**Constructor \_\_init\_\_**

Constructor \_\_init\_\_ pada Python adalah metode khusus yang digunakan untuk menginisialisasi objek dari suatu kelas. Metode ini otomatis dipanggil saat objek dibuat dari kelas tersebut. Nama \_\_init\_\_ merupakan singkatan dari "initialize" atau "inisialisasi". Ketika Anda membuat objek baru dari suatu kelas, \_\_init\_\_ akan dieksekusi secara otomatis.

Berikut adalah contoh programnya:

class Hero: # template

    def \_\_init\_\_(self):

        print("Hallo")

hero1 = Hero()

hero2 = Hero()

Program tersebut mendefinisikan sebuah kelas `Hero` sebagai template untuk membuat objek hero. Kelas ini memiliki constructor (`\_\_init\_\_`) yang mencetak pesan "Hallo" setiap kali objek dibuat menggunakan kelas ini.

Berikut adalah penjelasan singkat tentang program di atas:

1. **Definisi Kelas Hero:**

class Hero:

def \_\_init\_\_(self):

print("Hallo")

* Kelas `Hero` memiliki constructor (`\_\_init\_\_`) yang tidak memiliki parameter (selain `self` yang merujuk pada objek yang dibuat).
* Setiap kali objek dibuat menggunakan kelas ini, pesan "Hallo" akan dicetak.

1. **Membuat Objek Hero dan Mencetak Pesan:**

hero1 = Hero()

hero2 = Hero()

* Dua objek hero (`hero1` dan `hero2`) dibuat dengan menggunakan kelas `Hero`.
* Setiap kali objek dibuat, constructor (`\_\_init\_\_`) dijalankan, dan pesan "Hallo" dicetak.

Contoh hasil keluaran yang mungkin:

Hallo

Hallo

Program ini memberikan contoh sederhana tentang bagaimana constructor (`\_\_init\_\_`) dalam suatu kelas dijalankan setiap kali objek dari kelas tersebut dibuat. Pesan "Hallo" menjadi output karena itulah yang dijalankan dalam constructor. Program ini mungkin tidak melakukan banyak hal selain mencetak pesan, tetapi memberikan pemahaman dasar tentang cara konstruktor bekerja dalam konteks pembuatan objek.

class Hero: #template

    def \_\_init\_\_(self, inputName):

        self.name = inputName

hero1 = Hero("sniper")

hero2 = Hero("mirana")

hero3 = Hero("ucup")

print(hero1.name)

print(hero2.name)

print(hero3.name)

Program di atas mendefinisikan sebuah kelas `Hero` sebagai template untuk membuat objek hero. Kelas ini memiliki constructor (`\_\_init\_\_`) yang menerima satu parameter (`inputName`) untuk menginisialisasi atribut `name` pada objek. Setelah itu, tiga objek hero dibuat menggunakan kelas ini, masing-masing dengan nama yang berbeda.

Berikut adalah penjelasan singkat tentang program tersebut:

1. **Definisi Kelas Hero:**

class Hero:

def \_\_init\_\_(self, inputName):

self.name = inputName

* Kelas `Hero` memiliki constructor (`\_\_init\_\_`) yang menerima satu parameter (`inputName`) untuk inisialisasi atribut `name` pada objek.
* Saat objek dari kelas ini dibuat, nilai yang diberikan pada parameter `inputName` akan diassign ke atribut `name` pada objek tersebut.

1. **Membuat Objek Hero dengan Nama Berbeda:**

hero1 = Hero("sniper")

hero2 = Hero("mirana")

hero3 = Hero("ucup")

* Tiga objek hero (`hero1`, `hero2`, dan `hero3`) dibuat dengan menggunakan kelas `Hero` dan memberikan nama yang berbeda pada setiap objek.

1. **Mencetak Nama Hero:**

print(hero1.name)

print(hero2.name)

print(hero3.name)

* Menggunakan `print` untuk mencetak nama dari setiap objek hero.
* Atribut `name` dari masing-masing objek diakses dan dicetak.

Contoh hasil keluaran yang mungkin:

sniper

mirana

ucup

Program ini memberikan contoh sederhana tentang cara menginisialisasi dan mengakses atribut objek melalui constructor. Setiap objek hero memiliki nama yang berbeda, dan program mencetak nama dari masing-masing objek.

**Contoh \_\_init\_\_ dengan parameter inputName dan dengan parameter khusus self**

class Hero: #template

    def \_\_init\_\_(self, inputName):

        self.name = inputName

hero1 = Hero("sniper")

hero2 = Hero("mirana")

hero3 = Hero("ucup")

print(hero1.name)

print(hero2.name)

print(hero3.name)

Program di atas untuk mendefinisikan sebuah kelas Hero dengan sebuah constructor (\_\_init\_\_). Kelas ini digunakan untuk membuat objek hero dengan atribut-atribut seperti nama (name), health (health), power (power), dan armor (armor).

Berikut adalah penjelasan singkat tentang program tersebut:

1. **Definisi Kelas Hero:**

class Hero:

def \_\_init\_\_(self, inputName, inputHealth, inputPower, inputArmor):

self.name = inputName

self.health = inputHealth

self.power = inputPower

self.armor = inputArmor

* Kelas Hero memiliki empat atribut: name, health, power, dan armor.
* Constructor \_\_init\_\_ menerima empat parameter (inputName, inputHealth, inputPower, inputArmor), dan menginisialisasi atribut-atribut objek (self.name, self.health, self.power, self.armor) dengan nilai-nilai yang diberikan.

1. **Membuat Objek Hero:**

hero1 = Hero("snipper", 100, 10, 4)

hero2 = Hero("mirana", 100, 15, 1)

hero3 = Hero("ucup", 1000, 100, 0)

Tiga objek hero (hero1, hero2, dan hero3) dibuat dengan menggunakan kelas Hero dan memberikan nilai-nilai awal untuk setiap atribut.

1. **Mencetak Informasi Objek menggunakan \_\_dict\_\_:**

**print(hero1.\_\_dict\_\_)**

**print(hero2.\_\_dict\_\_)**

**print(hero3.\_\_dict\_\_)**

* \_\_dict\_\_ adalah atribut bawaan dalam Python yang berisi kamus (dictionary) yang berisi atribut dan nilai-nilai objek.
* Dengan mencetak \_\_dict\_\_ dari setiap objek, kita dapat melihat nilai-nilai dari setiap atribut untuk masing-masing objek.

Contoh hasil keluaran yang mungkin:

{'name': 'snipper', 'health': 100, 'power': 10, 'armor': 4}

{'name': 'mirana', 'health': 100, 'power': 15, 'armor': 1}

{'name': 'ucup', 'health': 1000, 'power': 100, 'armor': 0}

Program tersebut memberikan cara untuk membuat objek hero dengan atribut-atribut yang dapat diakses dan diubah. Penggunaan \_\_dict\_\_ mempermudah pengecekan nilai-nilai atribut dari objek-objek tersebut.