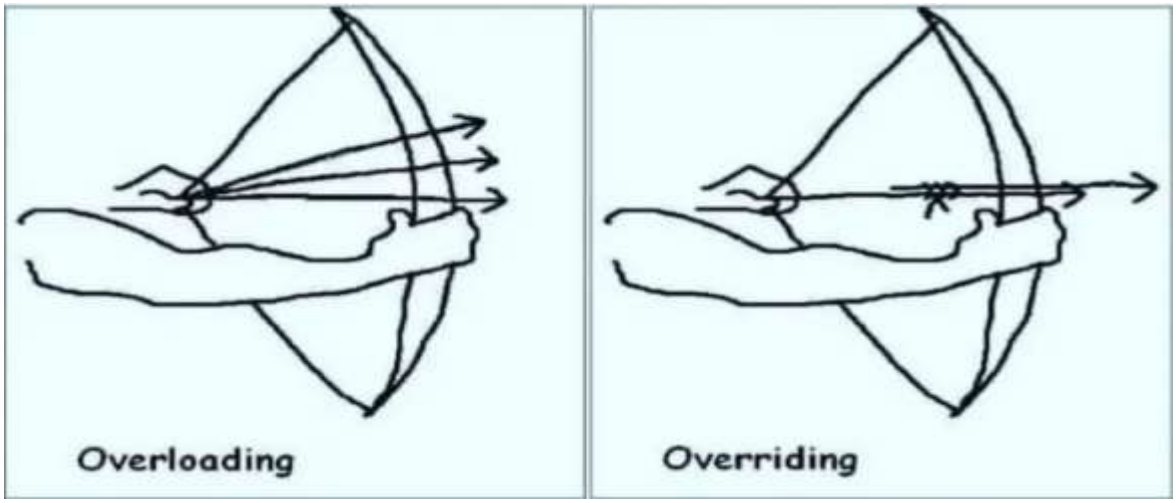
Types of Polymorphism:

1. **Method Overloading**

   Method with same name is declared in multiple instances with varying in arguments.

   Example:

```
def add(m,n):
   print(m+n)
def add(l,m,n):
   print(l+m+n)
add(2,3)
add(2,3,4)
```



2. **Method Overriding**

   If a new method is defined in a class with the existing method name, then the new method overwrites the previous method.

   Example:

```
class Department:
   def __init__(self,name):
      self.name = name
   def details(self):
      print(self.name)

e1 = Department("DPS")
e1.details()
```

```
class Employee(Department):
    def __init__(self,name,emp_name,emp_age):
        Department.__init__(self,name)
        #super().__init__(name)
        self.emp_name = emp_name
        self.emp_age = emp_age
    # def __init__(self,emp_name,emp_age):
    #     self.emp_name = emp_name
    #     self.emp_age = emp_age
    def display(self):
        print(f'{self.emp_name} of {self.emp_age} years old is
working in {self.name} department')
    def details(self):
        # super().details()
        print(self.emp_name)
e1 = Employee("DPS", "Arjun",32)
# e1 = Employee("Arjun",32)
# e1.display()
e1.details()
```

Method Resolution Order

The order in Python in which a method is looked for in the class hierarchy

Example

```
class A:
    def showA(self):
        print("Class A data")
    def display(self):
        print("A class data will be shown here")
class B:
    def showB(self):
        print("Class B data")
    def display(self):
        print("B class data will be shown here")
class C(A,B):
    def showC(self):
        print("Class C data")
    def displayC(self):
```

```
        print("C class data will be shown here")

obj = C()
obj.showC()
obj.showB()
obj.showA()
obj.displayC()
obj.display()
```

Class Methods

- Methods which use class variables inside method implementation called as Class Methods.
- Should explicitly declare @classmethod decorator.

Example:

```
class A:
   num = 9848022663
   @classmethod
   def my_num(cls,name):
    print(f'{name} phone number is {cls.num}')
obj = A
obj.my_num("Sivamani")
```

Static Methods

- General declared methods
- Will not be passing any self/cls keywords & using any instance/class variables inside the methods.
- Should explicitly declare @staticmethod decorator

Example:

```
class A:
   @staticmethod
   def add(x,y):
    print(x+y)
obj = A
obj.add(2,3)
```