Polymorphism

1. What is polymorphism?
> The same interface existing in different forms is called polymorphism.

Example :
An addition between two integers 2 + 2 return 4 whereas an addition between two strings "Hello" + "World" concatenates it to "Hello World"

2. What is operator overloading?
> Redefining how an operator operates its operands is called operator overloading.

Syntax:
```
def __operatorFunction__(operandOne, operandTwo):
# Define the operation that has to be performed
```

Example:
```
class Square:
     def __init__(self, side):
          self.side = side
     def __add__(sideOne, sideTwo):
          return(sideOne.side + sideTwo.side)

squareOne = Square(10)
squareTwo = Square(20)
# After overloading __add__ method, squareOne + squareTwo is
# interpreted as Square.__add__(squareOne, squareTwo)

print("Sum of sides of two squares = ", squareOne + squareTwo)
```

3. What is overriding ?
> Modifying the inherited behaviour of methods of a base class in a derived class is called overriding.
Syntax:
```
class BaseClass:
     def methodOne(self):
          class DerivedClass(baseClass):
     def methodOne(self):
          # Redefine the body of methodOne
```

Example:

```
class Shape:
	def area():
		return 0
class Square(Shape):
	def area(self, side):
		return (side * side)
```

4. Why is super() used ?

> super() is used to access the methods of base class.

Example:

```
class BaseClass:
	def baseClassMethod():
		print("This is BaseClassOne")
class DerivedClass(BaseClass):
	def __init__(self):
	# calls the base class method
	super().baseClassMethod()
```