I
What is Object-Oriented Programming in Python

Data Science Academy

Object-oriented programming

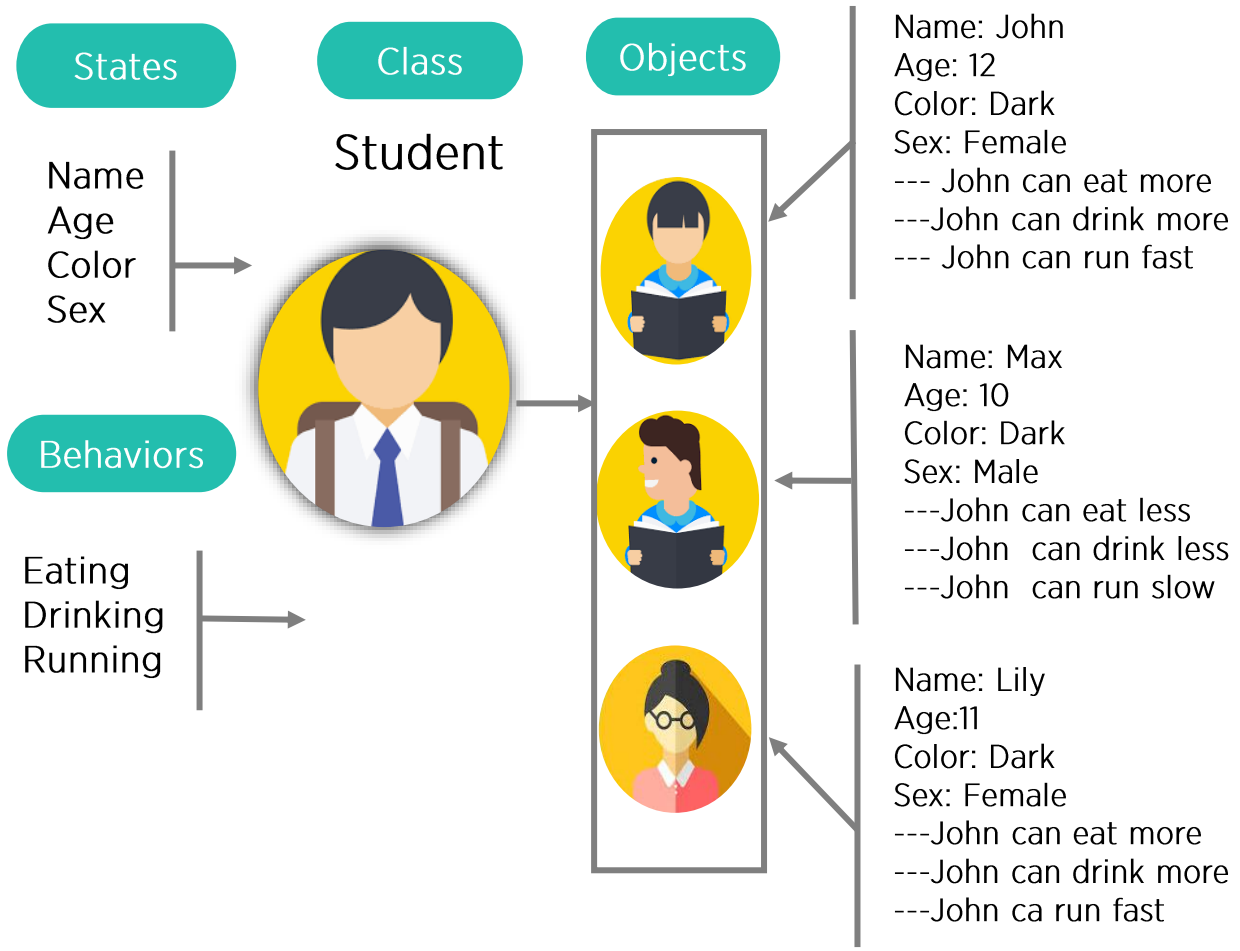
- Object-oriented programming (OOP) is a way of writing computer programs which is using the idea of "objects" to represent data and methods.

II
Define a Class and Instantiate an Object in Python

Data Science Academy

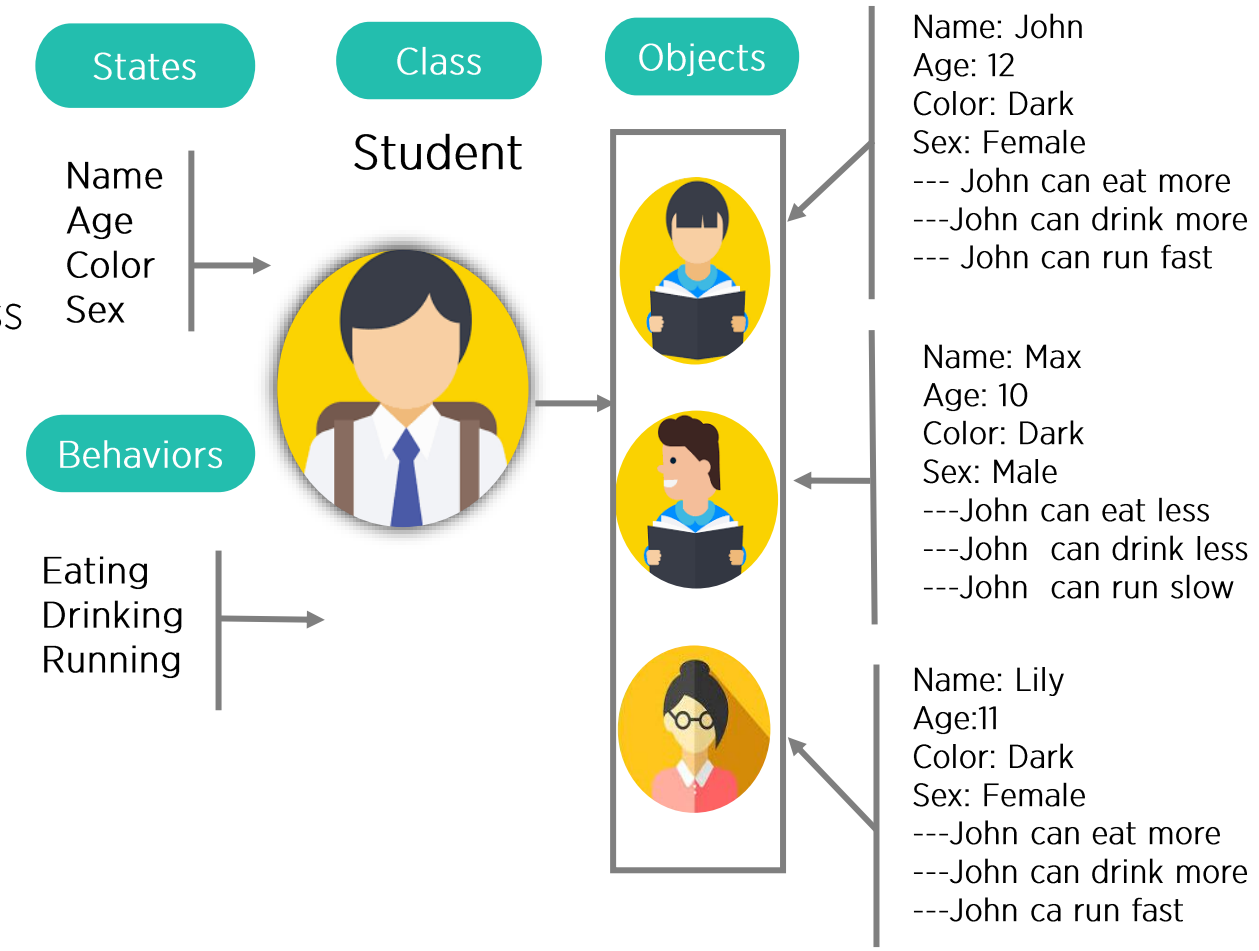
Defining a Class

- *class* stores some data items that are shared by all the instances of this class
- *instances* are created objects, which follow the definition given inside of the class



Attributes

- The non-method data stored by objects are called attributes
- Data attributes
  - Variable owned by a particular instance of a class
  - Each instance has it own value for it
  - These are the most common kind of attribute
- Class attributes
  - Owned by the class a whole
  - All class instances share the same value for it



Instantiating Objects

- Use the class name with ( ) notation and assign the result to a variable
- __init__ does some initialization work
- The arguments passed to the class name are given to its __init__ () method
- The __init__ method for student is passed “Bob” and 21 and the new class instance is bound to b:

```
In [ ]: st1 = student (“Bob”, 21)
```

Constructor __init__

- An __init__ method can take any number of arguments
- The arguments can be defined with default values, making them optional to the caller.
- However, the first argument **self** in the definition of __init__ is **special**...
- __init__ : The constructor for the class

Definition of student

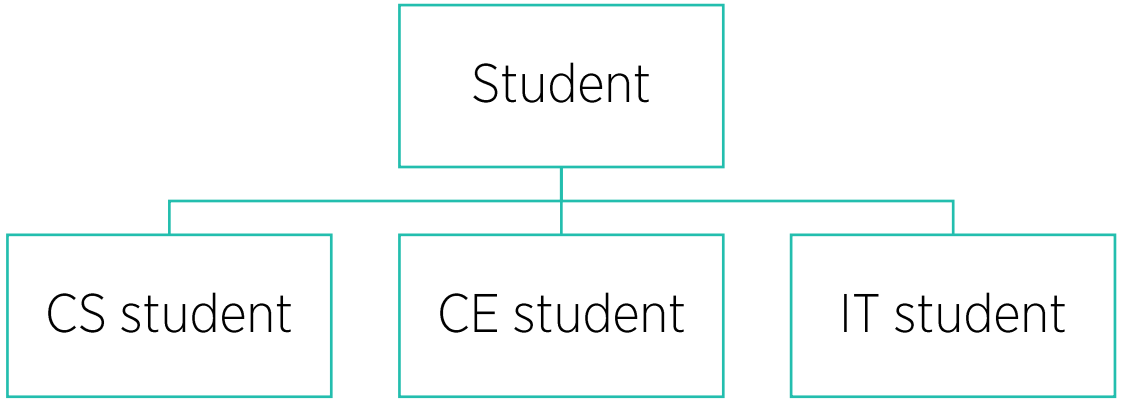
We create class named **student** with name and age parameters.

```
class student:

#A class representing a student
def __init__(self,name,age):
self.full_name = name
self.age = age
def get_age(self):
return self.age
st1 = student( "Bob“, 21)

>>>st1.full_name # Access a method
‘Bob’

>>> st1.get_age() # Access a method
21
```

Subclasses

- Classes can *extend* the definition of other classes
- To define a subclass, put the name of the superclass in parents after the subclass's name on the first line of the definition

In [ ]:
```
class Cs_student (student):
```



Definition of a class extending student

Creating student class:

```
class Student:

    #A class representing a student
    def __init__(self,name,gpa):
        self.full_name = name
        self.gpa = gpa
    def get_gpa(self):
        return self.gpa
```

Extending student class:

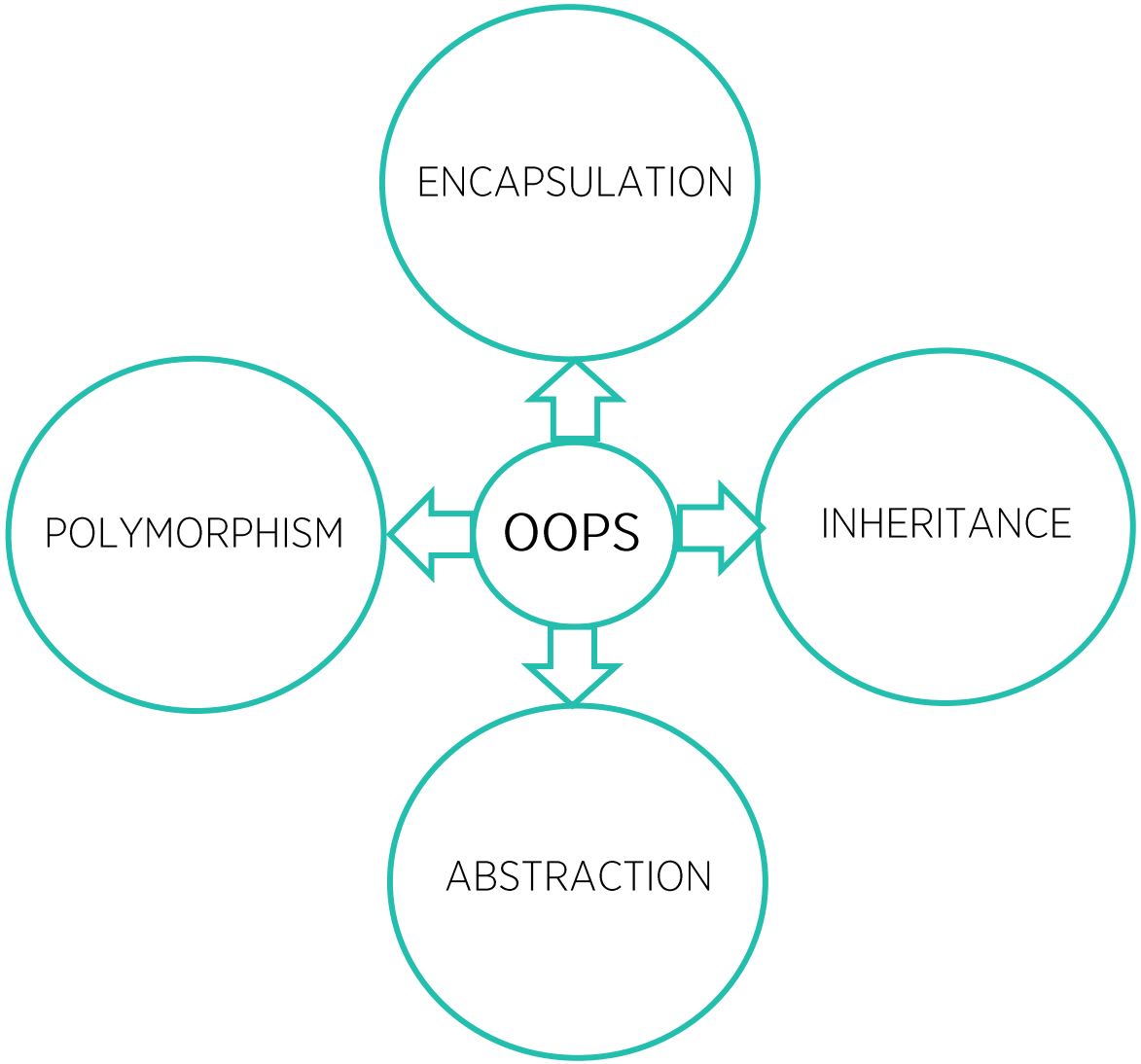
```
class Cs_student (student):
    #A class extending student
    def __init__ (self,name,gpa,sn):
        student.__init__ (self,name,gpa) #Call __init__ for student
        self.section_num = sn
    def get_gpa(self): #Redefines get_gpa method entirely
        print ("Gpa: " + str(self.gpa))
```

Built-In Members of Classes

- Classes contain many methods and attributes that are included by Python even if you do not define them explicitly
- All built-in members have double underscores around their names:

__init__ ; __doc __

Fundamentals Concepts of OOP

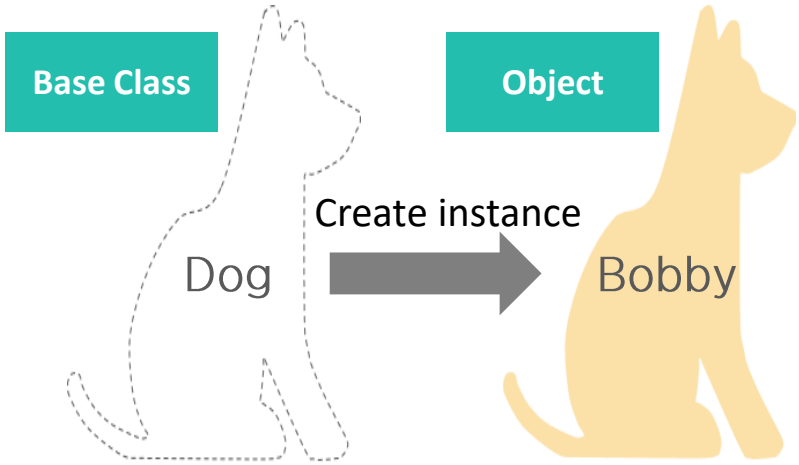


III
Inherit From Other Classes in Python

Data Science Academy

Inheritance: behaviors and characteristics

- Inheritance refers to defining a new class with little or no modification to an existing class.
- The new class is called derived (or child) class and the one from which it inherits is called the base (or parent) class.
- Derived class inherits features from the base class, adding new features to it.

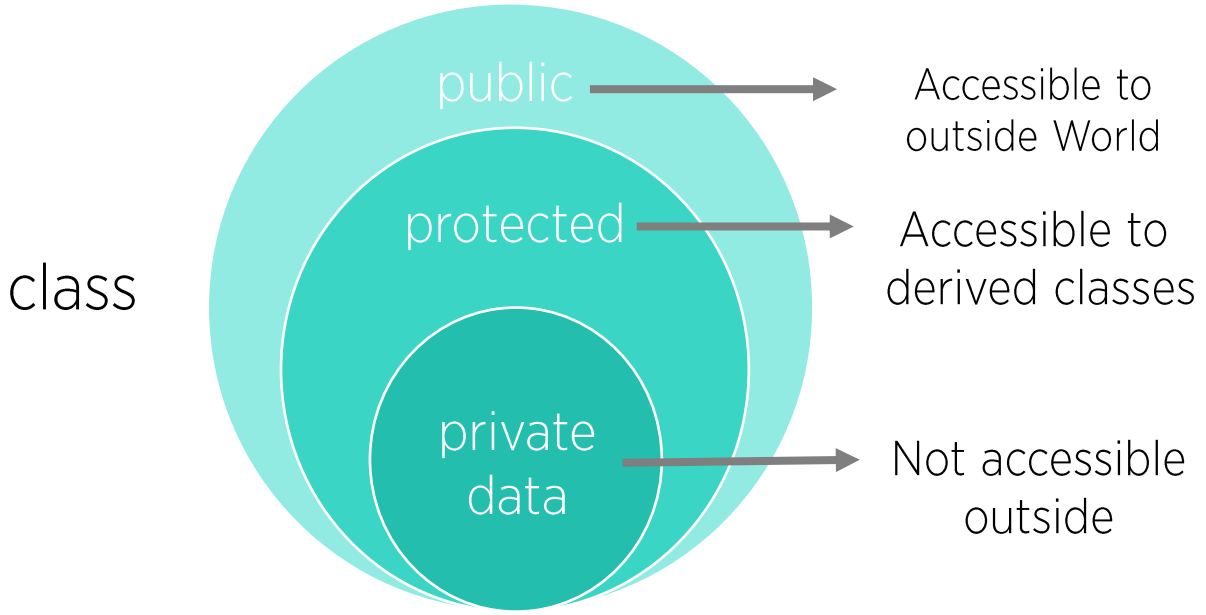


Properties	Methods
Color	Sit
Eye Color	Lay Down
Height	Shake
Length	Come
Weight	

Property Values	Methods
Color: Yellow	Sit
Eye Color: Brown	Lay Down
Height: 17 in	Shake
Length: 35 in	Come
Weight:24 pounds	

Encapsulation: Hiding Information

Encapsulation is a mechanism that restricts direct access to objects' data and methods. But at the same time, it facilitates operation on that data (objects' methods).



Polymorphism-existing in many forms

- The goals of Polymorphism in Object-oriented programming is to enforce simplicity, making codes more extendable and easily maintaining applications.
- A child class inherits all the methods from the parent class. However, in some situations, the method inherited from the parent class doesn't quite fit into the child class. In such cases, you will have to re-implement method in the child class

Polymorphism

```
class Animal(object):
    def __init__(self, name):
        self.name = name
    def talk(self):
        raise NotImplementedError
class Dog(Animal):
    def talk(self):
        return "Bow...Bow..."
class Cat(Animal):
    def talk(self): return "Meow...Meow..."
```

Abstraction

- Abstraction in Python is a programming methodology in which details of the programming codes are hidden away from the user, and only the essential things are displayed to the user.
- Abstraction is concerned with ideas rather than events. It's like a user running a program (Web Browser) without seeing the background codes.

Abstract

- ABC is a class from the abc module in Python. If we extend any class with ABC and include any abstraction methods, then the classes inherited from this class will have to **mandatorily** implement those abstract methods

In [ ]:

```
from abc import abstractmethod, ABC

class Vehicle(ABC):
    def __init__(self, speed, year):
        self.speed = speed
        self.year = year

    def start(self):
        print("Starting engine")

    def stop(self):
        print("Stopping engine")

    @abstractmethod
    def drive(self):
        pass

class Car(Vehicle):
    def __init__(self, canClimbMountains, speed, year):
        Vehicle.__init__(self, speed, year)
        self.canClimbMountains = canClimbMountains

    def drive(self):
        print("Car is in drive mode")
```

Thank you!

Data Science
Academy