Ques 1 What are the five concepts of OOPs ? ①

The five concept of OOP are —

(i) Abstraction → Shows only relevant details to the user, hide unexpected or irrelevant features

(ii) Encapsulation → Binds data variables and methods together in a class.

(iii) Inheritance → One class inherits the functions and properties of another class.

(iv) Polymorphism → Objects can take on more than one form and share behaviours.

(v) Classes & objects → Class is a coding template for creating objects.

→ Objects are variable that contain data and functions that can be used to manipulate the data.

Ques 3 Explain the difference Between instance method and class methods. Provide Example of each.

Instance Method → When creating an instance method, the first parameter is always (self).

→ Through the self parameter, instance methods can freely access attribute and other methods on the same object.

Ex →
```
Class K:
    def __init__ (self, name):
        self.name = name
    def introduce (self):
        print ("Hello", self, ~~name~~ self.name).
```

Class method → Instead of passing the instance hiddenly as a first parameter, we're now passing the class itself as a first parameter

Ex → class cls:
@classmethod
def introduce(cls)
print("Hello", cls)

Since, we are passing only a class to the method, no instance is involved.

Ques 4 How does Python implement method overloading?

→ Python overloading Two or more methods have the same name but different types of parameter or both.
→ Python does not support method overloading by default. different ways to achieve method overloading in Python.

Ex → def product(a, b):
p = (a*b)
print(p).

def product(a, b, c):
p = (a*b*c).
print(p)

product(4, 5, 5) ——→ 100

here, it is overloading happen.
↑
second product method will execute
↑

Ques 5 What are the three type of access modifiers in Python? How are they denoted

(i) Public → access by anyone, publically
The Public access modifier allows class member to be accessible anywhere in the progoam.

Ex → class MyClass:
def my-public(self)
print("I am CR7")

→ obj = myclass()
obj.my-public()
'I am CR7'

(2) Protected → The protected access modifier restricted the access of class member within the class and it's subclasses.

→ To denoted protected Member → use → underscore(-)

Ex → def _my_protected_method(self)
print("I am a Protected method")

(3) Private ⇒ it restricts the access of class member to within the class only..

→ Denote private member → use → double underscore prefix (__)

Ex → def __my_private(self)