Module 01

Class

Data Science Developer

Purwadhika
Startup and Coding School

Class

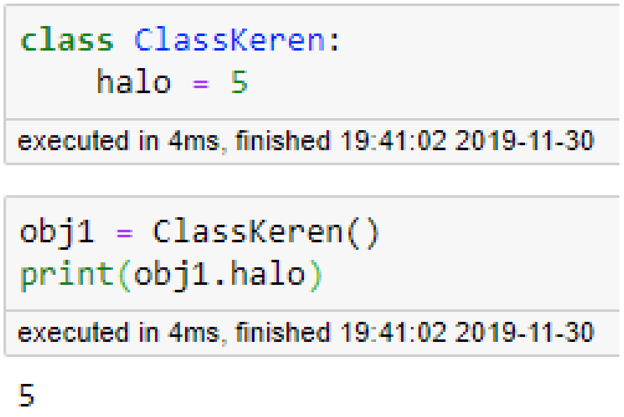
Python is an object oriented programming language.

Almost everything in Python is an object, with its properties and methods.

A Class is like an object constructor, or a "blueprint" for creating objects.

Create a Class and Object

To create a class, use the keyword class, and then we can use the class named ClassKeren to create objects:



```
class ClassKeren:
    halo = 5
```

```
executed in 4ms, finished 19:41:02 2019-11-30
```

```
obj1 = ClassKeren()
print(obj1.halo)
```

```
executed in 4ms, finished 19:41:02 2019-11-30
```

```
5
```

The __init__() Function

The examples above are classes and objects in their simplest form, and are not really useful in real life applications.

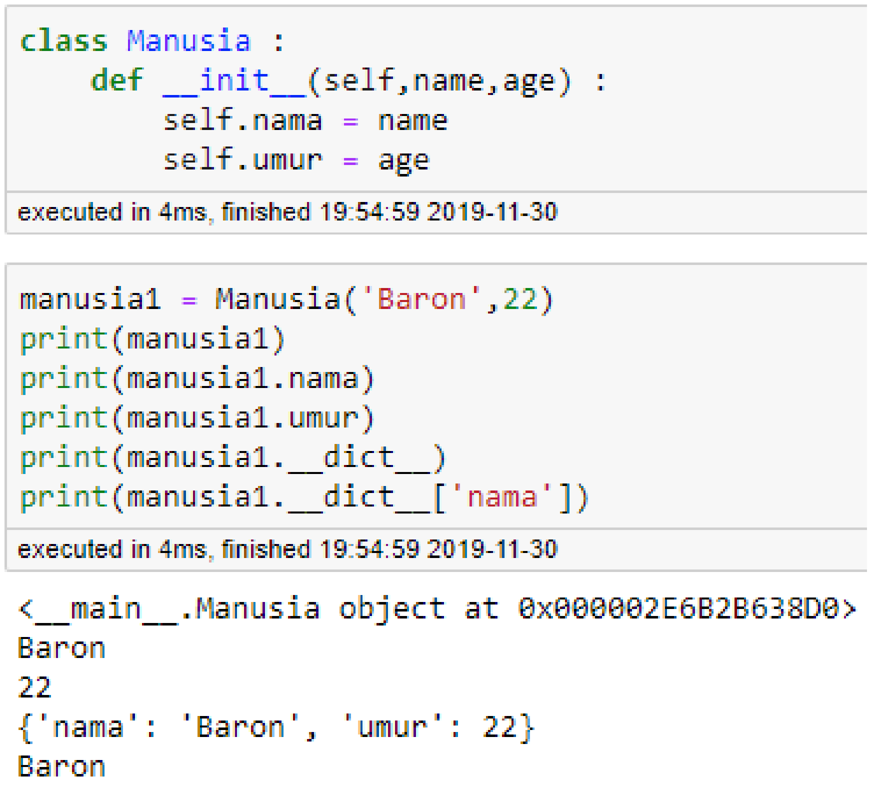
To understand the meaning of classes we have to understand the built-in __init__() function.

All classes have a function called __init__(), which is always executed when the class is being initiated.

Use the __init__() function to assign values to object properties, or other operations that are necessary to do when the object is being created:

Purwadhika
Startup and Coding School

The __init__() Function



```
class Manusia :
    def __init__(self,name,age) :
        self.nama = name
        self.umur = age
```

executed in 4ms, finished 19:54:59 2019-11-30

```
manusia1 = Manusia('Baron',22)
print(manusia1)
print(manusia1.nama)
print(manusia1.umur)
print(manusia1.__dict__)
print(manusia1.__dict__['nama'])
```

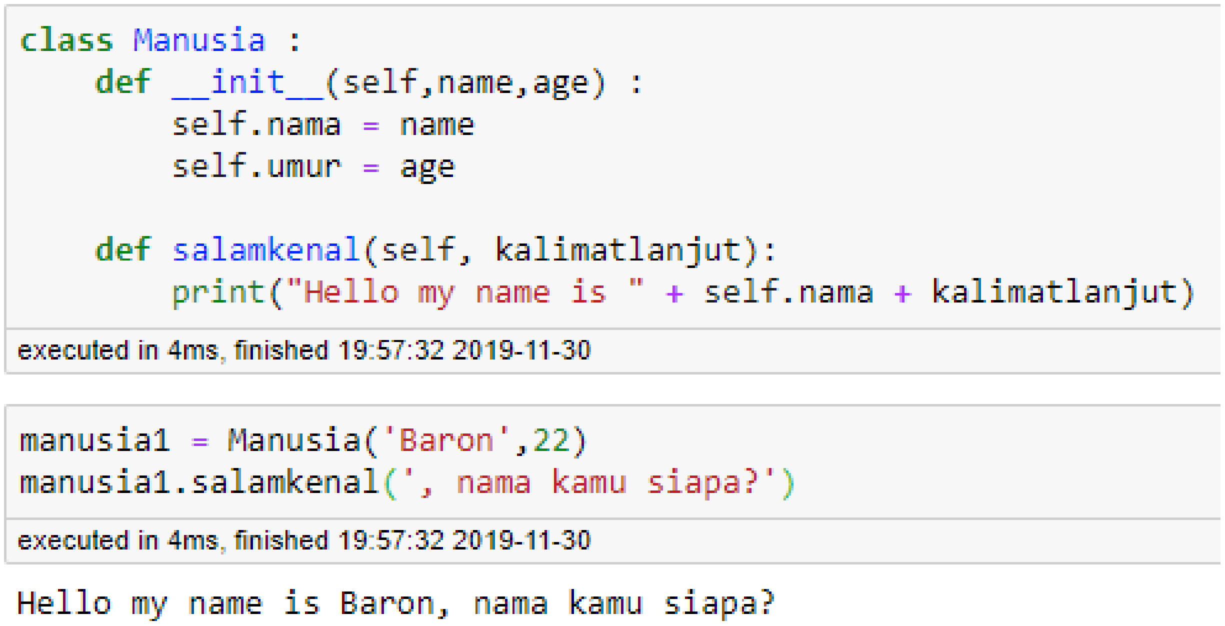
executed in 4ms, finished 19:54:59 2019-11-30

```
<__main__.Manusia object at 0x000002E6B2B638D0>
Baron
22
{'nama': 'Baron', 'umur': 22}
Baron
```

Object Methods

Objects can also contain methods. Methods in objects are functions that belong to the object.

Let us create a method in the Manusia class:



```python
class Manusia :
    def __init__(self,name,age) :
        self.nama = name
        self.umur = age

    def salamkenal(self, kalimatlanjut):
        print("Hello my name is " + self.nama + kalimatlanjut)
```

```
executed in 4ms, finished 19:57:32 2019-11-30
```

```python
manusia1 = Manusia('Baron',22)
manusia1.salamkenal(', nama kamu siapa?')
```

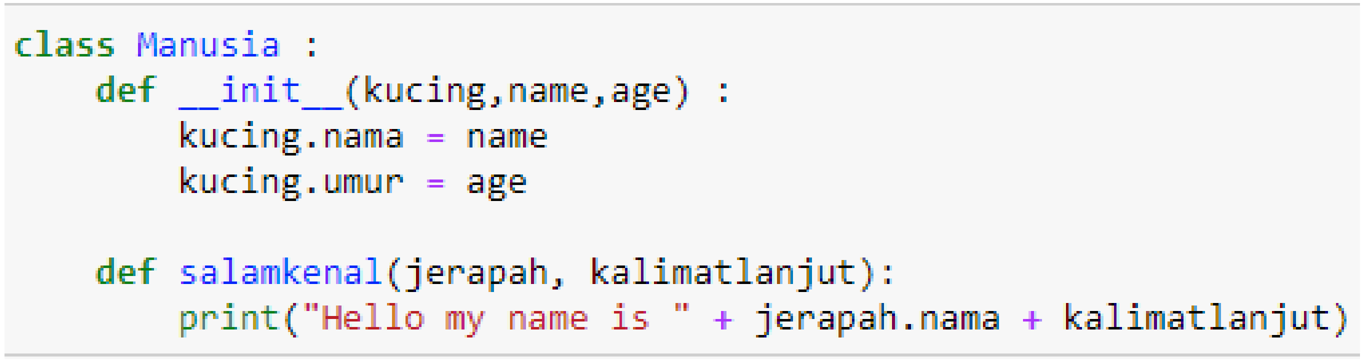
```
executed in 4ms, finished 19:57:32 2019-11-30
```

```
Hello my name is Baron, nama kamu siapa?
```

The self Parameter

The self parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.

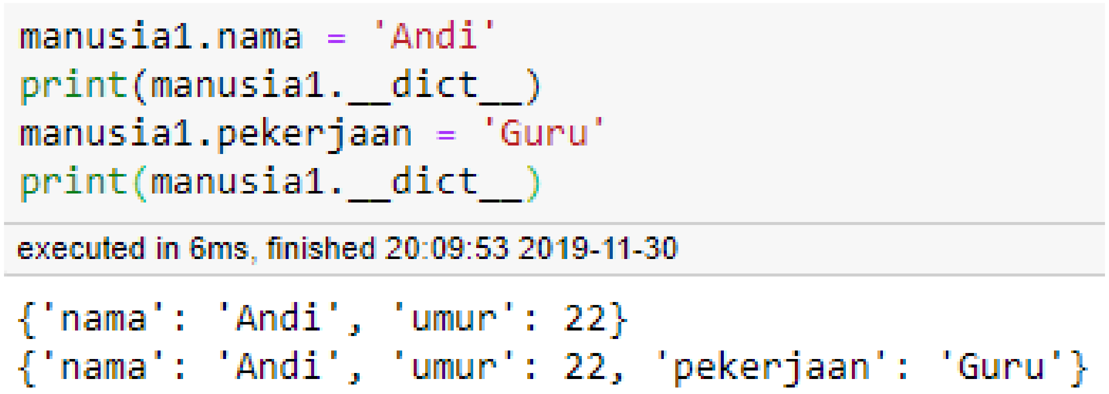
It does not have to be named self , you can call it whatever you like, but it has to be the first parameter of any function in the class:



```python
class Manusia :
    def __init__(kucing,name,age) :
        kucing.nama = name
        kucing.umur = age

    def salamkenal(jerapah, kalimatlanjut):
        print("Hello my name is " + jerapah.nama + kalimatlanjut)
```

Modify Object Properties

You can modify properties on objects like this, or even add a new one :



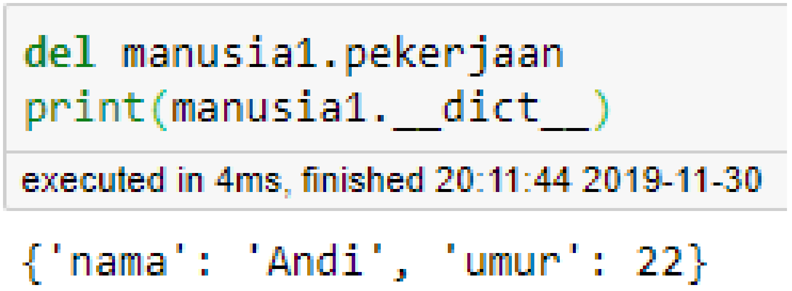
```
manusia1.nama = 'Andi'
print(manusia1.__dict__)
manusia1.pekerjaan = 'Guru'
print(manusia1.__dict__)
```

executed in 6ms, finished 20:09:53 2019-11-30

```
{'nama': 'Andi', 'umur': 22}
{'nama': 'Andi', 'umur': 22, 'pekerjaan': 'Guru'}
```

Delete Object Properties

You can delete properties on objects by using the del keyword:



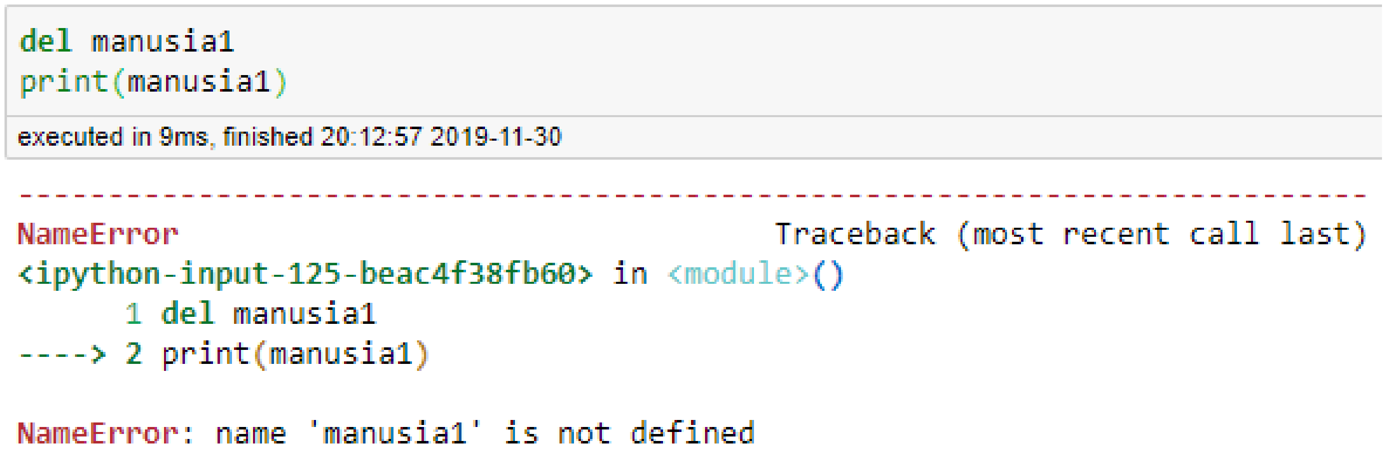
```
del manusia1.pekerjaan
print(manusia1.__dict__)
```

executed in 4ms, finished 20:11:44 2019-11-30

```
{'nama': 'Andi', 'umur': 22}
```

Delete Objects

You can delete objects by using the del keyword:



```
del manusia1
print(manusia1)
```

executed in 9ms, finished 20:12:57 2019-11-30

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-125-beac4f38fb60> in <module>()
      1 del manusia1
----> 2 print(manusia1)

NameError: name 'manusia1' is not defined
```

Inheritance

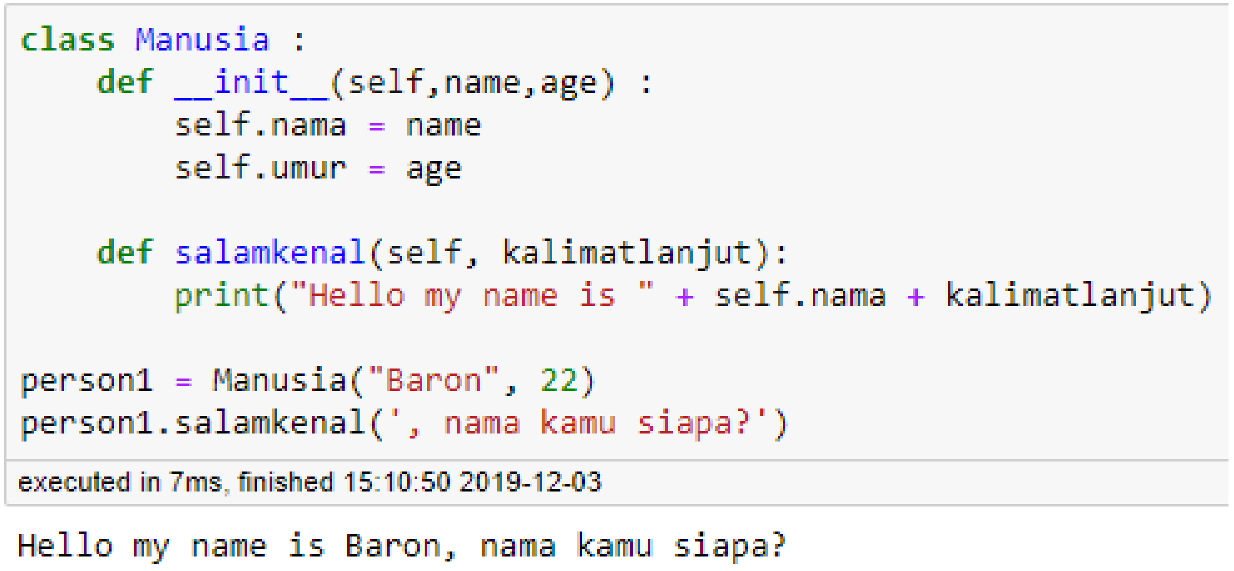
Inheritance allows us to define a class that inherits all the methods and properties from another class.

Parent class is the class being inherited from, also called base class.

Child class is the class that inherits from another class, also called derived class.

Create a Parent Class

Any class can be a parent class, so the syntax is the same as creating any other class:



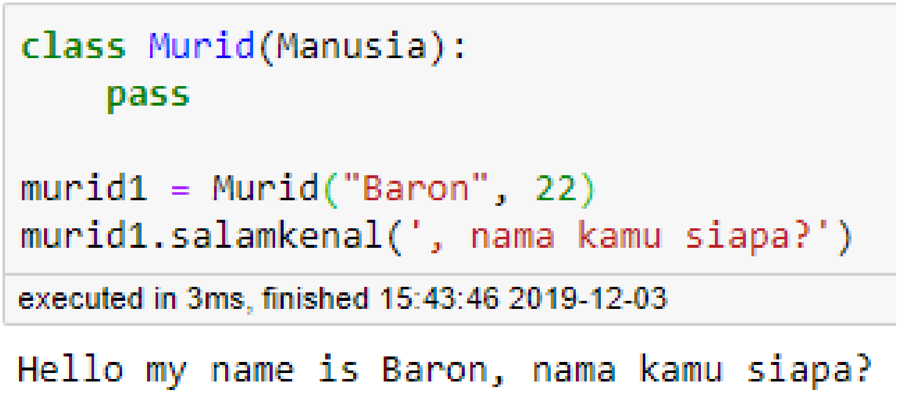
```
class Manusia :
    def __init__(self,name,age) :
        self.nama = name
        self.umur = age

    def salamkenal(self, kalimatlanjut):
        print("Hello my name is " + self.nama + kalimatlanjut)

person1 = Manusia("Baron", 22)
person1.salamkenal(', nama kamu siapa?')
```

executed in 7ms, finished 15:10:50 2019-12-03

```
Hello my name is Baron, nama kamu siapa?
```

Create a Child Class

To create a class that inherits the functionality from another class, send the parent class as a parameter when creating the child class:



```
class Murid(Manusia):
    pass

murid1 = Murid("Baron", 22)
murid1.salamkenal(', nama kamu siapa?')
```

executed in 3ms, finished 15:43:46 2019-12-03

```
Hello my name is Baron, nama kamu siapa?
```

Note: Use the pass keyword when you do not want to add any other properties or methods to the class.

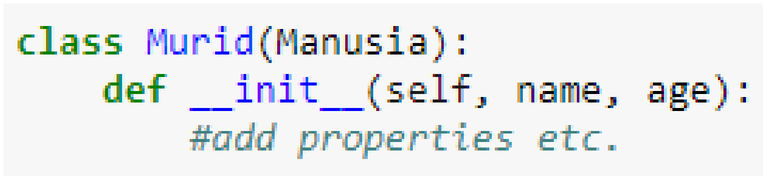
Now the Murid class has the same properties and methods as the Manusia class.

Add the __init__() Function

So far we have created a child class that inherits the properties and methods from its parent.

We want to add the __init__() function to the child class (instead of the pass keyword).

Note: The __init__() function is called automatically every time the class is being used to create a new object.

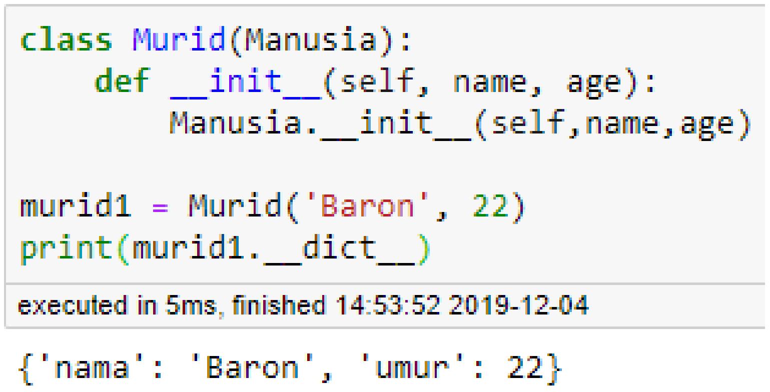


```python
class Murid(Manusia):
    def __init__(self, name, age):
        #add properties etc.
```

When you add the __init__() function, the child class will no longer inherit the parent's __init__() function.

Note: The child's __init__() function overrides the inheritance of the parent's __init__() function.

To keep the inheritance of the parent's __init__() function, add a call to the parent's __init__() function:



```
class Murid(Manusia):
    def __init__(self, name, age):
        Manusia.__init__(self,name,age)

murid1 = Murid('Baron', 22)
print(murid1.__dict__)
```

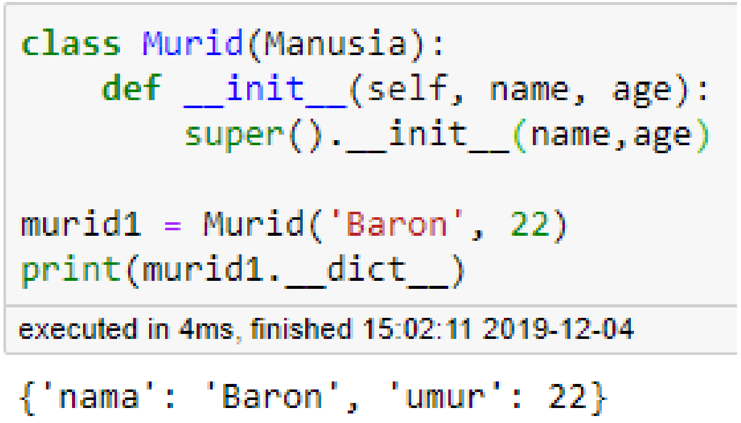
executed in 5ms, finished 14:53:52 2019-12-04

```
{'nama': 'Baron', 'umur': 22}
```

Now we have successfully added the __init__() function, and kept the inheritance of the parent class, and we are ready to add functionality in the __init__() function.

Use the super() Function

Python also has a super() function that will make the child class inherit all the methods and properties from its parent:



```
class Murid(Manusia):
    def __init__(self, name, age):
        super().__init__(name,age)

murid1 = Murid('Baron', 22)
print(murid1.__dict__)
```

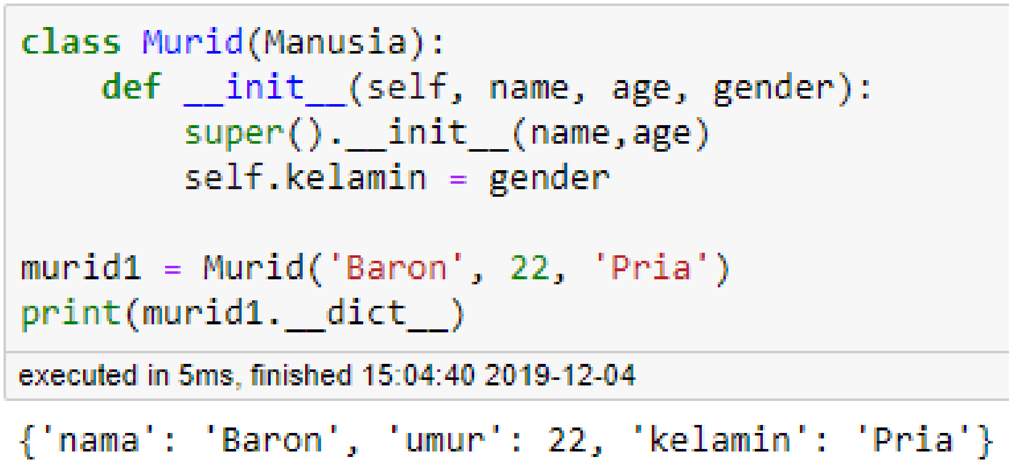
executed in 4ms, finished 15:02:11 2019-12-04

```
{'nama': 'Baron', 'umur': 22}
```

By using the super() function, you do not have to use the name of the parent element, it will automatically inherit the methods and properties from its parent.

Add Properties

we add new property (kelamin) to the Murid class, and add a new parameter to the __init__ function (gender) to initialize the kelamin value :

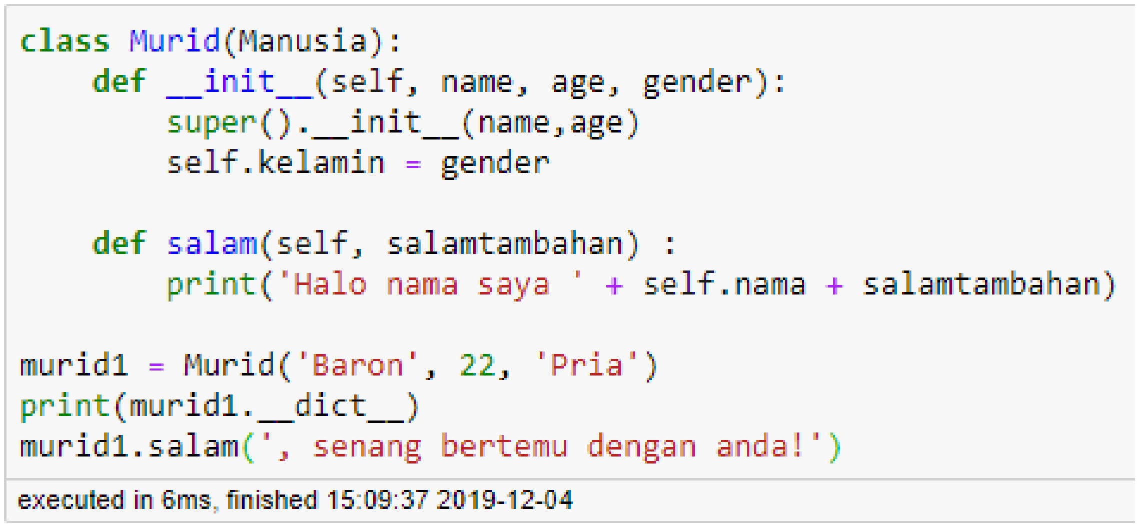


```
class Murid(Manusia):
    def __init__(self, name, age, gender):
        super().__init__(name,age)
        self.kelamin = gender

murid1 = Murid('Baron', 22, 'Pria')
print(murid1.__dict__)
```

executed in 5ms, finished 15:04:40 2019-12-04

```
{'nama': 'Baron', 'umur': 22, 'kelamin': 'Pria'}
```

Add Methods



```
class Murid(Manusia):
    def __init__(self, name, age, gender):
        super().__init__(name,age)
        self.kelamin = gender

    def salam(self, salamtambahan) :
        print('Halo nama saya ' + self.nama + salamtambahan)

murid1 = Murid('Baron', 22, 'Pria')
print(murid1.__dict__)
murid1.salam(', senang bertemu dengan anda!')
```

executed in 6ms, finished 15:09:37 2019-12-04

```
{'nama': 'Baron', 'umur': 22, 'kelamin': 'Pria'}
Halo nama saya Baron, senang bertemu dengan anda!
```

If you add a method in the child class with the same name as a function in the parent class, the inheritance of the parent method will be overridden.

Purwadhika
Startup and Coding School