

LEARNING PROGRESS REVIEW “WEEK 1”

BY : DREAM (KELOMPOK 1)
Data Realm Engineers And Maestros



ANGGOTA KELOMPOK

1

Afroh Fauziah

2

Althaf Taqiyah

3

Andrew Bintang
Pratama

4

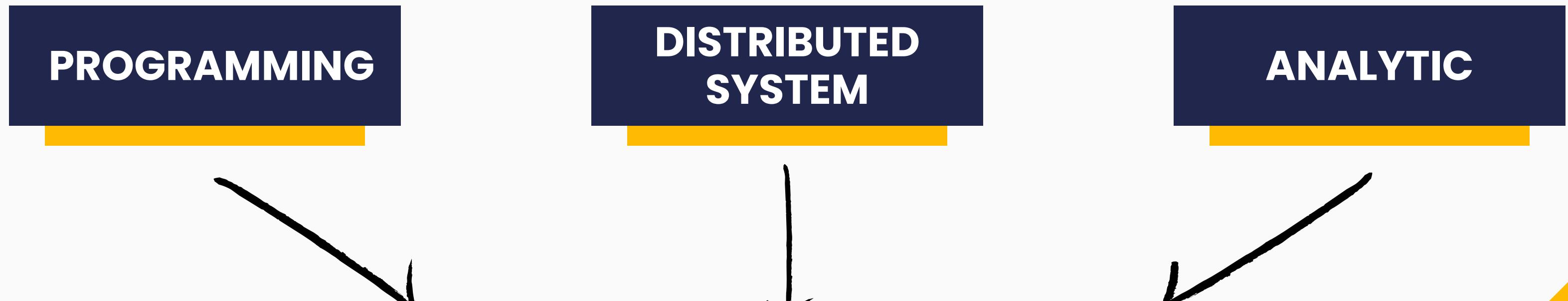
Andrew Suadnya

5

Andi Rosilala

DATA ENGINEER

- Bertugas mengatur, membuat, dan mengelola arsitektur data pada sebuah perusahaan; mendesign & membangun data pipeline dari hulu ke hilir.
- Membuat ETL / ELT untuk data transformasi juga batch & stream processing.



DATA ENGINEER

mensupport
(Data Analyst & Data Scientist)

HOW TO BECOME DATA ENGINEER

HARD SKILL

- Learn how database works
- Learn right programming language (Python, SQL, Java, etc)
- Learn cloud computing
- Learn automation tools & scripting
- Operating systems, networking

SOFT SKILL

- Conflict Management
- Project Management
- Time Management
- Business Knowledge
- Communication
- Coordination
- Story Telling



DATA ENGINEER

JOB DESC

Data Preparation

- Data Migration
- Data Ingestion
- Data Cleansing

ETL

- Extract Data
- Transforming Data
- Load Data

Architect

- Design Data Warehouse
- Design Data Lake
- Design Data Mart
- Design Data Model
- Design Data Platform Stack Architecture

Colaborate

- Create features data for Data Scientist
- Provide Adhoc data for Business & data analyst team
- Adopt & collaborate build new technologies with Backend / IT team

Managing

- Define Longterm & short term roadmap
- Hiring
- Define Team KPI
- Selecting the right tools
- Structuring an effective team
- Communicating with business Users

Governance

- Data Catalog
- Data Audit
- Build Data Convetion
- Data Quality check

DATA ENGINEER

SE (SOFTWARE ENGINEER)

Berfokus pada sistem / aplikasi khusus, berhubungan langsung kepada user

Digunakan untuk mendevelop data dari input customer

DE (DATA ENGINEER)

Berfokus pada membuat arsitektur dan automasi

Bekerja untuk membuat pipeline dari data masuk yang sudah digenerate

DATA ENGINEER

METHODOLOGY

mengcopy dari 1 file ke file yang lain

mengautomasi

TRADISIONAL WAY

database diambil
> masuk ke CSV file (download)
> masuk ke datawarehouse (upload)

MODERN WAY

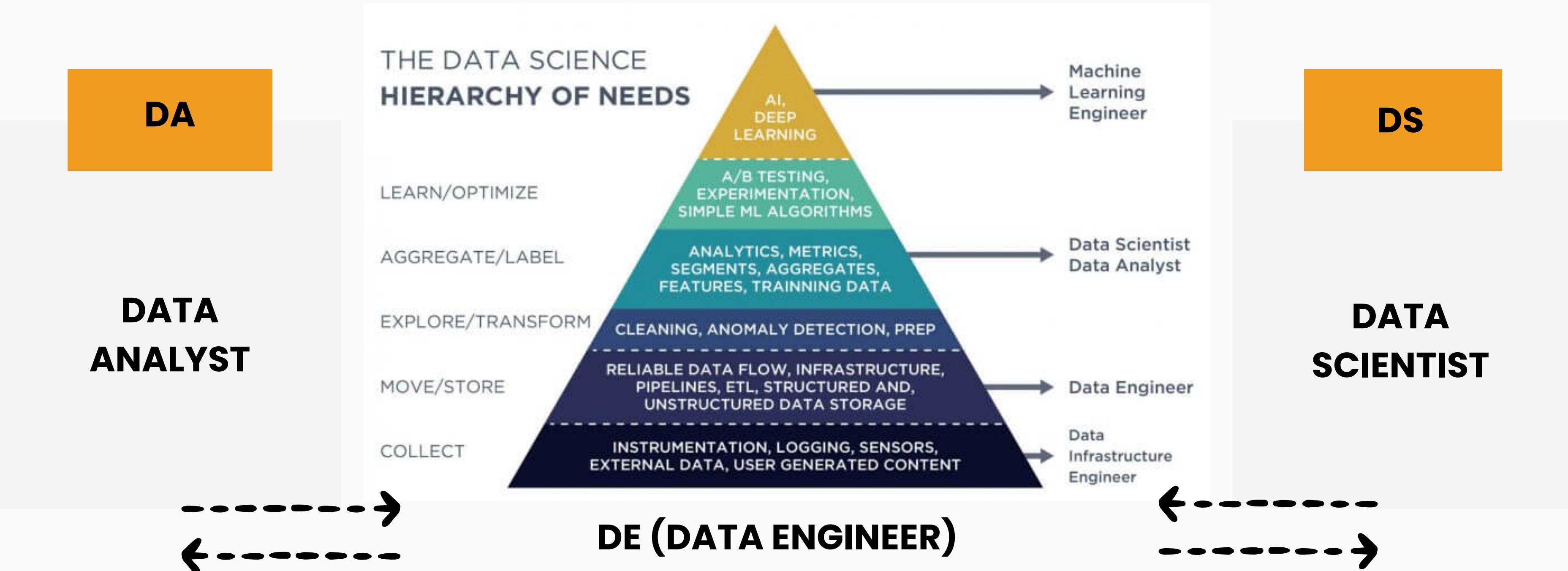
database diambil join dengan data lain
> masuk ke ETL (menextract)
> menload ke datawarehouse

ETL (Extract, Tranform, Load) bisa juga dengan ELT

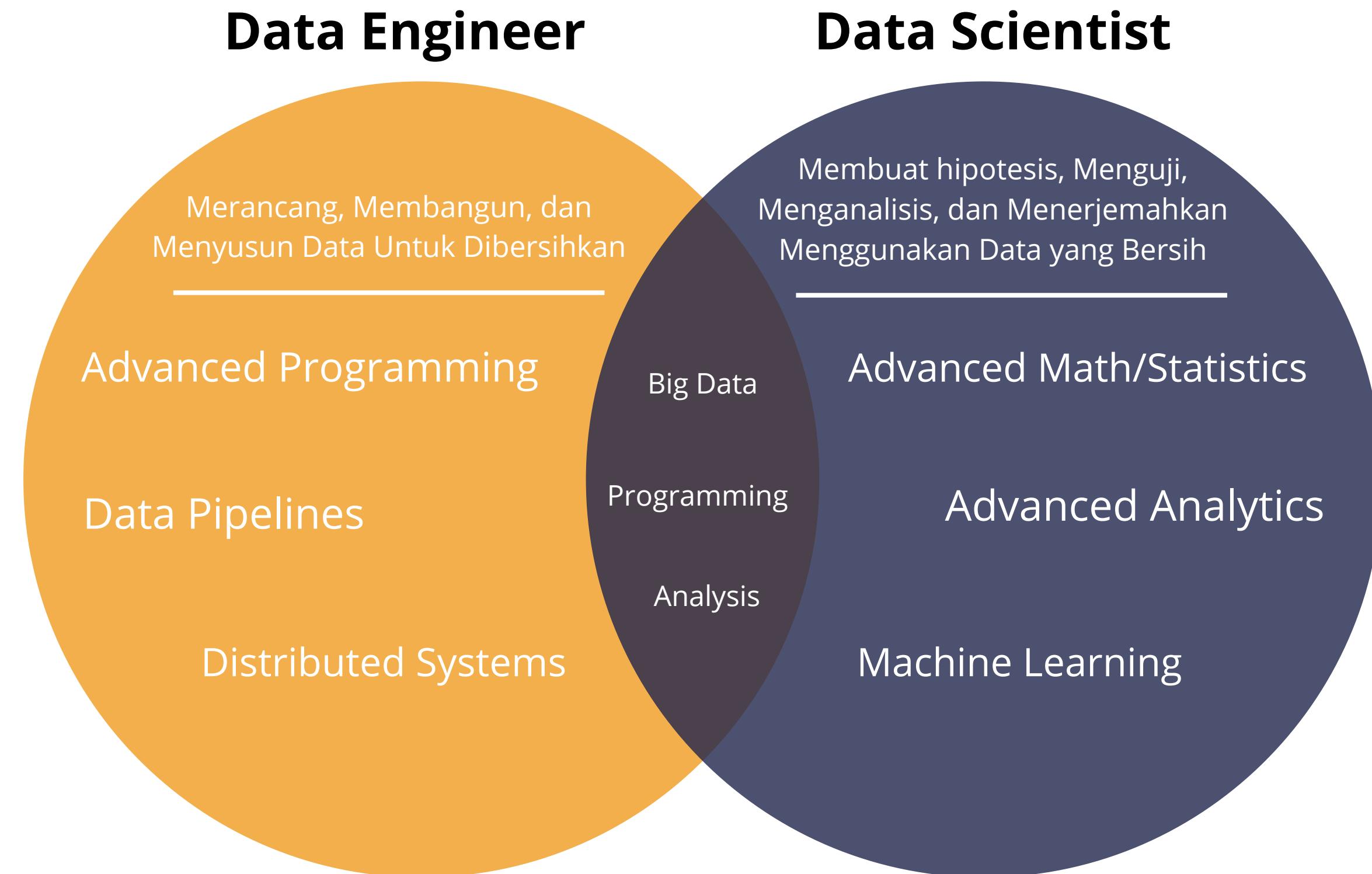
workflow



DATA TEAMS IN AN ORGANIZATION



PERBANDINGAN DATA ENGINEER & DATA SCIENTIST



BIG DATA

- Metodologi menyesuaikan zaman
- Dengan adanya perubahan, data yang semakin besar sulit diproses



- Big data muncul untuk mengatasi permasalahan
- Untuk menggabungkan data yang beragam, kompleks, dan besar dengan banyak komputer



BIG DATA TOOLS



SQL (STRUCTURED QUERY LANGUAGE)

- Data yang sudah terstruktur
- Data yg berpindah-pindah

GOOGLE CLOUD

- Data proc
- Cloud Composer

NO SQL

- Mengambil data tidak hanya 1 kolom
- Fleksibel

AWS

- Redshift
- DynamoDB

AIRFLOW, HADOOP, SPARK, HIVE, DOCKER, KAFKA



ETL & PYTHON

ETL

Di bidang data engineering terdapat istilah ETL atau ELT yang merupakan singkatan dari Extract, Transform, and Load. Untuk melakukan ETL kita memerlukan programming.

PYTHON

Bahasa yang digunakan di data engineering umumnya adalah python. Salah satu alasannya karena python mudah digunakan dan memiliki library yang lengkap khususnya terkait dengan pengolahan data.

BASIC PROGRAMMING IN PYTHON

VARIABLE

- Variabel adalah wadah yang digunakan untuk menyimpan suatu data.
- Contoh penggunaannya
`nama_kelompok = "DREAM"`
`jumlahAnggota = 5`
- `nama_kelompok` dan `jumlahAnggota` adalah contoh variabel. "DREAM" dan 5 adalah value atau nilai dari variabel tersebut.
- Salah satu sifat variable di python adalah tidak perlu mendeklarasikan tipe data.

DATA TYPE

- Tipe data dibagi menjadi 2, yaitu tipe data primitif dan tipe data non primitive.
- Primitive: contohnya adalah seperti Integer, Float, String, Boolean.
- Non-Primitive: contohnya adalah list, tuple, dictionary, set. Tipe data ini biasa disebut dengan collection, yaitu tipe data yang dapat menyimpan lebih dari satu data.

LIBRARY

- Library adalah kode yang dibuat yang dapat digunakan berulang. Python memiliki banyak sekali library yang bisa digunakan untuk berbagai keperluan.
- Jika ingin melakukan instalasi library, kita dapat menggunakan package manager seperti pip atau conda.

• • • • • • • • • • **CONTROL FLOW TYPE**

CONTROL FLOW

konsep Control Flow dalam pemrograman Python, yang mengatur bagaimana pernyataan-pernyataan dieksekusi berdasarkan kondisi tertentu.



CONTROL FLOW TYPE

SELECTION

Selection memungkinkan program untuk memilih jalur yang akan diambil berdasarkan kondisi yang diberikan. seperti : IF, IF-Else, switch.

ITERATION

memungkinkan program untuk menjalankan blok kode secara berulang selama kondisi tertentu terpenuhi. Seperti : For, While , Do-while, Foreach.

SEQUENCE

aliran kontrol yang paling sederhana di mana pernyataan dieksekusi satu per satu, dari atas ke bawah, sesuai dengan urutan yang diberikan dalam kode.

CONDITIONAL

IF

Digunakan untuk mengeksekusi blok kode jika kondisi tertentu terpenuhi.

IF-ELSE

Digunakan untuk mengeksekusi blok kode alternatif jika kondisi tidak terpenuhi.

NESTED IF-ELSE

Digunakan untuk menangani kondisi yang lebih kompleks dengan menyusun pernyataan if-else di dalam pernyataan if atau else.

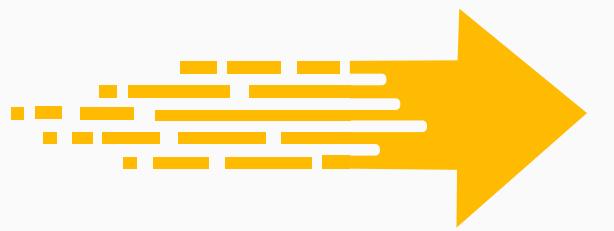
ELIF

Digunakan untuk menangani beberapa kondisi secara berturut-turut dengan menambahkan pernyataan elif (else if) setelah pernyataan if pertama.

EXAMPLE

```
1 # Contoh 1: Pernyataan If
2 x = 10
3 if x > 5:
4     print("Nilai x lebih besar dari 5")
5
6 # Contoh 2: Pernyataan If-Else
7 y = 3
8 if y % 2 == 0:
9     print("Nilai y adalah bilangan genap")
10 else:
11     print("Nilai y adalah bilangan ganjil")
12
13 # Contoh 3: Pernyataan Elif
14 nilai = 75
15 if nilai >= 90:
16     print("Nilai A")
17 elif nilai >= 80:
18     print("Nilai B")
19 elif nilai >= 70:
20     print("Nilai C")
21 elif nilai >= 60:
22     print("Nilai D")
23 else:
24     print("Nilai E")
25
26 # Contoh 4: Pernyataan Ternary Operator
27 umur = 20
28 status = "Dewasa" if umur >= 18 else "Anak-anak"
29 print("Status:", status)
30
```

```
PS C:\Users\ASUS> & C:/Python310/python.exe Untitled-1
C:/Python310/python.exe: can't open file 'C:\\\\Users\\\\ASUS\\\\Untitled-1': [Errno 2] No such file or directory
PS C:\Users\ASUS> & C:/Python310/python.exe c:/Users/ASUS/Untitled-1.py
Nilai x lebih besar dari 5
Nilai y adalah bilangan ganjil
Nilai C
Status: Dewasa
PS C:\Users\ASUS>
```



LOOPING

WHILE LOOP

perulangan "while" untuk mengeksekusi blok kode selama kondisi tertentu terpenuhi.

FOR LOOP

perulangan "for" untuk mengeksekusi blok kode secara berulang berdasarkan elemen-elemen dalam urutan atau koleksi.

NESTED LOOPS

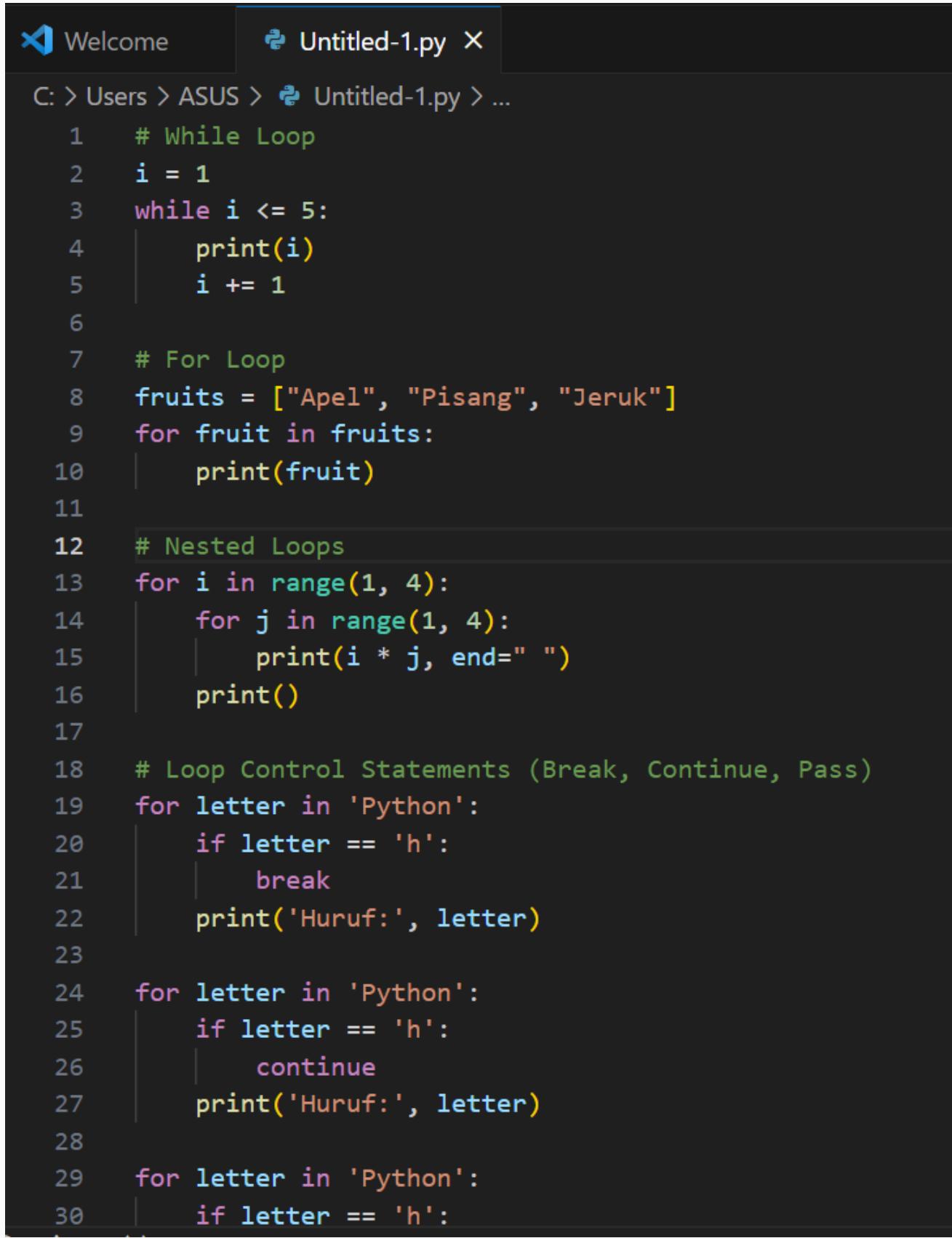
perulangan bersarang untuk menangani situasi yang lebih kompleks.

LOOP CONTROL

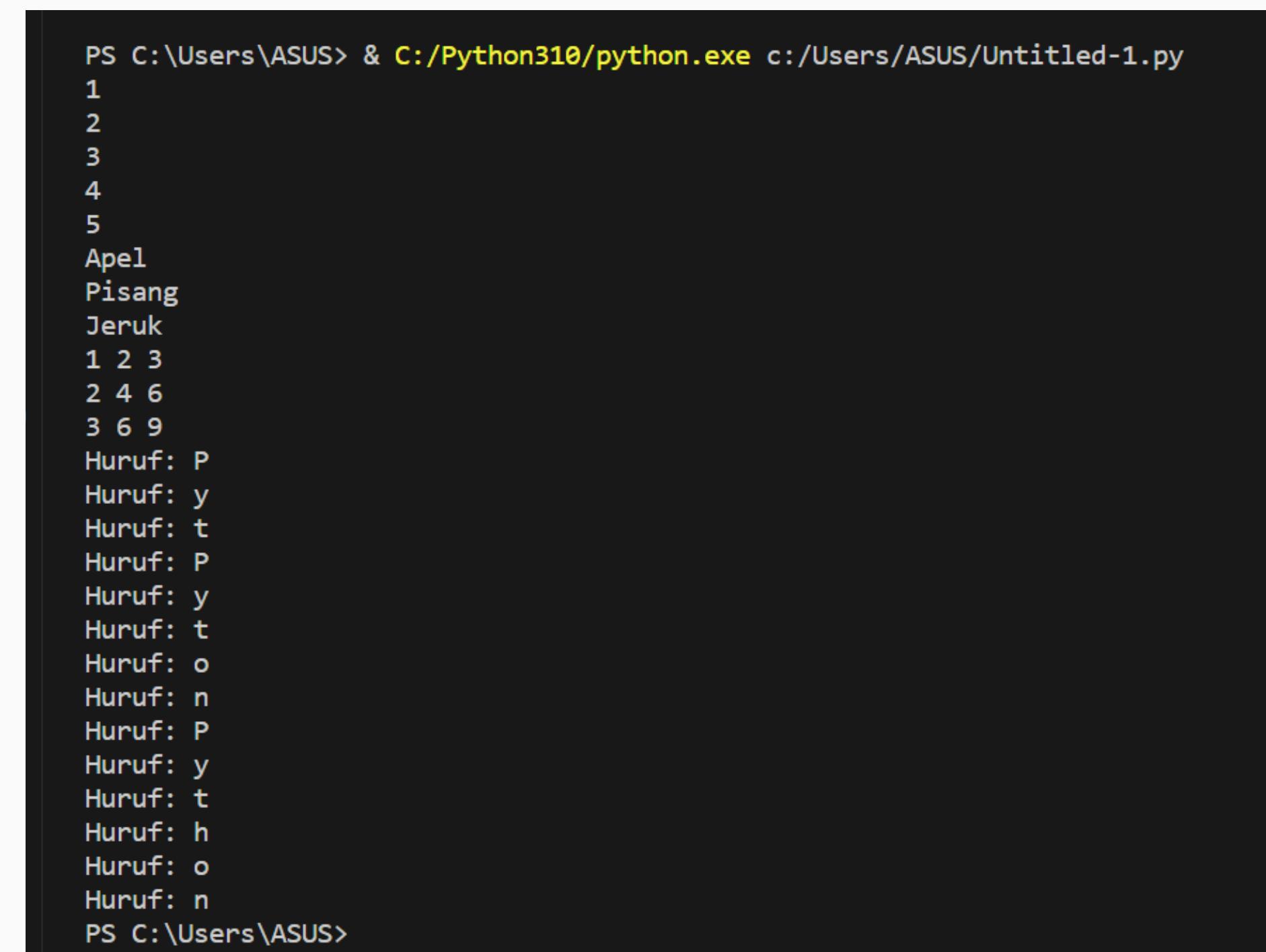
kontrol loop seperti "break", "continue", dan "pass" untuk mengontrol aliran eksekusi dalam perulangan.



EXAMPLE



```
C: > Users > ASUS > Untitled-1.py > ...
1 # While Loop
2 i = 1
3 while i <= 5:
4     print(i)
5     i += 1
6
7 # For Loop
8 fruits = ["Apel", "Pisang", "Jeruk"]
9 for fruit in fruits:
10    print(fruit)
11
12 # Nested Loops
13 for i in range(1, 4):
14     for j in range(1, 4):
15         print(i * j, end=" ")
16     print()
17
18 # Loop Control Statements (Break, Continue, Pass)
19 for letter in 'Python':
20     if letter == 'h':
21         break
22     print('Huruf:', letter)
23
24 for letter in 'Python':
25     if letter == 'h':
26         continue
27     print('Huruf:', letter)
28
29 for letter in 'Python':
30     if letter == 'h':
```



```
PS C:\Users\ASUS> & C:/Python310/python.exe c:/Users/ASUS/Untitled-1.py
1
2
3
4
5
Apel
Pisang
Jeruk
1 2 3
2 4 6
3 6 9
Huruf: P
Huruf: y
Huruf: t
Huruf: P
Huruf: y
Huruf: t
Huruf: o
Huruf: n
Huruf: P
Huruf: y
Huruf: t
Huruf: h
Huruf: o
Huruf: n
PS C:\Users\ASUS>
```

FUNCTION

Blok kode yang diberi nama dan dapat dipanggil secara berulang untuk menyelesaikan tugas tertentu. Fungsi dalam Python didefinisikan menggunakan kata kunci def, diikuti oleh nama fungsi, parameter (jika ada), dan blok kode yang menjalankan tugas yang diinginkan. Fungsi dapat mengembalikan nilai menggunakan kata kunci return.

DEF

Mendefinisikan sebuah fungsi. Ini memberitahu Python bahwa kita akan membuat sebuah fungsi.

FUNCTION_NAME

Menggambarkan tugas atau tujuan dari fungsi tersebut.

PARAMETERS (ARGUMENTS)

Nilai yang diterima oleh fungsi untuk melakukan tugas tertentu. Parameter bersifat opsional, tergantung pada kebutuhan dari fungsi yang dibuat.

TITIK DUA (:

Digunakan setelah nama fungsi dan parameter (jika ada) untuk menandakan bahwa blok kode fungsi akan dimulai.

DOKUMENTASI (DOCSTRING)

Komentar yang menjelaskan tujuan atau fungsionalitas dari fungsi tersebut.

RETURN

Digunakan untuk mengembalikan nilai dari fungsi.



EXAMPLE

```
def derajat0 (img):
    # Menghitung nilai piksel maksimum dalam citra
    max = np.max(img)

    # Inisialisasi matriks nol dengan ukuran (max+1) x (max+1)
    imgTmp=np.zeros([max+1,max+1])

    # Iterasi melalui piksel-piksel citra
    for i in range (len(img)):
        for j in range (len(img[i])-1):

            # Menghitung pasangan nilai intensitas piksel dan menyimpannya dalam imgTmp
            imgTmp[img[i,j],img[i,j+1]] +=1
```

Fungsi def digunakan untuk mendefinisikan sebuah fungsi. Dalam konteks kode yang Anda berikan, fungsi def derajat0(img): digunakan untuk melakukan pengolahan terhadap citra yang diberikan sebagai argumen img.

3 PARADIGMA PEMROGRAMAN YANG POPULER

PARADIGMA PROSEDURAL

Melakukan pengelompokan satu tugas tertentu yang bisa digunakan berkali-kali sebagai pendekatan dalam pemecahan suatu masalah.

PARADIGMA FUNGSIONAL

Hampir sama dengan prosedural, hanya saja paradigma ini lebih berorientasi terhadap “input-output” dari pada mengubah data secara langsung seperti pada paradigma prosedural.

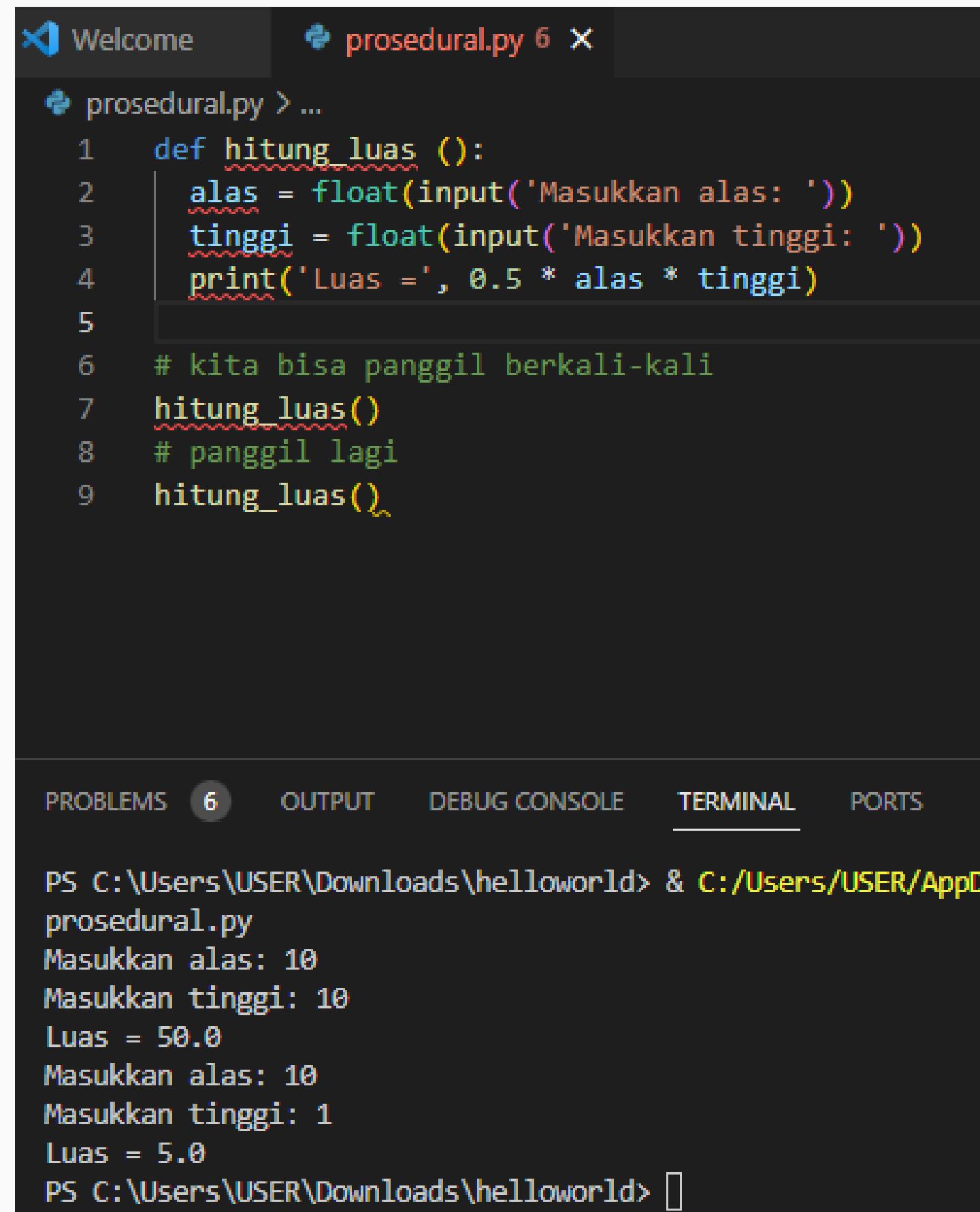
OBJECT ORIENTED PROG

Menjadikan semua komponen program sebagai objek. Yang mana setiap objek memiliki identitas dan tugasnya masing-masing.

FUNGSI VS PROSEDUR

Fungsi adalah fungsi yang mengembalikan nilai.
Prosedur adalah fungsi yang tidak mengembalikan nilai.

EXAMPLE PROSEDURAL



The screenshot shows a code editor window titled "Welcome" with a tab for "prosedural.py 6". The code defines a function "hitung_luas" that prompts the user for the base and height of a triangle, calculates the area, and prints it. It then demonstrates calling the function twice. The terminal below shows the execution of the script and its output.

```
1 def hitung_luas():
2     alas = float(input('Masukkan alas: '))
3     tinggi = float(input('Masukkan tinggi: '))
4     print('Luas =', 0.5 * alas * tinggi)
5
6 # kita bisa panggil berkali-kali
7 hitung_luas()
8 # panggil lagi
9 hitung_luas()

PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\USER\Downloads\helloworld> & C:/Users/USER/AppD
prosedural.py
Masukkan alas: 10
Masukkan tinggi: 10
Luas = 50.0
Masukkan alas: 10
Masukkan tinggi: 1
Luas = 5.0
PS C:\Users\USER\Downloads\helloworld>
```

EXAMPLE FUNGSIONAL

The screenshot shows a code editor interface with two tabs: 'procedural.py' (6 lines) and 'fungsional.py' (9 lines). The 'fungsional.py' tab is active, displaying the following code:

```
procedural.py 6          fungsional.py 9 X
fungsional.py > ...
1 def input_alas_dan_tinggi():
2     alas = float(input('Masukkan alas: '))
3     tinggi = float((input('Masukkan tinggi: ')))
4
5     return alas, tinggi
6
7 def hitung_luas(alas, tinggi):
8     return 0.5 * alas * tinggi
9
10 """
11     kalau fungsional, kita sendiri yang mengelola
12     hasil kembalinya"""
13
14 # satu fungsi bisa dipanggil secara independen
15 print(hitung_luas(5, 10))
16
17 # contoh dengan inputan alas dan tinggi
18 alas, tinggi = input_alas_dan_tinggi()
19 print(hitung_luas(alas, tinggi))
```

Below the code editor, there are navigation tabs: PROBLEMS (15), OUTPUT, DEBUG CONSOLE, TERMINAL (underlined), and PORTS.

In the terminal window, the following output is shown:

```
PS C:\Users\USER\Downloads\helloworld> & C:/Users/USER/AppData/Local
py
25.0
Masukkan alas: 5
Masukkan tinggi: 4
10.0
PS C:\Users\USER\Downloads\helloworld>
```



OBJECT ORIENTED PROG

BASIC

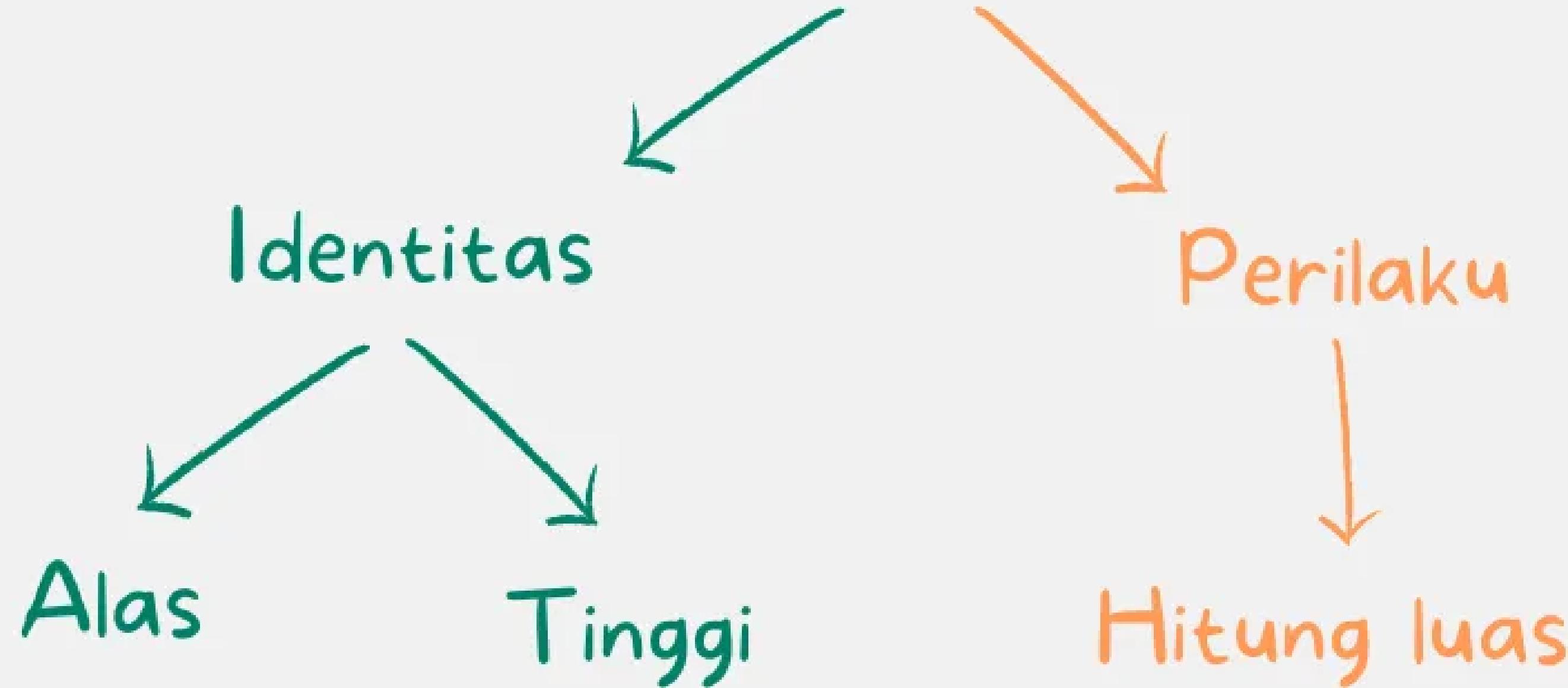
Pada konsep OOP, kita akan membuat semua komponen program seolah-olah adalah sebuah objek. Sebuah objek selalu memiliki identitas dan juga perilaku atau kemampuan untuk melakukan tugas tertentu.

Kita ambil contoh saja program yang ada di atas: cara menghitung segitiga. Maka cara berpikir kita akan seperti ini:

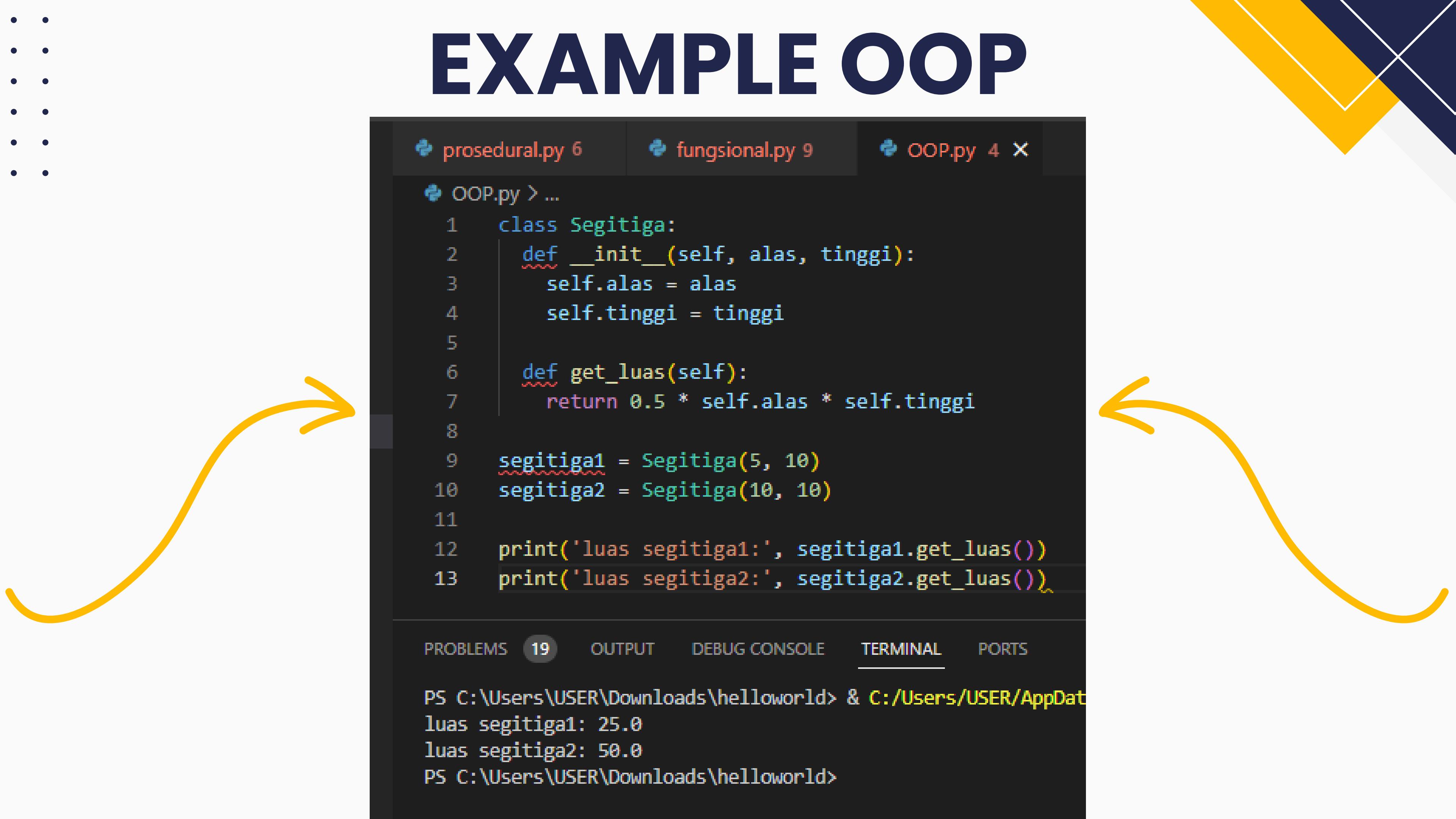
1. Segitiga adalah sebuah objek.
2. Objek segitiga memiliki 2 identitas berupa alas dan tinggi.
3. Objek segitiga memiliki "kemampuan" untuk menghitung luasnya sendiri.



Objek Segitiga



EXAMPLE OOP



The screenshot shows a code editor interface with several tabs at the top: 'procedural.py 6', 'fungisional.py 9', 'OOP.py 4 X', and 'OOP.py > ...'. The active tab is 'OOP.py > ...'. The code itself is as follows:

```
1  class Segitiga:
2      def __init__(self, alas, tinggi):
3          self.alas = alas
4          self.tinggi = tinggi
5
6      def get_luas(self):
7          return 0.5 * self.alas * self.tinggi
8
9  segitiga1 = Segitiga(5, 10)
10 segitiga2 = Segitiga(10, 10)
11
12 print('luas segitiga1:', segitiga1.get_luas())
13 print('luas segitiga2:', segitiga2.get_luas())
```

At the bottom of the editor, there are tabs for 'PROBLEMS' (with a count of 19), 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The 'TERMINAL' tab is currently selected. The terminal output window shows the following text:

```
PS C:\Users\USER\Downloads\helloworld> & C:/Users/USER/AppData
luas segitiga1: 25.0
luas segitiga2: 50.0
PS C:\Users\USER\Downloads\helloworld>
```

CLASS, INSTANCE, ATRIBUT & METHODS

CLASS

Cetak biru untuk membuat objek.

INSTANCE

Objek yang dibuat dari class.

ATRIBUT

Data yang dimiliki oleh instance.

METHODS

Fungsi yang dapat dilakukan oleh instance.



EXAMPLE CODE

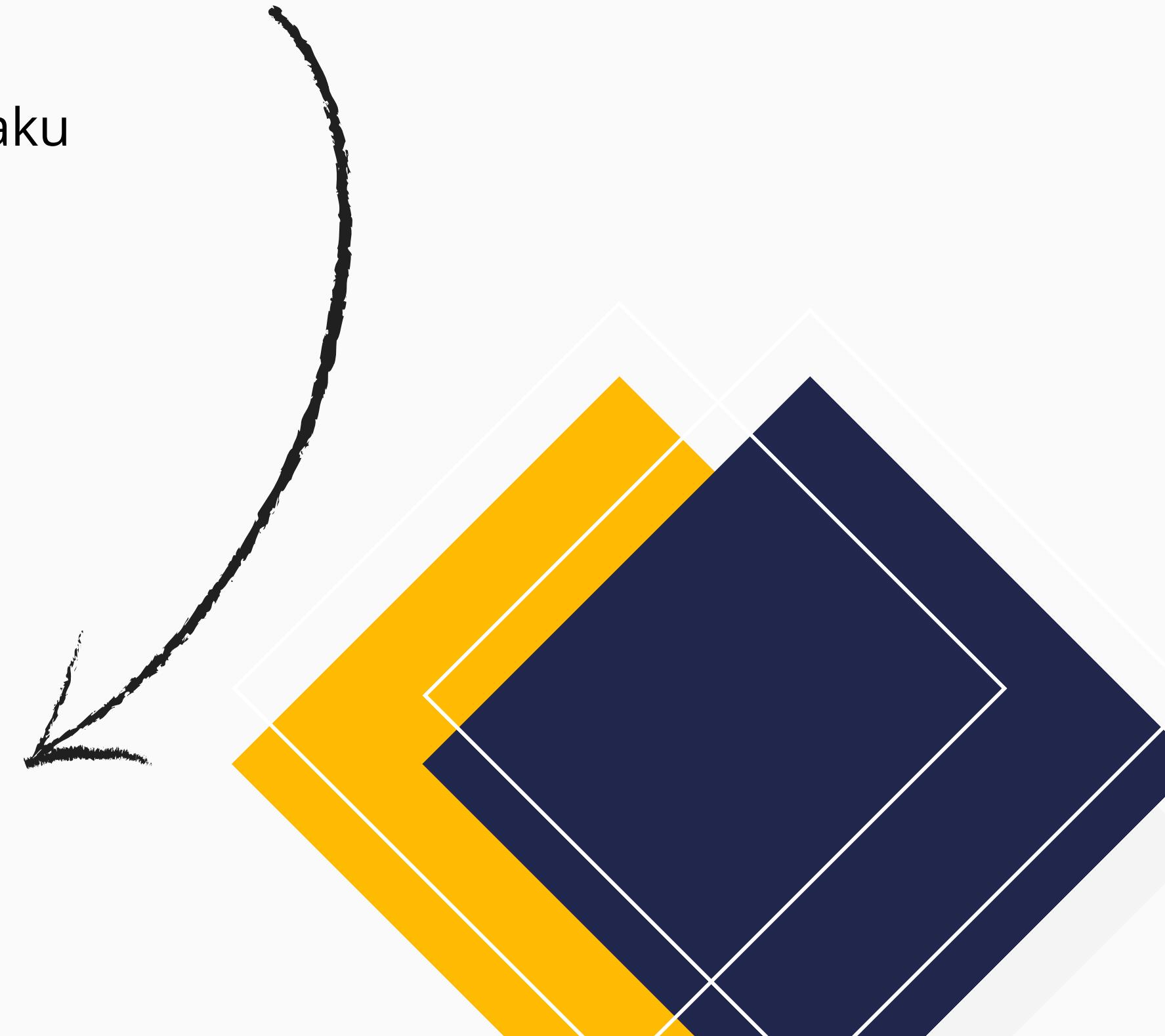
```
➊ class instance atribut method.py X  
➋ class instance atribut method.py  
➌ 1 ✓ class Animal:  
➍ 2 ✓     def __init__(self, name, species):  
➎ 3         self.name = name  
➏ 4         self.species = species  
➐ 5  
➑ 6 ✓     def speak(self):  
➒ 7         print(f"Saya {self.name}, seekor {self.species}")  
➓ 8  
➔ 9     # Membuat instance  
➕ 10    dog = Animal("Rex", "Anjing")  
➖ 11  
➗ 12    # Mengakses atribut  
➘ 13    print(dog.name) # Output: Rex  
➙ 14  
➚ 15    # Memanggil method  
➛ 16    dog.speak() # Output: Saya Rex, seekor Anjing
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\USER\Downloads\helloworld> & C:/Users/USER/AppData/Local  
rld/class instance atribut method.py"  
Rex  
Saya Rex, seekor Anjing  
PS C:\Users\USER\Downloads\helloworld> []
```

INHERITANCE

- 1 Mekanisme untuk mewariskan properti dan perilaku dari class ke class lain.
- 2 Meningkatkan reusability kode.
- 3 Class turunan mewarisi atribut dan methods dari class induk.
- 4 Inheritance memungkinkan untuk membuat hierarki class.



EXAMPLE CODE



```
class Animal:
    def __init__(self, name, species):
        self.name = name
        self.species = species

    def speak(self):
        print(f"Saya {self.name}, seekor {self.species}")

class Dog(Animal):
    def bark(self):
        print("Gukguk!")

# Membuat instance
dog = Dog("Rex", "Anjing")

# Mengakses method dari class induk
dog.speak() # Output: Saya Rex, seekor Anjing
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\USER\Downloads\helloworld> & C:/Users/USER/AppData/Local/P
.py
Saya Rex, seekor Anjing
Gukguk!
PS C:\Users\USER\Downloads\helloworld>
```



PYTHON: OVERRIDING (PENIMPAAN)

- Teknik overriding adalah fitur yang memungkinkan kita untuk mengimplementasikan "ulang" fungsi atau metode pada sebuah child class atau kelas turunan yang sebenarnya fungsi tersebut telah didefinisikan di dalam parent class atau kelas induk .

CONTOH OVERRIDING

Misalkan punya dua buah kelas yaitu Kendaraan dan Mobil di mana:

- Kelas Kendaraan adalah kelas induk
- Kelas Mobil adalah kelas turunan dari kelas Kendaraan
- Kelas Kendaraan punya kemampuan (fungsi) berjalan()
- Akan tetapi kita ingin bahwa kelas Mobil punya perilaku khusus untuk fungsi berjalan().

EXAMPLE CODE OVERRIDING

```
class Kendaraan:  
    def berjalan(self):  
        print('berjalan..')  
  
class Mobil(Kendaraan):  
    pass  
  
sepeda = Kendaraan()  
sedan = Mobil()  
  
sepeda.berjalan()  
sedan.berjalan()
```

OUTPUT:

berjalan..
berjalan..

Jika perhatikan lagi output di samping maka output-nya identik, tidak ada yang berbeda satu huruf pun. Itu karena kelas Mobil menurunkan semua hal dari kelas Kendaraan tanpa mengubah satu hal apapun.

MELAKUKAN OVERRIDING

Langkah berikutnya adalah melakukan overriding fungsi berjalan(), agar aksinya menjadi berbeda ketika dipanggil dari kelas turunan yaitu kelas Mobil. Ubah kode programnya menjadi seperti ini:

```
class Kendaraan:  
    def berjalan(self):  
        print('berjalan..')  
  
class Mobil(Kendaraan):  
    def berjalan(self):  
        print('Berjalan dengan cepat..')  
  
sepeda = Kendaraan()  
sedan = Mobil()  
  
sepeda.berjalan()  
sedan.berjalan()
```

OUTPUT:

```
berjalan..  
Berjalan dengan cepat..
```

PYTHON: ACCESS MODIFIERS (ENKAPSULASI)

Access Modifiers adalah sebuah konsep di dalam pemrograman berorientasi objek di mana kita bisa mengatur hak akses suatu atribut dan fungsi pada sebuah class. Konsep ini juga biasa disebut sebagai enkapsulasi, di mana kita akan mendefinisikan mana atribut atau fungsi yang boleh diakses secara terbuka, mana yang bisa diakses secara terbatas, atau mana yang hanya bisa diakses oleh internal kelas alias privat.

JENIS ACCESS MODIFIERS PADA PYTHON

1. PUBLIC ACCESS MODIFIER

Variabel atau atribut yang memiliki hak akses publik bisa diakses dari mana saja baik dari luar kelas maupun dari dalam kelas.

2. PROTECTED ACCESS MODIFIER

Variabel atau atribut yang memiliki hak akses protected hanya bisa diakses secara terbatas oleh dirinya sendiri (yaitu di dalam internal kelas), dan juga dari kelas turunannya.

3. PRIVATE ACCESS MODIFIER

Modifier selanjutnya adalah private. Setiap variabel di dalam suatu kelas yang memiliki hak akses private maka ia hanya bisa diakses di dalam kelas tersebut. Tidak bisa diakses dari luar bahkan dari kelas yang mewarisinya. Untuk membuat sebuah atribut menjadi private, kita harus menambahkan dua buah underscore sebagai prefix nama atribut



PYTHON: OPERATOR OVERLOADING

PENGERTIAN OVERLOADING

Overloading dalam dunia pemrograman adalah teknik untuk mengatur berbagai perilaku dari suatu fungsi berdasarkan dengan parameter yang diterimanya atau perilaku objek berdasarkan dengan operator yang sedang dioperasikan

Operator overloading adalah teknik di mana kita akan mengatur atau mendefinisikan perilaku sebuah kelas—yang kita buat, bagaimana ia akan berinteraksi dengan berbagai macam operator yang berbeda. contoh operator +.

EXAMPLE CODE OVERLOADING

```
class Angka:  
    def __init__(self, angka):  
        self.angka = angka  
  
    def __add__(self, objek):  
        return Angka(  
            self.angka + objek.angka  
        )  
  
x1 = Angka(5)  
x2 = Angka(20)  
x3 = x1 + x2  
print(x3.angka)
```

OUTPUT:

25



DAFTAR FUNGSI UNTUK OPERATOR ARITMATIKA

Operator	Nama Fungsi
+	<code>__add__()</code>
-	<code>__sub__()</code>
*	<code>__mul__()</code>
/	<code>__truediv__()</code>
//	<code>__floordiv__()</code>
%	<code>__mod__()</code>
**	<code>__pow__()</code>
>>	<code>__rshift__()</code>
<<	<code>__lshift__()</code>
&	<code>__and__()</code>
	<code>__or__()</code>
^	<code>__xor__()</code>

DAFTAR FUNGSI UNTUK OPERATOR KOMPARASI

Operator	Nama Fungsi
<	<code>__lt__()</code>
>	<code>__gt__()</code>
<=	<code>__le__()</code>
>=	<code>__ge__()</code>
==	<code>__eq__()</code>
!=	<code>__ne__()</code>

DAFTAR FUNGSI UNTUK OPERATOR ASSIGNMENT

Operator	Nama Fungsi
-=	<code>__isub__()</code>
+=	<code>__iadd__()</code>
*=	<code>__imul__()</code>
/=	<code>__idiv__()</code>
//=	<code>__ifloordiv__()</code>
%=	<code>__imod__()</code>
**=	<code>__ipow__()</code>
>>=	<code>__irshift__()</code>
<<=	<code>__ilshift__()</code>
&=	<code>__iand__()</code>
=	<code>__ior__()</code>
^=	<code>__ixor__()</code>

TERIMAKASIH

ATAS PERHATIANNYA

Mimpi yang diharapkan menjadi kenyataan
- DREAM



Table of Content

What will We Learn Today?

1. What is Data Engineer?
2. Data Engineer Methodology.
3. Data Engineer Workflow.
4. Data Engineer & Big Data.
5. Big Data Tools.
6. Data Teams In An Organization.
7. Conclusion.

DigitalSkola
Uncover The World Of Digital Skills With Us



© Copyright by Digital Skola 2022





Press esc to exit full screen



Table of Content

What will We Learn Today?

1. Apa Itu Programming ?
2. Kenapa Menggunakan Python ?
3. Apa itu Variable ?
4. Apa itu Data Type ?
5. Apa itu collection ?
6. Import Statement & Library Module
7. Menggunakan Module





4. Python III - Google Driv Copy of Python2.pptx Python2.ipynb - Colab Notebooks - Goog Colab Notebooks - Goog Python loops: Some beg Python Programs to Print Get Started: 3 Ways to L Python1.pptx - Google Slides

docs.google.com/presentation/d/1NoPKLg3D-IFNgSQpXwiWPjYkAE9w47bz/edit#slide=id.p2

Python2 .PPTX Slideshow Share Syarif Hidayatullah

View Insert Format Slide Arrange Tools Help

Fit Background Layout Theme Transition

1 2 3 4 5 6 7 8 9 10 11 12 13

DigitalSkola Python II: Loops & Conditional

Table of Content What will We Learn Today?

Control Flow Overview Control Flow Type Macam - Macam Control Flow

1. Control Flow Type
2. Conditional
3. Looping
4. Function

Copyright by Digital Skola 2022

Click to add speaker notes

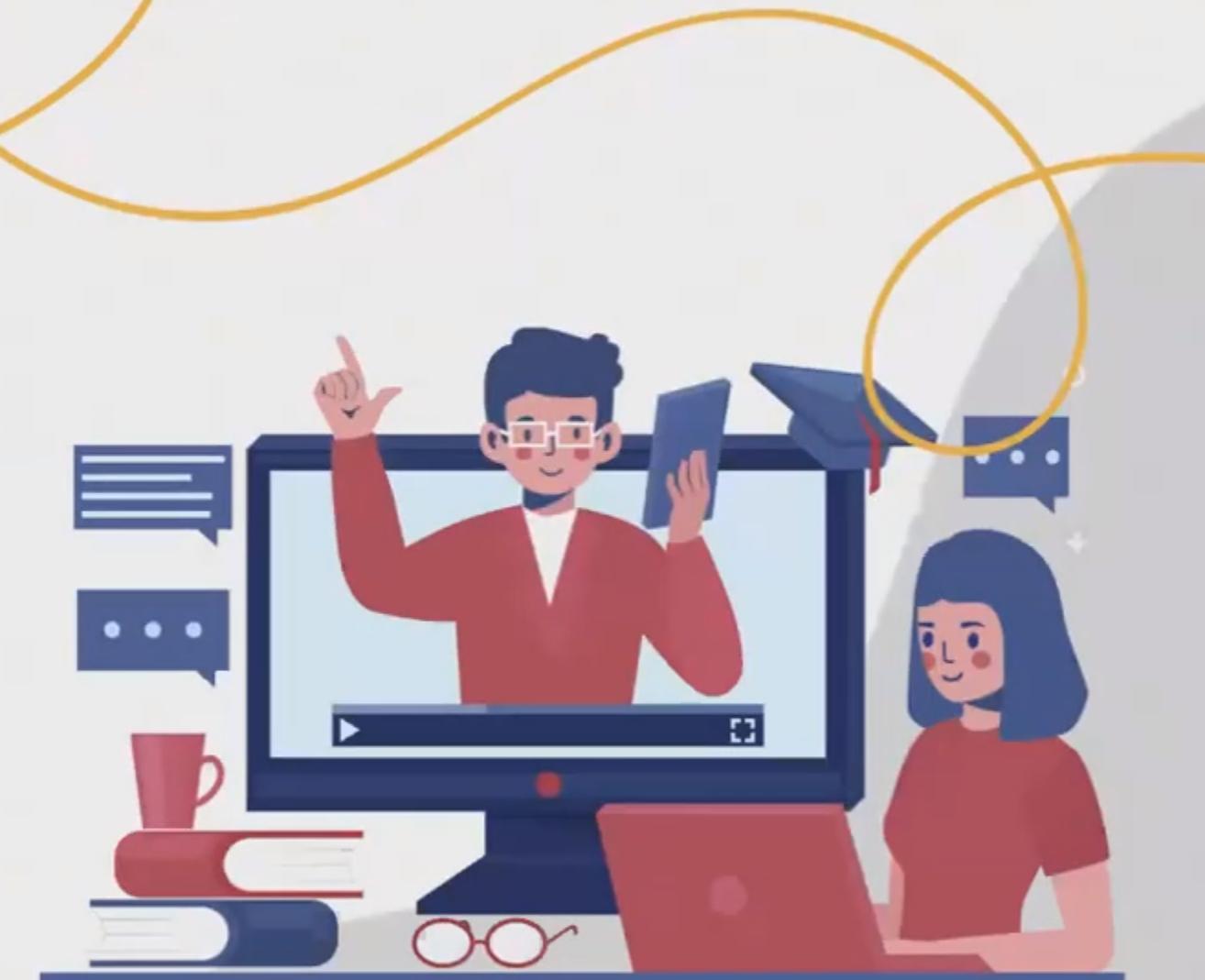




Table of Content

What will We Learn Today?

1. OOP
2. Why OOP?
3. Class
4. Instance
5. Atributes & Methods
6. Inheritance, Override, Overloading





digitalSkola [WSL: Debian]

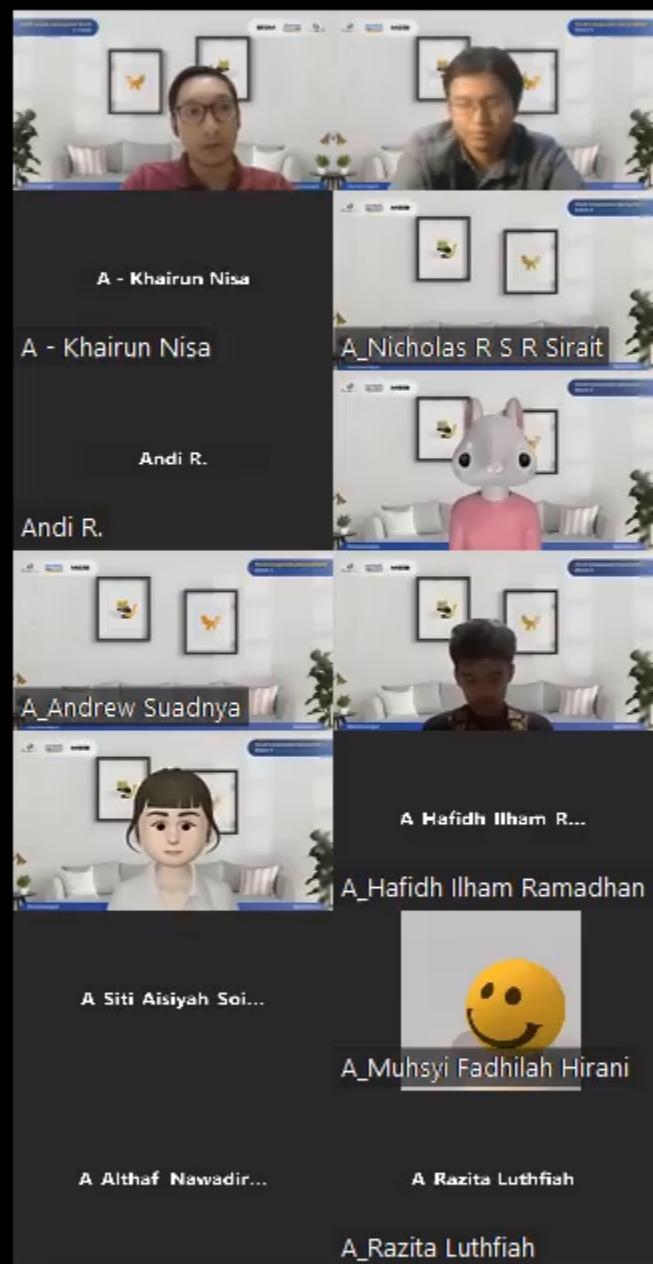
oop.py

```
1 # Library management system
2 - Books, Member,|
3
4 class Book:
5     def __init__(self, title, author, ISBN, available=True):
6         self.title = title
7         self.author = author
8         self.ISBN = ISBN
9         self.available = available
10
11     def checkout(self):
12         self.available = False
13
14     def return_book(self):
15         self.available = True
16
17 class EBook(Book):
18     def __init__(self, title, author, ISBN, format):
19         super().__init__(title, author, ISBN) # Inherit from Book
20         self.format = format
21
22     def download(self):
23         print(f"Downloading E-book: {self.title} in {self.format} format")
24
25 class Member:
26     def __init__(self, name, member_id):
27         self.name = name
28         self.member_id = member_id
29         self.books_checked_out = []
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

bash - digitalSkola

agung@Agung-Work:~/digitalSkola\$



```
j-Work: ~/digit  ×  +  ▾  
  files:  
    add <file>..." to include in what will be committed)  
meet_1.py  www.bing.com  
oop.py  
  added to commit but untracked files present (use "git add" to track)  
Agung-Work:~/digitalSkola$ vi meet_1.py  
Agung-Work:~/digitalSkola$ git status  
branch master  
  no commits yet  
  git conflict resolve  
  untracked files:  
    add <file>..." to include in what will be committed)  
meet_1.py  
oop.py  
  added to commit but untracked files present (use "git add" to track)  
Agung@Agung-Work:~/digitalSkola$ git add .  
Agung@Agung-Work:~/digitalSkola$ git status  
branch master  
  no commits yet  
  git conflict resolve  
Changes to be committed:  
(use "git rm --cached <file>..." to unstage)  
  new file:  meet_1.py  
  new file:  oop.py  
Github  Manage Your Search History  
Agung@Agung-Work:~/digitalSkola$ git commit -m "add file"
```