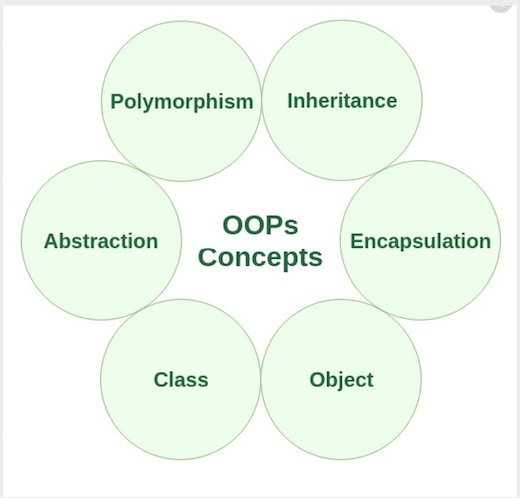
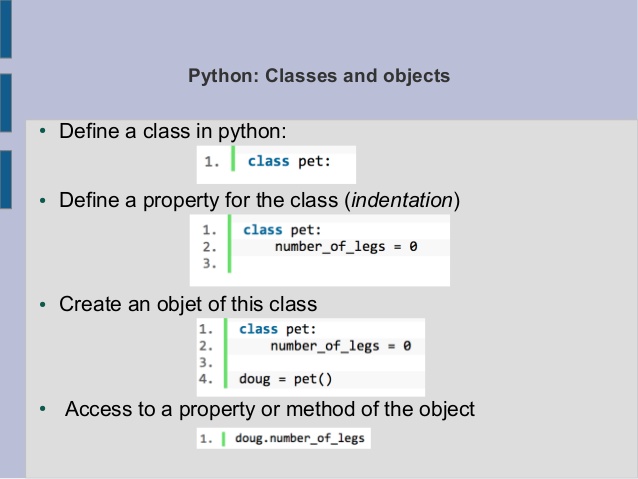
OBJECT ORIENTED PROGRAMMING LANGUAGE



**Class and Object:**



**CLASS:** The class can be defined as a collection of objects. It is a logical entity that has some specific attributes and methods. A Class is like an object constructor, or a "blueprint" for creating objects.

**Defining the class:**

A class in Python can be defined using the class keyword. As per the syntax above, a class is defined using the class keyword followed by the class name and : operator after the class name.

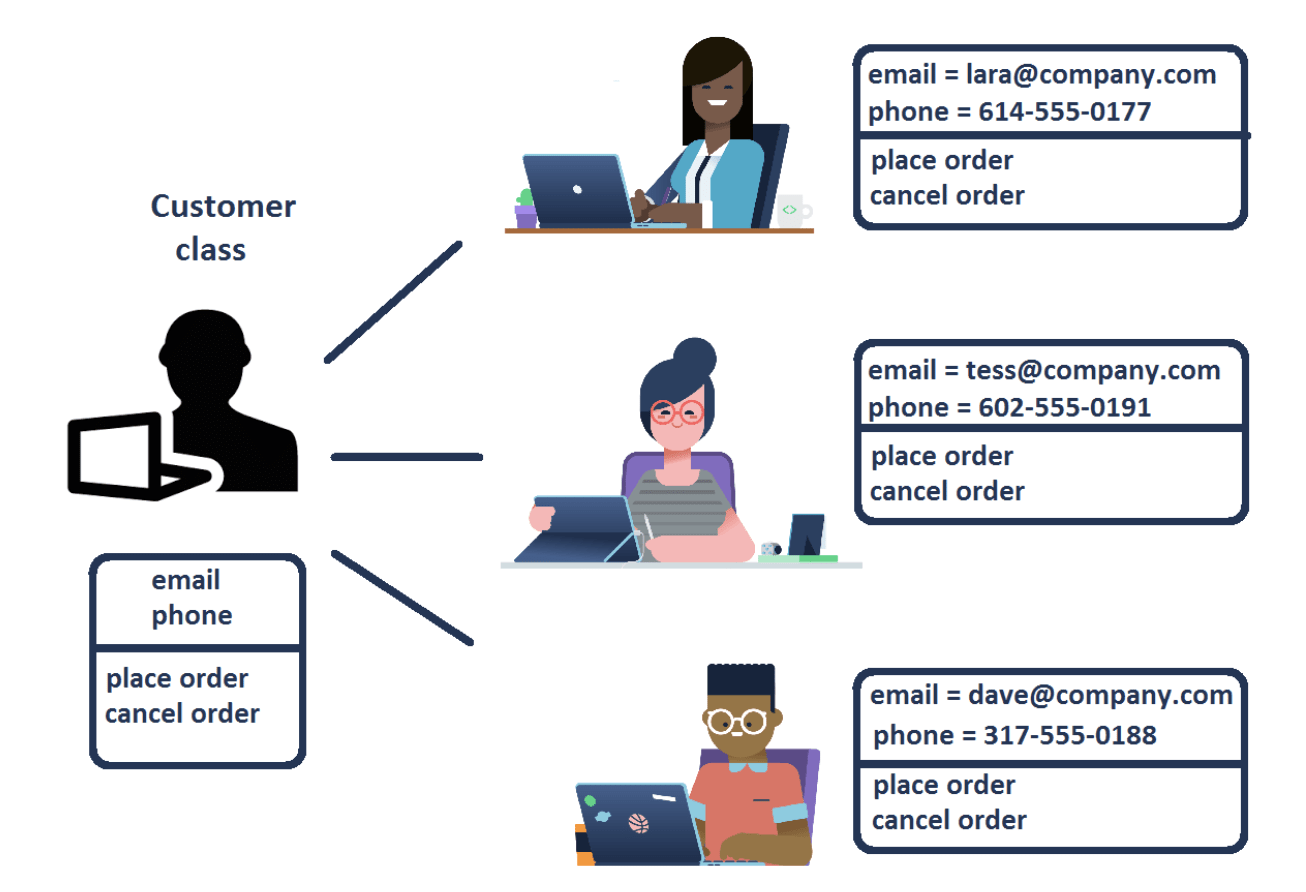
**Object:**

Everything is in Python treated as an object, including variable, function, list, tuple, dictionary, set, etc. Every object belongs to its class. For example - An integer variable belongs to integer class. An object is a real-life entity.

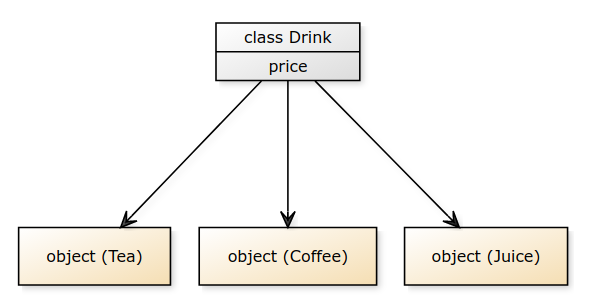
**Defining the object:**

“objects” represents data and methods. It is also, an approach used for creating neat and reusable code instead of a redundant one. the program is divided into self-contained objects or several mini-programs.

**Example of class:**



**Example of object:**



**Syntax of class:**

class ClassName:

#statements

.

.

.

#statements N

Syntax of object:

**Syntax of Object:**

class ClassName:

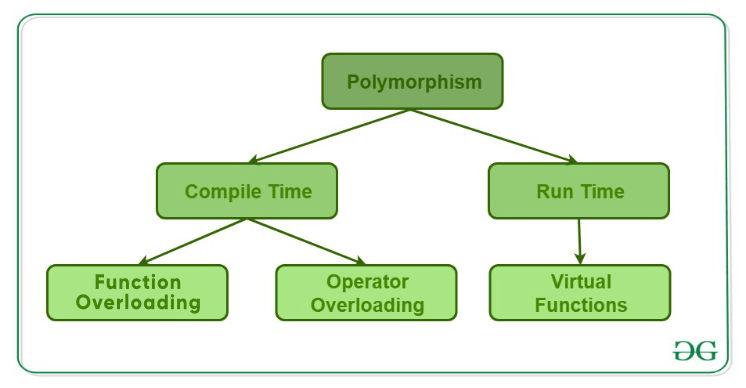
Def fun(self):

#statements

Object= ClssName()

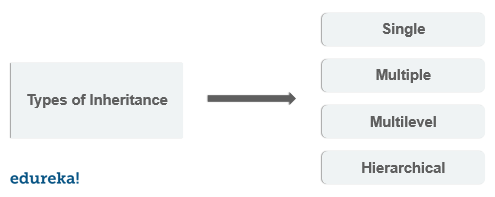
Object.fun()

**POLYMORPHISM:**

* Polymorphism is the ability to use an operator or function in different ways.
* Poly refers too many signifies the many uses of these operators and functions.
* A single function usage or an operator functioning in many ways can be called as polymorphism.
* It also refers to codes, operators or objects that behave differently in different contexts. 

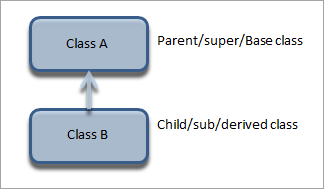
|  |  |  |
| --- | --- | --- |
| **Method Overloading** | **Method Overriding** | |
| * Having multiple methods with same name with different signatures * It is an example of compile time polymorphism * In this method, Inheritance may or may not be required * It is performed between methods with in the class * In this method, there is no need of more than one class | | * Subclass contains a method with same name and signature as in the super class * It is an example of runtime polymorphism * Inheritance is always required in this method * It’s is done between parent and child class * In this method, there is a need of at least of two classes |

**INHERITANCE:**

Inheritance is the process of inheriting the properties of the parent class into the child class. 

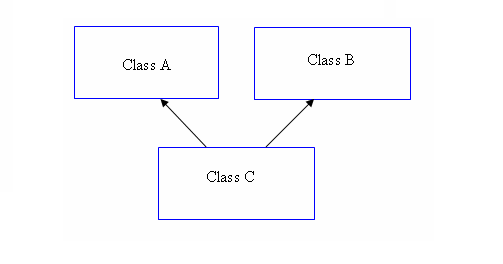
**Single Inheritance:**

Single inheritance enables a derived class to inherit properties from a single parent class, thus enabling code reusability and the addition of new features to existing code.



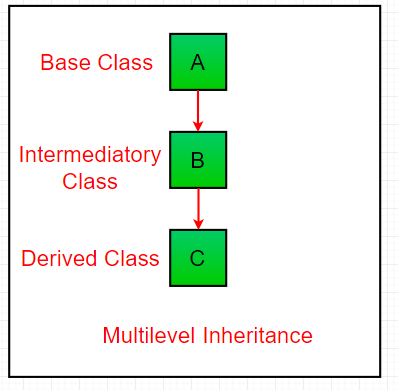
**Multiple Inheritance:**

A class can be derived from more than one base class in Python. This is called multiple inheritance. In multiple inheritance, the features of all the base classes are inherited into the derived class. The syntax for multiple inheritance is similar to single inheritance.



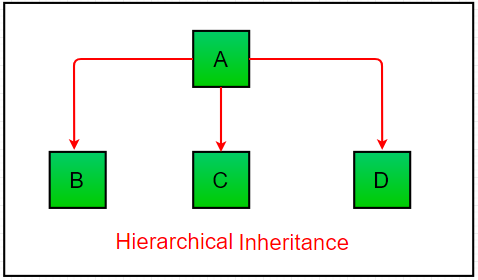
**Multilevel Inheritance:**

The multi-level inheritance includes the involvement of at least two or more than two classes. One class inherits the features from a parent class and the newly created sub-class becomes the base class for another new class.



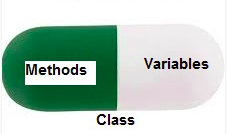
**Hierarchical Inheritance:**

When more than one derived classes are created from a single base.



**Encapsulation:**

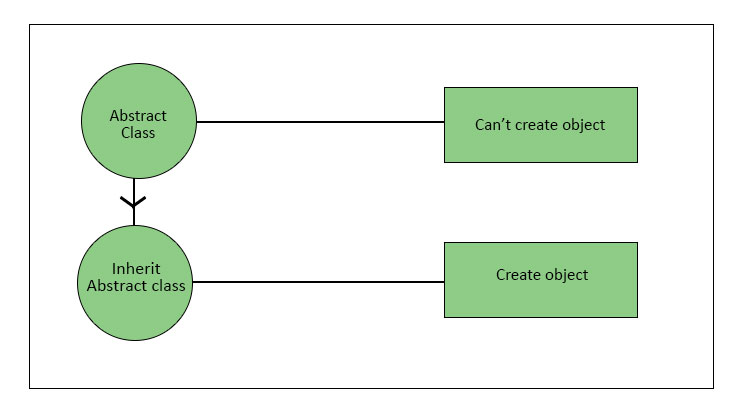
Encapsulation is the process of wrapping up variables and methods into a single entity.



**Abstraction:**

Abstraction is defined as a process of handling complexity by hiding unnecessary information from the user.

Abstraction means hiding the complexity and only showing the essential features of the object. So in a way, Abstraction means hiding the real implementation and we, as a user, knowing only how to use it. Real world example would be a vehicle which we drive without caring or knowing what all is going underneath.



---------------------------END---------------------------