Method Overloading

Method overloading is used for achieving polymorphism

What is polymorphism:

- One name different actions or multiple actions
- Python supports overloading writing one method which can act differently in different situations.

How python supports overloading ?

Class Arith:# creating class
Def sum(x, y)# takes two parameters
Return x+y # returning parameters by adding

a= arith()
print(a.sum(10,5))# calling sum
A.sum(8.5, 7.6)

print(a.sum("Hello", "world")

- It is adding int, float , string it is nothing but we are achieving polymorphism
- Python supports overloading implicitly.

Program:

Input:

```
class Arith:
    def sum(self,a,b):# takes two parameters
     return a+b
a= Arith()
print(a.sum(10, 5))
```

Output:

15

Input:

```
class Arith:
    def sum(self,a,b):# takes two parameters
     return a+b
a= Arith()
print(a.sum(8.5, 7.6))
```

Output:
16.1
Input:

```
class Arith:
    def sum(self,a,b):# takes two parameters
     return a+b
a= Arith()
print(a.sum('Hello  ', 'world'))
```

Output:

Hello World

- Concatenation means it will just attach first string to the second string

Input:

```
class Arith:
    def sum(self,a,b):# takes two parameters
     return a+b

    def sum(self, x, y, z):# takes three parameters
        return x + y + z

a= Arith()
print(a.sum('Hello  ', 'world'))
print(a.sum(5, 10, 3))# calling the sum
```

Output:

error

- Python does not allow two methods using same name.
- It will try to call out the second method and shadowed the second method
- We can never call the first method by calling two parameters

How single parameter works for both 2 and 3 parameters

Input:

```
class Arith:
    def sum(self, a, b, c=None):
        s = a + b
        if c==None:
            return s
        else:
            return s + c


a = Arith()
print(a.sum(5,10,3))

print(a.sum(8, 9))
```

Output:

18
17

- So python is having method overloading implicitly
- We don’t have to write multiple methods with the same name but the single method can behave in different type of method.