OOPs

What is OOP ?

OOP stands for Object-Oriented Programming.

Procedural programming is about writing procedures or functions that perform operations on the data, while object-oriented programming is about creating objects that contain both data and functions.

OOP - Object-Oriented Programming is a programming paradigm that is based on objects. (Objects - real-world entities like book, vehicle, tree, etc..)

There are 4 OOP concepts. They are,

1. Polymorphism
2. Inheritance
3. Encapsulation
4. Abstraction

Real world examples

<https://jeemariyana.medium.com/oop-concepts-with-real-world-examples-cda1cd277f4f>

structure vs class

* Structure=>
* Class=>
* Structure groups together multiple data types and it is considered as a structure variable.
* Class combines multiple data types into one group and its object is considered as an instance of a class.
* Structures do not support Polymorphism & inheritance
* Classes extensively support polymorphism and inheritance. It was developed to allow object-oriented design paradigms.
* By default, all the member variables in the structure are public
* Classes have member variables and functions by default private.
* It is normally used for the grouping of data
* It is normally used for data abstraction and further inheritance.

Access Modifiers

1. **Private**: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.

Syntax:-

Add \_\_ before variables

1. **Protected**: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.

Syntax:-

Add \_ before variables

1. **Public**: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

Constructor and destructors

Constructor

 a constructor is a special method that gets called when an object is created. It is used to initialize the object’s attributes with default or given values. The constructor is defined with the **\_\_init\_\_** method, which takes **self** as the first argument, followed by other arguments for initializing the object.

Destructor

a destructor is a special method that gets called when an object is about to be destroyed. It is used to perform cleanup actions such as closing files, releasing resources, or freeing memory. The destructor is defined with the **\_\_del\_\_** method, which takes the **self** as the first argument.

class vs object

|  |  |  |
| --- | --- | --- |
| **No.** | **Object** | **Class** |
| 1) | Object is an **instance** of a class. | Class is a **blueprint or template** from which objects are created. |
| 2) | Object is a **real world entity** such as pen, laptop, mobile, bed, keyboard, mouse, chair etc. | Class is a **group of similar objects**. |
| 3) | Object is a **physical** entity. | Class is a **logical** entity. |
| 4) | Object is created many times as per requirement. | Class is declared once. |
| 5) | Object allocates memory when it is created. | Class doesn't allocated memory when it is created. |

**Polymorphism**

Polymorphism is the ability to exist in many forms.

Types

Compile time polymorphism

In compile-time polymorphism, a function is called at the compile time. This type of polymorphism is also known as static binding or early binding. Function overloading and operator overloading are the two types of Compile time polymorphism and are used to accomplish compile-time polymorphism.

Method Overloading

 In this, more than one method of the same class shares the same method name having different signatures. Method overloading is used to add more to the behavior (number of arguments, type of arguments) of methods and there is no need of more than one class for method overloading.

Runtime polymorphism

In runtime polymorphism, a function is called at the run time or at the time of program execution. This type of polymorphism is also known as dynamic binding or late binding. Function overriding and virtual functions are the different types of run-time polymorphism.

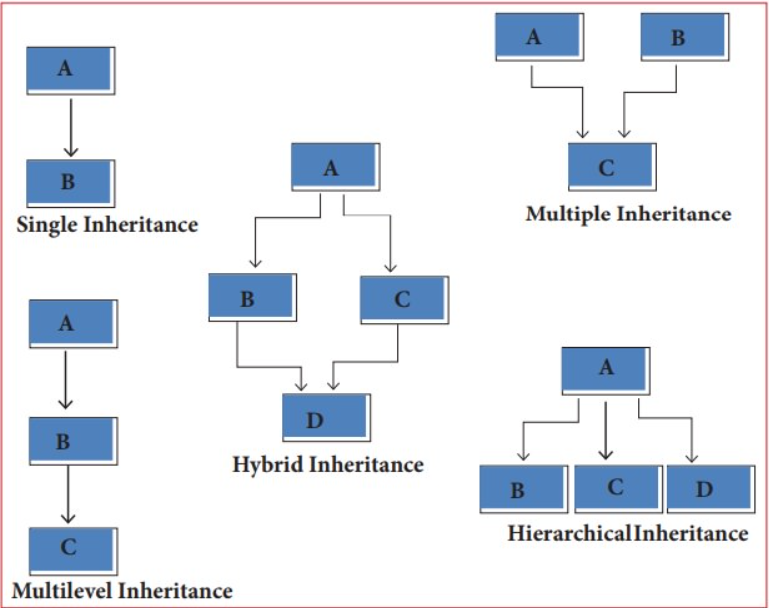
It is more flexible than compile-time polymorphism as the things are executed at run time.

[Method Overriding:](https://www.geeksforgeeks.org/method-overriding-in-python/)   
Method overriding is an example of run time polymorphism. In this, the specific implementation of the method that is already provided by the parent class is provided by the child class. It is used to change the behavior of existing methods and there is a need for at least two classes for method overriding. In method overriding, inheritance always required as it is done between parent class(superclass) and child class(child class) methods.

## Inheritance

Inheritance means it allows classes to inherit common properties from the parent class.

Types of Inheritance



## Encapsulation

* Encapsulation means it binds data and code together into one unit.
* It 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 variables.
* A class is an example of encapsulation as it encapsulates all the data that is member functions, variables, etc.

## Abstraction

In abstraction, it displays only the important information by hiding the implementation part.

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