Python OOP

(Object Oriented Programming)

Hello!

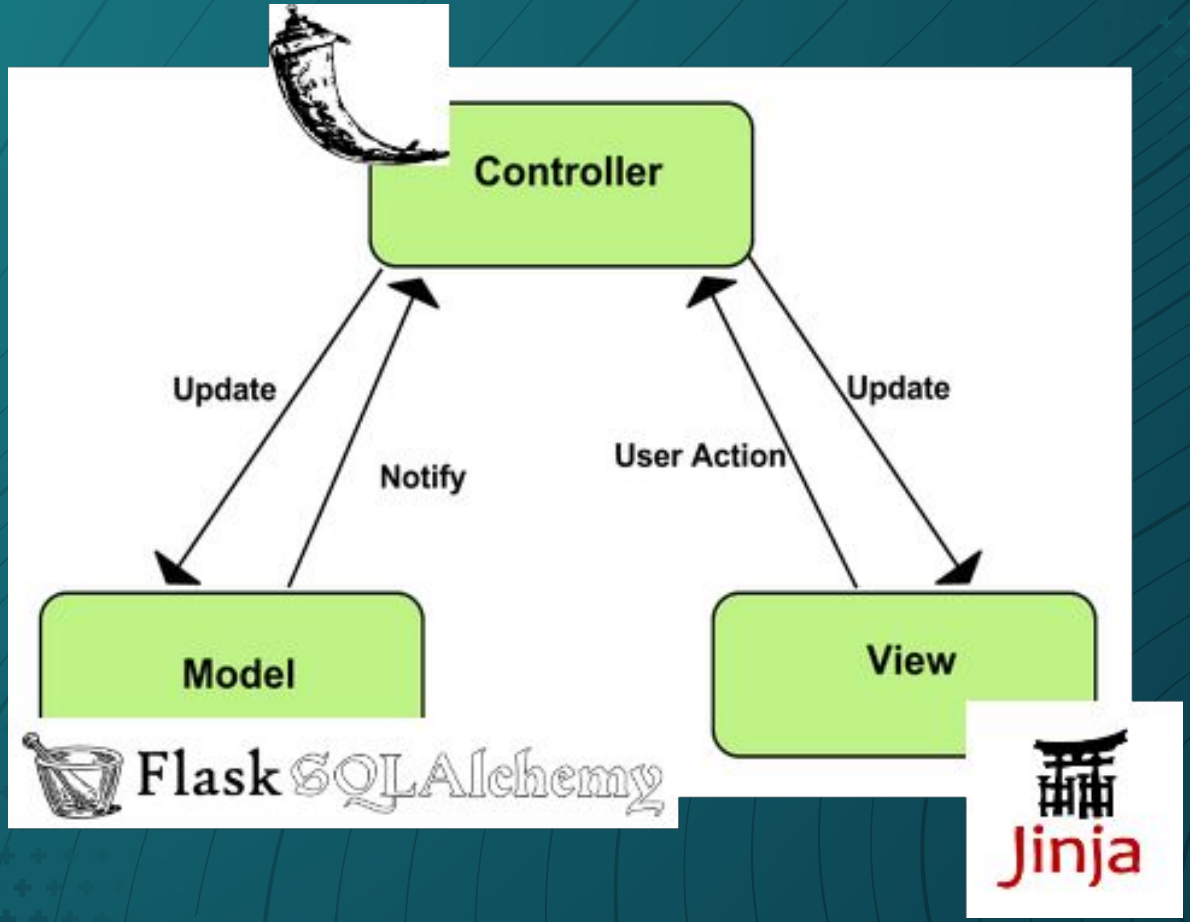
- Overview
- Constructor
- Magic Method
- Inheritance

Mengapa harus OOP ?

- Dengan OOP, kode-kode yang kita buat menjadi lebih rapih dan terstruktur.
- Konsep OOP memudahkan kita untuk menganalisa program yang kita akan buat. Ini akan sangat terasa kalau kita membuat program besar dan riwet.

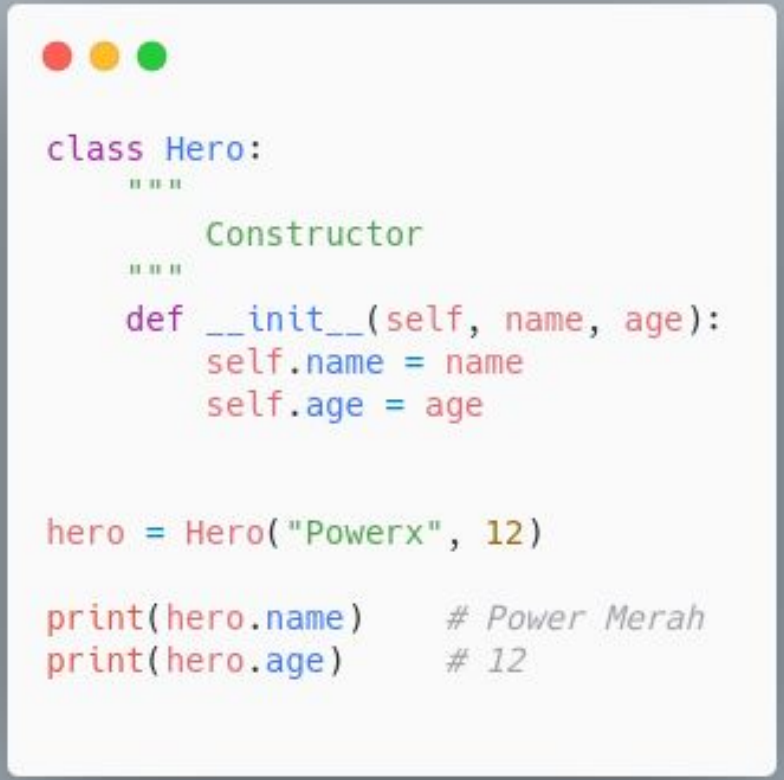
MVC

(Model View Controller)


Controller
Update
Notify
User Action
Update
Model
View
Flask SQLAlchemy
Jinja

1. Contructor

Konsktruktor adalah method yang pertama kali dijalankan pada saat sebuah objek pertama kali diciptakan.



```python
class Hero:
    """
        Constructor
    """
    def __init__(self, name, age):
        self.name = name
        self.age = age


hero = Hero("Powerx", 12)

print(hero.name)     # Power Merah
print(hero.age)      # 12
```

2. Magic Method

Method yang akan dipanggil secara otomatis pada kondisi tertentu.

Other standard Python operators

- Many standard operators and funtions:
  - https://docs.python.org/3/library/operator.html

- Common Arithmetic operators
  - object.__add__(*self, other*)
  - object.__sub__(*self*, *other*)
  - object.__mul__(*self*, *other*)
  - object.__floordiv__(*self*, *other*)
  - object.__truediv__(*self*, *other*)

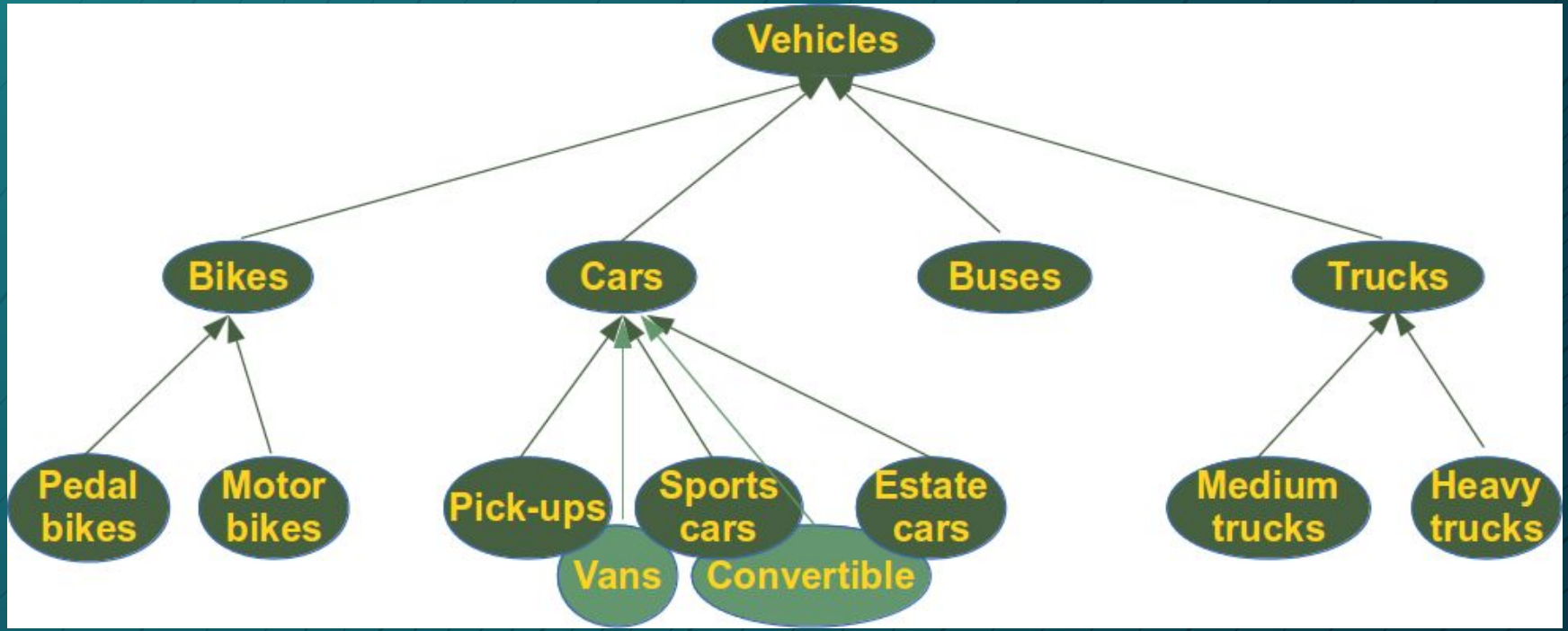
- Common Relational operators
  - object.__lt__(*self*, *other*)
  - object.__le__(*self*, *other*)
  - object.__eq__(*self*, *other*)
  - object.__ne__(*self*, *other*)
  - object.__gt__(*self*, *other*)
  - object.__ge__(*self*, *other*)

```python
class Operasi(object):
    def __init__(self, math_list):
        self.math_list = math_list

    def __sub__(self, other):
        minuslst = []
        zipped = zip(self.math_list, other.math_list)
        for tup in zipped:
            minuslst.append(tup[0] - tup[1])

        return Operasi(minuslst)

    def __add__(self, other):
        addlst = [x + y for x, y in zip(self.math_list, other.math_list)]
        return Operasi(addlst)

    def __mul__(self, other):
        mullst = [x * y for x, y in zip(self.math_list, other.math_list)]
        return Operasi(mullst)

    def __repr__(self):
        return str(self.math_list)


x = Operasi([100, 90, 80, 70, 60])
y = Operasi([10, 9, 8, 7, 6])

p = x - y
z = x + y
q = x * y

print('Pengurangan: ' + str(p))     # [90, 81, 72, 63, 54]
print('Tambah: ' + str(z))          # [110, 99, 88, 77, 66]
print('Perkalian: ' + str(q))       # [1000, 810, 640, 490, 360]
```

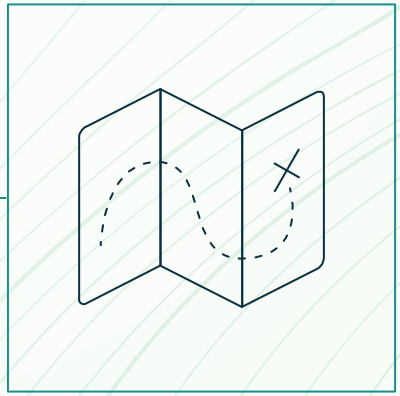
3. Inheritance (Perwarisan)

Inheritance atau Pewarisan/Penurunan adalah konsep pemrograman dimana sebuah class dapat 'menurunkan' property dan method yang dimilikinya kepada class lain.

Perwarisan


Vehicles
Bikes
Cars
Buses
Trucks
Pedal bikes
Motor bikes
Pick-ups
Sports cars
Estate cars
Vans
Convertible
Medium trucks
Heavy trucks

```python
class Hero:

    def __init__(self, name, age):
        self.name = name
        self.age = age

    def profil(self):
        print("Hi, i am " + self.name + " and ",  self.age,  " year old ")
        print("")


class AnakHero(Hero):
    pass


hero = Hero("Power Merah", 27)
anakHero = AnakHero("Anak Merah", 5)

hero.profil()       # Hi, i am Power Merah and  27  years old
anakHero.profil()   # Hi, i am Anak Merah and  5  years old
```



Thanks!

Any questions?