

AI Mastery Course


Module 1

Introduction to everything

Section

 OOPs and Common Packages

Learning Objectives

- Di akhir modul ini, kita akan dapat:
 - Memahami object-oriented programming dan inheritance di Python
 - Membuat custom modules dan packages di Python
 - Menggunakan built-in modules seperti os, time dan math
 - Memahami exception handling di Python
- 



Agenda

01

OOPS

- Object oriented programming
- OOPs terminology
- Inheritance di Python

02

MODULES I

- Python custom modules
- Python built in modules

03

MODULES II

- os
- time
- math

04

EXCEPTIONS

- Exception Handling
- Custom Exception

05

KESIMPULAN

- Kuis
- Ringkasan



01

OOPS

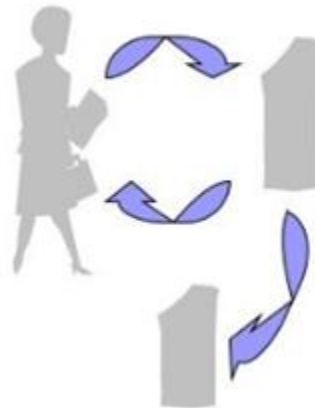
- Object oriented programming
- OOPs terminology
- Inheritance di Python

Object oriented programming

Object-Oriented Programming (OOP) adalah suatu metode pemrograman yang berorientasi objek.

Sederhananya, OOP adalah paradigma pemrograman yang **merepresentasikan objek** seperti objek di sekitar kita dalam bentuk kode program.

■ Procedural



Withdraw, deposit, transfer

■ Object Oriented



Customer, money, account

OOPs : class

Class merupakan blueprint atau prototipe yang digunakan untuk membuat objek. Biasanya, class terdiri dari beberapa atribut dan method.

Classes dibuat dengan keyword class.

Attributes adalah variables yang dimiliki sebuah class. Attributes selalu bersifat public dan bisa diakses menggunakan operator dot (.).

Method adalah fungsi yang melekat pada objek.

Syntax

```
class ClassName:  
  
    # Statement-1.  
  
    # Statement-N
```

OOPs terminology

Object	Instance	Method	Constructor method	Class variable	Instance variable
<ul style="list-style-type: none">• Instance unik dari struktur data yang ditentukan oleh kelasnya• terdiri dari anggota data (variabel kelas & variabel instan) & metode	<ul style="list-style-type: none">• Objek individu dari kelas tertentu	<ul style="list-style-type: none">• Jenis fungsi khusus yang didefinisikan di dalam kelas	<ul style="list-style-type: none">• Init() adalah keyword untuk membuat constructor• Method yang selalu dijalankan ketika kelas dibuat	<ul style="list-style-type: none">• Variabel yang dibagikan oleh semua instance kelas• didefinisikan di dalam kelas tetapi di luar method kelas manapun	<ul style="list-style-type: none">• Variabel yang didefinisikan di dalam method dan hanya dimiliki oleh instance kelas saat ini. Biasanya diakses menggunakan keyword self

Python class and objects

keyword class yang digunakan untuk
membuat sebuah class

Nama Class

```
1 class course:
2     # defining the class attributes
3     name = ""
4     duration = 0
```

Definisi
Class

Membuat
object dari
class yang
ada

```
1 c1 = course() # creatig an object of type class course
2 c1.name = "Python programming" # modifying the attribute
3 c1.duration = 25
4 print(c1.name)
5 print(c1.duration)
```

```
Python programming
25
```


Python class and objects

keyword class yang digunakan untuk membuat sebuah class

```
1 class Mobil:
2     # class attribute
3     Roda = 4
4     # initializer / instance attributes
5     def __init__(self, warna, merek):
6         self.warna = warna
7         self.merek = merek
8     # method 1
9     def showDescription(self):
10        print("Mobil ini warna ", self.warna, " merek ", self.merek)
11    # method 2
12    def changeColor(self, warna):
13        self.warna = warna
14 c = Mobil('Hitam', 'Tesla')
15 # call method 1
16 c.showDescription()
17 # Cetak Mobil ini warna Hitam merek Tesla
18 # call method 2 and set color
19 c.changeColor('Putih')
20 c.showDescription()
21 # Cetak Mobil ini warna Putih merek Tesla
```

Definisi Class

Membuat object dari
class yang ada

```
Mobil ini warna Hitam merek Tesla
Mobil ini warna Putih merek Tesla
```

OOPs terminology : self keyword

01

"Self" mewakili instance dari kelas.

02

Method dari sebuah Class harus memiliki parameter tambahan pertama dalam definisinya. Kita tidak harus memberikan nilai untuk parameter ini.

03

Dengan menggunakan keyword "self" kita dapat mengakses atribut dan metode Class dengan python

04

"self" mirip dengan pointer di C++ dan reference di Java.

Methods inside the class

Attribut Class

Method yang
didefinisikan di dalam
Class dengan keyword
self sebagai input

Mengakses attribute
dari Class,
Class attribute akan
dibagikan ke berbagai
instances dari Class
yang sama

```
1 class Employee:
2     "Common base class for all employees"
3     empCount = 0
4     def __init__(self, name, salary):
5         self.name = name
6         self.salary = salary
7         Employee.empCount += 1
8
9     def display_count(self):
10        print("Total Employee %d" % Employee.empCount)
11
12    def display_employee(self):
13        print("Name : ", self.name, ", Salary: ", self.salary)

```



```
1 # This would create first object of Employee class
2 emp1 = Employee("Johny", 9000)
3
4 #This would create second object of Employee class
5 emp2 = Employee("Shafi", 4000)
6
7 emp1.display_employee()
8 emp2.display_employee()
9 print("Total Employee %d" % Employee.empCount)

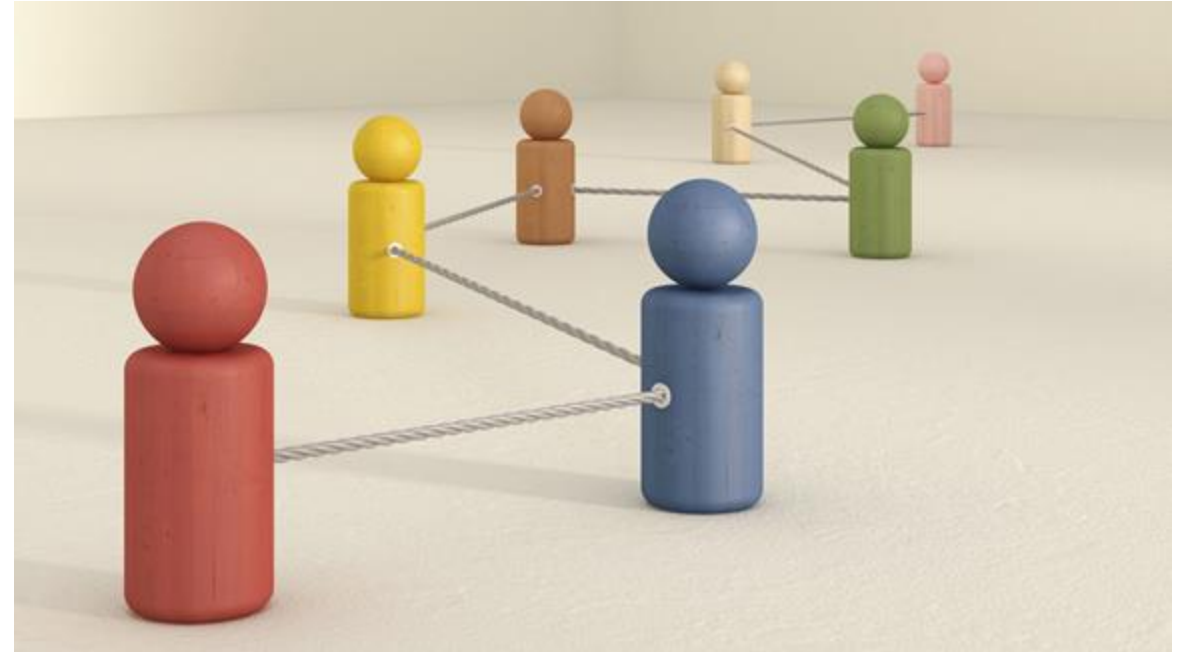
```

Name : Johny , Salary: 9000
Name : Shafi , Salary: 4000
Total Employee 2

Instance
attributes

Inheritance

- Inheritance adalah kemampuan satu kelas untuk memperoleh atau mewarisi properti dari kelas lain (yang sudah ada).
- Kelas yang baru terbentuk adalah kelas turunan (atau child class). Demikian pula kelas yang ada adalah kelas dasar (atau parent class).



Inheritance

Menggunakan inheritance
until kelas
Employee di
kelas
Manager

Child class / Derived
class

```
1 class Employee:
2     "Common base class for all employees"
3     empCount = 0
4     def __init__(self, name, salary):
5         self.name = name ; self.salary = salary
6         Employee.empCount += 1
7
8     def display_count(self):
9         print("Total Employee %d" % Employee.empCount)
10
11    def display_employee(self):
12        print("Name : ", self.name, ", Salary: ", self.salary)
13
14    class Manager(Employee):
15        manCount = 0
16        def __init__(self, name, salary, teamsize=0):
17            # invoking __init__() method of the parent class
18            super().__init__(name, salary)
19            self.teamsize = teamsize
20
21        def display_teamsiz(self):
22            print("Team size %d" % self.teamsize)
```

```
1 emp1 = Employee("Johny", 9000) # creating object of Employee class
2 mgr1 = Manager("Shafi", 12000) # creating object of Employee class
3
4 emp1.display_employee()
5 mgr1.display_employee()
6 mgr1.display_teamsiz()
7 print("Total Employee %d" % Employee.empCount)
```


```
Name : Johny , Salary: 9000
Name : Shafi , Salary: 12000
Team size 0
Total Employee 2
```

Mengakses
method dengan
nama yang sama
seperti class
parent nya



02

MODULES

- Python custom modules
 - Python built in modules
- 

Python modules



Python modules adalah file dengan ekstensi .py yang berisi kode python yang dapat diimpor ke dalam program python lain.



Modul Python berisi komponen berikut : _____→

- Function definitions
- Class implementations
- Variables



Untuk menggunakan modul python, kita dapat menggunakan keyword **import**

Python custom modules

Memuat
modul python
dalam kode
menggunakan
keyword
import

```
jupyter mycalculator.py a minute ago  
File Edit View Language  
1 def add(a,b):  
2     "this function can be used to add two numbers or concatenate two strings"  
3     return a+b  
4  
5 def mult(a,b):  
6     "this function can be used to multiply two numbers"  
7     return a*b  
  
1 import mycalculator  
1 mycalculator.add(4,5)  
9  
1 mycalculator.mult(7,2)  
14
```

.py file yang akan
digunakan modul
python

Harus ada di
direktori yang tepat

Mengakses
sebuah fungsi
dari module

Python built-in modules



Python memiliki banyak fungsi bawaan, selain itu banyak fungsi tersedia sebagai bagian dari perpustakaan yang dibundel dengan distribusi python, Dalam python ini dikenal sebagai built-in modules



Mayoritas modul python ditulis dalam bahasa C dan terintegrasi dengan python shell



Untuk melihat daftar semua modul yang tersedia, kita dapat menggunakan perintah `help('modules')`

Python built-in modules

Memanggil
built-in
module
dengan
keyword
import

```
1 import statistics
2
3 x = [4,5,4,2,6,9,4,7,6,5,6,4,3]
4 statistics.mean(x)
```

5

```
1 statistics.median(x)
```

5

```
1 statistics.mode(x)
```

4

```
1 statistics.stdev(x)
```

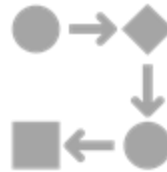
1.8257418583505538

Keuntungan dari modules



Reusability

Menggunakan modul python, meningkatkan tingkat penggunaan ulang kode dalam python, fungsi/implementasi kelas yang sama dapat diimpor dalam banyak kode.



Simplicity

Modul berfokus pada sebagian kecil dari masalah, daripada berfokus pada keseluruhan masalah.



Scoping

Namespace terpisah ditentukan oleh modul yang membantu menghindari tabrakan antara pengidentifikasi.



03

MODULES

- os
- time
- math

Python module: os

```
1 import os
2 print("Number of cores in Processor - ",os.cpu_count())
```

Number of cores in Processor - 8

```
1 os.getcwd() # accessing current working directory
```

'D:\\python'

```
1 os.path.isdir("mycodes") # checking whether directory exists or nor
```

False

```
1 os.mkdir("mycodes") # creating a new directory
```

```
1 os.path.isdir("mycodes") # checking whether directory exists or nor
```

True

```
1 os.path.getctime("mycodes")
```

1640915376.899972

Python module: time

```
1 import time
2
3 # seconds passed since epoch
4 local_time = time.ctime()
5 print("Local time:", local_time)
```

Local time: Thu Dec 30 14:54:33 2021

```
1 print("This is printed immediately.")
2 time.sleep(2.4)
3 print("This is printed after 2.4 seconds.")
```

This is printed immediately.
This is printed after 2.4 seconds.

Python module: math

```
1 import math    # Import math Library
2 print (math.pi)  # Print the value of pi
```

3.141592653589793

```
1 a = 2.3
2 print ("The ceil of 2.3 is : ", end="")
3 print (math.ceil(a)) # returning the ceil of 2.3
4
5 print ("The floor of 2.3 is : ", end="")
6 print (math.floor(a)) # returning the floor of 2.3
7
8 a = 5
9 print("The factorial of 5 is : ", end="")
10 print(math.factorial(a)) # returning the factorial of 5
```

The ceil of 2.3 is : 3
The floor of 2.3 is : 2
The factorial of 5 is : 120



04

EXCEPTIONS

- Exception Handling
- Custom Exception

Python syntax errors v/s python exceptions

Syntax error
Tidak
menggunakan titik
dua (:) setelah
kondisi tidak valid
di Python

```
1 # initialize the age variable
2 age = 18
3
4 # checking if you are eligible to cast your vote or not
5 if (age < 18)
6     print("You are NOT eligible to cast your vote.")

File "<ipython-input-42-85eadb0eae2a>", line 5
    if (age < 18)
        ^
SyntaxError: invalid syntax
```

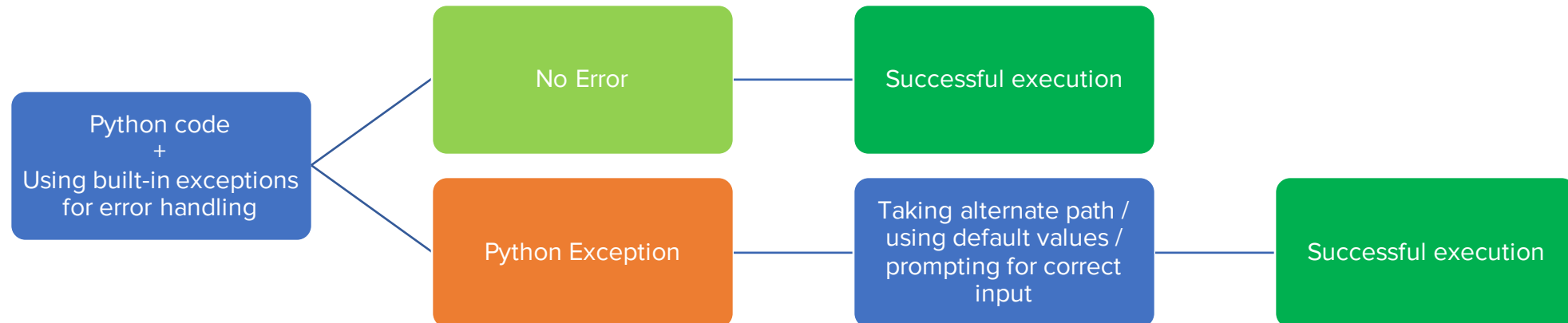
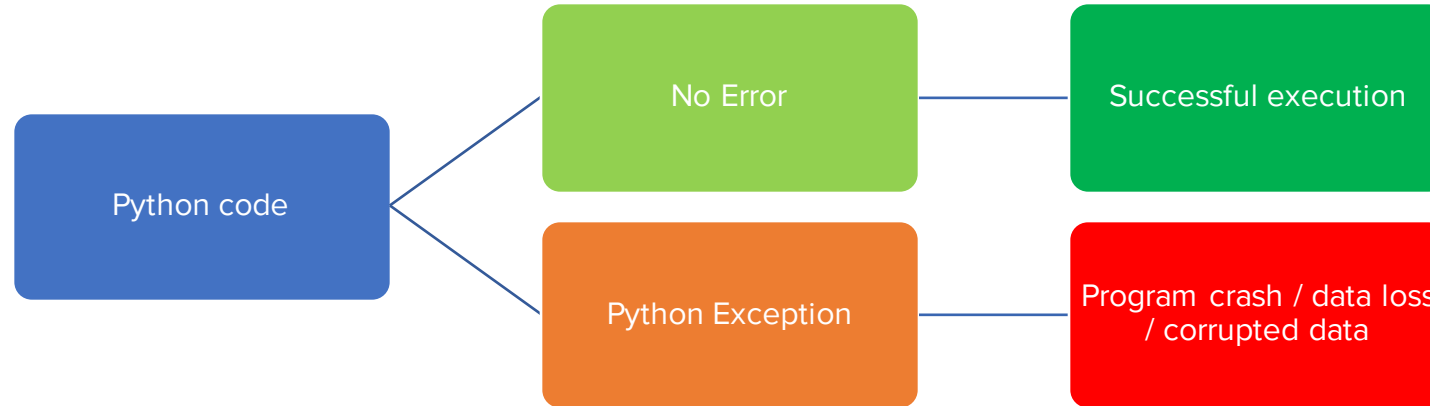
```
1 # initialize the age variable
2 age = 18
3
4 # dividng age by 0
5 out = age/0

-----
ZeroDivisionError                                Traceback (most recent call last)
<ipython-input-43-035f83291406> in <module>
      3
      4 # dividng age by 0
----> 5 out = age/0

ZeroDivisionError: division by zero
```

Membagi
dengan nol
akan memicu
python
exception

Exception handling pada Python



Custom exception in python

Raise dapat
digunakan untuk
memunculkan
python exception

```
1 from datetime import datetime
2 # accessing the current date
3 current_date = datetime.now()
4 print ("Current date is: " + current_date.strftime('%Y-%m-%d'))
5
6 dateinput = input("Enter your birth date in yyyy-mm-dd format: ")
7 date_provided = datetime.strptime(dateinput, '%Y-%m-%d')
8 print ("Date provided is: " + date_provided.strftime('%Y-%m-%d'))
9
10 if (date_provided.date() > current_date.date()):
11     raise Exception("Birth date can't be higher than today")
```

```
Current date is: 2021-12-30
Enter your birth date in yyyy-mm-dd format: 2022-01-05
Date provided is: 2022-01-05
```

```
-----
Exception                                 Traceback (most recent call last)
<ipython-input-44-52cf0bf6fef0> in <module>
      9
     10 if (date_provided.date() > current_date.date()):
--> 11     raise Exception("Birth date can't be higher than today")

Exception: Birth date can't be higher than today
```

```
1 if (date_provided.date() > current_date.date()):
2     raise ValueError("Birth date can't be higher than today")
```

```
-----
ValueError                                 Traceback (most recent call last)
<ipython-input-45-746d45a5943c> in <module>
      1 if (date_provided.date() > current_date.date()):
--> 2     raise ValueError("Birth date can't be higher than today")

ValueError: Birth date can't be higher than today
```

Kita juga
dapat
menentukan
jenis
exception
yang sesuai

Assertion Error Exception

Keyword
assert dapat
digunakan
untuk
melempar
kesalahan
pernyataan

```
1 from datetime import datetime
2 # accessing the current date
3 current_date = datetime.now()
4 print ("Current date is: " + current_date.strftime('%Y-%m-%d'))
5
6 dateinput = input("Enter your birth date in yyyy-mm-dd format: ")
7 date_provided = datetime.strptime(dateinput, '%Y-%m-%d')
8 print ("Date provided is: " + date_provided.strftime('%Y-%m-%d'))
9
10 assert date_provided.date() < current_date.date(), "Birth date can't be higher than today"
11 print("Your birthdate is accepted")
```

```
Current date is: 2021-12-30
Enter your birth date in yyyy-mm-dd format: 1995-05-05
Date provided is: 1995-05-05
Your birthdate is accepted
```

```
1 dateinput = input("Enter your birth date in yyyy-mm-dd format: ")
2 date_provided = datetime.strptime(dateinput, '%Y-%m-%d')
3 print ("Date provided is: " + date_provided.strftime('%Y-%m-%d'))
4
5 assert date_provided.date() < current_date.date(), "Birth date can't be higher than today"
6 print("Your birthdate is accepted")
```

```
Enter your birth date in yyyy-mm-dd format: 2022-05-05
Date provided is: 2022-05-05
```

```
-----
AssertionError                                Traceback (most recent call last)
<ipython-input-49-affb83e96d2d> in <module>
      3 print ("Date provided is: " + date_provided.strftime('%Y-%m-%d'))
      4
----> 5 assert date_provided.date() < current_date.date(), "Birth date can't be higher than today"
      6 print("Your birthdate is accepted")
```

```
AssertionError: Birth date can't be higher than today
```

Catching exceptions in python

Syntax

```
try:
    <--program code-->
except <--Exception Type 1-->:
    <--exception handling code-->
except <--Exception Type 2-->:
    <--exception handling code-->
else:
    <--program code to run if "try" block doesn't encounter any error-->
finally:
    <--program code that runs regardless of errors in the "try" or "else" block-->
```

Catching exceptions in python

Eksekusi
dimulai
dengan kode
di blok try

Blok else
akan
dieksekusi
jika tidak ada
kesalahan
dalam blok try

```
1 try:
2     f = open("testfile.txt", 'r')
3 except FileNotFoundError as fne:
4     print(fne)
5     print ('Creating file...')
6     f = open("testfile.txt", 'w')
7     f.write('2')
8 else:
9     data=f.readline(1)
10    print(data)
11 finally:
12    print ('Closing file')
13    f.close()
```

```
[Errno 2] No such file or directory: 'testfile.txt'
Creating file...
Closing file
```

Blok except
digunakan untuk
menentukan jalur
alternatif ketika
python exception
tertentu terjadi

Blok finally
selalu
dieksekusi
apakah
exception
terjadi atau
tidak

Agenda

05

KESIMPULAN

- Kuis
- Ringkasan

Pertanyaan

metode mana yang digunakan untuk membuat konstruktor di kelas?

- A. `__enter__()`
- B. `__init__()`
- C. `__doc__`
- D. `__myconstr__()`



Pertanyaan

metode mana yang digunakan untuk membuat konstruktor di kelas?

- A. `__enter__()`
- B. `__init__()`
- C. `__doc__`
- D. `__myconstr__()`

Answer - B



Ringkasan

- Object oriented programming memungkinkan pendekatan pemrograman modular dan struktural yang pada gilirannya meningkatkan kesederhanaan kode, memudahkan pemeliharaan kode
- Python module adalah file .py yang berisi implementasi kelas, atribut, dan fungsi, dan dapat dimuat menggunakan keyword import
- Python memiliki banyak modul bawaan yang berguna seperti matematika, os, time, statistics, dll.
- Python memiliki banyak built-in code exceptions yang dapat berguna dalam memprogram tindakan alternatif ketika pengecualian tertentu terjadi.





Orbit Future Academy

PT Orbit Ventura Indonesia
Center of Excellence (Jakarta Selatan)
Gedung Veteran RI, Lt.15
Unit Z15-002, Plaza Semanggi
Jl. Jenderal Sudirman Kav.50, Jakarta
12930, Indonesia

- 📍 Jakarta Selatan/Pusat
- 📍 Jakarta Barat/BSD
- 📍 Kota Bandung
- 📍 Kab. Bandung
- 📍 Jawa Barat

Hubungi Kami

Director of Sales & Partnership
ira@orbitventura.com
+62 858-9187-7388

Social Media



TERIMA KASIH