Python

OOP:

1. Class and objects: let's consider an example. To build a house, an architect must follow a blueprint. Using that blueprint all the house will look the same but the properties in that house, for example color, interior may not be same. So, the blueprint that used to make a house is known as a class and form that blueprint unlimited houses can be built. So the outcome from that blueprint is known as an object.

   -basic python class example

```
class ExampleClass:
      #non parameterized constructor
    def __init__(self):
        print("new object created")
e1 = ExampleClass()
e1
```

```
class Student:
      #parameterized constructor
    def __init__(self, name, dept):
        self.name = name
        self.dept = dept
s1 = Student("fahim", "cse")
print(s1.name)
print(s1.dept)
```

```
class Car:
    def __init__(self, color, brand):
        self.color = color
        self.brand = brand
      #instance method
    def display(self):
        print(self.brand+" is "+ self.color)

c1 = Car("black", "ford")
c2 = Car("red", "toyota")
c1.display()
c2.display()
```

```
class Car:
      #here, no constructor has been initialized.
      #a default constructor will work here.
    def display(self, color, brand):
        print(brand+" is "+color)

c1 = Car()
c2 = Car()
c1.display("black", "ford")
c2.display("red", "toyota")
```

2. 4 pillars of oop: inheritance, polymorphism, encapsulation, abstraction.

- Method overloading:
  -same method with different parameter
  -like other languages, Method overloading can not be achieved in python, that means, python doesn't support multiple methods with the same name. To achieve this, we can put some condition in the actual method which is not an actual way of doing method overloading.

```
class Calculator:
    def product(self, *nums):
        mul = 1;
        for i in nums:
            mul = mul * i
        print (mul)

c1=Calculator()
c1.product(3,2,4)
```

- Inheritance

```
class ParentStudent: #parent class
    def __init__(self, name, dept):
        self.name = name
        self.dept = dept
    def printInfo(self):
        print(self.name, self.dept)
```

```
class ChildStudent(ParentStudent):
    def __init__(self, name, dept):
      #super() helps to inherit all the properties from parent class
        super().__init__(name, dept)

s1=ChildStudent("fahim", "cse")
s1.printInfo()
```

```
class ParentStudent: #parent class
    def __init__(self, name, dept):
        self.name = name
        self.dept = dept
    def printInfo(self):
        print(self.name,self.age,self.dept,self.year)

class ChildStudent(ParentStudent):
    def __init__(self, name, dept, age, graduationYear): #add new
properties to the child class
        super().__init__(name, dept)
        self.age = age
        self.year = graduationYear

s1=ChildStudent("fahim", "cse", 23, 2023)
s1.printInfo()
```

- polymorphism

```
class Car:
  def move(self, brand, model):
    print(model, brand, "can Drive!")

class Boat:
  def move(self, brand, model):
    print(model, brand, "can Sail!")

class Plane:
  def move(self, brand, model):
    print(model, brand, "can Fly!")
```

```
car1 = Car()       #Create an object of Car class
boat1 = Boat() #Create an object of Boat class
plane1 = Plane()     #Create an object of Plane class

car1.move("Ford", "Mustang")
boat1.move("Ibiza", "Touring 20")
plane1.move("Boeing", "747")
```

- Inheritance, polymorphism and method overriding

```
class Vehicle:
  def move(self, brand, model):
    print(model, brand, "can Drive!")

class Car(Vehicle):
    pass #inherits properties and methods from parent class

class Boat(Vehicle):
    #inherits properties from parent
    #overrides move() method
  def move(self, brand, model):
    print(model, brand, "can Sail!")

class Plane(Vehicle):
    #inherits properties from parent
    #overrides move() method
  def move(self, brand, model):
    print(model, brand, "can Fly!")

car1 = Car()       #Create an object of Car class
boat1 = Boat() #Create an object of Boat class
plane1 = Plane()     #Create an object of Plane class

car1.move("Ford", "Mustang")
boat1.move("Ibiza", "Touring 20")
plane1.move("Boeing", "747")
```

```
class Animal:
    def eat(self, name):
        print(name,"eat meats")

class Dog(Animal):
    pass #inherits properties and methods from parent class
class Bird(Animal):
    #inherits properties from parent
    #overrides eat() method
    def eat (self, name):
        print(name, "eat insects")
    def fly (self, name):
        print(name, "can fly")
class Fish(Animal):
    #inherits properties from parent
    #overrides eat() method
    def eat (self, name):
        print(name, "eat aquas plant")
    def swim (self, name):
        print(name, "can swim")

dog1 = Dog()
bird1 = Bird()
fish1 = Fish()
dog1.eat("buzo")
bird1.eat("doyel")
bird1.fly("doyel")
fish1.eat("hilsha")
fish1.swim("hilsha")
```