**Training Report Day-8 not attended**

**15 June 2024**

**Abstraction in Python:**

Abstraction in Python is a fundamental concept in object-oriented programming (OOP) that focuses on hiding the complex implementation details of a class or function and exposing only the necessary parts to the user. This allows users to interact with objects at a higher level of abstraction without needing to understand the intricate details of their implementation.

Example

class Mobile:

    def \_\_init\_\_(self, brand, price):

        print("Inside constructor")

        self.brand = brand

        self.price = price

    def purchase(self):

        print("Purchasing a mobile")

        print("This mobile has brand", self.brand, "and price", self.price)

print("Mobile-1")

mob1=Mobile("Apple", 20000)

mob1.purchase()

print("Mobile-2")

mob2=Mobile("Samsung",3000)

mob2.purchase()

When we invoke the purchase() on a mobile object, we don’t have to know the details of the method to invoke it.

This ability to use something without having to know the details of how it is working is called as abstraction

**Encapsulation in Python:**

Encapsulation is the concept of Object Oriented Programming. It avoids the accidental change in any variable. The value or content of the variable can be accessed by any method of the object.

**Public data access:**

Public data can be accessed by the other functions and can be changed by any outer method. This can change any critical value in the code.

**Private data access:**

We can put a lock on that data by adding a double underscore in front of it, as shown in below code.

Adding a double underscore makes the attribute a private attribute. Private attributes are those which are accessible only inside the class. This method of restricting access to our data is called encapsulation.

class Customer:

    def \_\_init\_\_(self, cust\_id, name, age, wallet\_balance):

        self.cust\_id = cust\_id

        self.name = name

        self.age = age

        self.\_\_wallet\_balance = wallet\_balance

    def update\_balance(self, amount):

        if amount < 1000 and amount > 0:

            self.\_\_wallet\_balance += amount

    def show\_balance(self):

        print ("The balance is ",self.\_\_wallet\_balance)

c1=Customer(100, "Gopal", 24, 1000)

print(c1.wallet\_balance())

This code will give error.

Note: Private variable can be accessed in other method of same class but cannot accessed by any method outside the class.