**Method Overloading Examples**

An object is also created based on the class and we will call its method using zero and one parameter.

**Example 1:**

#!/usr/bin/env python

class Person:

def Hello(self, name=None):

if name is not None:

print('Hello ' + name)

else:

print('Hello ')

# Create instance

obj = Person()

# Call the method

obj.Hello()

# Call the method with a parameter

obj.Hello('Edureka')

**Output:**

Hello

Hello Edureka

To clarify method overloading, we can now call the method Hello() in two ways:

obj.Hello()

obj.Hello('Edureka')

In the above example, we have created a method that can be called with fewer arguments than it is defined to allow. Also, it is not limited to two [variables](https://www.edureka.co/blog/variables-and-data-types-in-python/) and your method can have more variables which are optional.

**Example 2:**

In the following example, we will overload the area method. If there is no argument then it returns 0. And, If we have one argument then it returns the square of the value and assumes you are computing the area of a square. Also, if we have two arguments then it returns the product of the two values and assumes you are computing the area of a

# class

class Compute:

# area method

def area(self, x = None, y = None):

if x != None and y != None:

return x \* y

elif x != None:

return x \* x

else:

return 0

# object

obj = Compute()

# zero argument

print("Area Value:", obj.area())

# one argument

print("Area Value:", obj.area(4))

# two argument

print("Area Value:", obj.area(3, 5))

The above code will give us the following **output:**

Area Value: 0

Area Value: 16

Area Value: 15