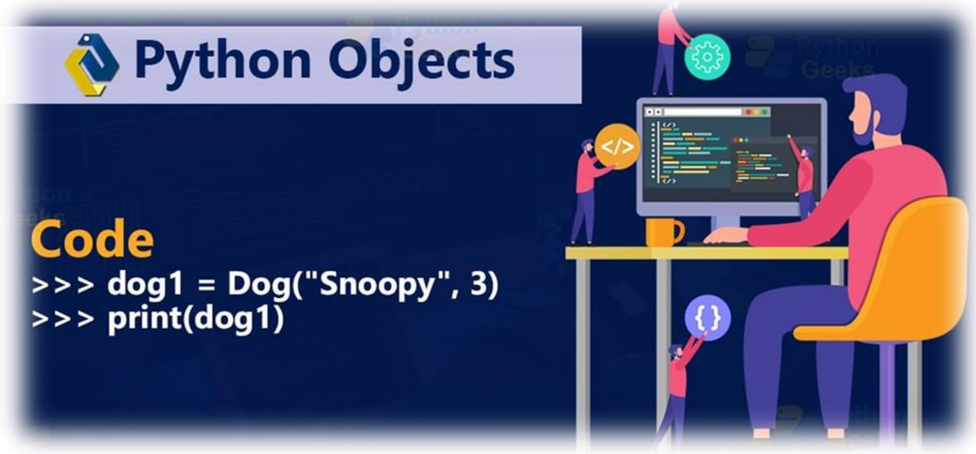

python

```python
dog1 = Dog("Buddy", 3)
dog2 = Dog("Max", 5)
```

- An object is a specific instance of a class. It represents a real-world entity created using the class as a template. Every object has its own set of attributes, which are initialized through the class constructor.

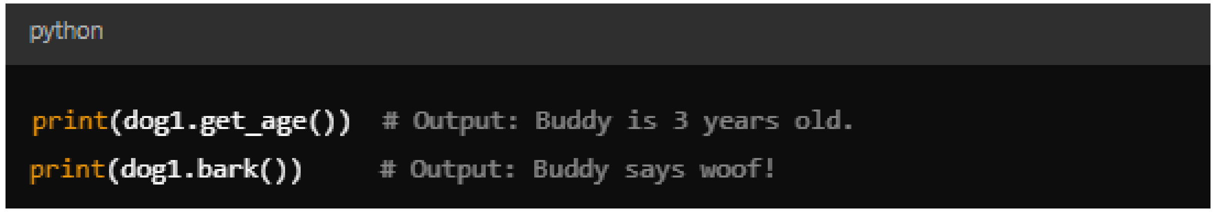
- Each object can have unique values for its attributes, even though both objects are instances of the same class.

- When creating an object, the __init__ method is automatically called to initialize the object's attributes.



```
>>> dog1 = Dog("Snoopy", 3)
>>> print(dog1)
```

## Accessing Attributes and Methods



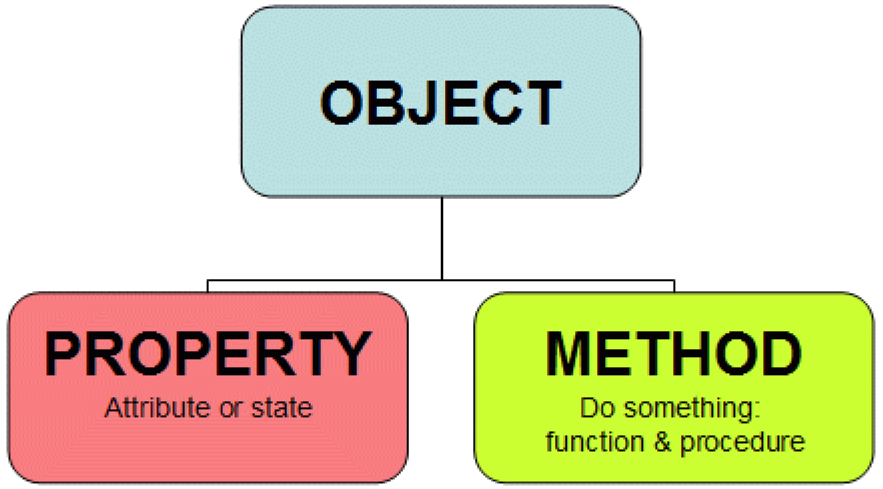
```python
print(dog1.get_age())  # Output: Buddy is 3 years old.
print(dog1.bark())     # Output: Buddy says woof!
```

- You can call methods of an object using the dot notation.

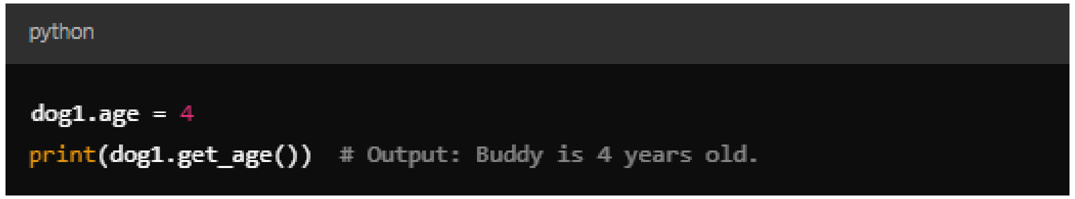
Calling Methods:

- dog1.get_age() accesses the get_age() method of the dog1 object, returning a string with the dog's name and age.
- dog1.bark() invokes the bark() method of dog1, returning the string "Buddy says woof!".

Dot Notation: In Python, you use the dot notation (.) to access methods or attributes of an object.



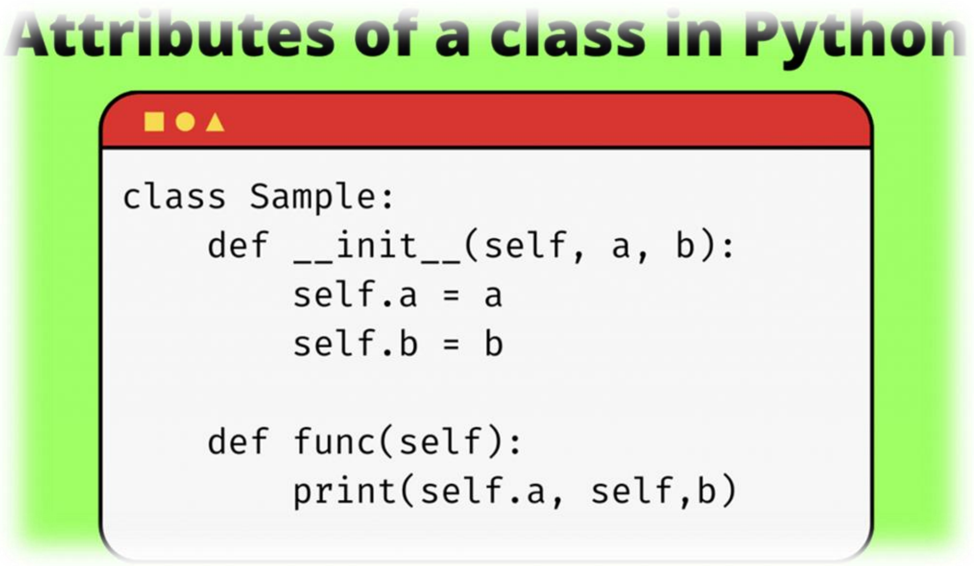
## Modifying Attributes



```python
dog1.age = 4
print(dog1.get_age())  # Output: Buddy is 4 years old.
```

- After creating an object, you can directly modify the object's attributes by assigning new values.

- The attribute age of dog1 was originally set to 3 when the object was created.
- Now, we’ve modified the age attribute of dog1 to 4. When we call dog1.get_age() again, it reflects the updated age.
- Modifying attributes allows us to change the internal state of an object dynamically after creation.


Attributes of a class in Python

```
class Sample:
    def __init__(self, a, b):
        self.a = a
        self.b = b

    def func(self):
        print(self.a, self,b)
```

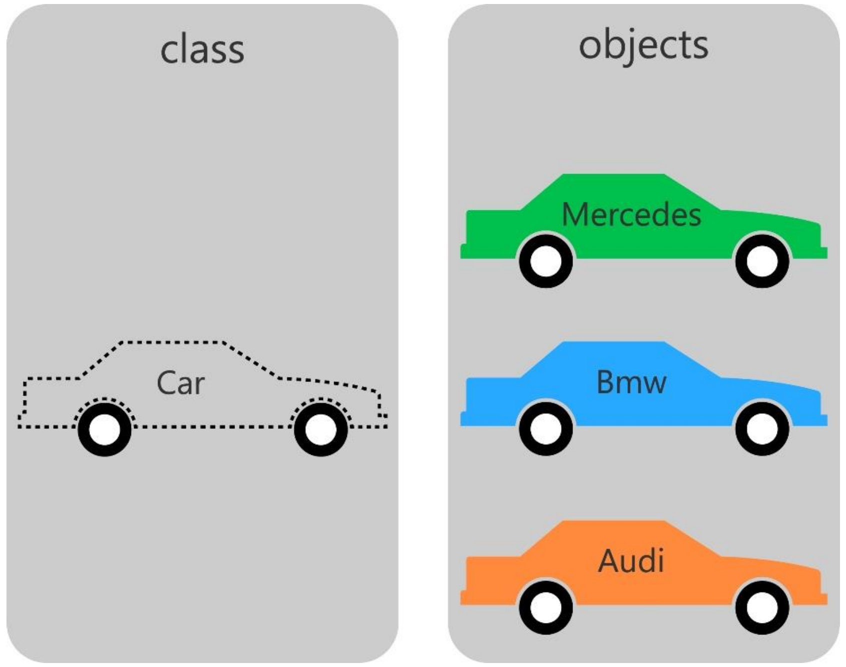
## Real-World Analogy

Class as a Blueprint:

- Think of a class as a blueprint for creating objects. For example, a blueprint for a car contains details like the car's design, components, and features, but it isn't an actual car.

Object as a Physical Entity:

- When the blueprint is used to create an actual car, that car is an object. Multiple cars (objects) can be created from the same blueprint (class), but each car may have its own unique features (attributes), like color, engine type, etc.



# Object-Oriented Programming (OOP)

**Reusability**: You can create multiple objects from the same class, reducing code duplication.

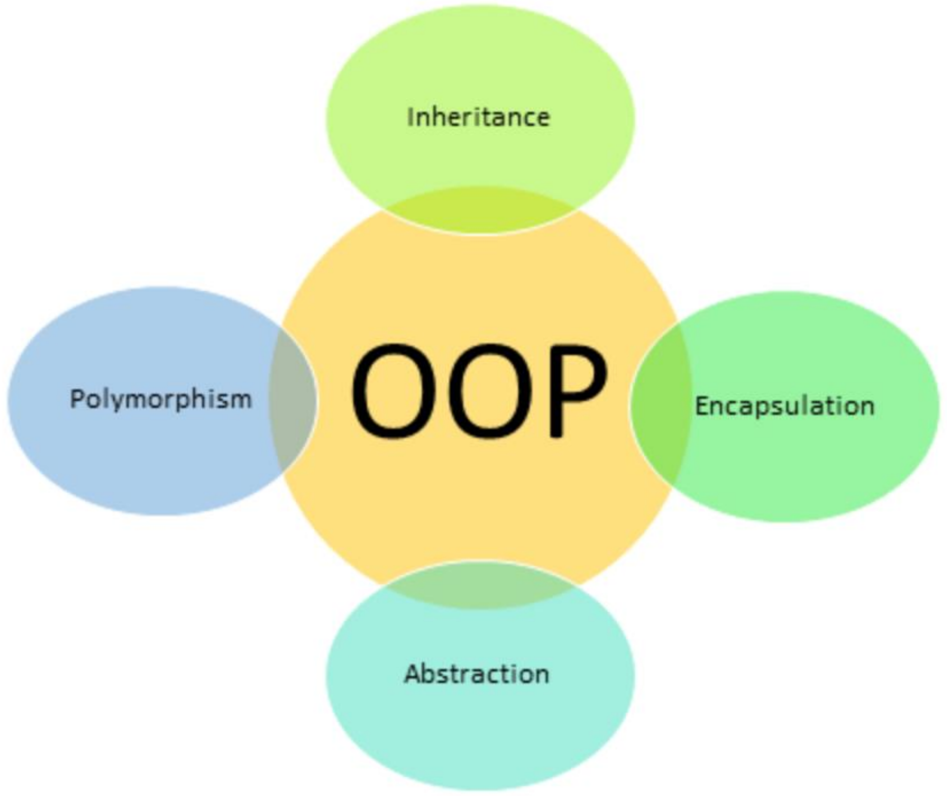
- Example: You can define the Dog class once and create multiple dog objects (dog1, dog2).

**Modularity**: Classes help you organize code into logical units.

- Example: A Car class can have separate classes for Engine or Wheel, making the code modular.

**Abstraction**: Focus on using the object's methods rather than its internal implementation.

- Example: You just call dog1.bark(), without worrying about how the bark() method is implemented.

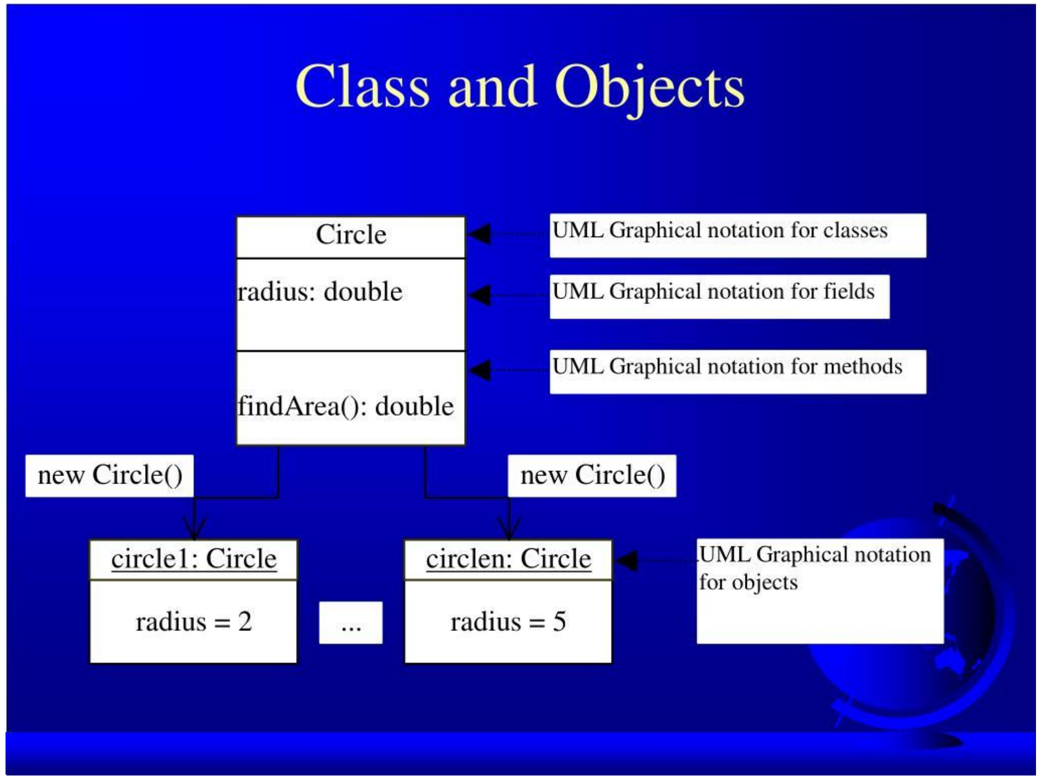


Step8Up Academy

## Conclusion

### Key Points

- Classes provide a structured way to define entities, allowing you to define their attributes and behavior.
- Objects are specific instances of classes, representing unique entities with their own data.
- Object-Oriented Programming (OOP) principles like encapsulation, abstraction, and reusability help organize and optimize your code for scalability.



Class and Objects
Circle
radius: double
findArea(): double
UML Graphical notation for classes
UML Graphical notation for fields
UML Graphical notation for methods
new Circle()
new Circle()
circle1: Circle
radius = 2
...
circlen: Circle
radius = 5
UML Graphical notation for objects