Python Classes and Objects in Hindi

Python - Python Classes and Objects

Python ये Object-Oriented Programming(Objects) Language है | इसके साथ-साथ ये Procedural-Oriented Programming(Functions) Language भी है |

ज्यादातर Programming Languages में OOP का concept होता है | Python के Object में variables, functions या methods होते है |
जैसे कि, अगर कोई प्राणी है तो उस प्राणी का behavior और properties जैसे कि, उसका भौकना, चलना, देखना , उसके शरीर की रचना को variables और functions या methods के जरिये लिखा जाता है |
हर एक से अधिक प्राणी को अलग-अलग नाम से उनके objects भी बनाये जाते है |

What is a Class ?

- Class के अन्दर कुछ functions या methods होते है और उससे निगडित कुछ variables दिए जाते है | उन functions और variables को एक ही class पर इकट्ठा किया जाता है और उन data को access करने के लिए उस class का object बनाया जाता है |
- Class अपने data को hold करने का काम करता है |
- Class ये Object की blueprint या layout होता है |
- एक Class के एक या एक से ज्यादा Object create किये जा सकते है |

Defining Class in Python

class को create करने के लिए पहले 'class' keyword का इस्तेमाल किया जाता है |

Syntax :

```
class MyClass:
   "I am a Docstring"
   #class_body
```

Syntax पर 'myClass' ये class का नाम है |
उसके बाद एक docstring लिया गया है | ये docstring class में optional होता है | docstring ये class के बारे में कुछ जानकारी देने के लिए लिया जाता है |
उसके बाद class_body में कुछ statements के रूप में कुछ variables, functions या methods हो सकते है |

Example for Creating a Class

Example पर Class के नाम से ही Object का इस्तेमाल किया गया है |इस class object; का इस्तेमाल class के अलग-अलग attributes access करने के लिए किया जाता है | निचे docstring को access करने के लिए '__doc__' इस class attribute का इस्तेमाल किया गया है और उसके बाद func ये attribute function object को return करता है |

Source Code :

```
class MyClass:        # Class Name
   "I am a Docstring"
   def func(self):
      print("Hello World")

print(MyClass.__doc__)   # docstring attribute
print(MyClass.func)      # func attribute
```

Output :

```
I am a Docstring
<function MyClass.func at 0x02423270>
```

Creating an Object in Python

Object को variable के रूप में ही create किया जाता है | उस variable पर जैसे function को call किया जाता है वैसे ही उस class को function के रूप में variable पर store किया जाता है |

Syntax :

```
Object_Name = Class_Name()
```

Example for Creating a Class

Example पर MyClass इस Class का Object 'obj' बनाया गया है |

Source Code :

```
class MyClass:
   "I am a Docstring"
 var = 1
   def func(self):
      print("Hello World")

obj = MyClass()  #Class Object
```

Accessing Class Variable and Function

Source Code :

```
class MyClass:
   "I am a Docstring"
   var = 1
   def func(self):
      print("Hello World")

obj = MyClass() #Object

print("MyClass.var :",MyClass.var)  #1
#same as
print("obj.var :",obj.var)          #2

print("MyClass.func :",MyClass.func)#3
print("obj.func :",obj.func)        #4

MyClass.func(obj)                   #5
#same as
obj.func()                          #6
```

Output :

```
MyClass.var : 1
obj.var : 1
MyClass.func : <function MyClass.func at 0x02993270>
obj.func : <bound method MyClass.func of <__main__.MyClass object at 0x0278AD50>>
Hello World
Hello World
```

Explaination of Above Example

- **Comment#1 :** यहाँ पर Class name वाले Object(MyClass) से class variable को access किया गया है |
- **Comment#2 :** यहाँ पर Class के बनाये हुए Object(obj) से class variable को access किया गया है |
- **Comment#3 :** यहाँ पर Class name वाले Object(MyClass) से func इस class function को access किया गया है लेकिन ये 'function object' को return करता है |
- **Comment#4 :** यहाँ पर Class के बनाये हुए Object(obj) से func इस class function को access किया गया है लेकिन ये 'method object' को return करता है |
- **Comment#5 :** यहाँ पर MyClass इस class के function पर जो definition पर 'self' इस parameter को pass किया गया है वैसे ही 'obj' इस class के object को argument पर pass किया गया है |
- **Comment#6 :** definition पर 'self' ये parameter pass किया गया है लेकिन यहाँ पर कोई भी argument को pass नहीं किया गया है इसका मतलब है कि जब बनाये हुए object द्वारा function को access किया जाता है तब खुद object ही argument पर pass हो जाता है | उसे अलग से देने की जरुरत होती है |

Class Attribute/Variable and Instance Attribute/Variable

Example पर class variable 'classVar' और instance variable 'inVar' लिया गया है |

class Variable अपने className या उसके object के जरिये access किये जा सकते है लेकिन instance variable को access करने के लिये class के object की जरुरत पड़ती है |

Instance Variable को constructor के अन्दर create किया जाता है |

Source Code :

```
class MyClass:
    classVar = 1 #classVar is a Class Variable

    def __init__(self, inVar):
        self.inVar = inVar    #inVar is a Instance Variable

obj1 = MyClass("Ramesh")
obj2 = MyClass("Rahul")


print("Access Class Variable using Class Name :", MyClass.classVar)
print("Access Class Variable using Class Object :", obj1.classVar)
print(obj1.classVar, obj1.inVar) #Value of inVar is Ramesh
print(obj2.classVar, obj2.inVar) #Value of inVar is Rahul
```

Output :

```
Access Class Variable using Class Name : 1
Access Class Variable using Class Object : 1
1 Ramesh
1 Rahul
```

Changing Class Variable's and Instance Variable's Values

Class के variable की value change करने के लिए ClassName या ClassObject को इस्तेमाल किया जा सकता है लेकिन instance variable की value को change करना हो तो ClassObject का ही इस्तेमाल किया जाता है |

Source Code :

```
class MyClass:
    classVar = 1 #classVar is a Class Variable

    def __init__(self, inVar):
        self.inVar = inVar    #inVar is a Instance Variable

obj1 = MyClass("Ramesh")
obj2 = MyClass("Rahul")

print("Before Changing :")
print(obj1.classVar, obj1.inVar)


MyClass.classVar = 2 #Changing Class Variable's Value by ClassName
obj1.inVar = "Rakesh" #Changing Instance Variable's Value

print("After Changing :")
print(obj1.classVar, obj1.inVar)
```

Output :

```
Before Changing :
1 Ramesh
After Changing :
2 Rakesh
```