

Specially Crafted by

# Hi All!



M Ammar Fauzan

Business Intelligence Mgr

E V E R M S



Muhammad Ammar Fauzan

<https://www.linkedin.com/in/muhammad-ammar-fauzan-748883117/>



**Sistem Informasi**  
(2014-2018)



**Data Scientist**  
(Sept 2018- August 2020)



**Business Intelligence .Mgr**  
(Sept 2020- Present)

# Python Programming

Function & Error  
Handling



# **Topik Python Programming**

**Topik 1 : *Conditional Statement***

**Topik 2 : *Loop and Iteration***

**Topik 3 : *Function and Error Handling***

# ***Hands-On Requirement:***

**Hands - On :**

**Python Programming 3 - Rakamin Academy.ipynb**

**Dataset :**

**monthly\_rakamin\_customer\_order.csv**

**summary\_buyer\_rakamin\_store\_special.csv**

**Klik disini untuk mengakses  
folder Hands-On dan  
Dataset**

## Topik Sebelumnya : Conditional Statement & Iteration



In [1]:

```
1 #Challenge Time #1
2 #bentuk for
3 n_target = 0 #awal gak ada participant
4 for i in range(101, 151):
5     if i%5 == 0 and i%2 == 0:
6         n_target+=1
7
8 print('Jumlah user_id yang sesuai ada',n_target)
```

Jumlah user\_id yang sesuai ada 5

## Topik Sebelumnya : Conditional Statement & Iteration



In [1]:

```
1 #Challenge Time #1
2 #bentuk for
3 n_target = 0 #awal gak ada participant
4 for i in range(101, 151):
5     if i%5 == 0 and i%2 == 0:
6         n_target+=1
7
8 print('Jumlah user_id yang sesuai ada',n_target)
```

Jumlah user\_id yang sesuai ada 5

Bagaimana jika persyaratannya diubah menjadi **kelipatan 6** dan **3**? atau **kelipatan 5** dan **7**?

## Topik Sebelumnya : Conditional Statement & Iteration



In [1]:

```
1 #Challenge Time #1
2 #bentuk for
3 n_target = 0 #awal gak ada participant
4 for i in range(101, 151):
5     if i%5 == 0 and i%2 == 0:
6         n_target+=1
7
8 print('Jumlah user_id yang sesuai ada',n_target)
```

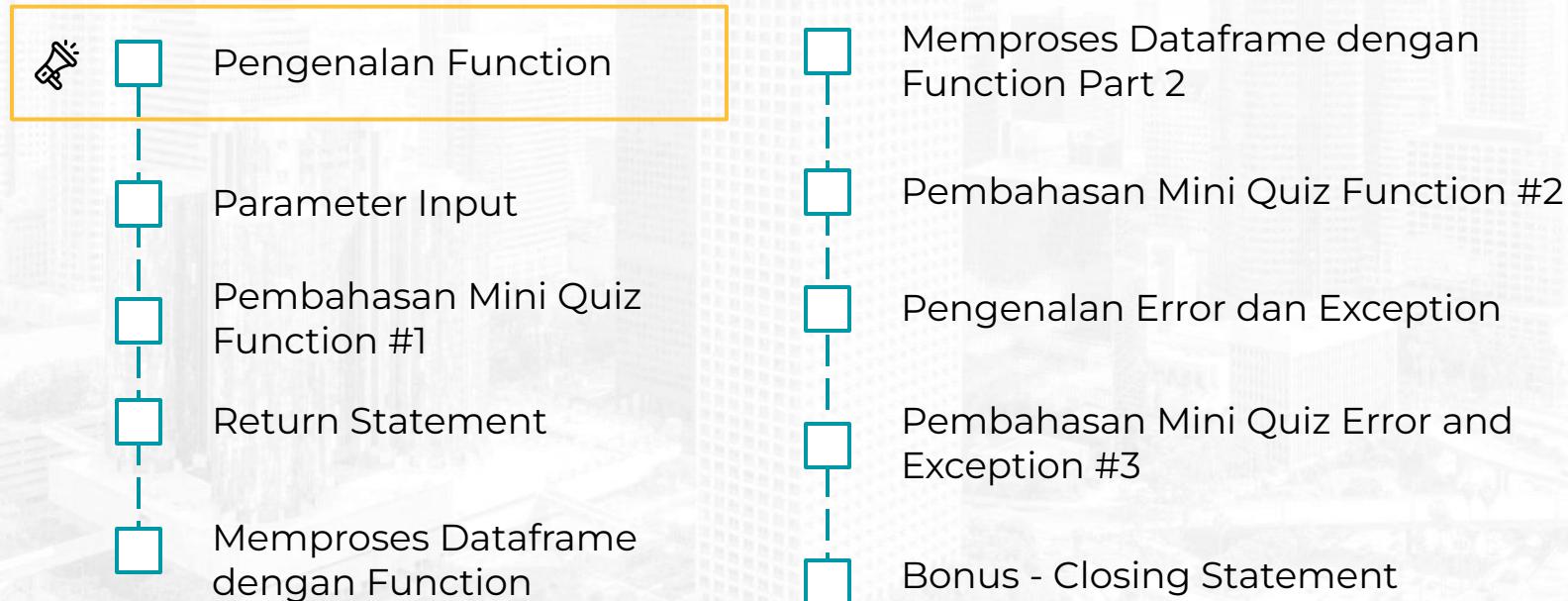
Jumlah user\_id yang sesuai ada 5

Apakah kita harus mengubah syntax setiap kali syaratnya berubah?

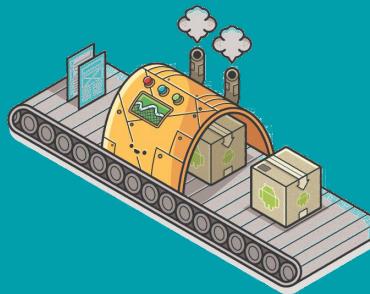
# Function and Error Handling

- Pengenalan Function
- Parameter Input
- Pembahasan Mini Quiz Function #1
- Return Statement
- Memproses Dataframe dengan Function
- Memproses Dataframe dengan Function Part 2
- Pembahasan Mini Quiz Function #2
- Pengenalan Error dan Exception
- Pembahasan Mini Quiz Error and Exception #3
- Bonus - Closing Statement

# Function and Error Handling



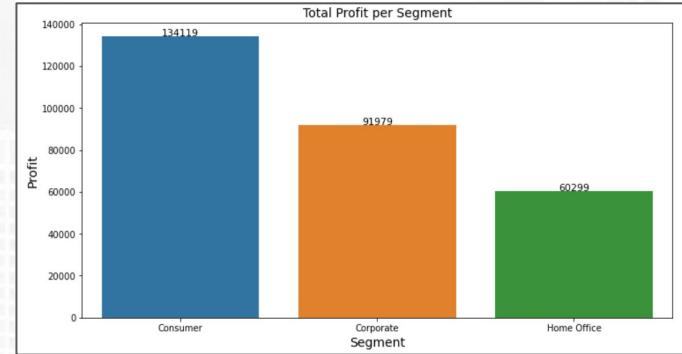
# Pengenalan *Function*



“**Function** adalah kumpulan **statement(s)** yang hanya berjalan jika dipanggil. Permintaan untuk menjalankan suatu function dikenal dengan istilah **function call**

**Function** sangat berguna ketika kita ingin melakukan suatu kumpulan **statement(s)** yang repetitif (berulang)”

## Bar Chart - Total Profit per Segment

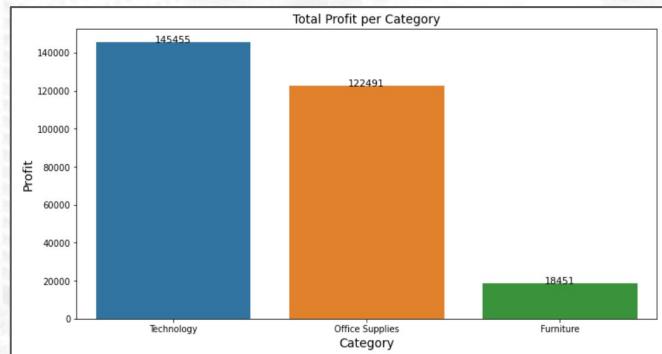


Untuk membuat **1 Bar Chart**  
membutuh ~25 baris kode.

Apakah untuk membuat 10 barchart dengan metric yang berbeda kita harus membuat  $25 \times 10 = \text{250 baris kode}$ ?

[Source Code](#)

## Bar Chart - Total Profit per Category



# Analogi Function - Memasak



Bahan Masakan  
(Input)



Chef Memasak  
(Proses)



Delicious Beef :  
(Output)

# Implementasi Function Sederhana

Contoh Code:

```
1 def function_name(parameter(s)):  
2     <statement(s)>
```

## Implementasi source code Function

- Header line pada function diawali dengan *syntax def* diikuti dengan *function name* kemudian kurung oval (...) dan titik dua :
- Di dalam kurung oval setelah nama function dapat diisi (optional) dengan suatu atau banyak *parameter* / argumen (input)
- Dalam *function body* bisa berisikan 1 atau lebih *statement*

# Implementasi Function Sederhana

Contoh Code:

```
1 def penjumlahan(a, b):  
2     hasil_jumlah = a+b #statement 1  
3     print(hasil_jumlah) #statement 2
```

Penjelasan contoh function di atas:

- Kita membuat function dengan nama “penjumlahan”
- Function memiliki 2 parameter a dan b
- Function memiliki 2 statement
  - Statement 1 → membuat variabel `hasil_jumlah` yang melakukan operasi  $a + b$
  - Statement 2 → melakukan operasi `print` dari variabel `hasil_jumlah`

# Melakukan Pemanggilan Function

Saat di **execute cell 2 tidak** memunculkan output apapun. Untuk membuat function berjalan kita perlu memanggilnya (**function call**)

**Contoh Code:**

```
[2]: 1 def penjumlahan(a, b):  
2     hasil_jumlah = a+b #statement 1  
3     print(hasil_jumlah) #statement 2
```

```
[ ]: 1
```

# Melakukan Pemanggilan Function

Saat di **execute cell 2 tidak** memunculkan output apapun. Untuk membuat function berjalan kita perlu memanggilnya (**function call**)

**Contoh Code:**

```
[2]: 1 def penjumlahan(a, b):  
      2     hasil_jumlah = a+b #statement 1  
      3     print(hasil_jumlah) #statement 2
```

```
[3]: 1 penjumlahan(2, 8)
```

10

Kita melakukan **function call** pada **cell 3** dengan **menuliskan nama function dan 2 parameter / inputan (a dan b)** yang dibutuhkan pada function penjumlahan

# Implementasi Function Sederhana dengan *Return Statement*

Contoh Code:

```
[4]: 1 def penjumlahan(a, b):
      2     hasil_jumlah = a+b #statement 1
      3     return hasil_jumlah #return statement
```

```
[ ]: 1
```

Pada baris kode ke 3, kita mendefinisikan *return statement*. Artinya saat function dijalankan dan berhasil menjalankan hingga *return statement* maka **function akan selesai dijalankan**

# Implementasi Function Sederhana dengan *Return Statement*

Contoh Code:

```
[4]: 1 def penjumlahan(a, b):  
      2     hasil_jumlah = a+b #statement 1  
      3     return hasil_jumlah #return statement
```

```
[5]: 1 penjumlahan(2, 8)|
```

```
[5]: 10
```

Pada cell ke 5, pada saat kita melakukan *function call* terdapat out [5] ketika kita menggunakan *return statement*. Coba bedakan dengan contoh sebelumnya!

# Global vs Local Variable



Dalam pemrograman python, terdapat istilah **global variable** dan **local variable**

- **Global Variable** - variabel yang dibuat diluar sebuah function, sehingga variabel tersebut dapat digunakan/diolah dimanapun
- **Local Variable** - variabel yang dibuat didalam sebuah function, sehingga hanya bisa diakses di dalam function tersebut

# Global vs Local Variable - Contoh #1

## Contoh Code:

```
[1]: 1 x = 'ini global variabel'  
  
[2]: 1 def belajar_global_local(x):  
2     x = 'ini local variabel' #nilai x dirubah value nya  
3     return x
```

# Global vs Local Variable - Contoh #1

## Contoh Code:

```
[1]: 1 x = 'ini global variabel'

[2]: 1 def belajar_global_local(x):
      2     x = 'ini local variabel' #nilai x dirubah value nya
      3     return x

[3]: 1 belajar_global_local(x) #function call

[3]: 'ini local variabel'
```

Meskipun **variabel x** diubah nilainya didalam function, namun nilainya tetap **tidak berubah** saat di jalankan di luar function (**cell 4**)

# Global vs Local Variable - Contoh #1

## Contoh Code:

```
[1]: 1 x = 'ini global variabel'

[2]: 1 def belajar_global_local(x):
      2     x = 'ini local variabel' #nilai x dirubah value nya
      3     return x

[3]: 1 belajar_global_local(x) #function call

[3]: 'ini local variabel'

[4]: 1 print(x)

      ini global variabel
```

Meskipun **variabel x** diubah nilainya didalam function, namun nilainya tetap **tidak berubah** saat di jalankan di luar function (**cell 4**)

# Global vs Local Variable - Contoh #2

## Contoh Code:

```
[1]: 1 a = 10 #inisiasi awal variable
[2]: 1 def pangkat(x, n):
      2     a = x**n #nilai a adalah operasi x pangkat n
      3     return a
```

# Global vs Local Variable - Contoh #2

## Contoh Code:

```
[1]: 1 a = 10 #inisiasi awal variable
[2]: 1 def pangkat(x, n):
      2     a = x**n #nilai a adalah operasi x pangkat n
      3     return a
[3]: 1 pangkat(2, 3)
[3]: 8
```

Meskipun **variabel a** diubah nilainya didalam function, namun nilainya tetap **tidak berubah** saat di jalankan di luar function (**Cell 4**)

# Global vs Local Variable - Contoh #2

## Contoh Code:

```
[1]: 1 a = 10 #inisiasi awal variable
[2]: 1 def pangkat(x, n):
      2     a = x**n #nilai a adalah operasi x pangkat n
      3     return a
[3]: 1 pangkat(2, 3)
[3]: 8
[4]: 1 print('Variabel a global', a)
Variabel a global 10
```

Meskipun **variabel a** diubah nilainya didalam function, namun nilainya tetap **tidak berubah** saat di jalankan di luar function (**Cell 4**)

# Function and Error Handling

-  Pengenalan Function
-  Parameter Input
-  Pembahasan Mini Quiz Function #1
-  Return Statement
-  Memproses Dataframe dengan Function Part 2
-  Pembahasan Mini Quiz Function #2
-  Pengenalan Error dan Exception
-  Pembahasan Mini Quiz Error and Exception #3
-  Bonus - Closing Statement

# Function and Error Handling



Pengenalan Function



Parameter Input



Pembahasan Mini Quiz  
Function #1



Return Statement



Memproses Dataframe  
dengan Function



Memproses Dataframe dengan  
Function Part 2



Pembahasan Mini Quiz Function #2



Pengenalan Error dan Exception



Pembahasan Mini Quiz Error and  
Exception #3



Bonus - Closing Statement

# Parameter **Input**



**“Parameter (input) pada function bersifat optional dan menjadi prasyarat jika didefinisikan saat membuat sebuah *function*. Parameter pada function disebut juga argument”**

## 4 macam parameter / input pada Function

Tanpa Parameter

1 atau lebih Parameter

Optional Parameter

Default Parameter

## 4 macam parameter / input pada Function

Tanpa Parameter

1 atau lebih Parameter

Optional Parameter

Default Parameter

# Function Tanpa Parameter

Contoh Code:

```
1 from datetime import date
2 from datetime import timedelta
3
4 def get_yesterday(): #tidak ada param
5     today = date.today()
6     yesterday = today - timedelta(days = 1)
7     return yesterday
8
9 print('Hari ini tanggal',date.today())
10
11 print('Kemarin tanggal',get_yesterday()) #tidak ada param
```

Saat membuat function **tidak definisikan parameter**. Sehingga saat melakukan **function call** cukup menggunakan nama functionnya saja dan kurung oval (..) tanpa inputan parameter.

# Function Tanpa Parameter

Contoh Code:

```
1 from datetime import date
2 from datetime import timedelta
3
4 def get_yesterday(): #tidak ada param
5     today = date.today()
6     yesterday = today - timedelta(days = 1)
7     return yesterday
8
9 print('Hari ini tanggal',date.today())
10
11 print('Kemarin tanggal',get_yesterday()) #tidak ada param
```

Saat membuat function **tidak definisikan parameter**. Sehingga saat melakukan **function call** cukup menggunakan nama functionnya saja dan kurung oval (..) tanpa inputan parameter.

# Function Tanpa Parameter

Contoh Code:

```
1 from datetime import date
2 from datetime import timedelta
3
4 def get_yesterday(): #tidak ada param
5     today = date.today()
6     yesterday = today - timedelta(days = 1)
7     return yesterday
8
9 print('Hari ini tanggal',date.today())
10
11 print('Kemarin tanggal',get_yesterday()) #tidak ada param
```

```
Hari ini tanggal 2022-08-21
Kemarin tanggal 2022-08-20
```

Saat membuat function **tidak definisikan parameter**. Sehingga saat melakukan **function call** cukup menggunakan nama functionnya saja dan kurung oval (..) tanpa inputan parameter.

# Function Tanpa Parameter

Contoh Code:

```
1 from datetime import date
2 from datetime import timedelta
3
4 def get_yesterday(): #tidak ada param
5     today = date.today()
6     yesterday = today - timedelta(days = 1)
7     return yesterday
8
9 print('Hari ini tanggal',date.today())
10
11 print('Kemarin tanggal',get_yesterday()) #tidak ada param
```

```
Hari ini tanggal 2022-08-21
Kemarin tanggal 2022-08-20
```

Saat membuat function **tidak definisikan parameter**. Sehingga saat melakukan **function call** cukup menggunakan nama functionnya saja dan kurung oval (..) tanpa inputan parameter.

**Contoh Kasus:** Mengambil data order 7 hari yang lalu / kemarin (**get\_order\_last\_7days / get\_order\_yesterday**). Hal ini dimungkinkan karena logik nya sudah pasti.

## 4 macam parameter / input pada Function

Tanpa Parameter

1 atau lebih Parameter

Optional Parameter

Default Parameter

# Function dengan 1 atau Lebih Parameter

Contoh Code:

```
1 def luas_segitiga(a, t): #membuat fungsi Luas segitiga dengan 2 parameter a dan t
2     hasil = (a*t)/2 #melakukan operasi Luas segita alas(a)*tinggi(t)*1/2
3     return hasil #mengerluarkan output variabel hasil
4
5 luas_segitiga(6,3) #menginputkan 2 parameter, dimana a = 6 dan t = 3
```

Contoh di atas, membuat **function luas segitiga dengan 2 parameter input a dan t**. Sehingga saat melakukan **function call wajib** menginputkan parameter a dan t sebagai prasyarat! Jika tidak akan **error**.

# Function dengan 1 atau Lebih Parameter

Contoh Code:

```
1 def luas_segitiga(a, t): #membuat fungsi Luas segitiga dengan 2 parameter a dan t
2     hasil = (a*t)/2 #melakukan operasi Luas segita alas(a)*tinggi(t)*1/2
3     return hasil #mengerluarkan output variabel hasil
4
5 luas_segitiga(6,3) #menginputkan 2 parameter, dimana a = 6 dan t = 3
```

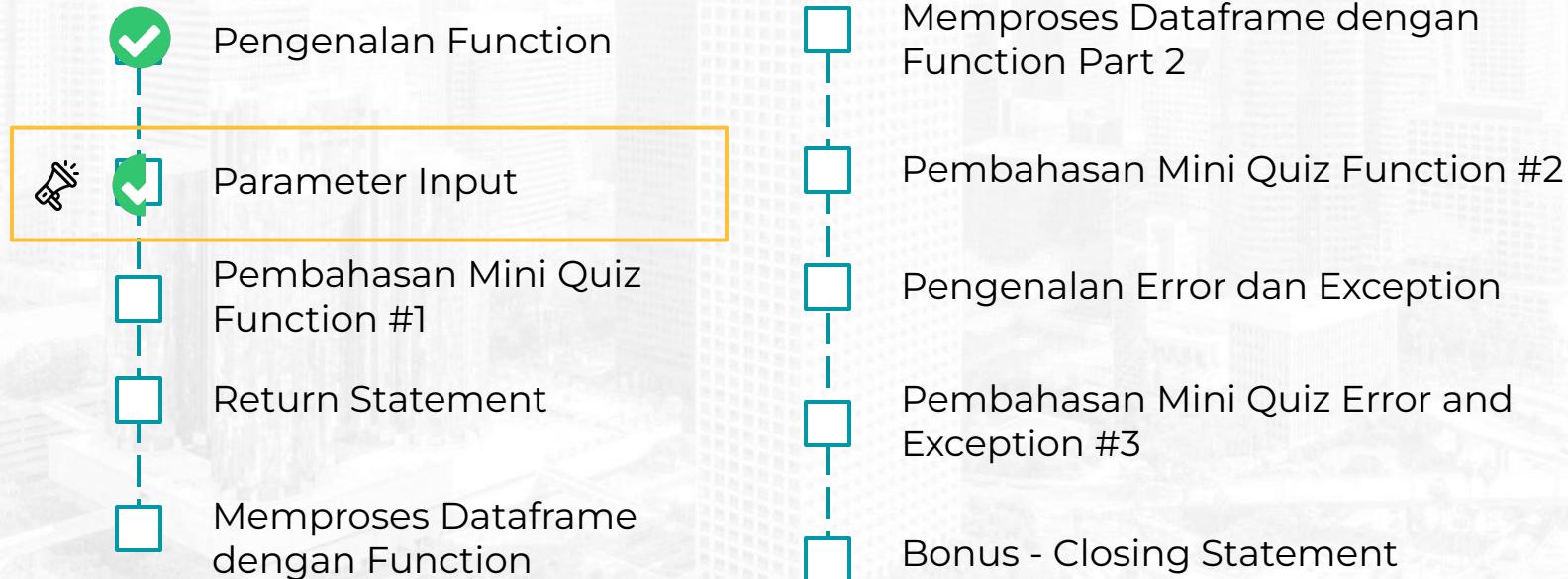
9.0

Contoh di atas, membuat **function luas segitiga dengan 2 parameter input a dan t**. Sehingga saat melakukan **function call wajib** menginputkan parameter a dan t sebagai prasyarat! Jika tidak akan **error**.

# Function and Error Handling

-  Pengenalan Function
-  Parameter Input
-  Pembahasan Mini Quiz Function #1
-  Return Statement
-  Memproses Dataframe dengan Function Part 2
-  Pembahasan Mini Quiz Function #2
-  Pengenalan Error dan Exception
-  Pembahasan Mini Quiz Error and Exception #3
-  Bonus - Closing Statement

# Function and Error Handling



## 4 macam parameter / input pada Function

Tanpa Parameter

1 atau lebih Parameter

Optional Parameter

Default Parameter

# Function dengan Optional Parameter

**Study Case :** Objective function **city\_order** adalah untuk mendapatkan total order pada suatu bulan tertentu dari 1-3 kota sekaligus.

```
1 import pandas as pd\n\n1 df = pd.read_csv('monthly_rakamin_customer_order.csv')\n2 df.head()
```

	order_month	full_name	user_id	phone	city	province		email	total_order	total_gmv	total_visit
0	2018-01-01	Ahmad Budiman	10000011	628785222211	Jakarta Pusat	DKI Jakarta		ahmad@gmail.com	125	3490000	230
1	2018-01-01	Azzam Haqq	10000012	628785222212	Surabaya	Jawa Timur		azzam@gmail.co.id	173	4757500	304
2	2018-01-01	Amrina Putri	10000013	628785222213	Jakarta Pusat	DKI Jakarta		amrina@yahoo.co.id	20	550000	62
3	2018-01-01	Aan Utama	10000014	628785222214	Medan	Sumatra Utara		aan@yahoo.com	18	495000	100

# Function dengan Optional Parameter

Contoh Code:

```
1 def city_order(month, city1, city2=None, city3=None):
2     #filter dataframe based on Month and City
3     df_selected_city = df[(df.order_month == month) & (df.city.isin([city1, city2, city3]))]
4
5     #get trx agregat function sum
6     total_trx = df_selected_city.total_order.sum()
7
8     #print
9     print('Total Trx pada Bulan', month, 'dari kota-kota tersebut adalah', total_trx)
```

Pada function **city\_order** didefinisikan dengan 2 parameter wajib (month dan city1) dan **2 parameter optional (city2 dan city3)**, artinya **tidak wajib** diinputkan saat melakukan *function call*.

# Function dengan Optional Parameter

Contoh Code:

```
1 def city_order(month, city1, city2=None, city3=None):
2     #filter dataframe based on Month and City
3     df_selected_city = df[(df.order_month == month) & (df.city.isin([city1, city2, city3]))]
4
5     #get trx agregat function sum
6     total_trx = df_selected_city.total_order.sum()
7
8     #print
9     print('Total Trx pada Bulan', month, 'dari kota-kota tersebut adalah', total_trx)
```

```
1 city_order('2018-08-01', 'Jakarta Pusat') #menginputkan 2 parameter saja untuk month dan city 1
```

Total Trx pada Bulan 2018-08-01 dari kota-kota tersebut adalah 45

Pada function **city\_order** didefinisikan dengan 2 parameter wajib (month dan city1) dan **2 parameter optional (city2 dan city3)**, artinya **tidak wajib** diinputkan saat melakukan *function call*.

# Function dengan Optional Parameter

Contoh Code:

```
1 def city_order(month, city1, city2=None, city3=None):
2     #filter dataframe based on Month and City
3     df_selected_city = df[(df.order_month == month) & (df.city.isin([city1, city2, city3]))]
4
5     #get trx agregat function sum
6     total_trx = df_selected_city.total_order.sum()
7
8     #print
9     print('Total Trx pada Bulan', month, 'dari kota-kota tersebut adalah', total_trx)
```

```
1 city_order('2018-08-01', 'Jakarta Pusat') #menginputkan 2 parameter saja untuk month dan city 1
```

Total Trx pada Bulan 2018-08-01 dari kota-kota tersebut adalah 45

Pada function **city\_order** didefinisikan dengan 2 parameter wajib (month dan city1) dan **2 parameter optional (city2 dan city3)**, artinya **tidak wajib** diinputkan saat melakukan *function call*.

# Function dengan Optional Parameter

Contoh Code:

```
1 city_order('2021-09-01', 'Jakarta Pusat', city2 = 'Bandung') #menginputkan 3 parameter untuk month, city1 dan city2
Total Trx pada Bulan 2021-09-01 dari kota-kota tersebut adalah 134
```

Saat melakukan **function call** dalam context input parameter bisa mendefinisikan dengan nama parameter nya atau tidak. Secara logic akan terdefinisikan **dari kiri ke kanan**.

# Function dengan Optional Parameter

Contoh Code:

month	city1	city2
-------	-------	-------

```
1 city_order('2021-09-01', 'Jakarta Pusat', city2 = 'Bandung') #menginputkan 3 parameter untuk month, city1 dan city2
```

```
Total Trx pada Bulan 2021-09-01 dari kota-kota tersebut adalah 134
```

Saat melakukan **function call** dalam context input parameter bisa mendefinisikan dengan nama parameter nya atau tidak. Secara logic akan terdefinisikan **dari kiri ke kanan.**

## 4 macam parameter / input pada Function

Tanpa Parameter

1 atau lebih Parameter

Optional Parameter

Default Parameter

# Function dengan Default Parameter

**Study Case :** Objective function **user\_order** adalah untuk mendapatkan total order pada suatu bulan tertentu dari specific user tertentu.

```
1 import pandas as pd\n\n1 df = pd.read_csv('monthly_rakamin_customer_order.csv')\n2 df.head()
```

	order_month	full_name	user_id	phone	city	province		email	total_order	total_gmv	total_visit
0	2018-01-01	Ahmad Budiman	10000011	628785222211	Jakarta Pusat	DKI Jakarta		ahmad@gmail.com	125	3490000	230
1	2018-01-01	Azzam Haqq	10000012	628785222212	Surabaya	Jawa Timur		azzam@gmail.co.id	173	4757500	304
2	2018-01-01	Amrina Putri	10000013	628785222213	Jakarta Pusat	DKI Jakarta		amrina@yahoo.co.id	20	550000	62
3	2018-01-01	Aan Utama	10000014	628785222214	Medan	Sumatra Utara		aan@yahoo.com	18	495000	100

# Function dengan Default Parameter

Contoh Code:

```
1 def user_order(user_id, month='2021-09-01'):
2     #filter dataframe based on Month and User
3     df_selected_user_monthly = df[(df.order_month == month)
4                                     & (df.user_id == user_id)].reset_index()
5
6     #get order agregat function sum
7     total_trx = df_selected_user_monthly.total_order.sum()
8
9     #print
10    print('Total Trx pada Bulan', month, 'untuk user_id', user_id, 'adalah', total_trx)
```

Pada function **user\_order** didefinisikan dengan **1 parameter wajib (user\_id)** dan **1 parameter default (month dengan value 2021-09-01)**. Hal ini mengakibatkan saat melakukan **function call tidak wajib mendefinisikan month.**

# Function dengan Default Parameter

Contoh Code:

```
1 def user_order(user_id, month='2021-09-01'):
2     #filter dataframe based on Month and User
3     df_selected_user_monthly = df[(df.order_month == month)
4                                     & (df.user_id == user_id)].reset_index()
5
6     #get order agregat function sum
7     total_trx = df_selected_user_monthly.total_order.sum()
8
9     #print
10    print('Total Trx pada Bulan', month, 'untuk user_id', user_id, 'adalah', total_trx)
```

```
1 user_order(10000038)
```

```
Total Trx pada Bulan 2021-09-01 untuk user_id 10000038 adalah 61
```

Pada function **user\_order** didefinisikan dengan **1 parameter wajib (user\_id)** dan **1 parameter default (month dengan value 2021-09-01)**. Hal ini mengakibatkan saat melakukan **function call tidak wajib mendefinisikan month.**

# Function dengan Default Parameter

Contoh Code:

```
1 user_order(10000038, '2020-01-01') #value default akan ter replace dengan input baru
```

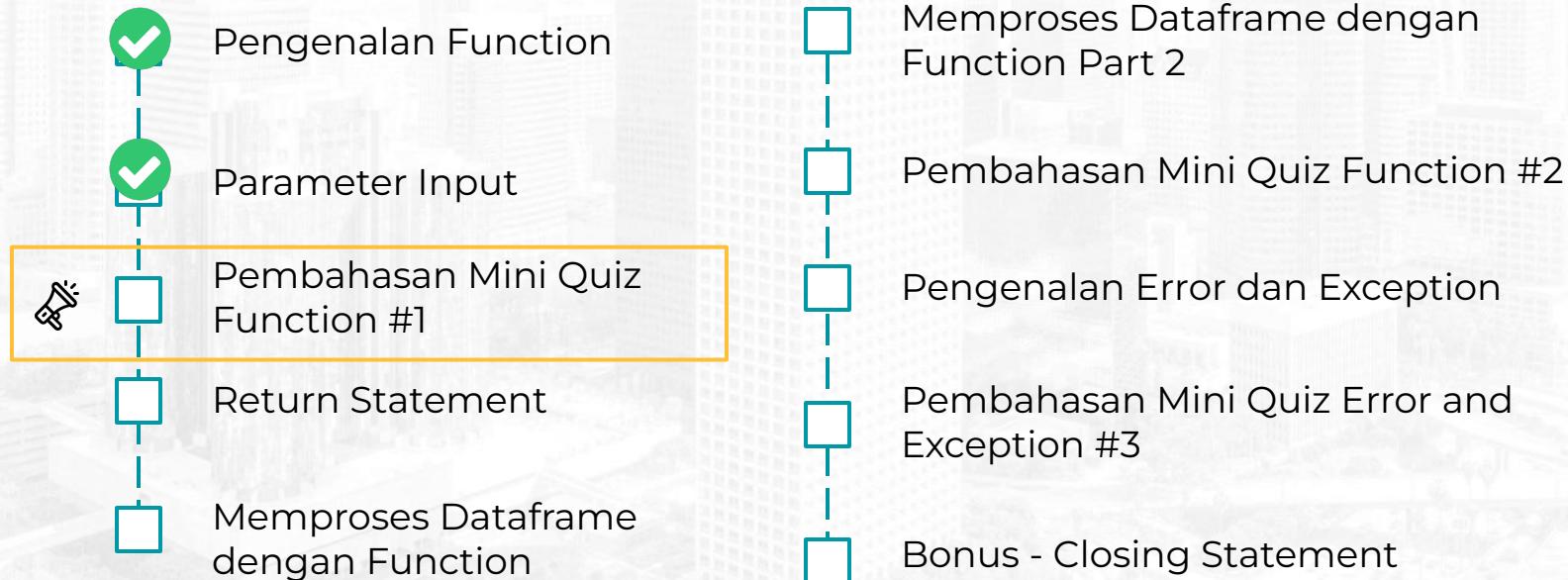
```
Total Trx pada Bulan 2020-01-01 untuk user_id 10000038 adalah 26
```

Jika parameter month **didefinisikan saat melakukan function call**, maka nilai default akan tergantikan / ter-replace

# Function and Error Handling

-  Pengenalan Function
-  Parameter Input
-  Pembahasan Mini Quiz Function #1
-  Return Statement
-  Memproses Dataframe dengan Function Part 2
-  Pembahasan Mini Quiz Function #2
-  Pengenalan Error dan Exception
-  Pembahasan Mini Quiz Error and Exception #3
-  Bonus - Closing Statement

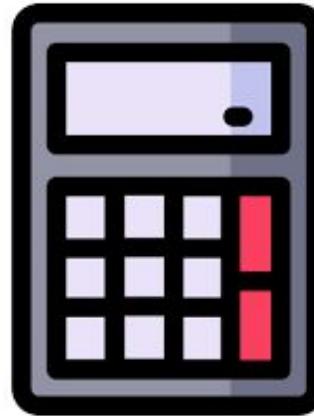
# Function and Error Handling





Buatlah **function my\_calculator** yang bisa menghitung 2 variabel angka dengan 4 alternatif metode operasi: (parameter **metode** by default Penjumlahan)

1. Penjumlahan
2. Pengurangan
3. Perkalian
4. Pembagian



# Function and Error Handling



Pengenalan Function



Parameter Input



Pembahasan Mini Quiz  
Function #1



Return Statement



Memproses Dataframe  
dengan Function



Memproses Dataframe dengan  
Function Part 2



Pembahasan Mini Quiz Function #2



Pengenalan Error dan Exception



Pembahasan Mini Quiz Error and  
Exception #3



Bonus - Closing Statement

# Return **Statement**



“*Return Statement* merupakan **output/keluaran** dari setiap function. Function akan otomatis selesai dijalankan jika *return statement* berhasil dijalankan. Penggunaan return **tidak bersifat wajib (optional).**”

# Mengapa perlu mendefinisikan Return Statement? #1

Kita memerlukan **output** dari **function** jika hasil dari output/balikan tersebut digunakan **lagi** dalam pemrosesan selanjutnya.

## Contoh Code:

```
1 def cek_genap(x):
2     if x%2 == 0:
3         hasil = 'Genap'
4     else:
5         hasil = 'Ganjil'
6     return hasil
```

```
1 y = 101
2 print('Bilangan',y, 'termasuk bilangan', cek_genap(y))
```

Bilangan 101 termasuk bilangan Ganjil



Output dari fungsi **cek\_genap** digunakan sebagai value pada operasi **print()**

## Mengapa perlu mendefinisikan Return Statement? #2

Kita memerlukan **output** dari **function** jika hasil dari output/balikan tersebut digunakan **lagi** dalam pemrosesan selanjutnya.

### Contoh Code:

```
1 # rumus: sisi x sisi
2 def luas_persegi(sisi):
3     luas = sisi * sisi
4     return luas
5
6
7 # rumus: sisi x sisi x sisi
8 def volume_persegi(sisi):
9     volume = luas_persegi(sisi) * sisi #menggunakan fungsi dalam fungsi
10    return volume
11
12 volume_persegi(5)
```

125



Output dari fungsi  
**luas\_persegi** digunakan  
untuk menghitung  
**volume**

# Implementasi Function dengan *Multi Return statement*

Contoh Code:

```
1 def kalkulator(metode, a, b):
2     metode = metode.lower()
3     if metode == 'penjumlahan':
4         hasil = a+b
5         return hasil #return statement 1
6     elif metode == 'pengurangan':
7         hasil = a-b
8         return hasil #return statement 2
9     else:
10        hasil = '-'
11    print('Inputan metode', metode, 'dengan inputan', a, 'dan', b, 'tidak sesuai system inputan', hasil)
```

1

**Return statement** bisa kita definisikan **lebih dari 1**. Function akan otomatis selesai dijalankan jika **return statement** berhasil dijalankan.

# Implementasi Function dengan *Multi Return statement*

Contoh Code:

```
def kalkulator(metode, a, b):
    metode = metode.lower()
    if metode == 'penjumlahan':
        hasil = a+b
        return hasil #return statement 1
    elif metode == 'pengurangan':
        hasil = a-b
        return hasil #return statement 2
    else:
        hasil = '-'
    print('Inputan metode', metode, 'dengan inputan', a, 'dan', b, 'tidak sesuai system inputan', hasil)

kalkulator('penjumlahan', 5, 15)
```

20

**Return statement** bisa kita definisikan **lebih dari 1**. Function akan otomatis selesai dijalankan jika **return statement** berhasil dijalankan.

# Implementasi Function dengan *Multi Return statement*

## Contoh Code:

```
1 def kalkulator(metode, a, b):
2     metode = metode.lower()
3     if metode == 'penjumlahan':
4         hasil = a+b
5         return hasil #return statement 1
6     elif metode == 'pengurangan':
7         hasil = a-b
8         return hasil #return statement 2
9     else:
10        hasil = '-'
11    print('Inputan metode', metode, 'dengan inputan', a, 'dan', b, 'tidak sesuai system inputan', hasil)
```

```
1 kalkulator('penjum', 5, 6)
```

Inputan metode penjum dengan inputan 5 dan 6 tidak sesuai system inputan -

**Function dijalankan hingga line code terakhir (line 11)**

## Output pada Return Statement bisa lebih dari 1

Sangat dimungkinkan jika value pada function bernilai **lebih dari 1**. Secara otomatis hasilnya akan **menjadi tuple**, sehingga setiap itemnya dapat diambil menggunakan **index**.

### Contoh Code:

```
1 def luas_volume_balok(p, l, t):
2     luas = p*l
3     volume = p*l*t
4     return luas, volume #output akan berwujud tuple
5
6 df_info = luas_volume_balok(2,4,6)
7 print('Luas', df_info[0], 'cm2 dan Volume', df_info[1], 'cm3')
```

Luas 8 cm<sup>2</sup> dan Volume 48 cm<sup>3</sup>

- **df\_info[0]** = untuk mendapatkan nilai luas
- **df\_info[1]** = untuk mendapatkan nilai volume

## Output pada Return Statement bisa lebih dari 1

Selain menggunakan index, kita juga bisa langsung **memberikan nilai pada masing-masing item nya kedalam beberapa variabel sekaligus**, seperti contoh berikut

### Contoh Code:

```
1 nilai_luas, nilai_volume = luas_volume_balok(2,4,6) #2 variabel Langsung
2 print('Ini Nilai Luas', nilai_luas)
3 print('Ini Nilai Volume', nilai_volume)
```

Ini Nilai Luas 8

Ini Nilai Volume 48

## Output pada Return Statement bisa lebih dari 1

Selain menggunakan index, kita juga bisa langsung **memberikan nilai pada masing-masing item nya kedalam beberapa variabel sekaligus**, seperti contoh berikut

### Contoh Code:

```
1 nilai_luas, nilai_volume = luas_volume_balok(2,4,6) #2 variabel Langsung
2 print('Ini Nilai Luas', nilai_luas)
3 print('Ini Nilai Volume', nilai_volume)
```

```
Ini Nilai Luas 8
Ini Nilai Volume 48
```

# Function and Error Handling

-  Pengenalan Function
-  Parameter Input
-  Pembahasan Mini Quiz Function #1
-  Return Statement
-  Memproses Dataframe dengan Function Part 2
-  Pembahasan Mini Quiz Function #2
-  Pengenalan Error dan Exception
-  Pembahasan Mini Quiz Error and Exception #3
-  Bonus - Closing Statement

# Function and Error Handling



Pengenalan Function



Parameter Input



Pembahasan Mini Quiz  
Function #1



Return Statement



Memproses Dataframe  
dengan Function



Memproses Dataframe dengan  
Function Part 2



Pembahasan Mini Quiz Function #2



Pengenalan Error dan Exception



Pembahasan Mini Quiz Error and  
Exception #3



Bonus - Closing Statement

# Memproses **Dataframe** dengan Function



“ Objective dari pemrosesan Dataframe dengan function adalah kita mampu untuk membuat kolom baru menggunakan fungsi apply lambda. ”

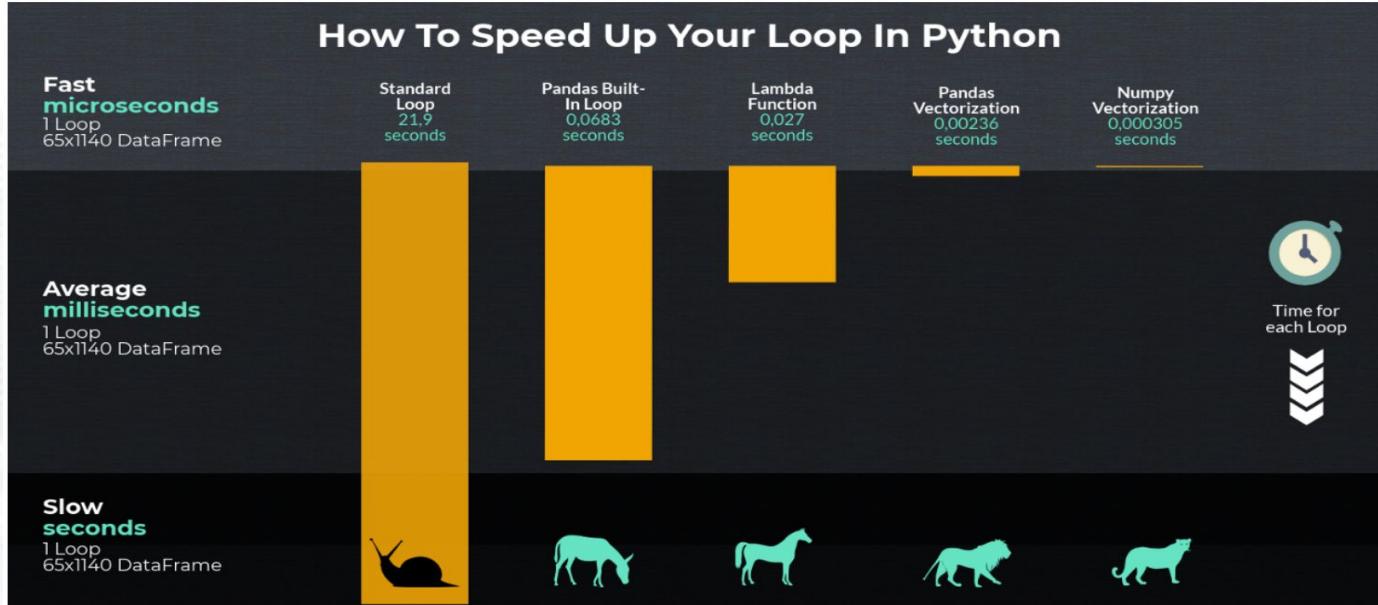
# Ada banyak cara dalam membuat kolom baru

Salah satu nya bisa menggunakan **for iteration** tiap baris pada dataframe menggunakan **.iterrows()**

## Contoh Code:

```
1 list_segment = [] #membuat list kosong
2 for i, kolom in df.iterrows(): #melakukan iterasi pada setiap baris dataframe
3     if kolom['total_order'] >= 100:
4         segment = 'high'
5     elif kolom['total_order'] >= 20 and kolom['total_order'] < 100:
6         segment = 'mid'
7     else:
8         segment = 'low'
9     list_segment.append(segment) #menambahkan list kosong dari item tiap row
10
11 df['segment_for'] = list_segment
12 df.head()
```

	order_month	user_id	full_name	phone	city	province	email	total_order	total_gmv	total_visit	segment_for
0	2018-01-01	10000011	Ahmad Budiman	628785222211	Jakarta Pusat	DKI Jakarta	ahmad@gmail.com	127	3492500	235	high
1	2018-01-01	10000012	Azzam Haqq	628785222212	Surabaya	Jawa Timur	azzam@gmail.co.id	173	4757500	304	high
2	2018-01-01	10000013	Amrina Putri	628785222213	Jakarta Pusat	DKI Jakarta	amrina@yahoo.co.id	20	550000	62	mid
3	2018-01-01	10000014	Aan Utama	628785222214	Medan	Sumatra Utara	aan@yahoo.com	18	495000	100	low
4	2018-01-01	10000015	Adam Adi	628785222215	Jakarta Timur	DKI Jakarta	adam@gmail.com	159	4372500	524	high



[Reference](#)

# Lambda Function

**Lambda function** adalah fungsi anonim (tidak bernama) yang **hanya memiliki 1 statement saja**. Namun, lambda function mampu memiliki lebih dari satu parameter input.

**Contoh Code:**

```
1 lambda parameter(s) : expression
```

# Lambda Function

**Lambda function** adalah fungsi anonim (tidak bernama) yang **hanya memiliki 1 statement saja**. Namun, lambda function mampu memiliki lebih dari satu parameter input.

## Contoh Code:

### Contoh 1

1 **lambda** parameter(s) : expression

```
1 a1 = lambda x, y: x+y #2 parameter x dan y
1 a1(2, 3)
```

# Lambda Function

**Lambda function** adalah fungsi anonim (tidak bernama) yang **hanya memiliki 1 statement saja**. Namun, lambda function mampu memiliki lebih dari satu parameter input.

## Contoh Code:

Contoh 1

1 **lambda** parameter(s) : expression

```
1 a1 = lambda x, y: x+y #2 parameter x dan y
```

```
1 a1(2, 3)
```

5

Contoh 2

```
1 a2 = lambda x : 'Genap' if x%2==0 else 'Ganjil'
```

```
1 a2(18)
```

'Genap'

## Sekilas Penjelasan .apply()

	A	B
0	2	8
1	2	8
2	2	8

```
1 df_apply.apply(np.sum, axis=0)
```

A	6
B	24

Axis = 0 menjalankan fungsi  
agregat sum pada setiap kolom

```
1 df_apply.apply(np.sum, axis=1)
```

0	10
1	10
2	10

Axis = 1 menjalankan fungsi agregat  
sum pada setiap baris

# Function and Error Handling

-  Pengenalan Function
-  Parameter Input
-  Pembahasan Mini Quiz Function #1
-  Return Statement
-  Memproses Dataframe dengan Function
- Memproses Dataframe dengan Function Part 2
- Pembahasan Mini Quiz Function #2
- Pengenalan Error dan Exception
- Pembahasan Mini Quiz Error and Exception #3
- Bonus - Closing Statement

# Function and Error Handling

-  Pengenalan Function
-  Parameter Input
-  Pembahasan Mini Quiz Function #1
-  Return Statement
-  Memproses Dataframe dengan Function

-   Memproses Dataframe dengan Function Part 2
- Pembahasan Mini Quiz Function #2
- Pengenalan Error dan Exception
- Pembahasan Mini Quiz Error and Exception #3
- Bonus - Closing Statement

## Alternatif yang lebih efisien untuk membuat kolom baru Apply Lambda

```
1 #membuat function segment
2 def segment(x):
3     if x['total_order'] >= 100:
4         segment = 'high'
5     elif x['total_order'] >= 20 and x['total_order'] < 100:
6         segment = 'mid'
7     else:
8         segment = 'low'
9     return segment
```

Jika secara processing **complex**, perlu diinisiasi suatu function terlebih dahulu

## Alternatif yang lebih efisien untuk membuat kolom baru Apply Lambda

```

1 #membuat function segment
2 def segment(x):
3     if x['total_order'] >= 100:
4         segment = 'high'
5     elif x['total_order'] >= 20 and x['total_order'] < 100:
6         segment = 'mid'
7     else:
8         segment = 'low'
9     return segment

```

Buat **Function Def** dengan output **segmen** terlebih dulu

```

1 df['segment_apply'] = df.apply(lambda x: segment(x), axis=1)
2 df.head()

```

	order_month	user_id	full_name	phone	city	province	email	total_order	total_gmv	total_visit	segment_apply
0	2018-01-01	10000011	Ahmad Budiman	628785222211	Jakarta Pusat	DKI Jakarta	ahmad@gmail.com	127	3492500	235	high
1	2018-01-01	10000012	Azzam Haqq	628785222212	Surabaya	Jawa Timur	azzam@gmail.co.id	173	4757500	304	high
2	2018-01-01	10000013	Amrina Putri	628785222213	Jakarta Pusat	DKI Jakarta	amrina@yahoo.co.id	20	550000	62	mid
3	2018-01-01	10000014	Aan Utama	628785222214	Medan	Sumatra Utara	aan@yahoo.com	18	495000	100	low
4	2018-01-01	10000015	Adam Adi	628785222215	Jakarta Timur	DKI Jakarta	adam@gmail.com	159	4372500	524	high

## Alternatif yang lebih efisien untuk membuat kolom baru Apply Lambda

```

1 #membuat function segment
2 def segment(x):
3     if x['total_order'] >= 100:
4         segment = 'high'
5     elif x['total_order'] >= 20 and x['total_order'] < 100:
6         segment = 'mid'
7     else:
8         segment = 'low'
9     return segment

```

Buat **Function Def** dengan output **segmen** terlebih dulu

```

1 df['segment_apply'] = df.apply(lambda x: segment(x), axis=1)
2 df.head()

```

**Kolom baru**

	order_month	user_id	full_name	phone	city	province	email	total_order	total_gmv	total_visit	segment_apply
0	2018-01-01	10000011	Ahmad Budiman	628785222211	Jakarta Pusat	DKI Jakarta	ahmad@gmail.com	127	3492500	235	high
1	2018-01-01	10000012	Azzam Haqq	628785222212	Surabaya	Jawa Timur	azzam@gmail.co.id	173	4757500	304	high
2	2018-01-01	10000013	Amrina Putri	628785222213	Jakarta Pusat	DKI Jakarta	amrina@yahoo.co.id	20	550000	62	mid
3	2018-01-01	10000014	Aan Utama	628785222214	Medan	Sumatra Utara	aan@yahoo.com	18	495000	100	low
4	2018-01-01	10000015	Adam Adi	628785222215	Jakarta Timur	DKI Jakarta	adam@gmail.com	159	4372500	524	high

# Penjelasan Script Apply Lambda

2| .apply() pada pandas untuk memproses suatu fungsi setiap baris pada dataframe (axis = 1)

```
1 df['segment_apply'] = df.apply(lambda x: segment(x), axis=1)
```

1| Mendefinisikan  
Nama Kolom Baru

3| Lambda digunakan untuk menjalankan  
function segment pada setiap barisnya

**Notes:** x di definisikan sebagai alias dari dataframe yang jadikan sebagai parameter / inputan function segment

## Penjelasan Script Apply Lambda #2

```
1 #direct function without ()
2 df['segment_apply'] = df.apply(lambda x: segment, axis=1)
```

```
1 #spesific kolom input
2 df['segment_apply'] = df.apply(lambda x: segment(x['total_order']), axis=1)
```

**Notes:** beberapa cara lain dalam menuliskan syntax apply lambda

# Implementasi Apply Lambda tanpa Function def

Kita bisa menjalankan apply lambda **tanpa membuat suatu function def terlebih dulu**. Namun dalam case ini parameternya **wajib berupa series** bukan dataframe (**fokus 1 kolom**)

Contoh Code:

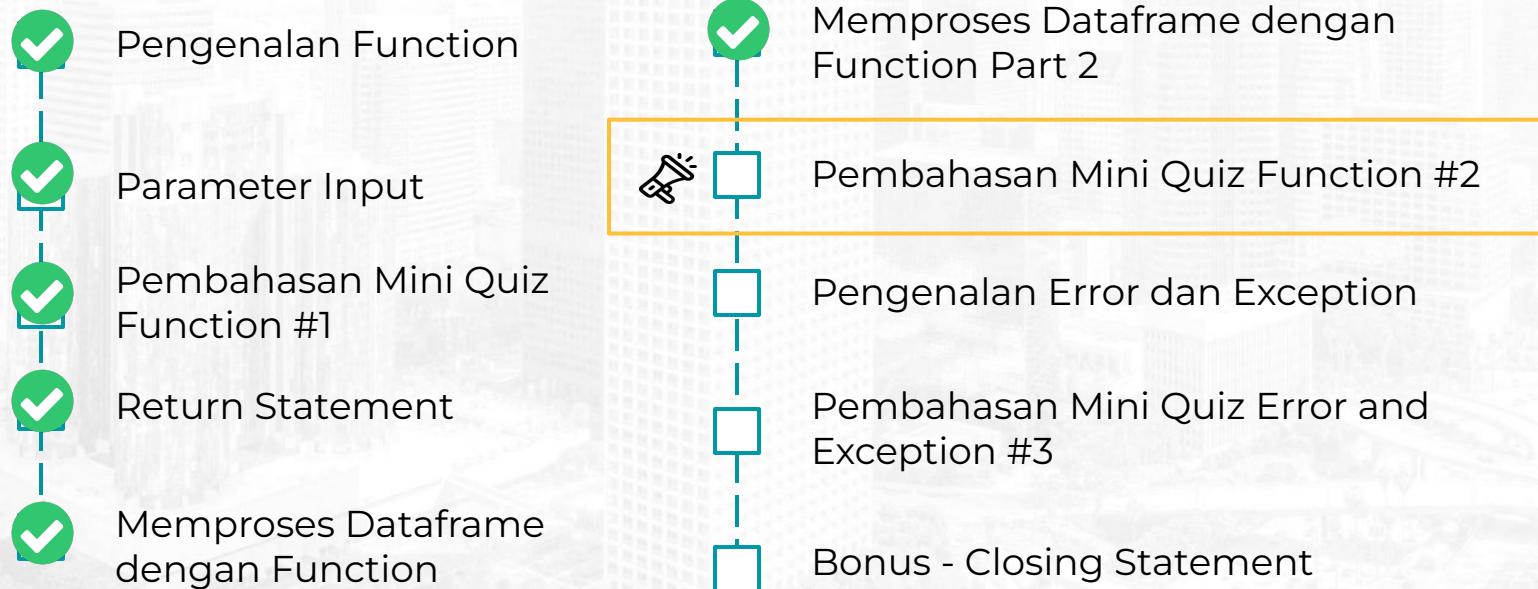
```
1 df['jakpus'] = df['city'].apply(lambda x: True if x == 'Jakarta Pusat' else False)
2 df.head()
```

	order_month	user_id	full_name	phone	city	province	email	total_order	total_gmv	total_visit	segment_apply	jakpus
0	2018-01-01	10000011	Ahmad Budiman	628785222211	Jakarta Pusat	DKI Jakarta	ahmad@gmail.com	127	3492500	235	high	True
1	2018-01-01	10000012	Azzam Haqq	628785222212	Surabaya	Jawa Timur	azzam@gmail.co.id	173	4757500	304	high	False
2	2018-01-01	10000013	Amrina Putri	628785222213	Jakarta Pusat	DKI Jakarta	amrina@yahoo.co.id	20	550000	62	mid	True
3	2018-01-01	10000014	Aan Utama	628785222214	Medan	Sumatra Utara	aan@yahoo.com	18	495000	100	low	False
4	2018-01-01	10000015	Adam Adi	628785222215	Jakarta Timur	DKI Jakarta	adam@gmail.com	159	4372500	524	high	False

# Function and Error Handling

-  Pengenalan Function
-  Parameter Input
-  Pembahasan Mini Quiz Function #1
-  Return Statement
-  Memproses Dataframe dengan Function
-  Memproses Dataframe dengan Function Part 2
-  Pembahasan Mini Quiz Function #2
-  Pengenalan Error dan Exception
-  Pembahasan Mini Quiz Error and Exception #3
-  Bonus - Closing Statement

# Function and Error Handling





Menggunakan dataset `monthly_rakamin_customer_order`, buatlah kolom baru dengan nama **provider\_email** dengan ketentuan berdasarkan domain emailnya:

1. gmail → Google
2. yahoo → Yahoo

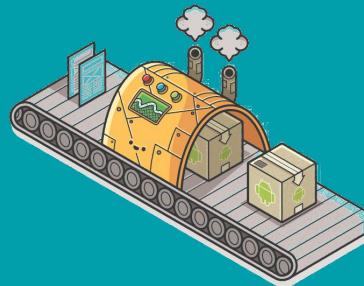
Gunakan syntax **apply lambda dan .split('@')!** Tentukan provider email apa yang paling banyak digunakan customer pada tahun 2021

# Function and Error Handling

-  Pengenalan Function
-  Parameter Input
-  Pembahasan Mini Quiz Function #1
-  Return Statement
-  Memproses Dataframe dengan Function

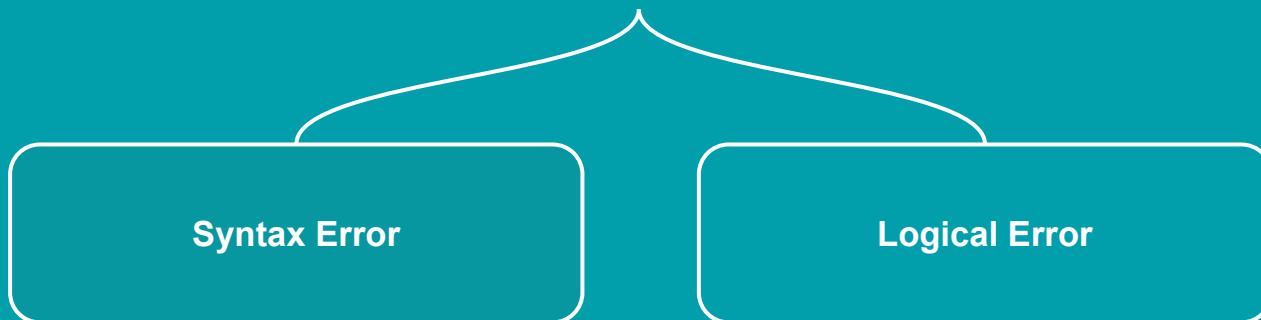
-  Memproses Dataframe dengan Function Part 2
-  Pembahasan Mini Quiz Function #2
-   Pengenalan Error dan Exception
-  Pembahasan Mini Quiz Error and Exception #3
-  Bonus - Closing Statement

# Pengenalan **Error** dan **Exception**



**“Penanganan *Error* penting untuk dilakukan agar membuat kode program lebih *robust*, sehingga menghindari potensi kegagalan yang mengakibatkan program menjadi berhenti atau tidak terkontrol ”**

# Kategori Error



# Kategori Error



# Syntax Error

Error yang terjadi saat **syntax yang dituliskan tidak sesuai** dengan standar penulisan python programming. Sehingga kode program belum berhasil dijalankan. **Contoh error:** Syntax Error dan Indentation Error

1

```
1 print'Hi! All'

File "<ipython-input-29-077be39d2fce>", line 1
print'Hi! All'
^
SyntaxError: invalid syntax
```

2

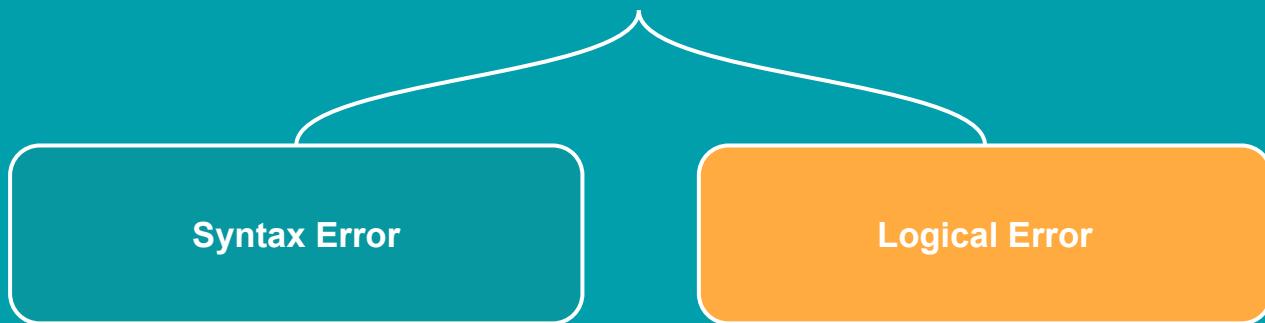
```
1 if metode == 'Penjumlahan':
2 result = a+b

File "<ipython-input-31-fcac8b070bf6>", line 2
result = a+b
^
IndentationError: expected an indented block
```

## Notes

Solusi dari perbaikan error ini adalah merubah syntax nya sehingga sesuai dengan kaidah penulisan pada Python programming

# Kategori Error



# Logical Error

Terjadinya error setelah berhasil mengeksekusi syntax test disebut exception / logical type.

**Contoh error:** IndexError, ZeroDivisionError, dll (Link [Reference](#))

1

```
1 a = [1, 2, 3, 4]
2
3 a[5]
```

-----  
**IndexError** Traceback (most recent call last)  
<ipython-input-33-788b56e163ff> in <module>  
 1 a = [1, 2, 3, 4]  
 2  
----> 3 a[5]  
  
**IndexError**: list index out of range

2

```
1 a = 1
2 b = 0
3
4 c = a/b
```

-----

```
ZeroDivisionError Traceback (most recent call last)
<ipython-input-34-4f83a7e9259d> in <module>
      2 b = 0
      3
----> 4 c = a/b

ZeroDivisionError: division by zero
```

## Notes

Solusi dari perbaikan error logical ini bisa di handle dengan **try .. except** statement

# Daftar beberapa Logical Error #1

Type	Deskripsi	Contoh Code
<b>IndexError</b>	Error terjadi saat salah memberikan nilai index dari suatu list	<pre> 1 a = [1, 2, 3, 4] 2 3 a[5]  ----- IndexError   Traceback &lt;ipython-input-33-788b56e163ff&gt; in &lt;module&gt;       1 a = [1, 2, 3, 4]       2 ----&gt; 3 a[5]  IndexError: list index out of range </pre>
<b>ZeroDivisionError</b>	Error terjadi saat suatu bilangan dibagi dengan 0	<pre> 1 a = 1 2 b = 0 3 4 c = a/b  ----- ZeroDivisionError                                     Traceback &lt;ipython-input-34-4f83a7e9259d&gt; in &lt;module&gt;       2 b = 0       3 ----&gt; 4 c = a/b  ZeroDivisionError: division by zero </pre>
<b>NameError</b>	Error terjadi saat variable yang dieksekusi belum terdefinisi	<pre> 1 print(company_name)  ----- NameError   Traceback &lt;ipython-input-35-77a0ee43dab7&gt; in &lt;module&gt; ----&gt; 1 print(company_name)  NameError: name 'company_name' is not defined </pre>

# Daftar beberapa Logical Error #2

Type	Deskripsi	Contoh Code
<b>TypeError</b>	Error terjadi saat suatu function dan operasi yang dijalankan type nya tidak sesuai. (ex. Int ditambah string)	<pre> 1 angka = 12 2 kata = 'Ammar' 3 4 angka + kata </pre> <pre> TypeError                                 Traceback (most recent &lt;ipython-input-36-4018a8df0009&gt; in &lt;module&gt;       2 kata = 'Ammar'       3 ----&gt; 4 angka + kata  TypeError: unsupported operand type(s) for +: 'int' and 'str' </pre>
<b>AttributeError</b>	Error terjadi saat suatu atribut yang diberikan pada suatu variabel tidak sesuai. (ex Int namun di append)	<pre> 1 angka = 12 2 3 angka.append(13) </pre> <pre> AttributeError                            Traceback (most &lt;ipython-input-37-7c48323ade3a&gt; in &lt;module&gt;       1 angka = 12       2 ----&gt; 3 angka.append(13)  AttributeError: 'int' object has no attribute 'append' </pre>
<b>ImportError</b>	Error terjadi saat modul yang di import tidak ditemukan, dikarenakan belum terinstall	<pre> 1 import tensorflow </pre> <pre> ModuleNotFoundError                       Traceback &lt;ipython-input-39-d6579f534729&gt; in &lt;module&gt; ----&gt; 1 import tensorflow  ModuleNotFoundError: No module named 'tensorflow' </pre>

# Handling Exception (Try - Except - Finally Statement)

## Contoh Code:

```
1 # put unsafe operation in try block
2 try:
3     print("Start")
4
5     # unsafe operation perform
6     print(1 / 0)
7
8 # if error occur the it goes in except block
9 except:
10    print("Muncul Error")
11
12 # final code in finally block
13 finally:
14    print("Finish!")
```

# Handling Exception (Try - Except - Finally Statement)

## Contoh Code:

```
1 # put unsafe operation in try block
2 try:
3     print("Start")
4
5     # unsafe operation perform
6     print(1 / 0)
7
8 # if error occur the it goes in except block
9 except:
10    print("Muncul Error")
11
12 # final code in finally block
13 finally:
14     print("Finish!")
```

```
Start
Muncul Error
Finish!
```

# Handling Exception (Try - Except - Finally Statement)

## Contoh Code:

```
1 # put unsafe operation in try block
2 try:
3     print("Start")
4
5     # unsafe operation perform
6     print(1 / 0)
7
8 # if error occur the it goes in except block
9 except:
10    print("Muncul Error")
11
12 # final code in finally block
13 finally:
14    print("Finish!")
```

Kondisi yang memungkinkan terjadi nya  
error exceptional

Kondisi penanganan  
saat ada terjadinya  
error exceptional

```
Start
Muncul Error
Finish!
```

# Handling Exception (Try - Except - Finally Statement)

## Contoh Code:

```
1 # put unsafe operation in try block
2 try:
3     print("Start")
4
5     # unsafe operation perform
6     print(1 / 0)
7
8 # if error occur the it goes in except block
9 except:
10    print("Muncul Error")
11
12 # final code in finally block
13 finally:
14     print("Finish!")
```

Optional condition

```
Start
Muncul Error
Finish!
```

# Function and Error Handling

-  Pengenalan Function
-  Parameter Input
-  Pembahasan Mini Quiz Function #1
-  Return Statement
-  Memproses Dataframe dengan Function
-  Memproses Dataframe dengan Function Part 2
-  Pembahasan Mini Quiz Function #2
-  Pengenalan Error dan Exception
-  Pembahasan Mini Quiz Error and Exception #3
-  Bonus - Closing Statement

# Function and Error Handling

-  Pengenalan Function
-  Parameter Input
-  Pembahasan Mini Quiz Function #1
-  Return Statement
-  Memproses Dataframe dengan Function

-  Memproses Dataframe dengan Function Part 2
-  Pembahasan Mini Quiz Function #2
-  Pengenalan Error dan Exception
-   Pembahasan Mini Quiz Error and Exception #3
-  Bonus - Closing Statement



Setelah berhasil melakukan pembuatan kolom **provider\_email** pada Mini Quiz sebelumnya, coba lakukan hal yang sama pada Mini Quiz #2 pada dataset **summary\_buyer\_rakamin\_store\_special.csv**!

Apakah mengalami **Error**?



Setelah berhasil melakukan pembuatan kolom **provider\_email** pada Mini Quiz sebelumnya, coba lakukan hal yang sama pada Mini Quiz #2 pada dataset **summary\_buyer\_rakamin\_store\_special.csv**!

Apakah mengalami **Error**?

Coba lakukan solve issue pembuatan kolom baru pada dataset **summary\_buyer\_rakamin\_store\_special.csv** dengan **try and except statement**!

# Function and Error Handling

-  Pengenalan Function
-  Parameter Input
-  Pembahasan Mini Quiz Function #1
-  Return Statement
-  Memproses Dataframe dengan Function
-  Memproses Dataframe dengan Function Part 2
-  Pembahasan Mini Quiz Function #2
-  Pengenalan Error dan Exception
-  Pembahasan Mini Quiz Error and Exception #3
-  Bonus - Closing Statement

# Function and Error Handling

-  Pengenalan Function
-  Parameter Input
-  Pembahasan Mini Quiz Function #1
-  Return Statement
-  Memproses Dataframe dengan Function
-  Memproses Dataframe dengan Function Part 2
-  Pembahasan Mini Quiz Function #2
-  Pengenalan Error dan Exception
-  Pembahasan Mini Quiz Error and Exception #3
-   Bonus - Closing Statement

le

# Bonus - Closing Statement



Muhammad Ammar Fauzan

<https://www.linkedin.com/in/muhammad-ammar-fauzan-748883117/>

# Function and Error Handling

-  Pengenalan Function
-  Parameter Input
-  Pembahasan Mini Quiz Function #1
-  Return Statement
-  Memproses Dataframe dengan Function
-  Memproses Dataframe dengan Function Part 2
-  Pembahasan Mini Quiz Function #2
-  Pengenalan Error dan Exception
-  Pembahasan Mini Quiz Error and Exception #3
-  Bonus - Closing Statement

le

# Danke Schön!



Muhammad Ammar Fauzan

<https://www.linkedin.com/in/muhammad-ammar-fauzan-748883117/>