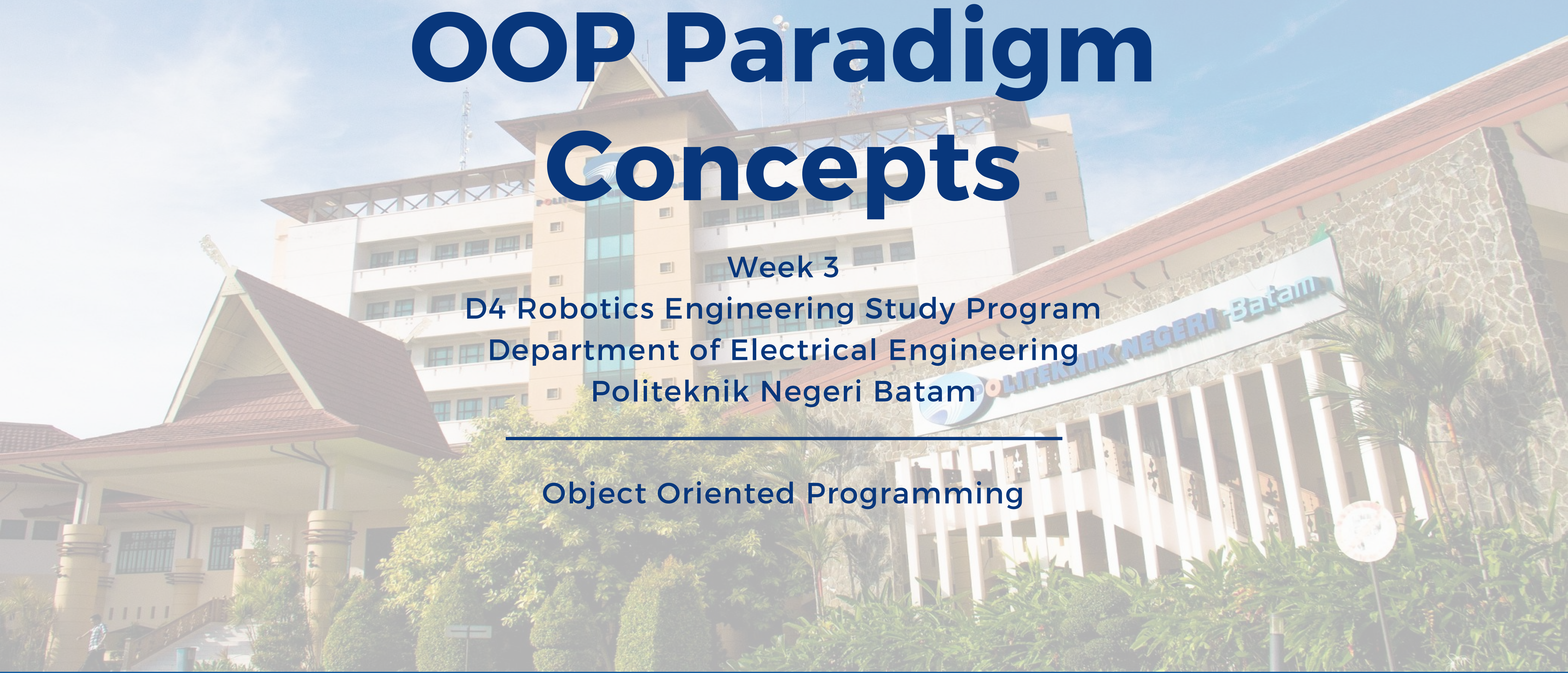
Politeknik
Negeri
Batam

OOP Paradigm Concepts

Week 3

D4 Robotics Engineering Study Program
Department of Electrical Engineering
Politeknik Negeri Batam

Object Oriented Programming



Course Learning Outcome

Students are able to know about object oriented programming concepts

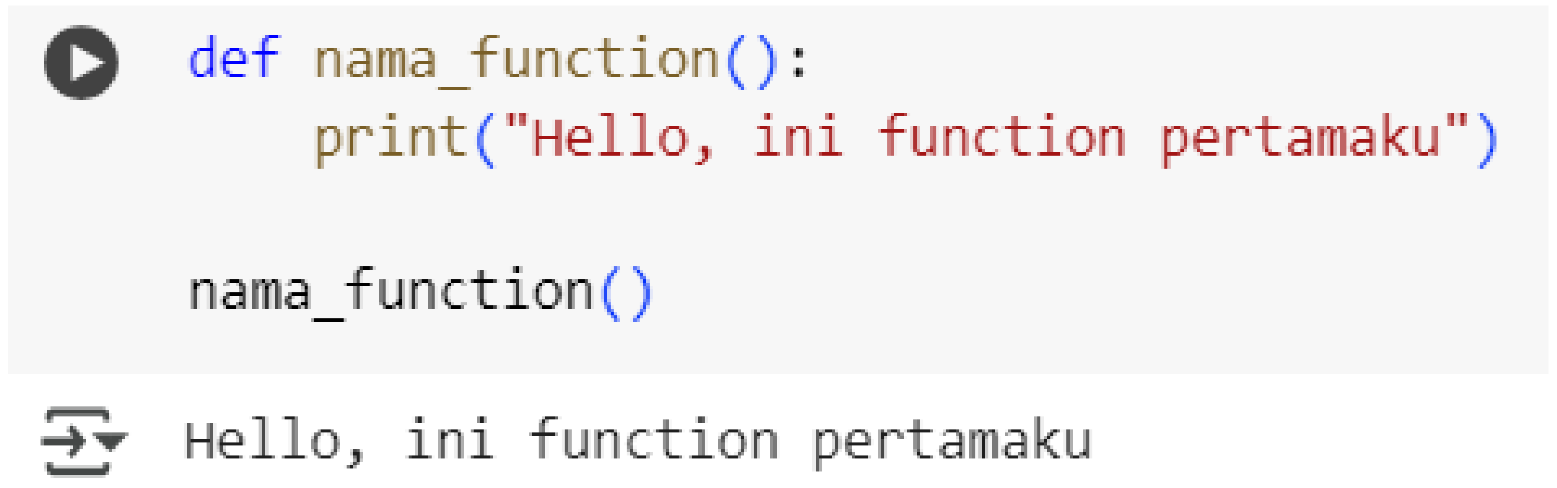
Table of Contents

01 Python Function
02 What is OOP?
03 Advantages of OOP
04 Programming Terms in OOP
05 Constructor

Python Function

A function is a block of code which only runs when it is called. Functions are defined using the def keyword, followed by the function name, parentheses (), and the block of code that is executed when the function is called.

How to Create a Function in Python:

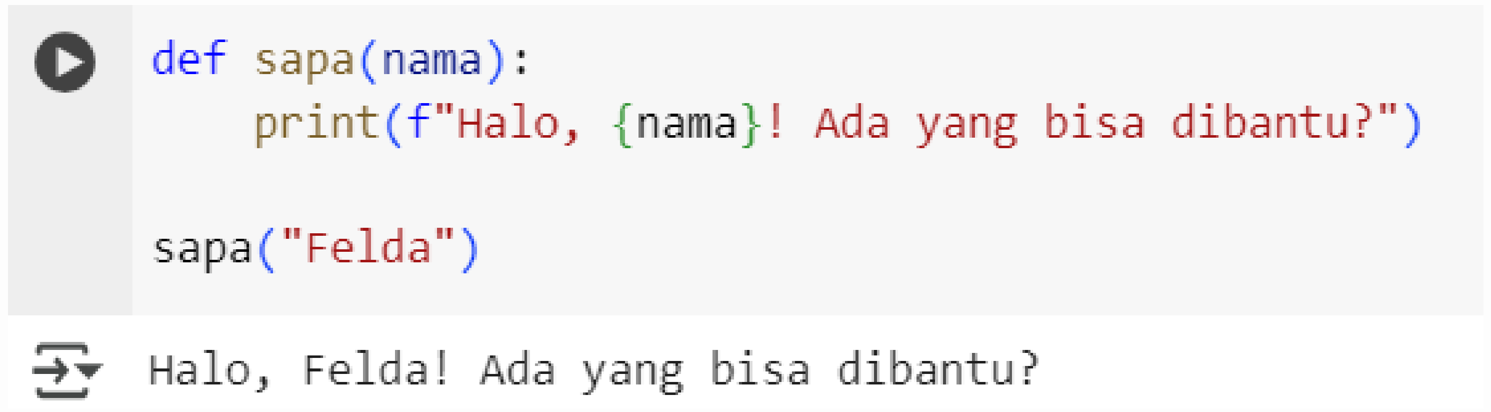


```
def nama_function():
    print("Hello, ini function pertamaku")

nama_function()
```

```
Hello, ini function pertamaku
```

Functions with Parameter

Parameters are specified after the function name, in parentheses ‘(..)’. We can add as many commands as we want, just separate them with commas.

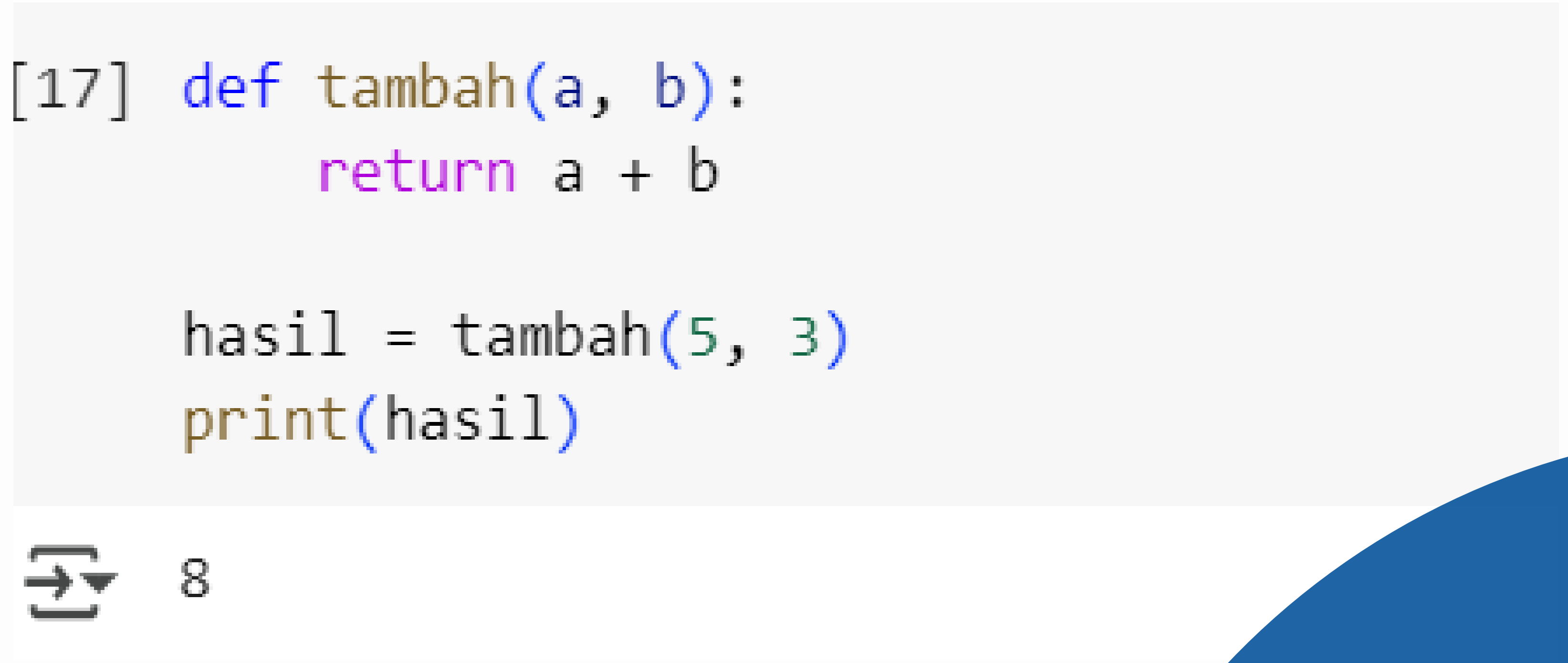


```
def sapa(nama):
    print(f"Halo, {nama}! Ada yang bisa dibantu?")

sapa("Felda")
```

```
Halo, Felda! Ada yang bisa dibantu?
```

Functions with Return Value

Functions can return values with return



```
[17] def tambah(a, b):
         return a + b

     hasil = tambah(5, 3)
     print(hasil)
```

```
8
```

What’s OOP?

These objects interact with each other to perform tasks and represent real-world entities, making OOP particularly intuitive for modeling complex systems.

Languages that Support OOP

Ruby
PROGRAMMING
Language
Java
python
{AS}3
C++
C#
PHP 5
php

Programming Terms in OOP

Class

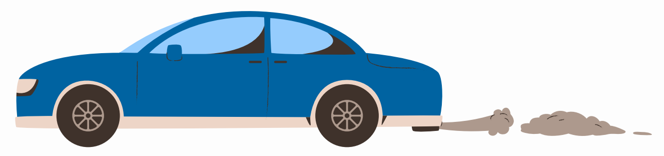
Blueprint or template for creating object

Object

are instances of a class

Attribute

represent the state of an object



Method

representing behavior

OOP Analogy

Attribute

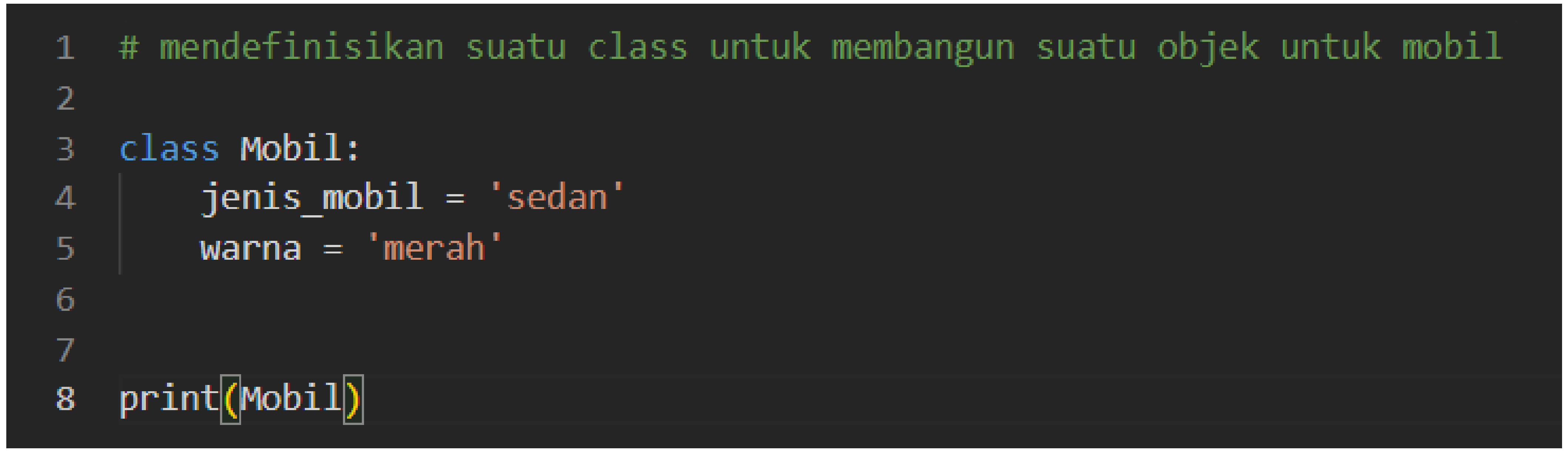
- Car color
- Brand
- Steering wheel
- Pedal
- Brake
- Car type

- Walk fast
- Go slow
- Back off
- Braking
- Honking

Behavior

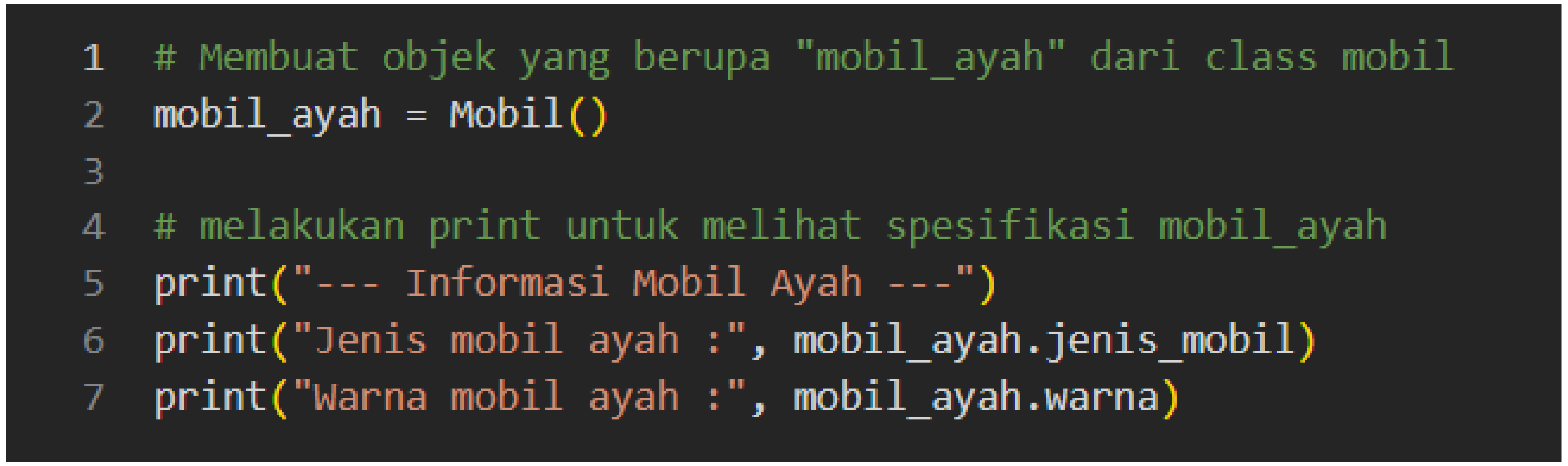
Class and Object initiation

Class is a *blueprint* or template for creating object.



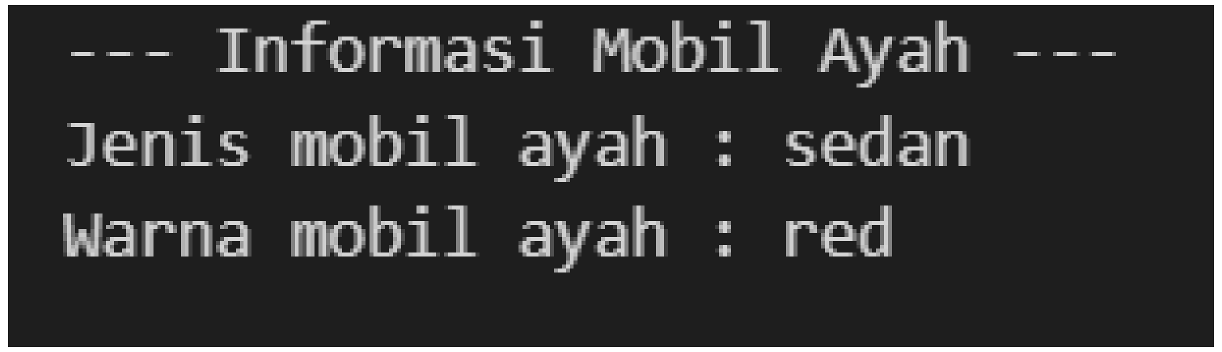
```python
# mendefinisikan suatu class untuk membangun suatu objek untuk mobil

class Mobil:
    jenis_mobil = 'sedan'
    warna = 'merah'


print(Mobil)
```

Meanwhile, to initialize an object, we can assign a class that has been defined.



```python
# Membuat objek yang berupa "mobil_ayah" dari class mobil
mobil_ayah = Mobil()

# melakukan print untuk melihat spesifikasi mobil_ayah
print("--- Informasi Mobil Ayah ---")
print("Jenis mobil ayah :", mobil_ayah.jenis_mobil)
print("Warna mobil ayah :", mobil_ayah.warna)
```

Output :

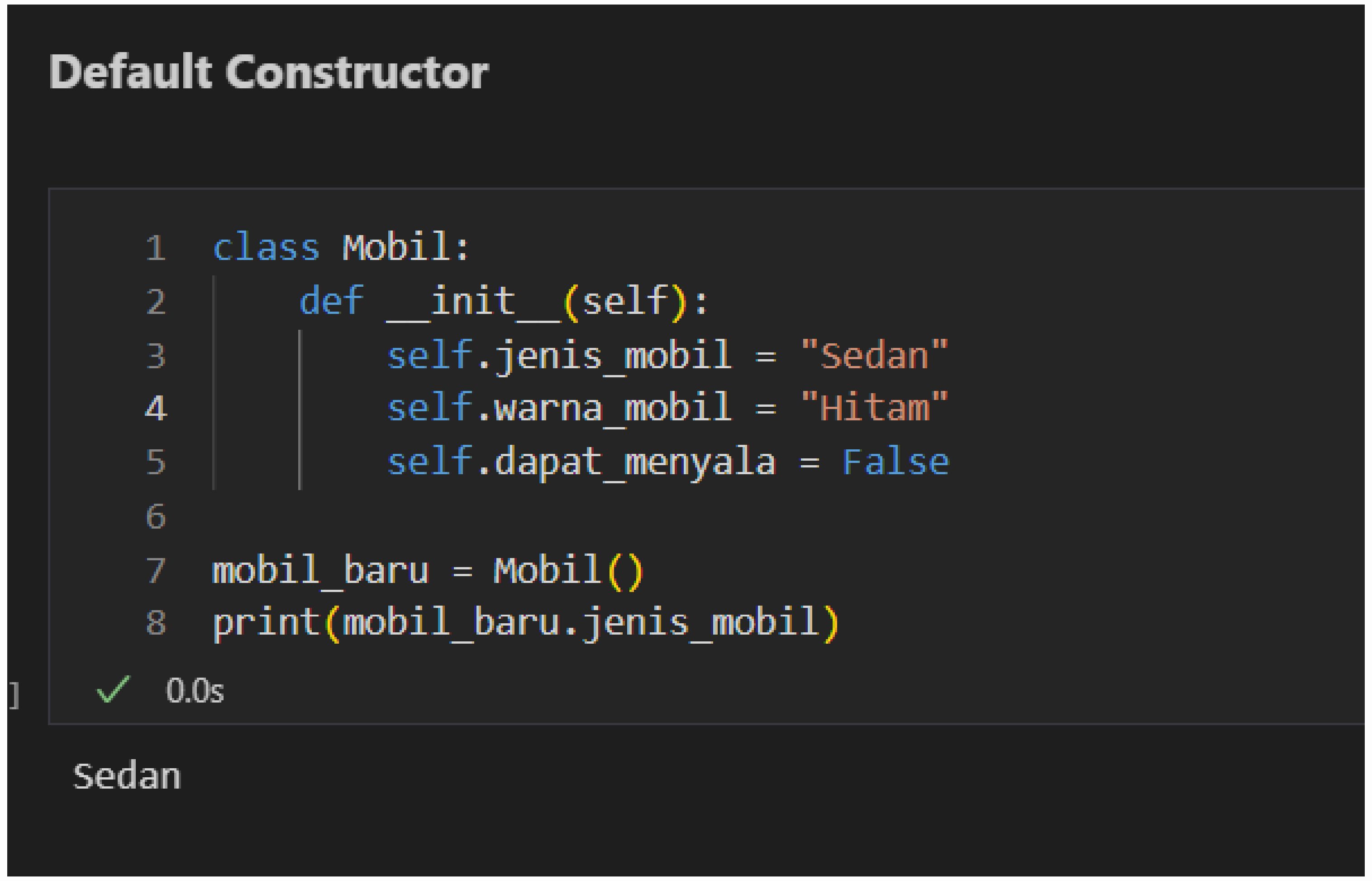


```
--- Informasi Mobil Ayah ---
Jenis mobil ayah : sedan
Warna mobil ayah : red
```

Constructor

A constructor is a special method that can initialize an object when it is created. In python, constructors are created with the __init__() function. The `__init__` function is used **to initialize a class that will create an object**. Types of Constructors in Python:

1



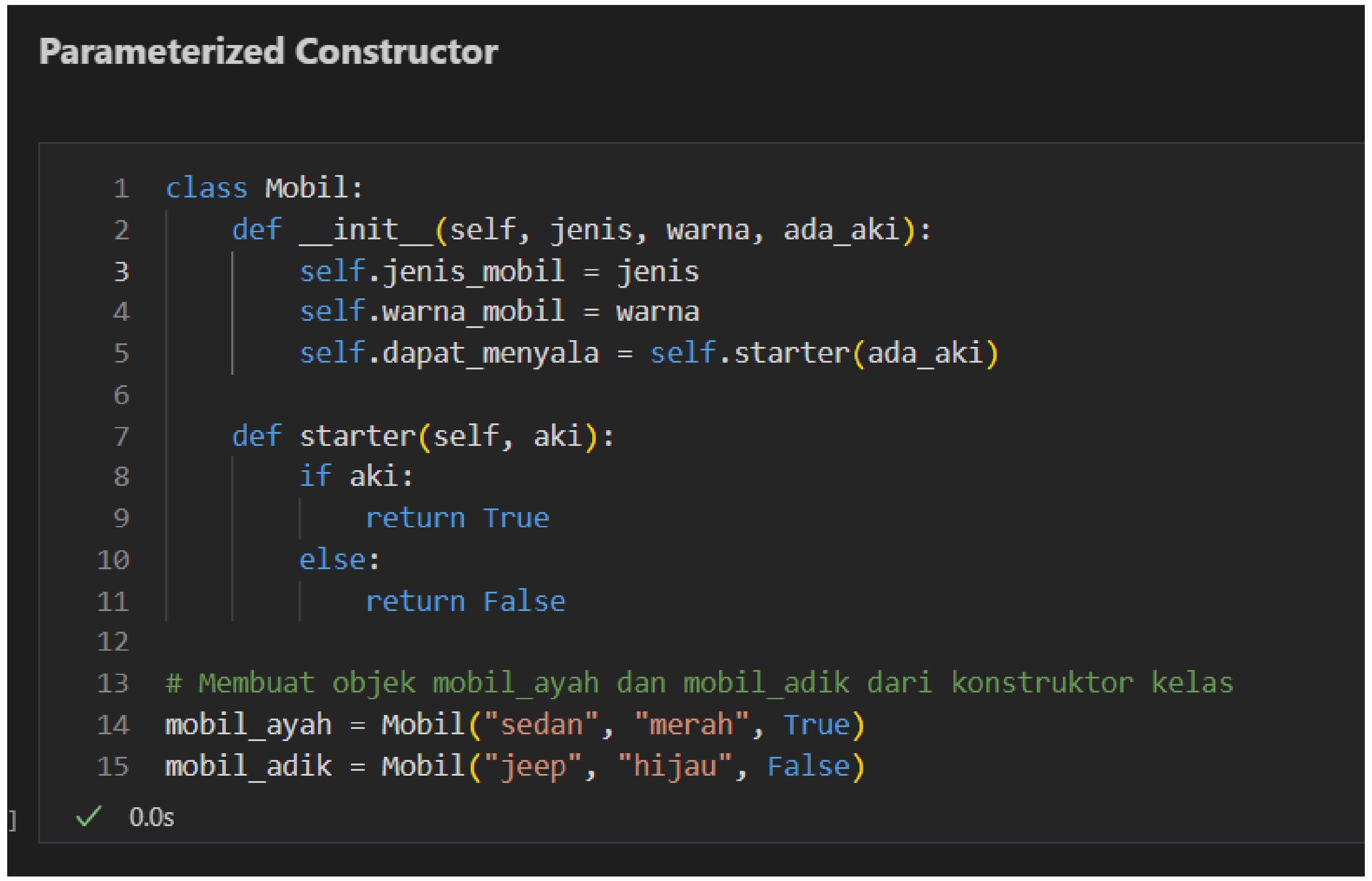
Default Constructor

```
class Mobil:
    def __init__(self):
        self.jenis_mobil = "Sedan"
        self.warna_mobil = "Hitam"
        self.dapat_menyala = False

mobil_baru = Mobil()
print(mobil_baru.jenis_mobil)
```

✓ 0.0s

```
Sedan
```

2



Parameterized Constructor

```
class Mobil:
    def __init__(self, jenis, warna, ada_aki):
        self.jenis_mobil = jenis
        self.warna_mobil = warna
        self.dapat_menyala = self.starter(ada_aki)

    def starter(self, aki):
        if aki:
            return True
        else:
            return False

# Membuat objek mobil_ayah dan mobil_adik dari konstruktor kelas
mobil_ayah = Mobil("sedan", "merah", True)
mobil_adik = Mobil("jeep", "hijau", False)
```

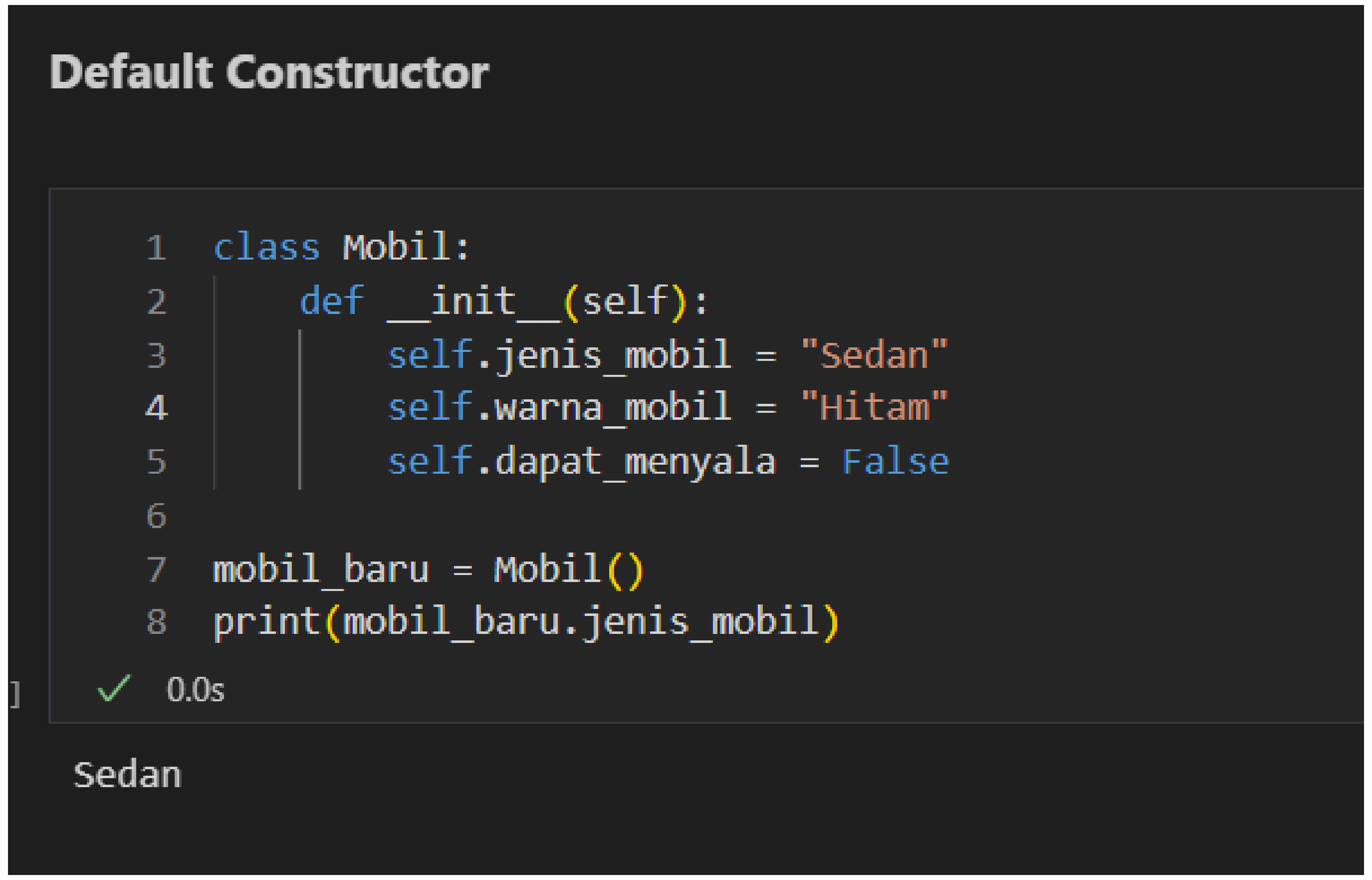
✓ 0.0s

Constructor

Types of Constructors in Python

☑ Default Constructor : Has no additional parameters other than self.

☑ Parameterized Constructor : Has additional parameters to initialize object attributes.

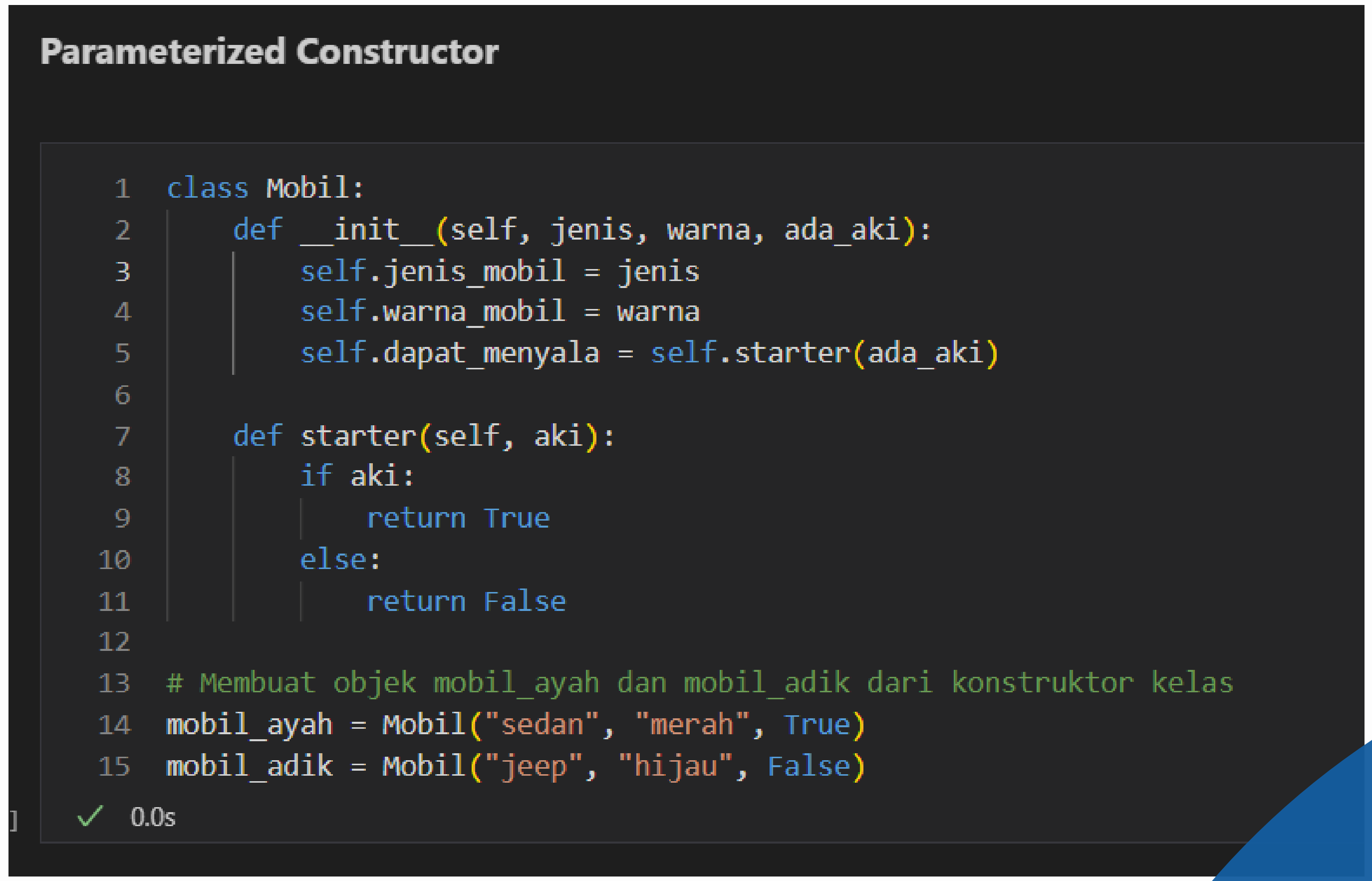


Default Constructor

```python
class Mobil:
    def __init__(self):
        self.jenis_mobil = "Sedan"
        self.warna_mobil = "Hitam"
        self.dapat_menyala = False

mobil_baru = Mobil()
print(mobil_baru.jenis_mobil)
```

✓ 0.0s

```
Sedan
```



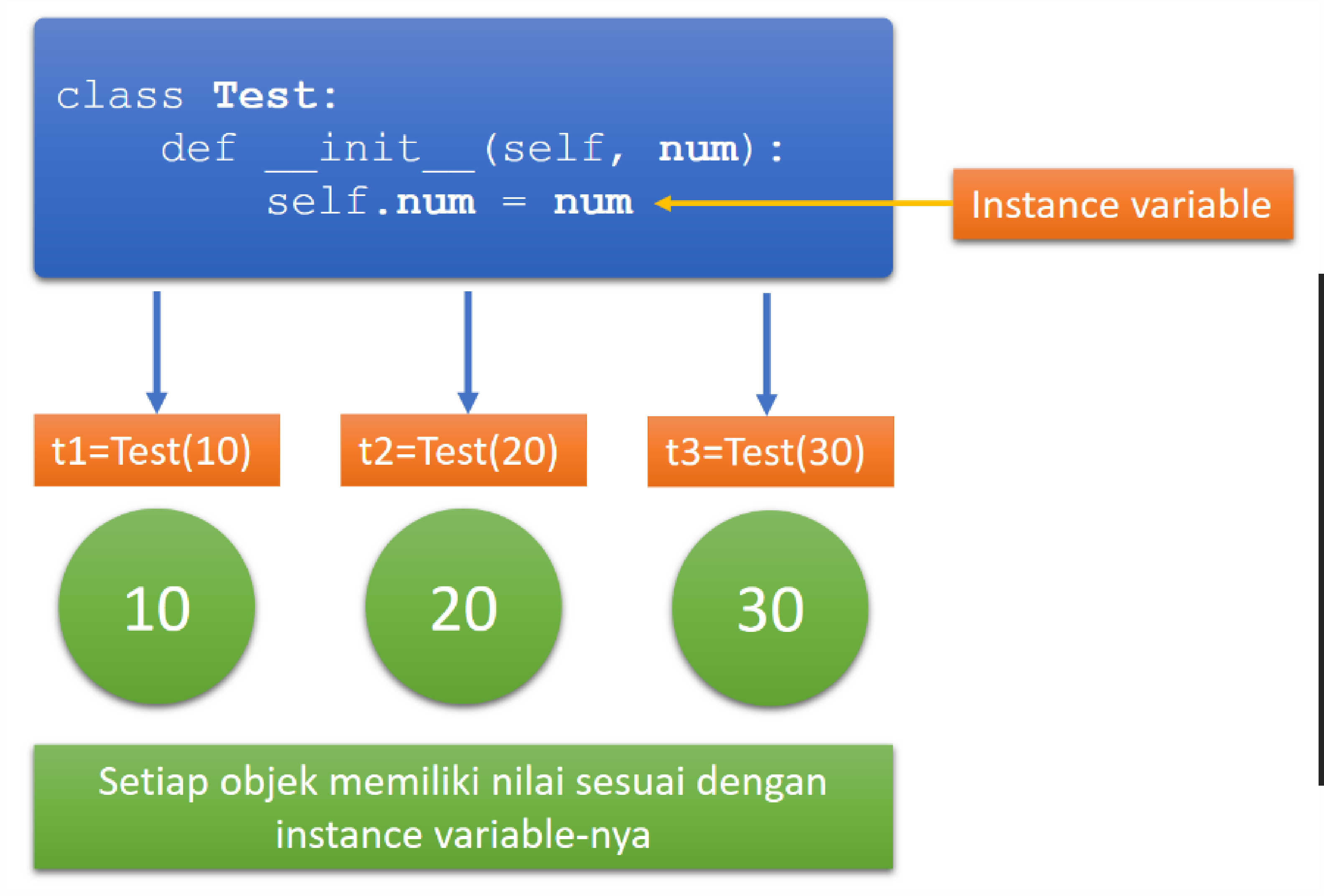
Parameterized Constructor

```python
class Mobil:
    def __init__(self, jenis, warna, ada_aki):
        self.jenis_mobil = jenis
        self.warna_mobil = warna
        self.dapat_menyala = self.starter(ada_aki)

    def starter(self, aki):
        if aki:
            return True
        else:
            return False

# Membuat objek mobil_ayah dan mobil_adik dari konstruktor kelas
mobil_ayah = Mobil("sedan", "merah", True)
mobil_adik = Mobil("jeep", "hijau", False)
```

✓ 0.0s

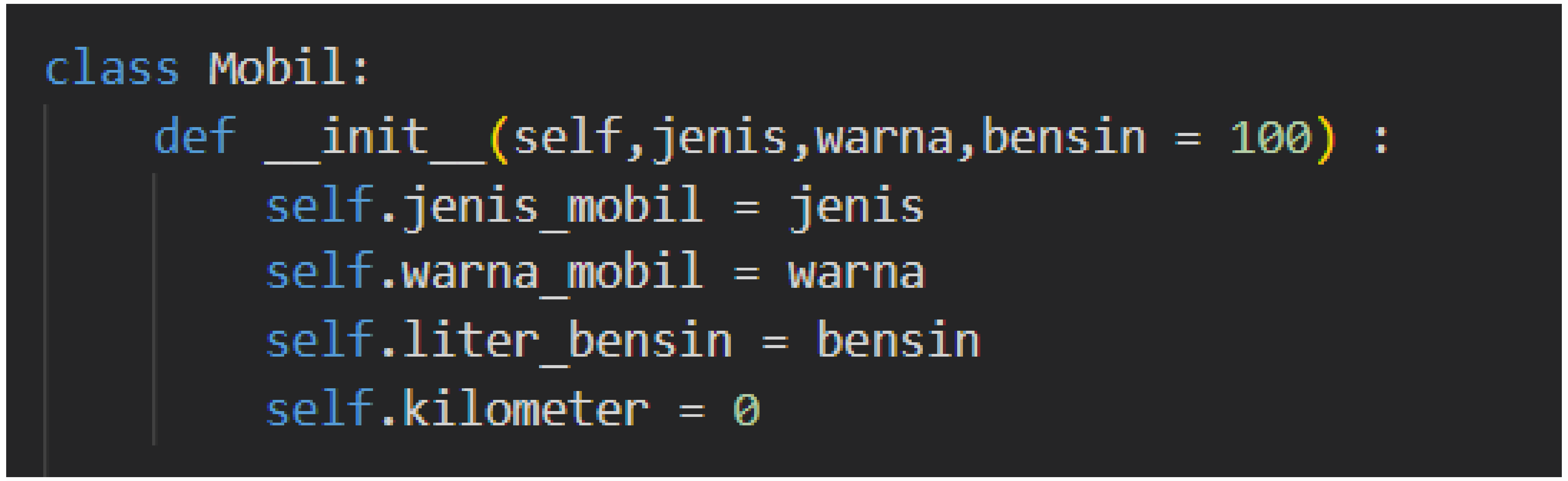
Instance Variable

Instance variables belong to objects that are instances of a particular class. This means that each object of that class can have a different instance variable value.



```
class Test:
    def __init__(self, num):
        self.num = num
```

Instance variable
t1=Test(10)
t2=Test(20)
t3=Test(30)
10
20
30
Setiap objek memiliki nilai sesuai dengan instance variable-nya



```
class Mobil:
    def __init__(self,jenis,warna,bensin = 100) :
        self.jenis_mobil = jenis
        self.warna_mobil = warna
        self.liter_bensin = bensin
        self.kilometer = 0
```

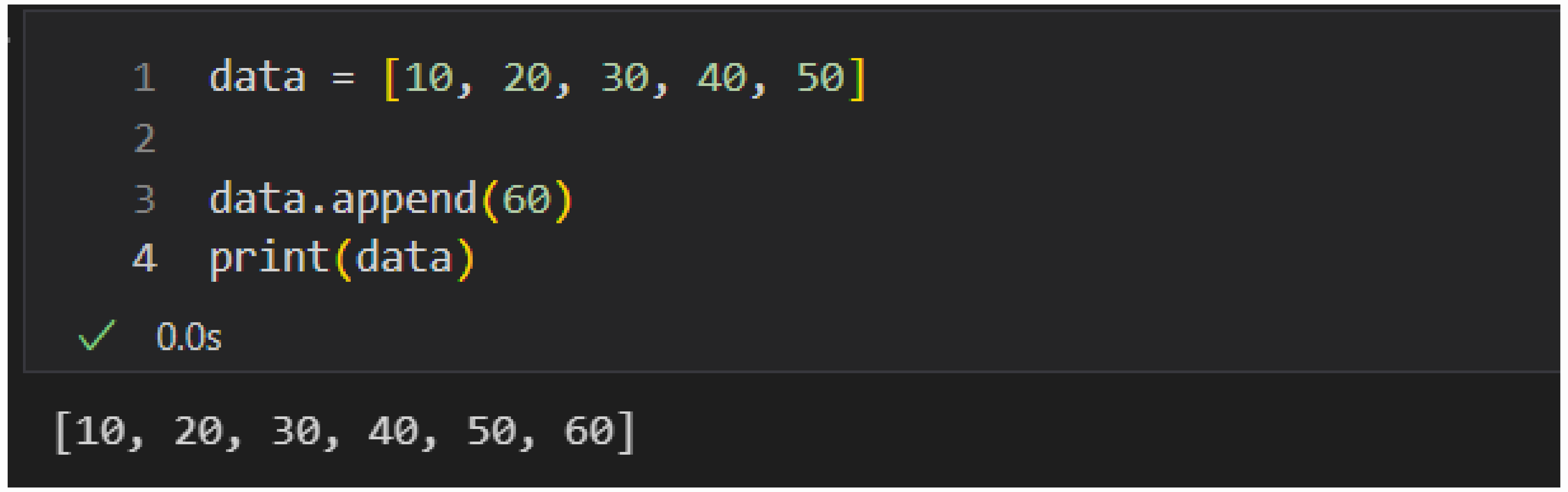
Method

Methods in python are functions that relate to objects of a class. These methods are called within or together with objects.

<u>Syntax for calling a method:</u>

Object.method_name()

To call a method, after the object, <u>add a dot (.) followed by the method name and end with the call operator ( ).</u>



```
data = [10, 20, 30, 40, 50]

data.append(60)
print(data)
```

```
✓ 0.0s
```

```
[10, 20, 30, 40, 50, 60]
```

Instance Method

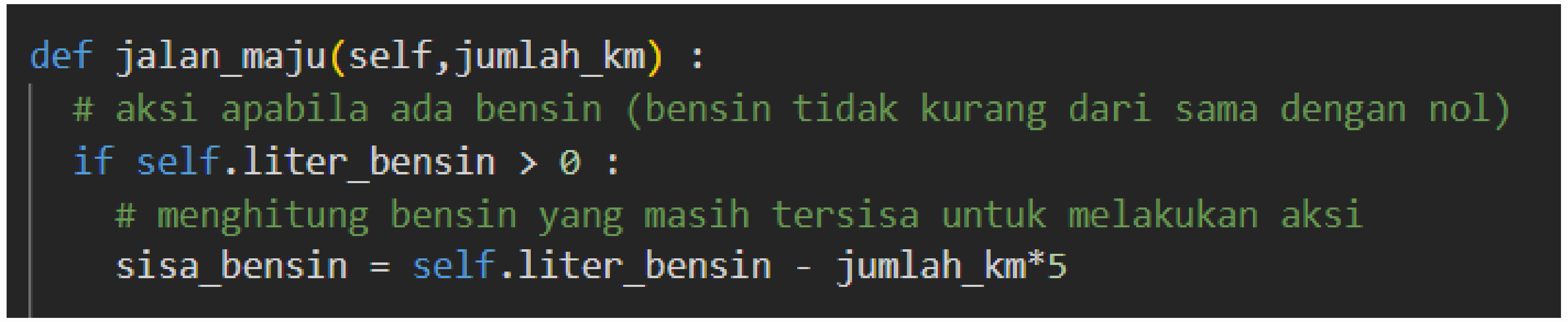
Instance methods are functions defined in a class and can only be called by an object.

📌 Characteristics:

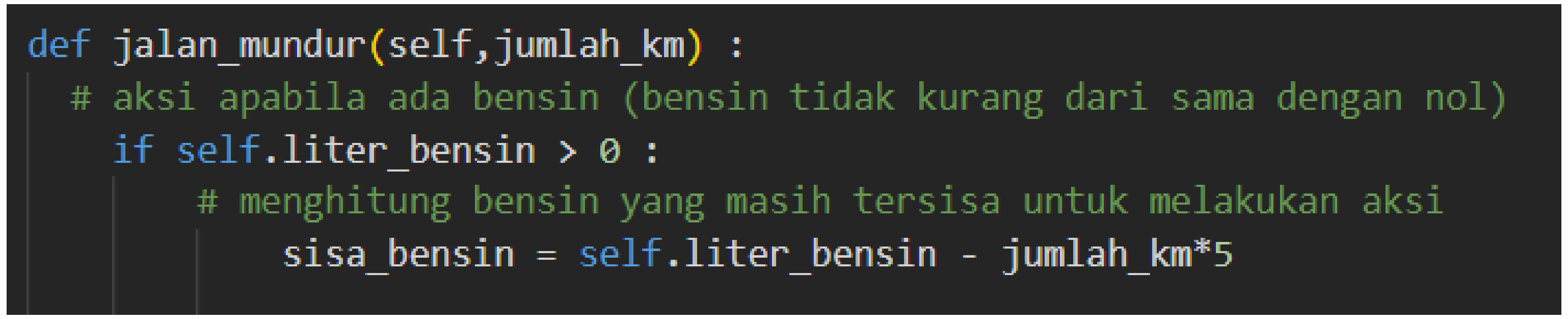
✓ Declared with self as the first parameter.

✓ Can only be accessed by objects (not classes directly).

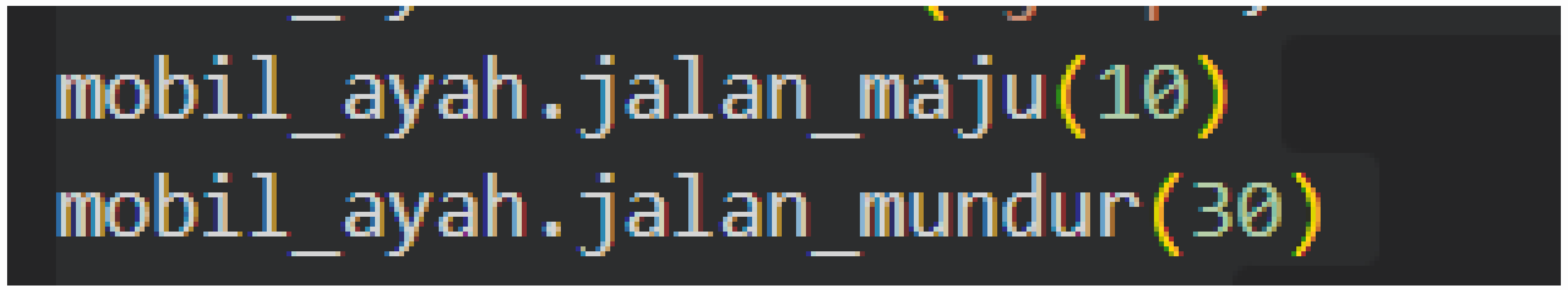
✓ Can access instance variables (self.variable_name).



```
def jalan_maju(self,jumlah_km) :
    # aksi apabila ada bensin (bensin tidak kurang dari sama dengan nol)
    if self.liter_bensin > 0 :
        # menghitung bensin yang masih tersisa untuk melakukan aksi
        sisa_bensin = self.liter_bensin - jumlah_km*5
```



```
def jalan_mundur(self,jumlah_km) :
    # aksi apabila ada bensin (bensin tidak kurang dari sama dengan nol)
        if self.liter_bensin > 0 :
            # menghitung bensin yang masih tersisa untuk melakukan aksi
                sisa_bensin = self.liter_bensin - jumlah_km*5
```



```
mobil_ayah.jalan_maju(10)
mobil_ayah.jalan_mundur(30)
```

jalan_maju() is an instance method because it can only be called from an object (mobil_ayah.jalan_maju(10)).

Next Topic: Main Principles of object oriented programming

Reference

1. Mark Lutz - Learning Python_ Powerful Object-Oriented Programming-O'Reilly Media (2009)

2.Mv Annela - Understanding Object-Oriented Programming (OOP): A Comprehensive Guide

polibatam

THANK YOU!