**Materi Lab 8**

**Object Oriented Programming (OOP) dan Class**

Object Oriented Programming atau yang juga dikenal dengan OOP adalah salah satu prinsip pemrograman yang berbasis objek. OOP dapat memudahkan kita karena prinsip **encapsulation**, **inheritance**, dan **polymorphism**.

**Mengecek Jenis Class**

Sebenarnya selama ini kalian sudah bermain dengan class dan object tanpa kalian sadari. Ya, int, str, float, bool, list, dict, dan lain-lain sebenarnya adalah class. Kalian bisa menggunakan method type(x) atau isinstance(x, class) untuk **mengecek jenis class** suatu nilai.

|  |
| --- |
| >>> type(123)<class 'int'>>>> type("ddp asyiik")<class 'str'>>>> type(True or False)<class 'bool'>>>> isinstance(["H", "T", "T", "T", "H"], list)True>>> isinstance((10, -4), tuple)True>>> isinstance({"Budi": 628123123123, "Andi": 628321321321}, dict)True |

**Membuat Class dan Object**

Kalian juga bisa **membuat class** kalian sendiri dengan menggunakan keyword class. Harus diperhatikan bahwa class dan object **bukan** hal yang sama. Class hanyalah cetakannya dan object adalah hasil cetakannya. Ibarat membuat kue, class adalah cetakan kuenya dan object adalah kuenya itu sendiri. Ingat bahwa parameter self harus menjadi parameter pertama di suatu class method.

**Atribut dari Object**

Sebuah **object** memiliki **atribut-atribut**, yaitu :

1. Tempat untuk menyimpan informasi mengenai object tersebut, yaitu **instance variables/data fields**.
2. Fungsi yang “hidup” di dalam sebuah object yang disebut dengan **methods**

|  |
| --- |
| class Komputer: ''' Ini adalah contoh constructor method. Method ini bisa dipanggil ketika kalian  ingin membuat sebuah object Komputer baru Di dalamnya didefinisikan instance variable ''' def \_\_init\_\_(self, pemilik, processor, ram, gpu): self.pemilik = pemilik  self.processor = processor  self.ram = ram self.gpu = gpu ''' Ini adalah method string,  jika kalian melakukan print(komputer\_object), maka hasil return dari method inilah yang akan dicetak ''' def \_\_str\_\_(self): res = f"=== {self.pemilik}'s PC ===\n" res += f"Processor: {self.processor}\n" res += f"RAM: {self.ram} GB\n" res += f"Graphics Card: {self.gpu}" return res |

|  |
| --- |
| >>> pc1 = Komputer(“Mr. X”, “Intel Pentium”, 1, “None”)>>> pc2 = Komputer(“Ms. Y”, “Intel Core i7”, 16, “RTX 2080”)>>> pc3 = Komputer(“Mrs. Z”, “Intel Core i3”, 4, “MX 110”)>>> print(pc1)=== Mr. X's PC ===Processor: Intel PentiumRAM: 1 GBGraphics Card: None>>> print(pc2)=== Ms. Y's PC ===Processor: Intel Core i7RAM: 16 GBGraphics Card: RTX 2080>>> print(pc3)=== Mrs. Z's PC ===Processor: Intel Core i3RAM: 4 GBGraphics Card: MX 110>>> print(type(pc1))<class '\_\_main\_\_.Komputer'>>>> print(type(pc2))<class '\_\_main\_\_.Komputer'>>>> print(type(pc2) == Komputer)True>>> print(type(pc3) == Komputer)True |

**Memanggil Instance Variable**

Jika telah didefinisikan, maka sebuah instance variabel dapat dipanggil dengan cara komputer\_object.instance\_variable

|  |
| --- |
| >>> pc1 = Komputer(“Mr. X”, “Intel Pentium”, 1, “None”)>>> pc2 = Komputer(“Ms. Y”, “Intel Core i7”, 16, “RTX 2080”)>>> pc3 = Komputer(“Mrs. Z”, “Intel Core i3”, 4, “MX 110”)>>> nama\_pemilik = pc1.pemilik>>> print(nama\_pemilik)Mr. X>>> print(pc2.processor)Intel Core i7>>> print(pc3.ram + 6)10>>> print(pc3.processor.split())[“Intel”, “Core”, “i3”] |

**Membuat dan Memanggil Class Method**

Selain method \_\_init\_\_ dan \_\_str\_\_ kalian juga bisa menambahkan method lain. Jangan lupa untuk menambahkan parameter self sebagai **parameter pertama** suatu method yang membuat suatu method tersebut dapat mengikat dirinya ke object yang memanggil method tersebut. Nama parameter pertama tersebut sebenarnya bebas, namun self yang paling umum digunakan.

|  |
| --- |
|  ... masih di class Komputer …  def cetak\_pemilik(self): print(self.pemilik) def has\_bigger\_ram(self, other): return self.ram > other.ram def swap\_ram(self, other): self.ram, other.ram = other.ram, self.ram |

|  |
| --- |
| >>> pc1 = Komputer(“Mr. X”, “Intel Pentium”, 1, “None”)>>> pc2 = Komputer(“Ms. Y”, “Intel Core i7”, 16, “RTX 2080”)>>> pc3 = Komputer("Mrs. Z", "Intel Core i3", 4, "MX 110")>>> Komputer.cetak\_pemilik(pc1)Mr. X>>> pc2.cetak\_pemilik()Ms. Y>>> pc1.has\_bigger\_ram(pc2)False>>> pc3.has\_bigger\_ram(pc1)True>>> Komputer.has\_bigger\_ram(pc2, pc3)True>>> pc1.ram1>>> pc2.ram16>>> pc1.swap\_ram(pc2)>>> pc1.ram16>>> pc2.ram1>>> Komputer.swap\_ram(pc2, pc1)>>> pc1.ram1>>> pc2.ram16 |