



LEARNING PROGRESS REVIEW

Week 3

Entropy Team

OUR TEAM

Entropy Team



Adhang Muntaha Muhammad

<https://www.linkedin.com/in/adhangmuntaha/>



Aziz Fauzi

<https://www.linkedin.com/in/aziz-fauzi-a6904711b/>



Iwan Wahyu

<https://www.linkedin.com/in/iwan-wahyu-setyawan-506809183>



Marcellina Alvita F

<https://www.linkedin.com/in/marcellina-alvita-faustina-63a284226>



Ramadhan Luthfan

<https://www.linkedin.com/in/luthfan-mahathir-91369b18b>

DAFTAR ISI

1.

Advanced SQL

Materi SQL lanjutan
(PostgreSQL)

2.

Versioning/ Version Control

Version control dengan
Git dan GitHub

3.

Introduction to Python and Programming

Pengenalan pemrograman
dan bahasa Python

01

ADVANCED SQL

Materi SQL lanjutan
(PostgreSQL)

Function – CURRENT_DATE

- Digunakan untuk mendapatkan data **tanggal saat ini**
- Format:
YYYY-MM-DD

sekarang	besok	kemarin
2022-01-20	2022-01-21	2022-01-19

Syntax

```
SELECT CURRENT_DATE
```

Contoh

```
SELECT  
CURRENT_DATE sekarang,  
(CURRENT_DATE + 1) besok,  
(CURRENT_DATE - 1) kemarin
```

Function – CURRENT_TIME

- Digunakan untuk mendapatkan data **waktu saat ini**
- Format:
HH:MM:SS

sekarang	nanti	tadi
14:43:19	16:43:19	14:13:19

Syntax

```
SELECT CURRENT_TIME
```

Contoh

```
SELECT
```

```
CURRENT_TIME sekarang,  
(CURRENT_TIME + INTERVAL '2 HOURS') nanti,  
(CURRENT_TIME - INTERVAL '30 MINUTES') tadi
```

Function – CURRENT_TIMESTAMP

- Digunakan untuk mendapatkan data **tanggal** dan **waktu saat ini**
- Format:
YYYY-MM-DD HH:MM:SS

sekarang

2022-01-20 14:55:14.701

Syntax

SELECT CURRENT_TIME

Contoh

SELECT CURRENT_TIMESTAMP sekarang

Function – EXTRACT

- Digunakan untuk **mengekstrak** suatu bagian (*subfield*) dari data **waktu** (*timestamp*)

waktu	jam	tahun
2022-01-20 15:00:54.815	15	2022

Syntax

SELECT EXTRACT(subfield FROM timestamp)

Contoh

SELECT

CURRENT_TIMESTAMP waktu,
EXTRACT('hour' FROM CURRENT_TIME) jam,
EXTRACT('year' FROM CURRENT_DATE) tahun

Function – DATE_TRUNC

- Digunakan untuk **memotong** data **waktu** (*timestamp*) sesuai bagian (*subfield*) yang ditentukan
- Misal dipotong berdasarkan '**jam**', maka data **menit** dan **detik** akan jadi **nilai terkecil**

waktu	jam	tahun
2022-01-20 15:06:36.404	2022-01-20 15: 00:00.000	2022- 01-01 00:00:00.000

Syntax

```
SELECT DATE_TRUNC(subfield, timestamp)
```

Contoh

```
SELECT  
    CURRENT_TIMESTAMP waktu,  
    DATE_TRUNC('hour', CURRENT_TIMESTAMP) jam,  
    DATE_TRUNC('year', CURRENT_TIMESTAMP) tahun
```

Key



- Key adalah **atribut** (nilai pada suatu kolom) yang dapat digunakan untuk **mengidentifikasi baris** pada sebuah tabel
- Key juga dapat digunakan untuk menentukan **relasi antartabel**

Super Key

Tabel Mahasiswa
NIM
nama
jenis_kelamin
tanggal_lahir
alamat
NIK
no_hp
email
tanggal_daftar

- **Sebuah** atribut atau **kumpulan atribut** yang dapat digunakan untuk **mengidentifikasi** setiap baris menjadi **unique**
- **Superset** dari **candidate key**
- Contoh:
(NIM)
(NIM, NIK), key berupa gabungan dari NIM dan NIK

Candidate Key

Tabel Mahasiswa
NIM
nama
jenis_kelamin
tanggal_lahir
alamat
NIK
no_hp
email
tanggal_daftar

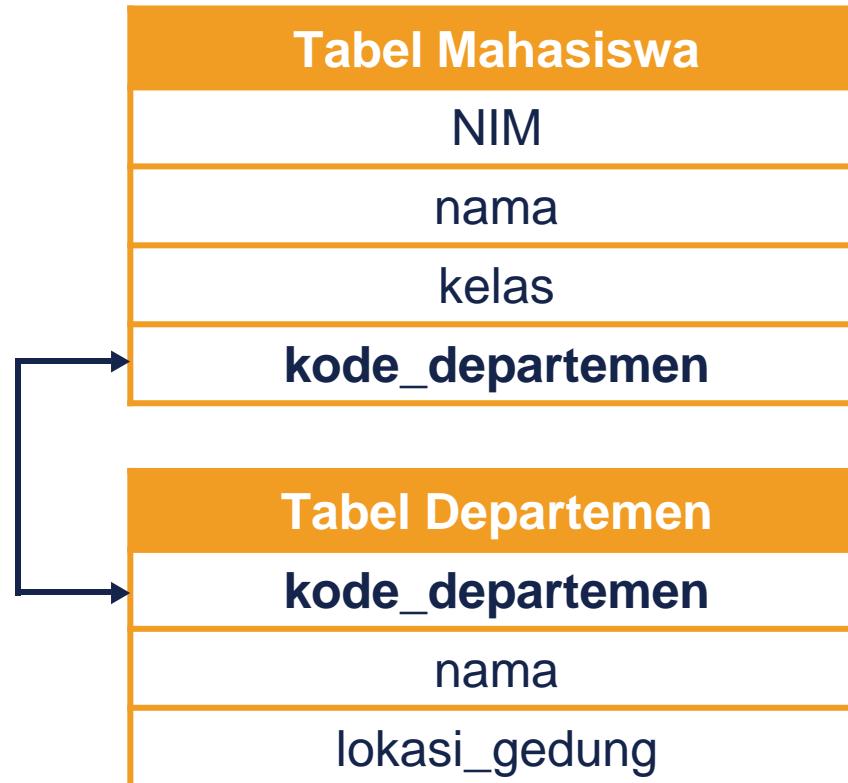
- Kumpulan **atribut paling sedikit** yang dapat digunakan untuk **mengidentifikasikan** setiap baris menjadi **unique**
- Contoh:
(NIM)
(NIK)
(no_hp)
(email)

Primary Key

Tabel Mahasiswa	
NIM	
nama	
jenis_kelamin	
tanggal_lahir	
alamat	
NIK	
no_hp	
email	
tanggal_daftar	

- Salah **satu** bagian dari **candidate key** yang **paling cocok** digunakan untuk membedakan data antarbaris
- Contoh:
NIM, karena lebih cocok dijadikan *primary key* untuk membedakan data antarmahasiswa

Foreign Key



- Atribut yang digunakan untuk **membuat hubungan** antara 2 **tabel**
- Contoh:
kode_departemen menjadi *foreign key* pada tabel mahasiswa, dan menjadi *primary key* pada tabel departemen

Secondary/ Alternative Key

Tabel Mahasiswa
NIM
nama
jenis_kelamin
tanggal_lahir
alamat
NIK
no_hp
email
tanggal_daftar

- Atribut dari **candidate key** yang **tidak dijadikan** sebagai **primary key**
- Contoh:
Jika NIM menjadi *primary key*, maka NIK, nomor HP, dan email menjadi *alternative key*

Composite Key

Tabel Mahasiswa
NIM
nama
jenis_kelamin
tanggal_lahir
alamat
NIK
no_hp
email
tanggal_daftar

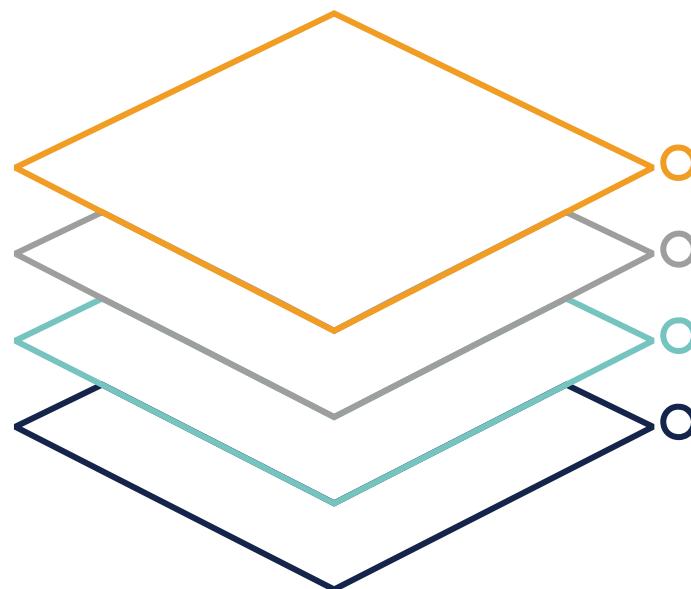
- **Kumpulan atribut** yang dapat digunakan untuk **mengidentifikasi** bahwa setiap baris menjadi **unique**
- Atribut tersebut **tidak dapat berdiri sendiri** untuk menjadi key
- Contoh:
(nama, tanggal lahir, alamat)

Non-prime Attributes

Tabel Mahasiswa
NIM
nama
jenis_kelamin
tanggal_lahir
alamat
NIK
no_hp
email
tanggal_daftar

- **Atribut tidak** dapat digunakan untuk **mengidentifikasi** setiap baris menjadi **unique**
- **Atribut** yang **tidak** dijadikan sebagai ***candidate key***
- Sering disebut sebagai ***non-key attribute***

Database Normalization



- Digunakan untuk **mengurangi redundancy** dari suatu hubungan antardata
- **Redundancy** dapat **menyebabkan anomali** penambahan, penghapusan, dan pengubahan
- **Normalisasi** akan **membagi tabel** yang besar menjadi beberapa tabel kecil yang memiliki keterkaitan

Normal Form

First Normal Form (1NF)

- **Setiap cell** hanya memiliki **1 nilai**

Second Normal Form (2NF)

- Harus berada pada 1NF
- Tidak boleh ada **partial dependency**, yaitu *non-prime attribute* hanya bergantung pada sebagian dari *candidate key*

Third Normal Form (3NF)

- Harus berada pada 2NF
- Tidak boleh ada **transitive dependency**, yaitu *non-prime attribute* bergantung pada *non-prime attribute* lainnya

First Normal Form (1NF)

NIM	Nama	Kelas
04	John	DS, DA
01	Doe	DS, MLE



NIM	Nama	Kelas
04	John	DS
04	John	DA
01	Doe	DS
01	Doe	MLE

- Pada atribut “Kelas”, terdapat cell yang memiliki lebih dari 1 nilai
- Atribut “Kelas” harus dipecah agar setiap cell hanya memiliki 1 nilai

Second Normal Form (2NF)

NIM	Kelas	Dosen	Nilai
04	DE-01	John	90
04	DS-03	Doe	80
01	DE-01	John	85
01	AI-02	Loem	75



NIM	Kelas	Nilai
04	DE-01	90
04	DS-03	80
01	DE-01	85
01	AI-02	75



Kelas	Dosen
DE-01	John
DS-03	Doe
AI-02	Loem

- Awalnya berada pada 1NF
- Primary key: (NIM + Kelas)
- Kombinasi tersebut menjadi primary key karena “NIM” saja tidak bisa digunakan untuk mengetahui “Dosen” dan “Nilai”
- Terjadi partial dependency karena “Dosen” hanya bergantung pada “Kelas”
- Atribut “Dosen” harus dipisah

Third Normal Form (3NF)

NIM	Nama	DPA	Ruang DPA
04	John	Lorem	05-01
01	Doe	Ipsum	05-09



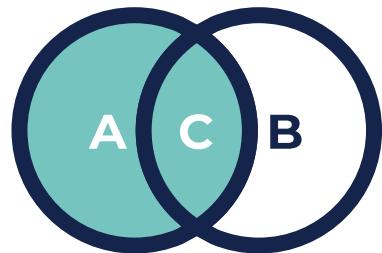
NIM	Nama	Kode DPA
04	John	01
01	Doe	02

Kode DPA	Nama	Ruang
01	Lorem	05-01
02	Ipsum	05-09

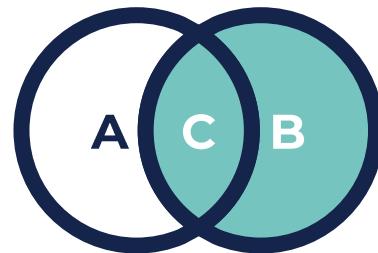


- Awalnya berada pada 2NF
- “NIM” menjadi *primary key*
- “DPA” dan “Ruang DPA” merupakan *non-prime attribute*
- Terjadi *transitive dependency* karena “Ruang DPA” bergantung pada “DPA”
- Atribut “DPA” dan “Ruang DPA” harus dipisah

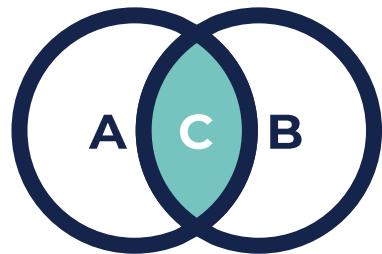
Join



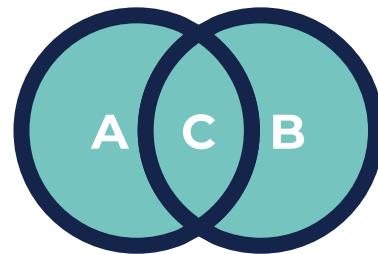
LEFT JOIN
(A + C)



RIGHT JOIN
(C + B)



INNER JOIN
(C)



FULL OUTER JOIN
(A + C + B)

- Digunakan untuk **menggabungkan** beberapa tabel dari segi **kolom**
- Tabel dapat digabungkan karena adanya **foreign key**

Function – JOIN

id	city
421833	Depok
125389	Depok
192320	Depok
337829	Depok

quantity
1
40
87
38

id	city	quantity
421833	Depok	1
125389	Depok	40
192320	Depok	87
337829	Depok	38

Syntax

```
SELECT t1.col1, t1.col2, ..., t2.col1, t2.col2, ...  
FROM table_name t1  
JOIN table_name t2  
ON t1.col = t2.col
```

Contoh

```
SELECT c.id, c.city, t.quantity  
FROM customer c  
JOIN transactions t  
ON c.id = t.customer_id  
LIMIT 4
```

Merge

TABEL A			
NIM	Nama	DPA	Kelas
04	John	Lorem	DS
01	Doe	Ipsum	DE



TABEL B			
NIM	Nama	DPA	Kelas
04	John	Lorem	DA
01	Doe	Ipsum	DE

- Digunakan untuk **menggabungkan** beberapa tabel dari segi **baris**
- **UNION**
Hanya menggabungkan data yang berbeda (***distinct value***)
- **UNION ALL**
Menggabungkan **semua data**, boleh ada data duplikat

Function – UNION

id	city
1	Jakarta
2	Tangerang

id	city
3	Depok
4	Bogor

id	city
1	Jakarta
2	Tangerang
3	Depok
4	Bogor

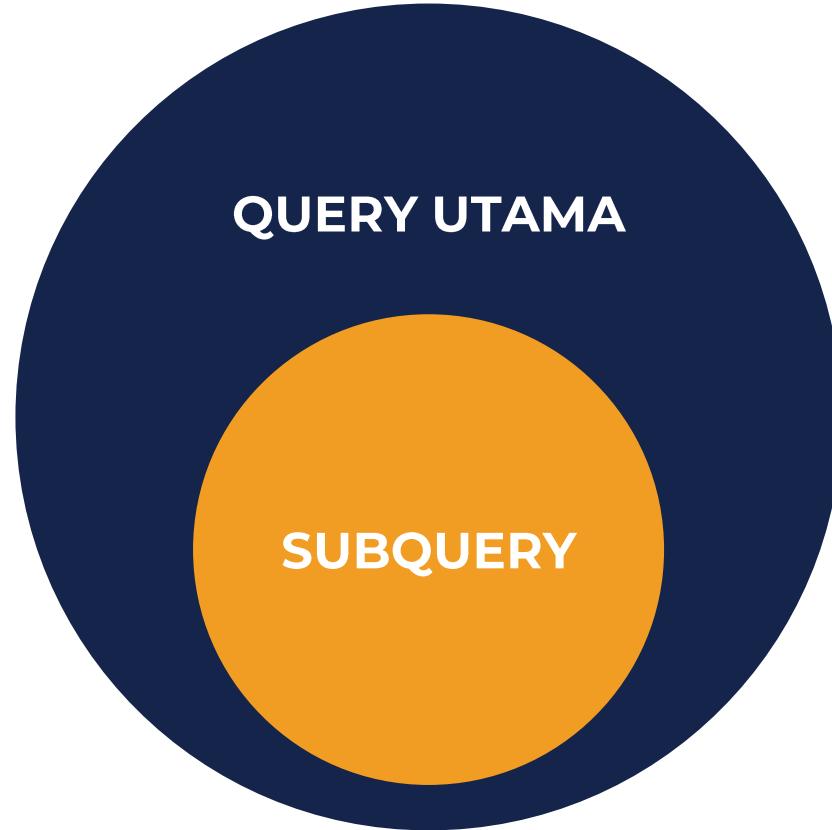
Syntax

```
SELECT t1.col1, t1.col2  
FROM table_name t1  
UNION  
SELECT t2.col1, t2.col2  
FROM table_name t2
```

Contoh

```
SELECT t1.id, t1.city  
FROM table_name t1  
UNION  
SELECT t2.id, t2.city  
FROM table_name t2
```

Subquery



- Digunakan untuk melakukan **query di dalam query**
- Hasil dari sebuah **query** dapat berupa **nilai tunggal** (value), **list**, atau **tabel**
- Subquery dapat diletakkan pada
 - **SELECT**, harus value
 - **WHERE**, bisa value/ list
 - **JOIN**, bisa list/ tabel
 - **FROM**, bisa list/ tabel

Contoh Subquery

- Tim *marketing* ingin mengetahui jumlah **customer** dari **Jakarta** yang memiliki total **transaksi** lebih dari **10 kali**
- Data *customer* berada pada tabel “customer”
- Data transaksi berada pada tabel “transactions”

Subquery pada JOIN

```
SELECT  
    c.id customer_id,  
    COUNT(t.*) jumlah_transaksi  
FROM transactions t  
JOIN  
    (SELECT *  
     FROM customer  
     WHERE city = 'Jakarta') c  
ON c.id = t.customer_id  
GROUP BY 1  
HAVING COUNT(t.*) > 10
```

Subquery – WITH

- Digunakan untuk membuat **tabel sementara** (*temporary table*) yang hanya ada **pada 1 query** tersebut
- Penggunaan WITH dapat membuat *script query* menjadi **lebih rapi dan mudah dipahami**

Subquery dengan WITH

WITH

```
customer_jakarta AS (
    SELECT *
    FROM customer
    WHERE city = 'Jakarta'
)
```

SELECT

```
c.id customer_id,
COUNT(t.*) jumlah_transaksi
FROM transactions t
JOIN customer_jakarta c
ON c.id = t.customer_id
GROUP BY 1
HAVING COUNT(t.*) > 10
```

Effective Query Tips – Technical



- **Script query** yang **panjang** tidak membuatnya selalu menjadi lebih bagus
- Berhati-hati dengan penggunaan **fungsi** yang **tidak diperlukan**
- Pilih **kolom** yang hanya **dibutuhkan**, jangan menggunakan (*) di setiap SELECT
- **Urutkan data** hanya ketika dibutuhkan
- Sebisa mungkin, **kurangi** penggunaan **CASE – WHEN**
- **Jangan** terlalu **bergantung** pada **subquery**
- Selalu **gunakan LIMIT** jika hanya ingin mengecek isi tabel

Effective Query Tips – Business



- Gunakan pendekatan **top down** untuk menerjemahkan permintaan/ pertanyaan dari tim bisnis
- **Ekstrak tujuan** utama dari pertanyaan
- Buat **daftar semua kolom** yang dibutuhkan berdasarkan tujuan
- Uraikan semua **sumber data** untuk kolom yang diperlukan
- Tulis **script query** pada setiap sumber data untuk mendapatkan kolom yang dibutuhkan
- **Satukan** semuanya menjadi 1 script

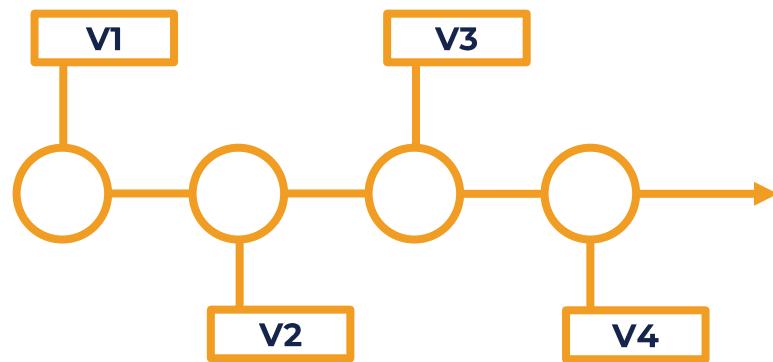


02

VERSIONING/ VERSION CONTROL

*Version control dengan
Git dan GitHub*

Version Control System (VCS)



- Sebuah sistem yang **merekam perubahan** pada sebuah berkas atau sekumpulan berkas **dari waktu ke waktu**
- Oleh karena itu, kita dapat **melihat kembali** versi tertentu di lain waktu

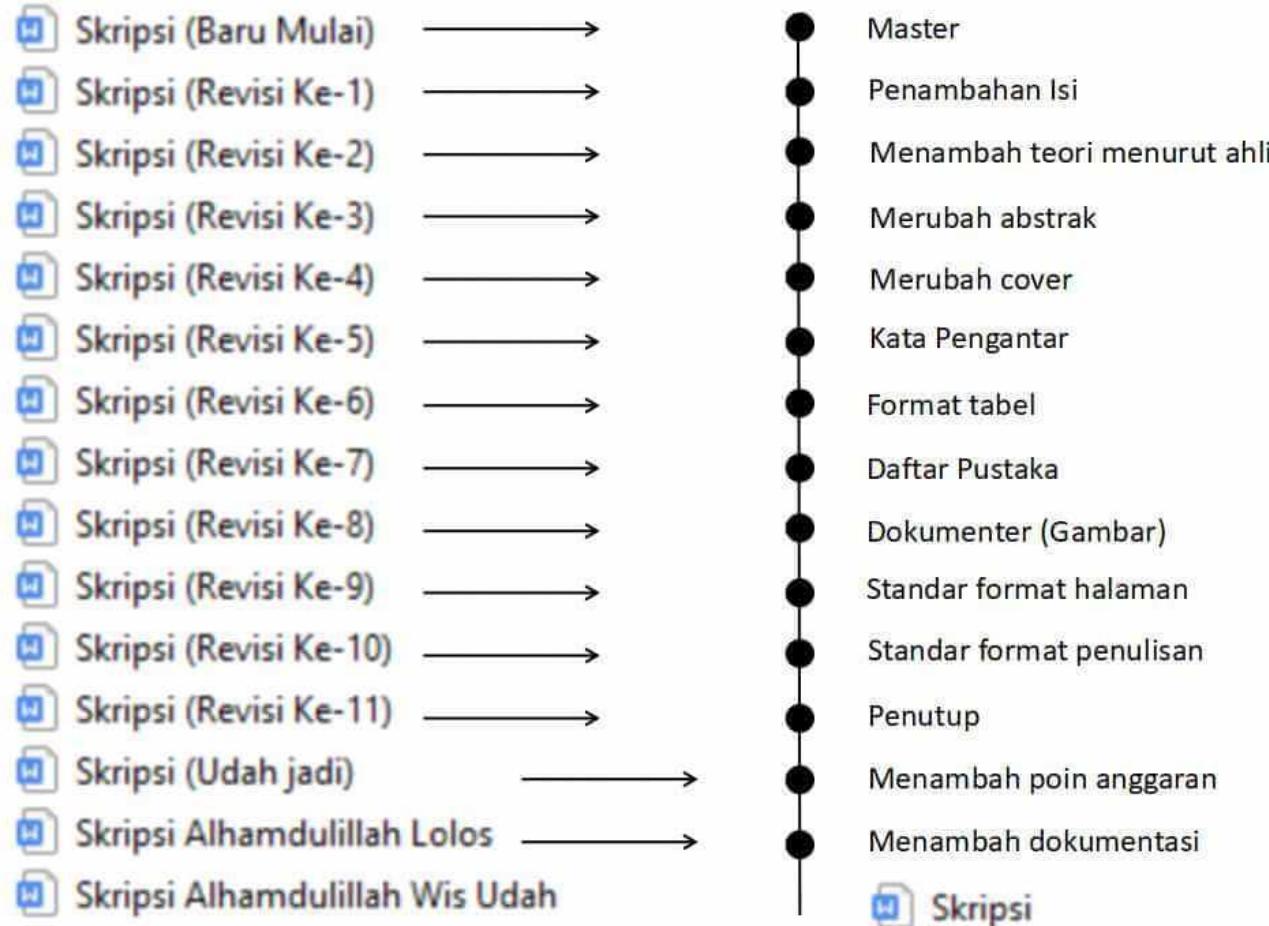
Sumber:

<https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>

Fungsi VCS

- Memberikan **kemudahan** dalam **berkolaborasi**, khususnya untuk proyek besar
- **Mengetahui perubahan** yang terjadi dalam source code
- **Menghindari banyak versi file** yang tersimpan
- **Menyediakan cadangan** terhadap file yang sedang dikerjakan

Ilustrasi Penggunaan VCS



- **Sebelum** menggunakan VCS, terdapat **banyak file** dengan berbagai versi
- **Setelah** menggunakan VCS, hanya terdapat **satu file**.
- Meski hanya 1 file, semua **riwayat perubahan** masih tersimpan dan dapat **dilihat kembali**

Sumber:

<https://www.dicoding.com/blog/perbedaan-git-dan-github/>

Git

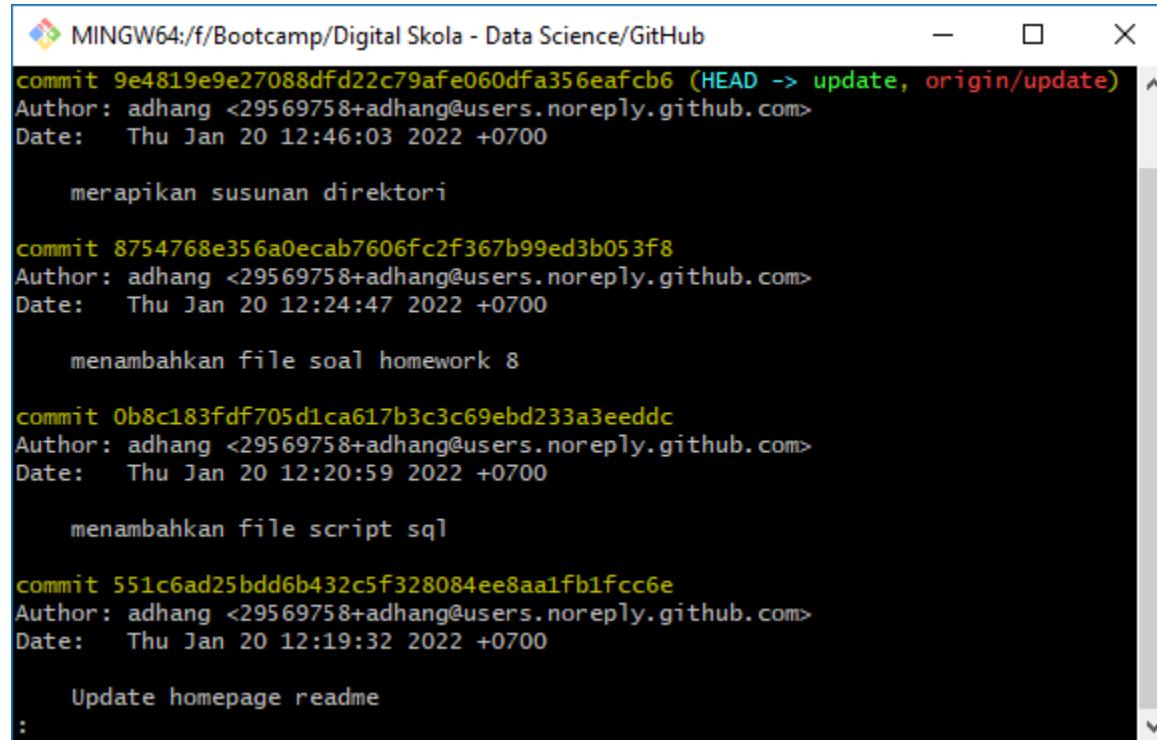


Software berbasis VCS yang bertugas untuk **mencatat perubahan** seluruh *file* atau *repository* suatu *project*

Sumber:

<https://www.dicoding.com/blog/perbedaan-git-dan-github/>

Fitur Utama GIT – 1



The screenshot shows a terminal window titled 'MINGW64:/f/Bootcamp/Digital Skola - Data Science/GitHub'. It displays a history of five Git commits made by the user 'adhang' on Thursday, January 20, 2022, at various times between 12:46:03 and 12:19:32. Each commit message includes a brief description of the changes made:

- commit 9e4819e9e27088dfd22c79afe060dfa356eaefcb6 (HEAD -> update, origin/update)
Author: adhang <29569758+adhang@users.noreply.github.com>
Date: Thu Jan 20 12:46:03 2022 +0700

merapikan susunan direktori
- commit 8754768e356a0ecab7606fc2f367b99ed3b053f8
Author: adhang <29569758+adhang@users.noreply.github.com>
Date: Thu Jan 20 12:24:47 2022 +0700

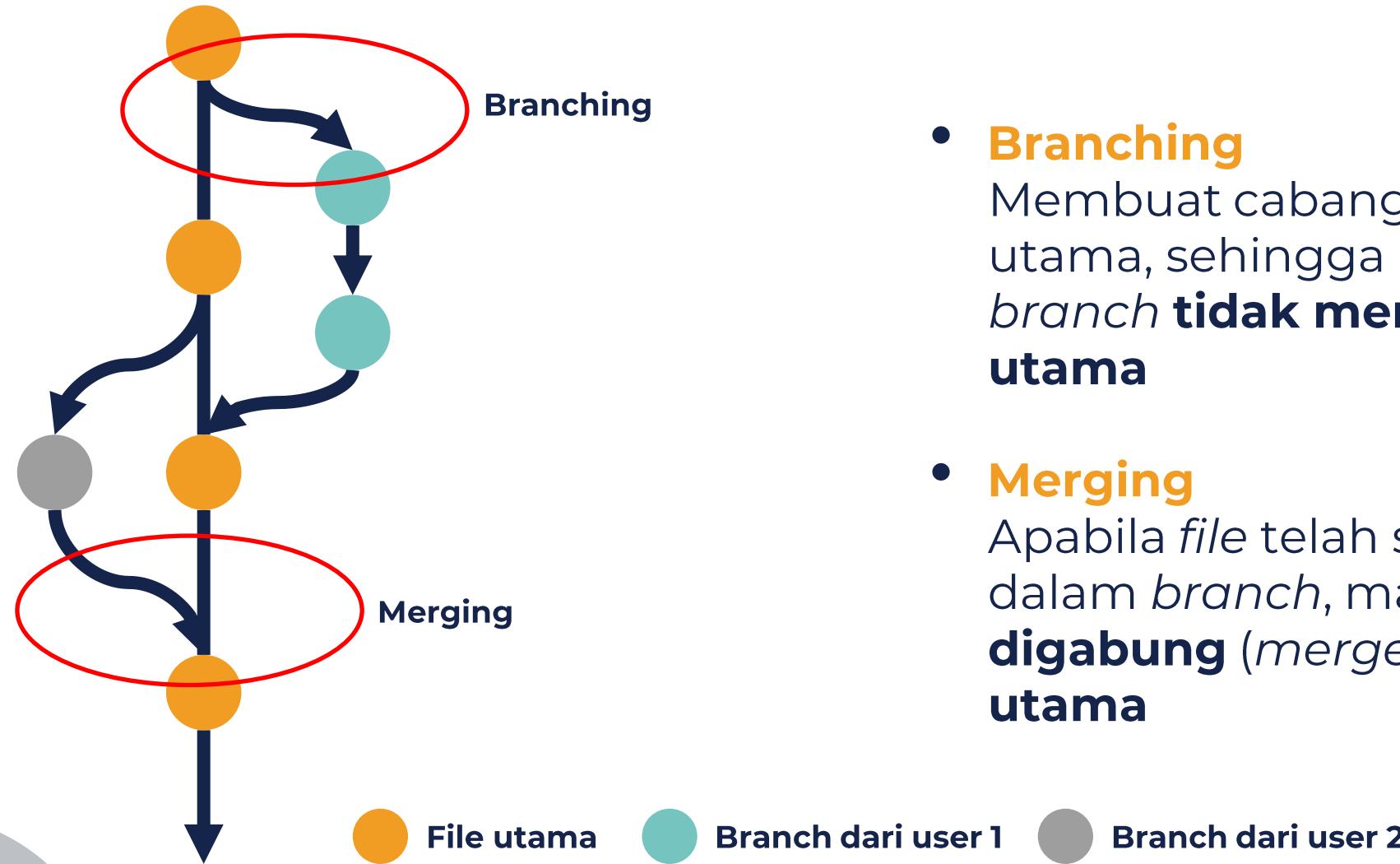
menambahkan file soal homework 8
- commit 0b8c183fdf705d1ca617b3c3c69ebd233a3eeddc
Author: adhang <29569758+adhang@users.noreply.github.com>
Date: Thu Jan 20 12:20:59 2022 +0700

menambahkan file script sql
- commit 551c6ad25bdd6b432c5f328084ee8aa1fb1fcc6e
Author: adhang <29569758+adhang@users.noreply.github.com>
Date: Thu Jan 20 12:19:32 2022 +0700

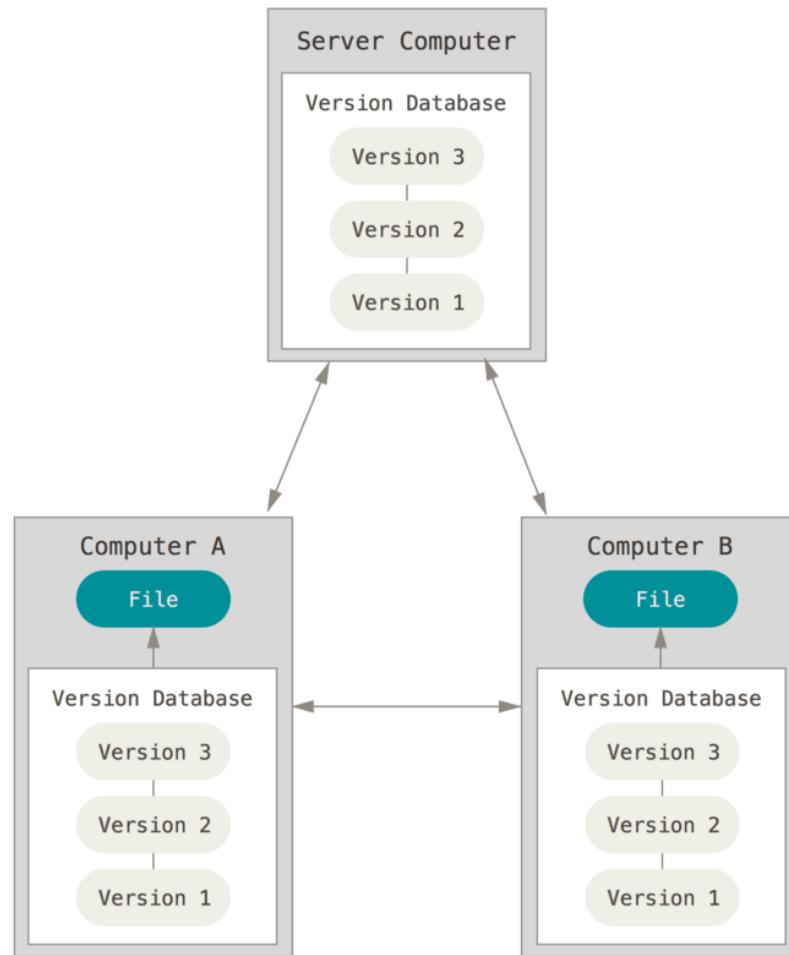
Update homepage readme
- :

- **Riwayat dari semua file**
Setiap **perubahan** file yang telah terjadi dari awal dibuat sampai dengan perubahan terakhir **dapat ditelusuri**
- **Traceability**
Detail perubahan dapat ditelusuri, mulai dari **author, alasan, waktu** perubahan, dan **perubahan** yang terjadi

Fitur Utama GIT – 2



Fitur Utama GIT – 3



- **Distributed system**
 - Setiap *user* mendapatkan masing-masing *repository* lokal
 - Memudahkan dalam hal *back up*
 - Meminimalisasi *error* pada jaringan yang biasanya terdapat dalam *centralized system*

Sumber:

<https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>

GitHub



Layanan **cloud** yang berguna untuk **menyimpan** dan **mengelola** sebuah *project* yang dinamakan **repository**

Sumber:

<https://www.dicoding.com/blog/perbedaan-git-dan-github/>

Git VS GitHub

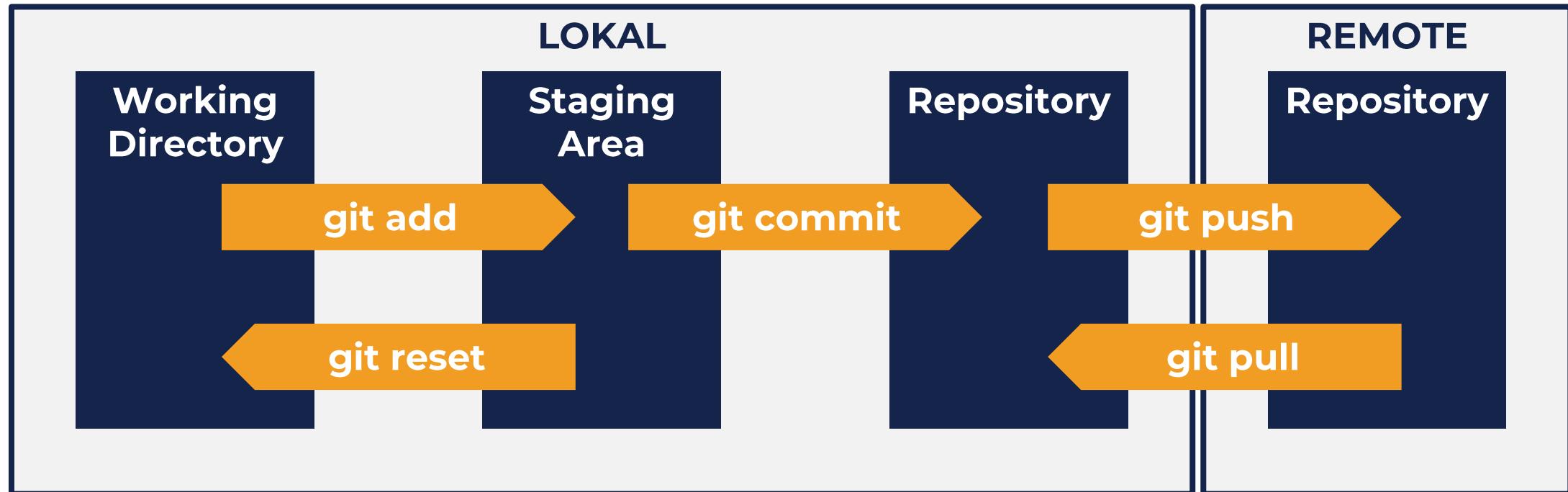


Pada dasarnya, **konsep** Git dan GitHub **sama**. Hal yang paling membedakan adalah:

- **Cara pengaksesan**
Git diakses secara *offline*, sedangkan GitHub harus *online*
- **User interface**
Secara umum, UI dari GitHub lebih menarik dan mudah dipahami oleh pengguna awal

Git & GitHub

Git dan GitHub dapat dihubungkan sehingga dapat **digunakan** secara **bersamaan**, di mana GitHub digunakan sebagai *remote server*



Git Cheatsheet – 1

Git Configuration	
Perintah	Deskripsi
git --version	Mengetahui versi Git
git config --global user.name "entropy"	Mengatur nama author
git config --global user.email "entropy@team.com"	Mengatur email author
git config user.name	Melihat nama author
git config user.email	Melihat email author
Git Repository	
Perintah	Deskripsi
git init	Membuat repository
git status	Melihat perubahan status
git log	Melihat riwayat commit
git log --p	Melihat riwayat commit

Git Cheatsheet – 2

Git Branching	
Perintah	Deskripsi
git branch "nama_branch"	Membuat branch baru
git branch -d "nama_branch"	Menghapus branch
git branch -D "nama_branch"	Menghapus branch
git checkout "nama_branch"	Masuk ke dalam branch
git chekcout -b "nama_branch"	Membuat dan masuk ke branch baru
git branch	Melihat daftar branch

Git Cheatsheet – 3

Git Execution	
Perintah	Deskripsi
git add .	Menambah semua file ke staging area
git add "nama file"	Menambah file ke staging area
git rm -cached .	Menghapus semua file dari staging area
git rm -cached "nama file"	Menghapus file dari staging area
git commit -m "pesan commit"	Melakukan commit
git commit -a -m "pesan commit"	Menambah semua file ke staging area sekaligus melakukan commit

Git Cheatsheet – 4

Git Remote	
Perintah	Deskripsi
git remote add origin "link https dari github"	Mengintegrasikan dengan GitHub
git clone "link https dari github"	Menyalin repository dari GitHub
git push origin "nama_branch"	Push semua commit ke GitHub
git pull origin "nama_branch"	Pull semua commit dari GitHub

03

INTRODUCTION TO PYTHON AND PROGRAMMING

Pengenalan pemrograman
dan bahasa Python

Python

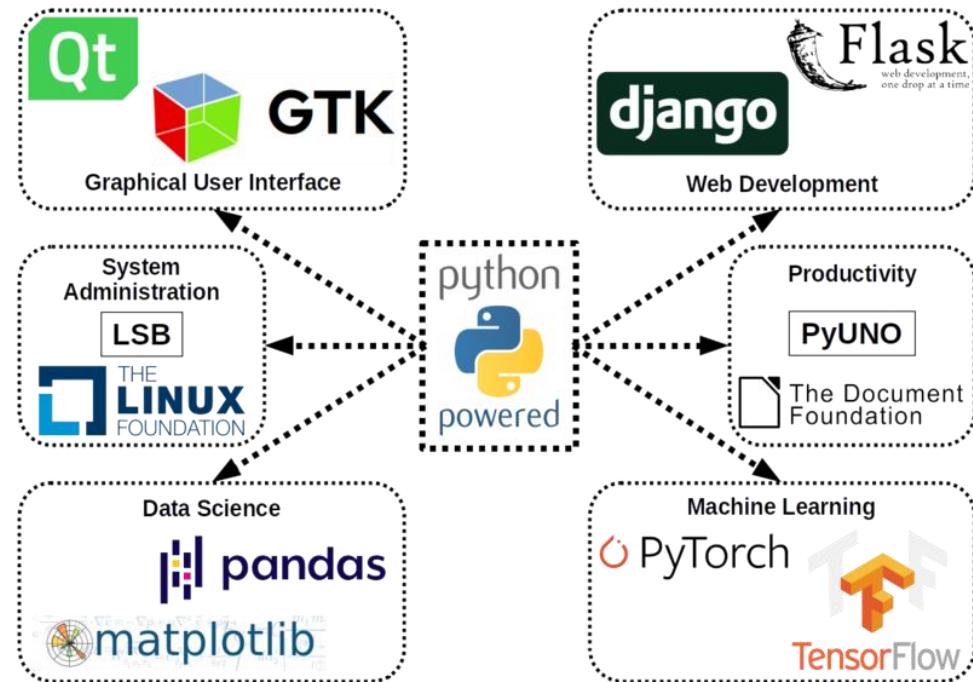


Python adalah **bahasa pemrograman tingkat tinggi** yang dapat digunakan untuk berbagai macam implementasi, seperti:

- *Web & internet development*
- *Scientific & numeric*
- *Education*
- *Desktop GUI*
- *Software development*
- *Business application*

Sumber:
<https://www.python.org/about/apps/>

Kelebihan Python



- **Banyak *library*** yang tersedia
- **Syntax** yang sederhana
- **Komunitas** yang besar
- **General-purpose**, artinya Python dapat digunakan di berbagai macam implementasi

Sumber:

[https://en.wikipedia.org/wiki/Python_\(programming_language\)](https://en.wikipedia.org/wiki/Python_(programming_language))

Python dalam Data Science



Python memiliki banyak library yang mendukung proses dalam Data Science, seperti:

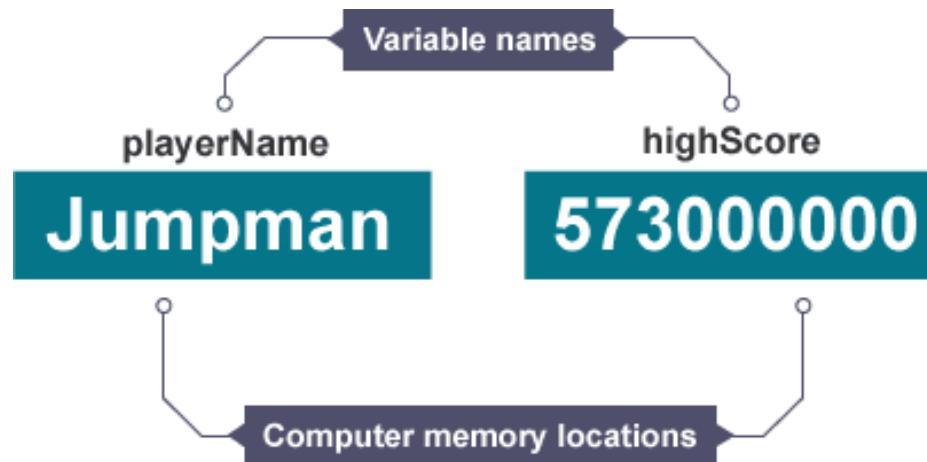
- Pandas
- NumPy
- SciPy
- Scikit-learn
- Matplotlib
- Seaborn

Jupyter Notebook



- Jupyter Notebook adalah REPL (read-eval-print loop) **berbasis browser**
- Terdapat berbagai **cell** yang dapat berisikan **kode** (*input dan output*), **teks** (*markdown*), gambar **plot**, dan **rich media**

Variabel dalam Python



- Variabel adalah suatu **nama** yang melekat pada (alamat memori) **data**
- Nama tersebut digunakan untuk mempermudah dalam penggunaan selanjutnya
- **Aturan penulisan:**
 - Harus dimulai dengan huruf atau *underscore* (_)
 - Tidak bisa dimulai dengan angka
 - Hanya mengandung *alphanumeric* dan underscore (A-Z, a-z, 0-9, _)
 - Case-sensitive

Tipe Data Dasar – 1

```
# Integer  
x = 25  
print(x) # 25
```

```
# Float  
x = 25.4  
print(x) # 25.4
```

```
# Complex  
x = complex(25, 2)  
print(x) # 25 + 2j
```

- **Integer**

Digunakan untuk menyimpan data **bilangan bulat** (tidak ada pecahan)

- **Float**

Digunakan untuk menyimpan data **bilangan cacah** (ada koma)

- **Complex**

Digunakan untuk menyimpan data **bilangan kompleks**, yaitu bilangan yang memiliki bagian nyata dan imajiner

Tipe Data Dasar – 2

```
# String  
  
Nama = 'Entropy'  
  
print(Nama) # Entropy  
  
  
# Boolean  
  
x = True  
  
print(x) # True  
  
  
# List  
  
x = [1, 2, 2, 3]  
  
print(x) # [1, 2, 2, 3]
```

- **String**

Digunakan untuk menyimpan data **teks**, ditandai dengan menggunakan **petik satu** ('isi teks') atau **petik dua** ("isi teks")

- **Boolean**

Digunakan untuk menyimpan nilai **TRUE** atau **FALSE**

- **List**

Digunakan untuk menyimpan **beberapa data**, di mana data tersebut **dapat diubah**

Tipe Data Dasar – 3

```
# Tuple  
x = (1, 2, 2, 3)  
print(x) # (1, 2, 2, 3)  
  
# Set  
x = {1, 2, 2, 3}  
print(x) # {1, 2, 3}  
  
# Dictionary  
x = {'nama': 'Team'}  
print(x['nama']) # Team
```

- **Tuple**

Digunakan untuk menyimpan **beberapa data**, di mana data tersebut **tidak dapat diubah**

- **Set**

Digunakan untuk menyimpan **beberapa data**, di mana data tersebut **tidak bernilai sama** (*distinct value*)

- **Dictionary**

Digunakan untuk **memetakan nilai** (*value*) dengan **kuncinya** (*key*) menjadi **key:value**

Operasi Aritmatika

Operator	Keterangan	Contoh
+	Penjumlahan	$5 + 2 = 7$
-	Pengurangan	$5 - 2 = 3$
*	Perkalian	$5 * 2 = 10$
/	Pembagian	$5 / 2 = 2.5$
%	Modulo, sisa bagi	$5 \% 2 = 1$
**	Pangkat	$5 ** 2 = 25$
//	Pembagian bulat	$5 // 2 = 2$

Seperti dalam matematika, penggerjaan operasi aritmatika pada Python mengikuti urutan PEMDAS:

- *Parentheses*
- *Exponent*
- *Multiplication & Division*
- *Addition & Subtraction*

Assignment

Operator	Contoh	Ekuivalen
=	X = 5	X = 5
+=	X += 5	X = X + 5
-=	X -= 5	X = X - 5
*=	X *= 5	X = X * 5
/=	X /= 5	X = X / 5
%=	X %= 5	X = X % 5
**=	X **= 5	X = X ** 5
//=	X //= 5	X = X // 5

Operator assignment digunakan untuk **memberikan nilai** pada suatu **variabel**

Whitespace

```
x = 5

# tanpa memperhatikan whitespace

Y=2*(3*x)+0.25*(2.5*x)+2

# whitespace ditambahkan

Y = 2 * (3 * x) + 0.25 * (2.5 * x) + 2

# penambahan backslash

Y2 = 2 * (3 * x) \
    + 0.25 * (2.5 * x) + 2
```

- Pada Python, **spasi** dan **tab** merupakan **whitespace**
- Whitespace dapat digunakan untuk **meningkatkan readability** dari source code yang ditulis
- Jika suatu **code** sangat **panjang**, simbol **backslash** (\) dapat digunakan untuk meneruskan code pada **baris baru**

Indentation

```
x = 5

# dengan indentation
if x >= 2:
    print(x)

# tanpa indentation, akan error
if x >= 2:
print(x)
```

- *Indentation* adalah **leading whitespace**, yaitu spasi atau tab di depan code
- *Indentation* pada Python digunakan untuk menandakan suatu **block of code**
- Jika suatu *block of code* tidak menggunakan *indentation*, maka akan muncul **IndentationError**

THANKS

Entropy Team

CREDITS: This presentation template was originally created by [Slidesgo](#),
including icons by [Flaticon](#), and infographics & images by [Freepik](#)