Page :
Date : / /

→ Procedural Programming :-

OOPS
↳ class (Blueprint)
- → object
- → method
- → func^n
- → Encapsulation
- → polymorphism
- → Abstraction.

Procedural programming is defined as a model which is derived from structured programming based on the concept of step by step execution of program.

→ procedural programming follow the up-down approch where as O.O.P follow bottom up approach.

Ex:- FORTRAN, ALGOL, COBOL, BASIC, Pascal and C

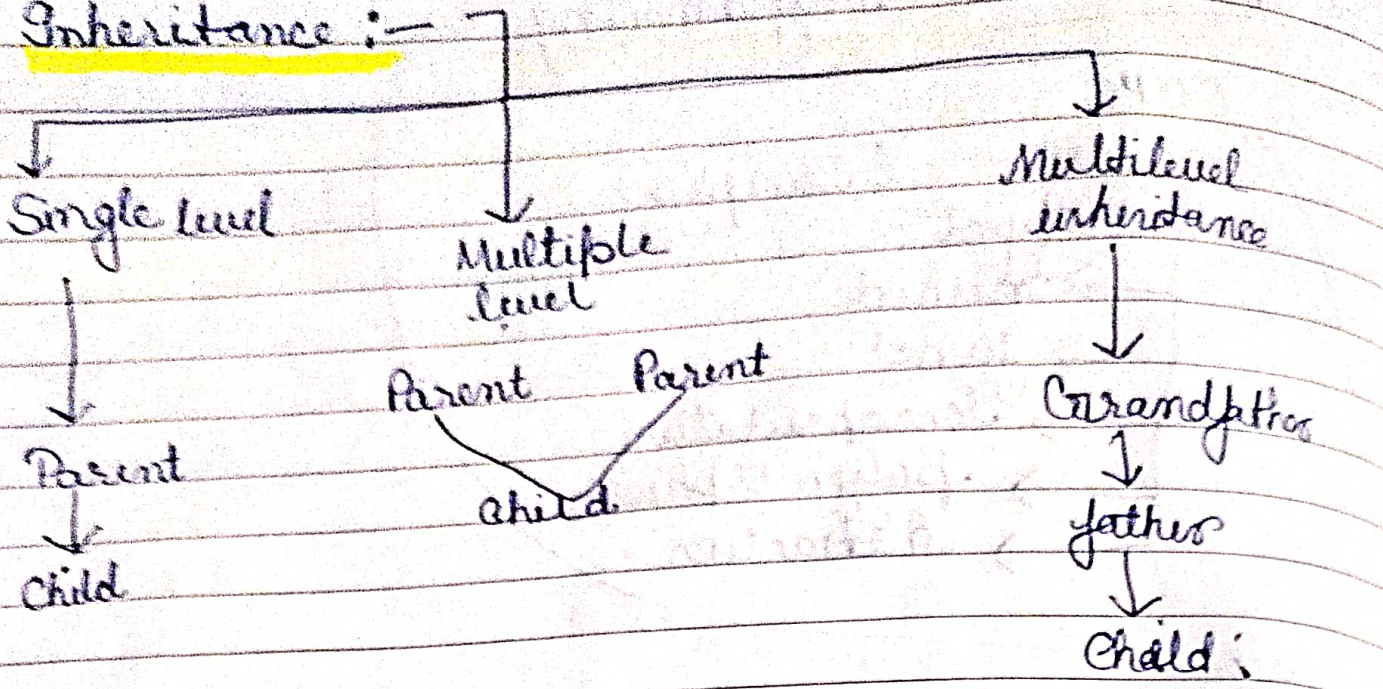
→ Object oriented programming can be defined as a programming model which is based upon the concept of objects.

→ Objects contains data in the form of attributes and code in the form of methods.

Ex:- java, C++, C#, Python, PHP, Ruby, Perl, Objective-C, Dart, Swift, Scala.

Class :— Class is user define datatype that act as the blueprint for individual objects, attributes and methods.

Inheritance :-



Polymorphism:

i) Method overriding

ii) " Overloading

iii) " Overloaded

→ Constructors in Python:- "__init__" dunder or magic method, it is constructor it begins with "__" and end with "__".

when an object is created than the code or instruction inside constructors are must be executed.

```
Class person:
    # init method or Constructor
    def __init__(self, name)
```

→ This variable is uses to intialise value of other variable

each and every there when constructore is created than their must be one variable i.e: self or any other name is

self.name = name
print ("ghfjfkl")

# Sample method
def say_hi(self):
    print("Hello my name is", self.name)

p = person("niharika") [OP :- ghfjk
→ Object created of class 'person'
'niharika' → is parameter which is accessed through 'self.name = name'

p.say_hi() O/P :— My name is niharika

p1 = person("chhaya") OP :— ghfjk
[Constructor executed when object is created]

p1.say_hi() OP → My name is chhaya.

Class :—

- class is a blueprint
- Here 'a' is an object of integer class.
  a = 2
- In python object and variable are same i.e :— each variable is the object of their 'datatype class'
  Ex — a = 2 (a is object of 'int class')
  b = 'chaya' (b is " " 'string class')
  etc.

→ In python 'datatype' is basically a class, when we create a variable of that datatype, it means we are creating object of that class.

→ In class we have two things :
1. Data or property or Attributes
2. funcs or behaviours or method

→ The name of class should be in 'pascal case'
Ex :— ThisIsPascalCase
('eachword 1st letter is capital')

Camel Case : Ex: thisIsCamelCase
In this 1st letter of 1st word. Small and 1st letter of other word capital and rest letter small.

→ The name of method should be in snake case
Ex :— this_is_snake_case
In this each word first letter small and each word is seprated with underscore (_).

→ Structure of class:-

→ Pascalcase

```
class car:
    color = "blue"  # data
    model = "sports"  # "
```

→ Snakecase

```
    def calculate_avg_speed(km, time):
        ----  ----
        ---  ---  ---  {Code of method}
```

Self :- 'Self' is basically a variable name that contain the 'object it self'. We can name it as any variable name.

```
obj = __init__(self):
    self.pin = ""
    self.balance = 0
    self.menu()
```

⇒ We use it to ~~call~~ fetch the attribute or method.

id(self) > Both same because both is same variable or 'obj' is same as 'self'
id(obj)

⇒ We use or pass 'self' in each method because any 'attributes' or 'method' can't call any other 'attributes' or 'method()' to establish relation only their 'obj' can call or fetch the 'attributes' or 'methods()' that why we have to pass the 'obj' as the variable 'self' to establish relation between all methods and attributes.