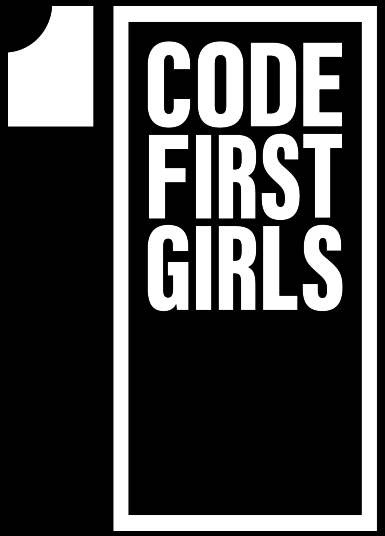
OOP & PYTHON CLASS PART 1

LESSON 1



NANODEGREE → ENGINEERING MODULE

AGENDA

01 Object Oriented Programming

02 Python Class

03 Inheritance

04 Polymorphism

05 Encapsulation

06 Types of class methods

OBJECT ORIENTED PROGRAMMING

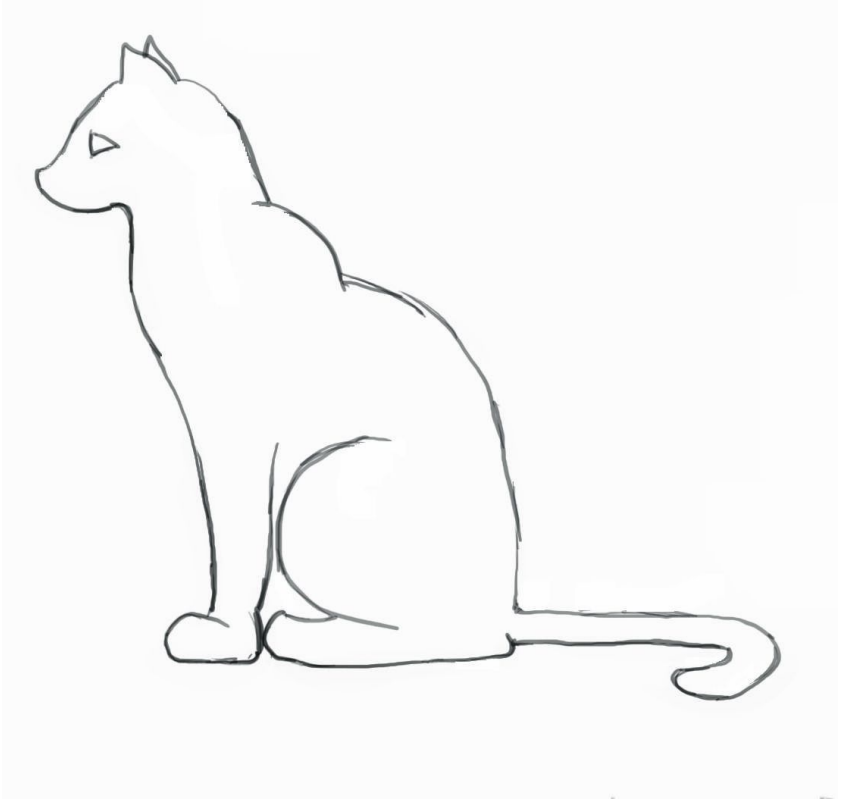
KEY FEATURES

- OOP is a programming concept that provides patterns for structuring programs where properties and behaviors are bundled into **individual objects**.
- Advantages of OOP: development is faster and cheaper, better software maintainability, which results in higher-quality software. However, the learning curve is steeper.
- Examples of OOP programming languages: C, C++, Java, Go, Ruby and Python.
- OOP uses the **concept of objects and classes**.
- A class is a 'blueprint' for objects.
- SUMMARY: OOP is an approach for modeling concrete, real-world things, like ATM machine, a car, as well as relations between things, like companies and employees, students and teachers, and so on.

OOP

BASE CLASS

- An example of a class is a general class Cat.
- Don't think of it as a specific cat, or your own cat.
- We're describing what a cat is and can do, in general.
- Cats usually have a name and age and colour; these are instance attributes.
- Cats can also meow; this is a method.



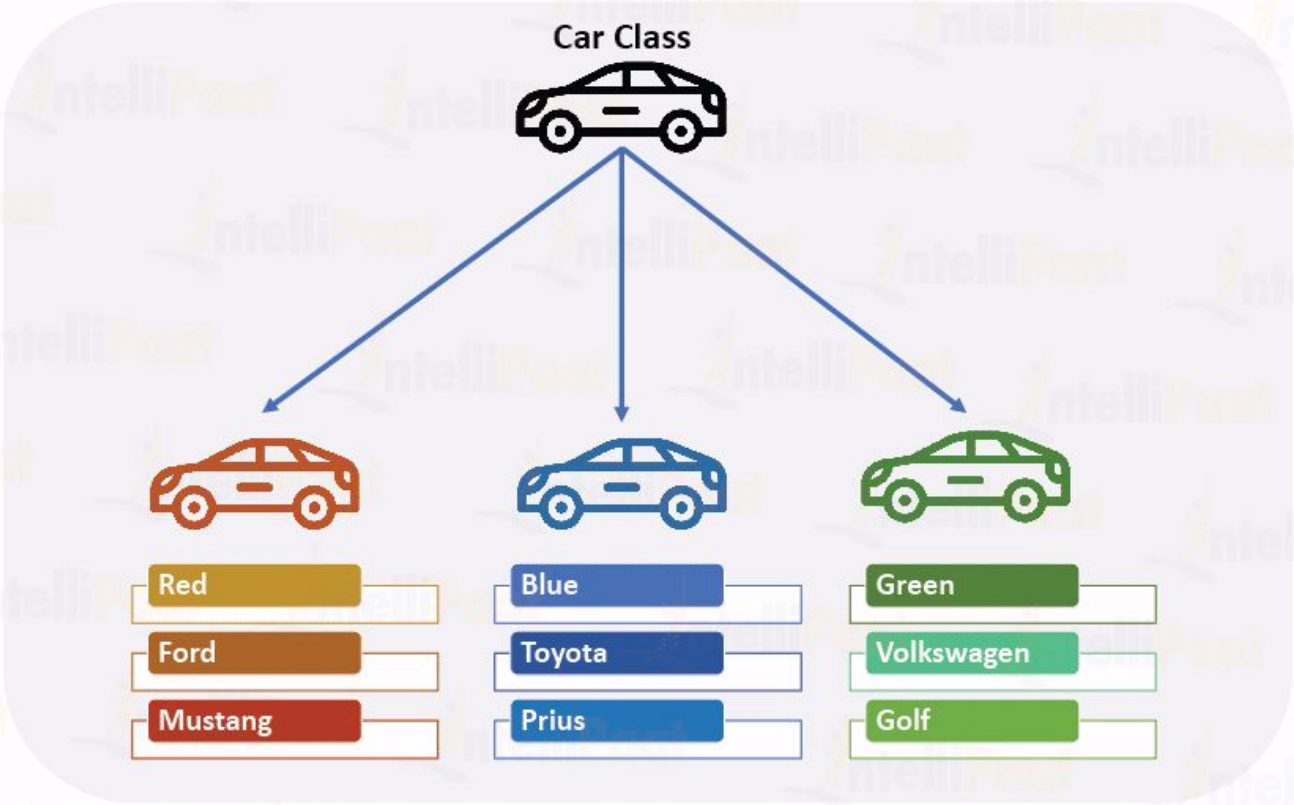
OOP

CONCRETE CLASS

- When we talk about a specific cat, you would have an object in programming.
- An object is an instantiation of a class.
- This is the basic principle on which object-oriented programming is based.
- **Our example:** this cat belongs to the class Cat. His attributes are name = 'Jake' and age = '5'.
- A different cat will have different attributes.

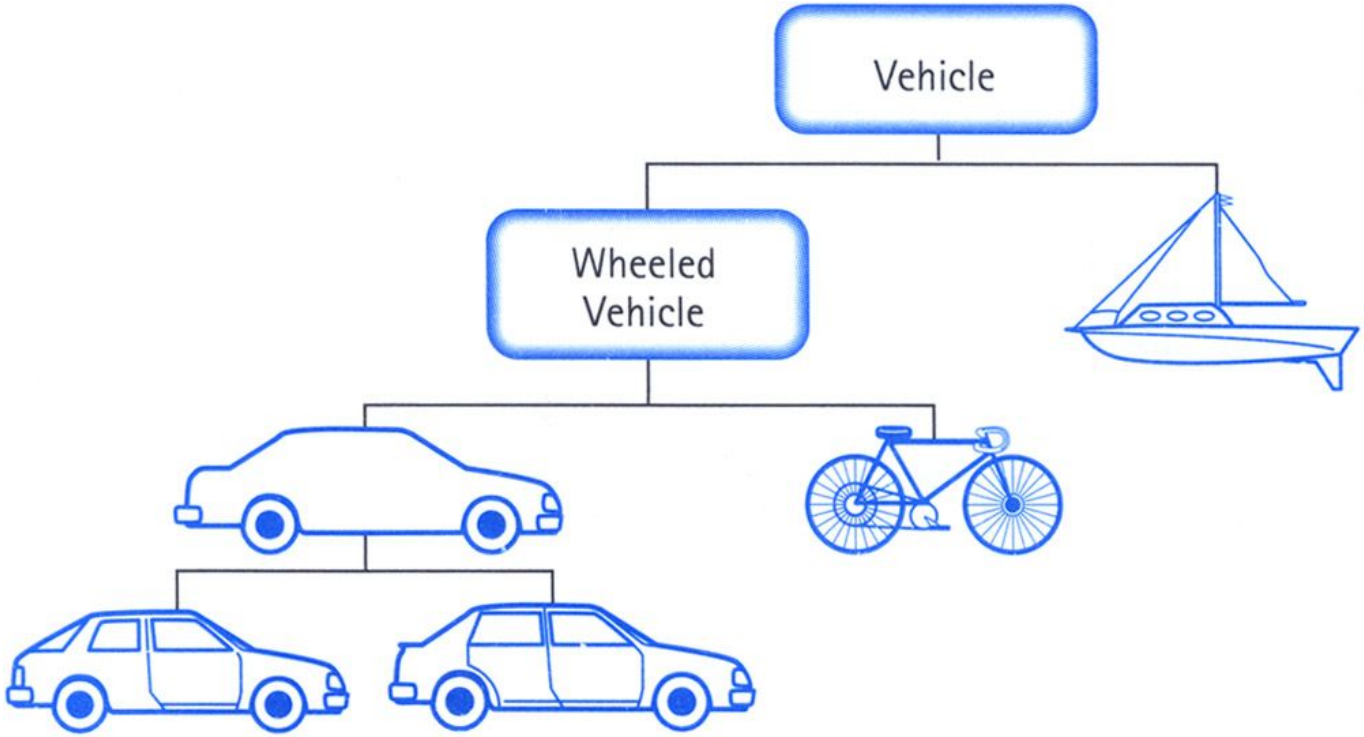
OOP

CAR CLASS EXAMPLE



OOP

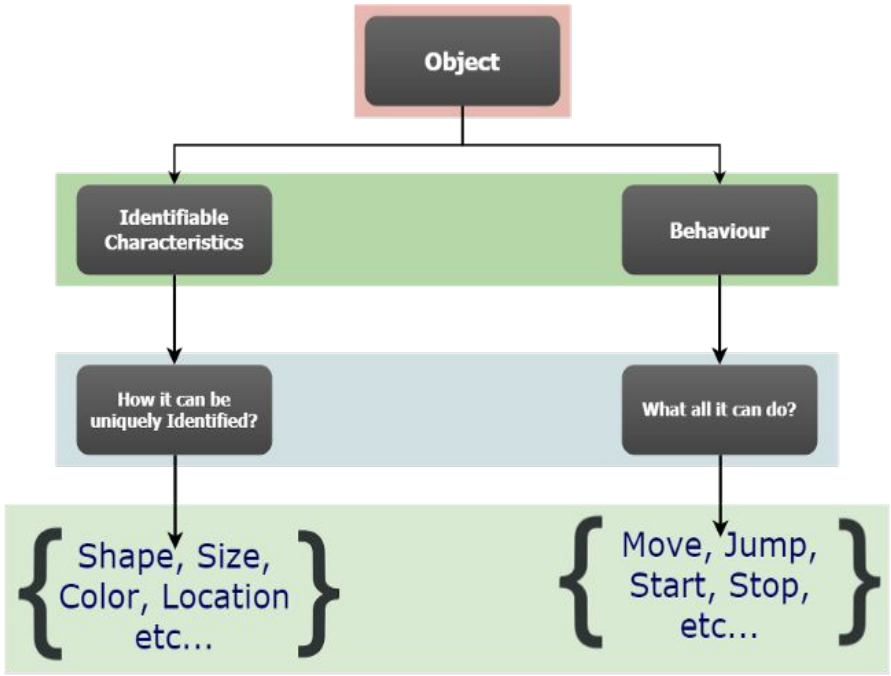
COMPLEX CLASS EXAMPLE

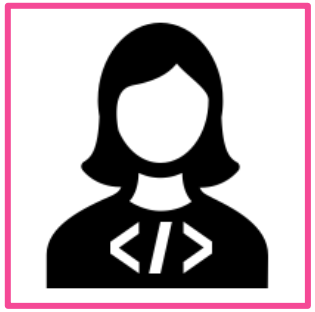


OOP

CLASSES & OBJECTS

- A class defines the blueprint, which can be instantiated to create Object(s)
- A class is a code template for creating objects.
- Objects have member variables and have behaviour associated with them. In python a class is created by the keyword class.





DEMO & EXERCISES

CREATE & INSTANTIATE A CLASS

CLASS INHERITANCE

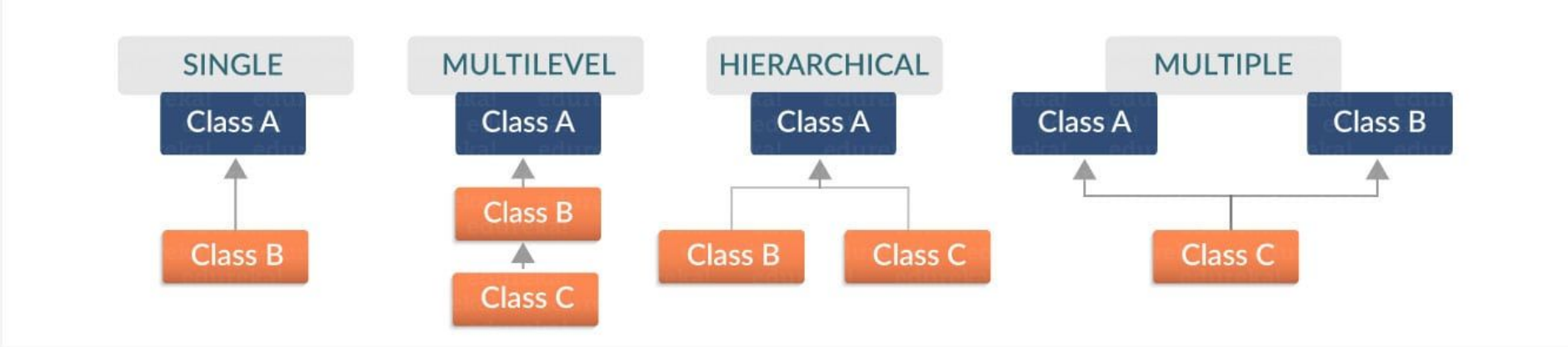
DEFINITION

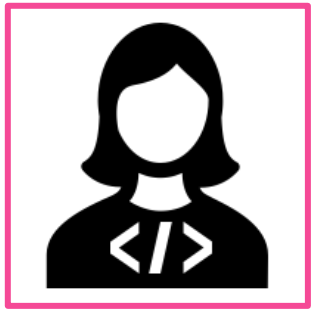
- Inheritance is the process by which one class takes on the attributes and methods of another.
- Newly formed classes are called **child classes**
- The classes that child classes are derived from are called **parent classes**.
- Child classes can override or extend the attributes and methods of parent classes.
- Child classes inherit all of the parent's attributes and methods but can also specify attributes and methods that are unique to themselves.

EXAMPLE

TYPES OF INHERITANCE

EXAMPLE





DEMO & EXERCISES

INHERIT FROM A CLASS

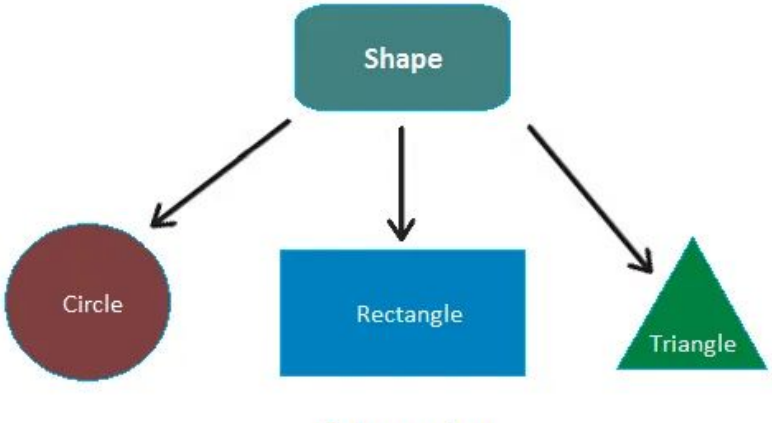
POLYMORPHISM

DEFINITION

- Polymorphism is the condition of occurrence in different forms.

- In programming. It refers to the use of a single type entity (method, operator or object) to represent different types in different scenarios.
- EXAMPLE: len() function is compatible to run with multiple data types

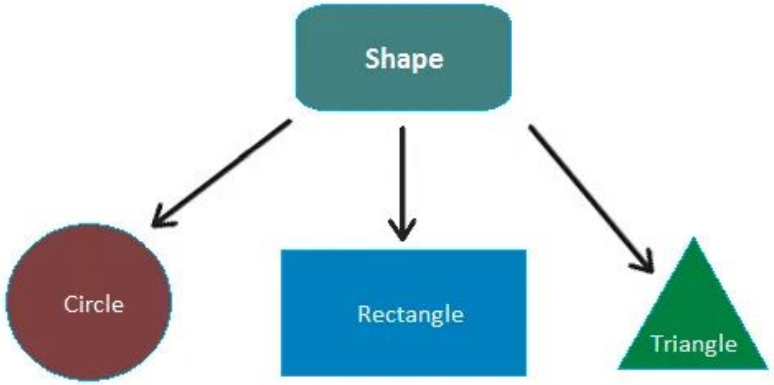
```
print(len("CodeFirstGirls"))
print(len(["Python", "API", "SQL"]))
print(len({"Name": "George", "Age": 10}))
```



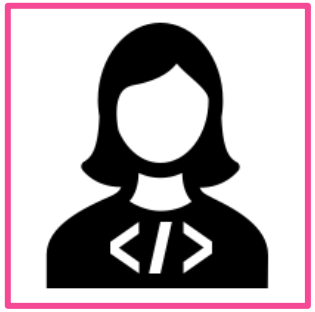
POLYMORPHISM & INHERITANCE

DEFINITION

- The child classes in Python inherit methods and attributes from the parent class.
- We can redefine certain methods and attributes specifically to fit the child class, which is known as Method Overriding.
- Polymorphism allows us to access these overridden methods and attributes that have the same name as the parent class.



FORMULA TO CALCULATE AREA FOR EACH SHAPE



DEMO & EXERCISES

POLYMORPHISM EXAMPLES

ENCAPSULATION

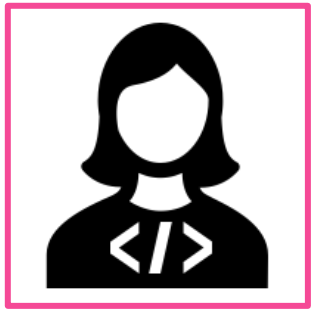
DEFINITION

- Encapsulation describes the idea of wrapping data and the methods that work on data within one unit.
- This puts restrictions on accessing variables and methods directly and can prevent the accidental modification of data.
- To prevent accidental change, an object's variable can only be changed by an object's method.
- Those types of variables are known as private variable.
- A class is an example of encapsulation as it encapsulates all the data that is member functions, variables, etc.

ENCAPSULATION

PROTECTED MEMBERS

- Protected members are those members of the class that cannot be accessed outside the class but can be accessed from within the class and its subclasses.
- In Python, we need to follow **the convention** by prefixing the name of the member by a **single underscore "_"**.
- **Note:** The __init__ method is a constructor and runs as soon as an object of a class is instantiated.



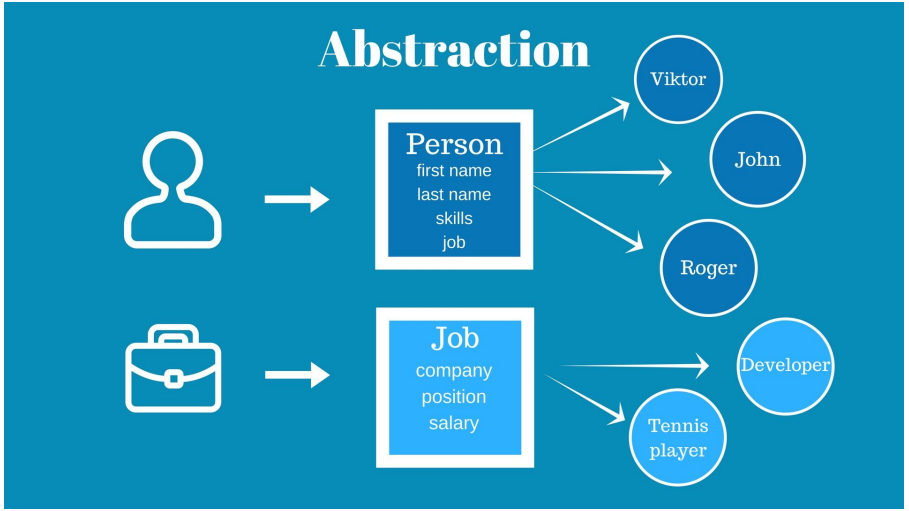
DEMO & EXERCISES

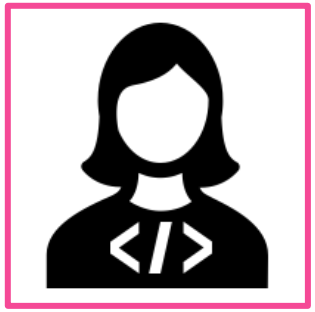
ENCAPSULATION EXERCISE

ABSTRACTION

DEFINITION

- An abstract class can be considered as a blueprint for other classes.
- It allows you to create a set of methods that must be created within any child classes built from the abstract class.
- A class which contains one or more abstract methods is called an abstract class.
- An abstract method is a method that has a declaration but does not have an implementation.
- When we want to provide a common interface for different implementations of a component, we use an abstract class.





DEMO & EXERCISES

ABSTRACTION EXERCISE

CLASS METHODS

DEFINITION

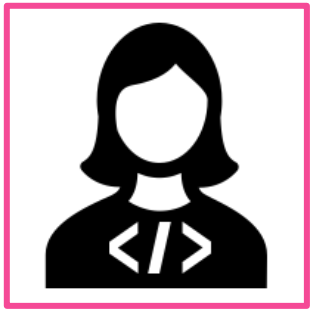
```
Class <Name>():

        variable = 'text'

        def __init__():
                some logic

        @classmethod
        def method_one()
                some logic

        @staticmethod
        def method_two()
                some logic

        def method_three()
                some logic
```

@classmethod

- Declares a class method.
- It can access class attributes, but not the instance attributes.
- It can be called using the ClassName.MethodName() or object.MethodName().
- It can be used to declare a factory method that returns objects of the class.

@staticmethod

- Declares a static method.
- It cannot access either class attributes or instance attributes.
- It can be called using the ClassName.MethodName() or object.MethodName().
- It cannot return an object of the class.



DEMO & EXERCISES

METHOD TYPES EXERCISE

OOP TERMINOLOGY

PART 1

- **Class** − A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.
- **Class variable** − A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.
- **Data membe**r − A class variable or instance variable that holds data associated with a class and its objects.
- **Function overloading** − The assignment of more than one behavior to a particular function. The operation performed varies by the types of objects or arguments involved.
- **Instance variabl**e − A variable that is defined inside a method and belongs only to the current instance of a class.

OOP TERMINOLOGY

PART 2

- **Inheritance** − The transfer of the characteristics of a class to other classes that are derived from it.
- **Instance** − An individual object of a certain class. An object obj that belongs to a class Circle, for example, is an instance of the class Circle.
- **Instantiation** − The creation of an instance of a class.
- **Method** − A special kind of function that is defined in a class definition.
- **Object** − A unique instance of a data structure that's defined by its class. An object comprises both data members (class variables and instance variables) and methods.
- **Operator overloading** − The assignment of more than one function to a particular operator.

THANK YOU!