modul modul pembelajaran
VedulPro

BELAJAR PYTHON 12

Object & Class Python

Python telah menjadi bahasa berorientasi objek sejak bahasa Python sendiri dibuat. Untuk membuat dan menggunakan kelas dan objek pada Python benar-benar mudah. Pada tutorial ini Anda akan dibantu untuk menjadi ahli dalam penggunaan pemrograman berorientasi objek Python.

Jika Anda tidak memiliki pengalaman sebelumnya dengan pemrograman berorientasi objek (OOP), Anda mempelajarinya terlebih dahulu agar Anda dapat memahami konsep dasarnya.

Jika memang sudah mengerti konsep dasar OOP berikut ini adalah pengenalan dari Object-Oriented Programming (OOP) untuk membantu Anda.

Istilah Dalam OOP

Istilah	Penjelasan
Class	Prototipe yang ditentukan pengguna untuk objek yang mendefinisikan seperangkat atribut yang menjadi ciri objek kelas apa pun. Atribut adalah data anggota (variabel kelas dan variabel contoh) dan metode, diakses melalui notasi titik.
Class variable	Sebuah variabel yang dibagi oleh semua contoh kelas. Variabel kelas didefinisikan dalam kelas tapi di luar metode kelas manapun. Variabel kelas tidak digunakan sesering variabel contoh.
Data member	Variabel kelas atau variabel contoh yang menyimpan data yang terkait dengan kelas dan objeknya.
Function overloading	Penugasan lebih dari satu perilaku ke fungsi tertentu. Operasi yang dilakukan bervariasi menurut jenis objek atau argumen yang terlibat.
Instance variable	Variabel yang didefinisikan di dalam sebuah metode dan hanya dimiliki oleh instance kelas saat ini.
Inheritance	Pengalihan karakteristik kelas ke kelas lain yang berasal darinya.
Instance	Objek individu dari kelas tertentu. Obyek obj yang termasuk dalam Lingkaran kelas, misalnya, adalah turunan dari Lingkaran kelas.
Instantiation	Penciptaan sebuah instance dari sebuah kelas.
Method	Jenis fungsi khusus yang didefinisikan dalam definisi kelas.

Object	Contoh unik dari struktur data yang didefinisikan oleh kelasnya. Objek terdiri dari kedua anggota data (variabel kelas dan variabel contoh) dan metode.
Operator overloading	Penugasan lebih dari satu fungsi ke operator tertentu.

Membuat Class Python

Statement class digunakan untuk membuat definisi kelas baru. Nama kelas segera mengikuti kelas kata kunci diikuti oleh titik dua sebagai berikut.

```
class ClassName: 'Optional class documentation string' class_suite
```

Dibawah ini adalah contoh cara membuat class dan penggunaanya :

```
class Employee:
'Common base class for all employees'
empCount = 0

def **init**(self, name, salary):
self.name = name
self.salary = salary
Employee.empCount += 1

def displayCount(self):
print "Total Employee %d" % Employee.empCount

def displayEmployee(self):
print "Name : ", self.name, ", Salary: ", self.salary
```

Membuat Instance Objects

Untuk membuat instances kelas, Anda memanggil class menggunakan nama class dan meneruskan argumen apa pun yang metode init terima.

```
This would create first object of Employee class
emp1 = Employee("Zara", 2000)
This would create second object of Employee class
emp2 = Employee("Manni", 5000)
```

Mengakses Atribut

Anda mengakses atribut objek menggunakan dot operator dengan objek. Variabel kelas akan diakses dengan menggunakan nama kelas sebagai berikut :

```
emp1.displayEmployee()
emp2.displayEmployee()
print ("Total Employee %d" % Employee.empCount)
```

Contoh lengkapnya, silahkan lihat kode dibawah ini.

```
class Employee:
'Common base class for all employees'
empCount = 0

def **init**(self, name, salary):
self.name = name
self.salary = salary
Employee.empCount += 1

def displayCount(self):
print ("Total Employee %d" % Employee.empCount)

def displayEmployee(self):
print ("Name : ", self.name, ", Salary: ", self.salary)

#This would create first object of Employee class"
emp1 = Employee("Zara", 2000)
#This would create second object of Employee class"
emp2 = Employee("Manni", 5000)
emp1.displayEmployee()
emp2.displayEmployee()
print ("Total Employee %d" % Employee.empCount)
```