CBS Department

Classes and Objects, Attributes and Methods

Soma Soleiman Zadeh

Object-Oriented Programming (CBS 215)
Fall 2025 - 2026
Week 5
November 5-6, 2025

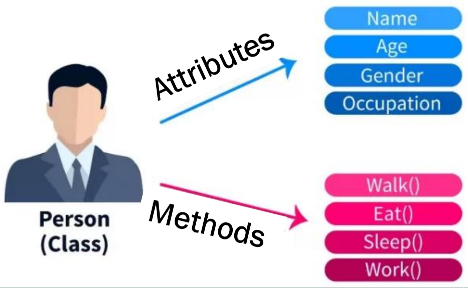
Outline

Programming Paradigms

- Two popular programming paradigms are:
  - **Procedural Programming:**
    - ✓ Step-by-step sequence of instructions to perform a task.
    - ✓ Organizing programs around **functions**.
  - **Object-Oriented Programming (OOP):**
    - ✓ combining data and functionality and wrapping it inside an **object**.

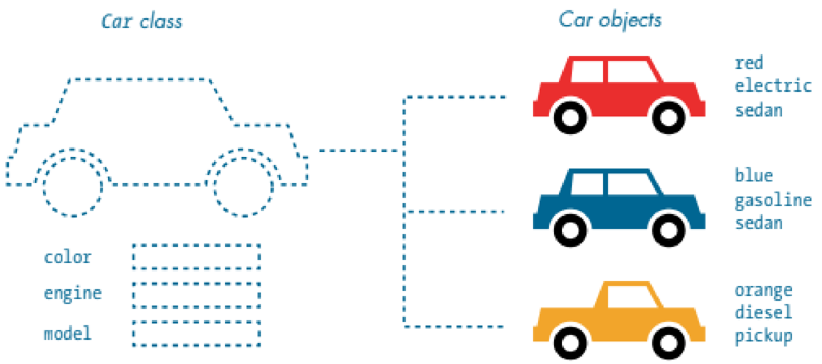
What is an Object?

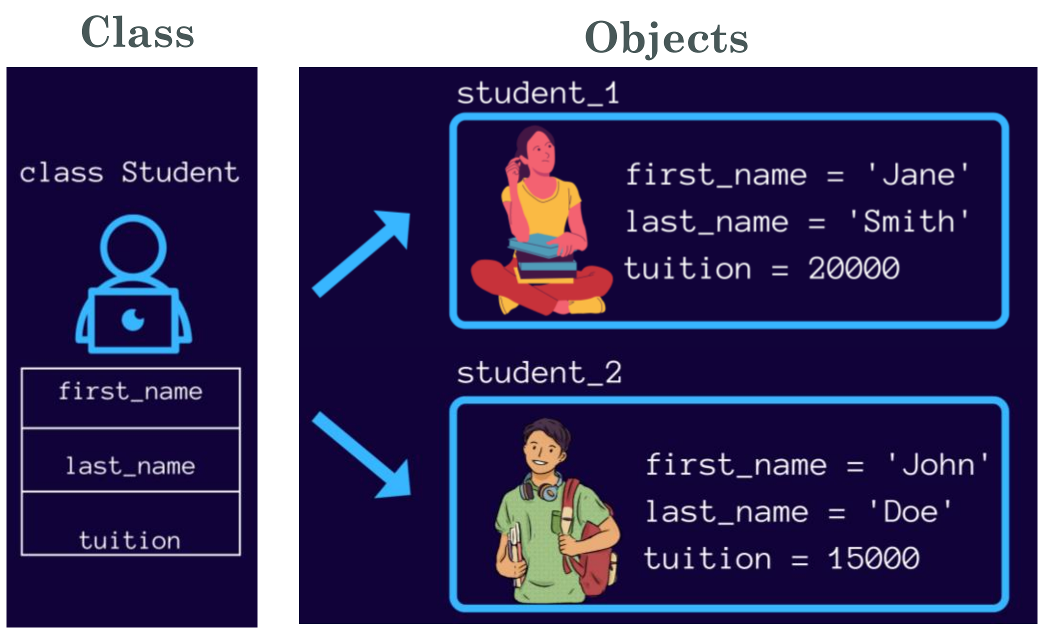
- **Objects** are the building blocks of an object-oriented program.
  - A program that uses objects is basically a collection of **objects.**
  - **Objects** interact much like things in the real world do.
- **Objects** have two components:
  - Data (**Attributes**),
  - Behaviors (**Methods**)



Object-Oriented Programming

- **Classes** and **Objects** are the two main aspects of Object-Oriented Programming.





Objects in Python

- Everything in Python is an object.
- We've worked on objects already.

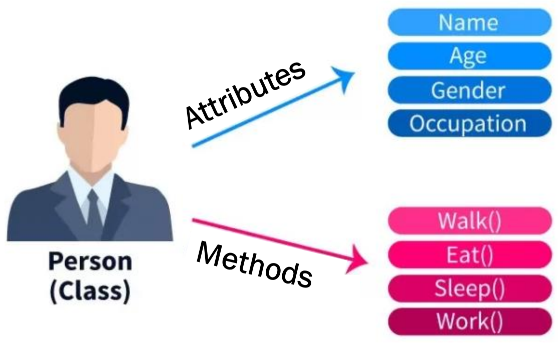
'PYTHON'.lower( ) → An **object** of str **class**

[4, 7, 2].append(3) → An **object** of list **class**

- Programming in Python is done in an **object-oriented** fashion.

Class in Python

- A **class** is a blueprint(template) for creating objects by specifying their ***attributes*** and ***methods***. It encapsulates data and the methods to manipulate that data.



Attributes and Methods of an Object

- **Attributes** and **methods** define an object's behavior and encapsulate data within a class.
- **Attributes** represent the properties or characteristics of an object.
- **Methods** define the actions or behaviors that an object can perform.

Creating a Class

- A **class** is created using the ***class*** keyword.
- The **attributes** and **methods** of the class are listed in an indented block.
- **Classes** are high-level data types.
- Given a **class** description, we can instantiate objects of that class.

Creating a Class

- To create an **object**, first you need to create a **class** using the keyword ***class***.
- The following is an example of creating a ***class*** named **Student**:

```
class  Student:
      def  __init__(self, name, age):
            self.name = name
            self.age = age
```

An attribute; you can have as many attributes as you want.

Class Constructor

- All classes in Python have a special method called **__init__()**, which is always executed when the class is being initiated (i.e., an object of it is created).

```
class  Student:
      def  __init__(self, name, age):
            self.name = name
            self.age = age
```

Class Constructor

Creating Objects of Classes

- After defining a **class**, you can create any number of **objects** of it.
- Accessing an attribute/method of an object → objectName.attribute

  objectName.method

s1 and **s2** are two <u>objects</u> of **Student** class.

```
s1 = Student( “Ali”, 20 )
print(s1.name)
print(s1.age)
```

```
s2 = Student( “Saif”, 25 )
print(s2.name)
print(s2.age)
```

Class Constructor and Methods

- You can also define other ***methods*** alongside **__init__()**.
- **setName()** is a user_defined **method** for setting a new name for a student object.

```
class  Student:
      def   __init__(self, name, age):
              self.name = name
              self.age = age

      def   setName(self, newName):
              self.name = newName
```

Creating Objects of a Class

```
class  Student:
     def  __init__(self, name, age):
            self.name = name
            self.age = age

     def  setName(self, newName):
            self.name = newName
```

```
s1 = Student("Ahmed", 21)
print(s1.name)
```

$\longrightarrow$ *Ahmed*

```
s1.setName("Ahmed Karim")
print(s1.name)
```

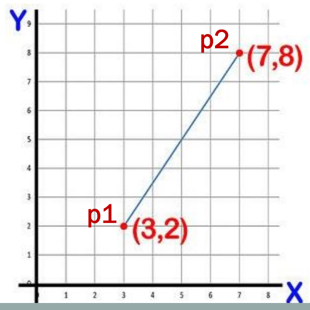
$\longrightarrow$ *Ahmed Karim*

s1 is an object of Student class. When we created this object, we put name='Ahmed' and age=21.

*Since student class has a method for changing student name (**setName**), we can use it later for setting new name for any student object.*

Class Example

- Create a class named **Point** for defining a point with **x** and **y** coordinates. (**x** and **y** coordinates are point attributes.)
  - In the **Point** class, define a **method** to <u>calculate the distance of the current point to any other point</u>.



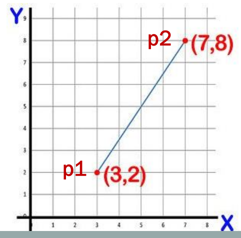
Answer

```
import math

class point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def distance(self, other):
        return math.sqrt((self.x - other.x)**2 +
                         (self.y - other.y)**2)
```

Creating Point Class

```
p1 = point(3,2)
p2 = point(7,8)
dist = p1.distance(p2)
print("The distance between two points is", dist)
```

Creating Two Objects (p1 and p2) from Point Class


Y
X
p2
(7,8)
p1
(3,2)