PAGE NO :
DATE : / /

☆ OOPs

① There can be many number of object. Each object is unique.

② Object is a instance of class.

③ Function can return value.
Method can also return value.

Method Example	Function example

class Raglaw:
def __init__(self, color)
self.color = color
def ram(self)
self.color = Red
obj = Raglaw("Red")
obj.ram
	class Raglaw:
	def __init__(self, color):
	self.color = color
	def ram(_):
	color = Red
	obj = Raglaw("Red")
	obj.ram

→ A method is related to class, it can access var of class only.

→ Function don't have access limitation.

④ OOPs:-

1. Object
2. Class
3. Abstraction
4. Polymorphism
5. Inheritance
6. Encapsulation

} APIE

* Constructor is always public, has same name as class and do not return anything.
* C++, by default, everything is private.

Scanned with CamScanner

Attribute → value → classname, outside class
constructor is always public (no return value)

PAGE NO :
DATE : / /

★ Abstraction → We focus on only important information. &
unimportant information is not shown.
ex:- Person class is there, important info. is only name, height, weight

★ Encapsulation :- Restrict the data access like Private, Public, Protect.
- It decreases dependency in code & reduces the chance of crash.

Public :- outside class & accessed
Private :- No access outside class
Protected :- Can be accessed by inherited only.

★ Inheritance :- Multiple inheritance supported by C++ & python only.
Java supports single inheritance

Is a relationship

★ Polymorphism :- Having many forms

① Dynamic Polymorphism :- (Run Time)
→ Use same interface for methods on diff types of objects i.e. diff class.
→ ~~Function~~ ~~overloading~~ ~~overriding~~ method overriding.

normal virtual function
→ redefine in derived class
→ defined also in base class

② (Compile Time) Static Polymorphism :-
→ method overloading :- implement multiple method with same name, but diff parameters in same class

Pure virtual function
→ declared in abstract class, defined in derived class.

Overloading → params diff
overriding → params same.

PAGE NO :
DATE : / /

★ Abstract class : → → Exist for other classes to inherit from it. It is superclass.
→ Cannot be instanted (No objects)
→ Contain atleast one abstract method. (No method in it, just it is declared)
→ Then those abstract method is declared & written in the child class., its method is also written.
- Object is created for child class.

Pure abs

★ Concrete class : → ① Object can be created.
② Cannot be inherited ~~from~~.
③ All methods are implemented in it

★ Interface : List methods of class to be implemented.
But don't contain any actual behaviour
Only function is declared but it donot contain ~~only~~ any code init.

ex: class Raylaw:
def demo (x, y):
pass.

Interface is similar to abstract but interface contains all methods as abstract

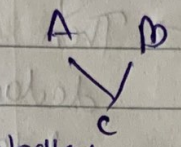
- Interface show capability.
- Abstract shows type.

- Interface class is inherited in child class, and those abstract ~~functions~~ methods are coded in those child class
- Interface donot have object

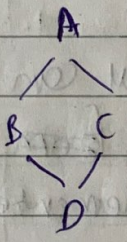
Scanned with CamScanner

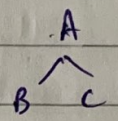
PAGE NO :
DATE : / /

- Composition : Building complex object out of other objects.
- Inheritance :
  - Multilevel A → B → C
  - Multiple Inheritance



  - Protected
  - Hybrid
  - Hierarchical





- Packages contain classes, interfaces
- Class → default Private
  Structure → default Public

Scanned with CamScanner