

The background features a white canvas with several large, colorful circles in teal, lime green, orange, and yellow. Some circles have smaller circles inside them, and some are dashed. A thin, light blue dashed line curves across the page, passing behind the main text.

LEARNING PROGRESS REVIEW WEEK 3

OMICRON

OMICRON

Anggota Kelompok 3 :

Anugrah Yazid Ghani

Fajar Achmad

Muhammad Fikri Fadila

Edo Mohammad Hadad Gibran



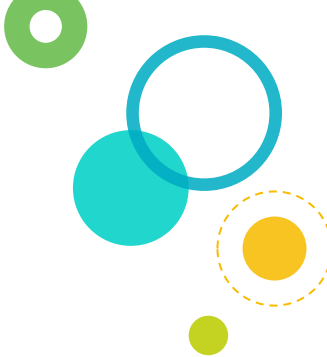
1

Advanced SQL





Content

1. **Date Function**
 2. **Primary Key, Auto increment and Foreign Key**
 3. **Data Normalization**
 4. **Join and Merge**
 5. **Subquery**
- 



1

Date Function

How to manipulate Date and/or Time
data effectively



Formats

- ◎ Date Format: YYYY-MM-DD
- ◎ Time Format: HH:MI:SS

Function

- ◎ CURRENT_DATE
- ◎ CURRENT_TIME
- ◎ EXTRACT
- ◎ DATE_TRUNC

CURRENT_DATE

Menjawab
tanggal terkini.

Example:

Input : SELECT CURRENT_DATE

Ouput : '2022-01-24'

CURRENT_TIME

Menjawab waktu
terkini.

Example:

Input : SELECT CURRENT_TIME

Ouput : '20:31:54'

EXTRACT

Fungsi EXTRACT mengambil subbidang seperti tahun atau jam dari nilai tanggal/waktu. Sumber harus berupa ekspresi nilai dari jenis timestamp, waktu, atau interval. Field adalah pengidentifikasi atau string yang memilih bidang apa yang akan diekstraksi dari nilai sumber. Fungsi EXTRACT mengembalikan tipe nilai presisi ganda.

Contoh daftar Field:

day : hari ke- dalam bulan (1 - 31) doy : hari ke- dalam tahun (1 – 365)

dow : hari ke- dalam minggu (0 – 6) month : bulan ke- dalam tahun (1 – 12)

Example:

Input :

```
SELECT EXTRACT (DECADE FROM TIMESTAMP '2022-01-24 20:34:32')
```

Ouput : 200

DATE_TRUNC

Sama seperti fungsi TRUNC pada nomor. Sumbernya adalah ekspresi nilai dari jenis cap waktu atau interval. (Nilai dari tipe tanggal dan waktu di CAST secara otomatis, ke timestamp atau interval masing-masing.) Field memilih presisi nilai mana yang akan dipotong dari input. Nilai yang dikembalikan bertipe stempel waktu atau interval.

Contoh daftar Field:

second, minute, hour, day, week, month, quarter, year, decade dst.

Example:

Input :

```
SELECT date_trunc ('hour' TIMESTAMP '2022-01-24 20:34:32')
```

Ouput : 2022-01-24 20:00:00

Date Function Logics

- $\text{date} + \text{integer} \rightarrow \text{date}$
- $\text{date} - \text{date} \rightarrow \text{integer}$
- $\text{date} + \text{interval} \rightarrow \text{timestamp}$
- $\text{date} + \text{time} \rightarrow \text{timestamp}$
- $\text{interval} + \text{interval} \rightarrow \text{interval}$
- $\text{timestamp} + \text{interval} \rightarrow \text{timestamp}$
- $\text{time} + \text{interval} \rightarrow \text{time}$
- $\text{date} - \text{integer} \rightarrow \text{date}$
- $\text{date} - \text{interval} \rightarrow \text{timestamp}$
- $\text{time} - \text{time} \rightarrow \text{interval}$
- $\text{time} - \text{interval} \rightarrow \text{time}$
- $\text{timestamp} - \text{interval} \rightarrow \text{timestamp}$
- $\text{interval} - \text{interval} \rightarrow \text{interval}$
- $\text{timestamp} - \text{timestamp} \rightarrow \text{interval}$
- $\text{interval} * \text{double precision} \rightarrow \text{interval}$
- $\text{interval} / \text{double precision} \rightarrow \text{interval}$



2

Key

An unique attribute which helps you to identify a row in a table.

Key

Atribut atau kumpulan atribut unik dengan kombinasi satu atau lebih yang membantu mengidentifikasi baris dalam tabel.

Key sangat berguna untuk mengidentifikasi setiap baris data dalam sebuah table, memeriksa duplikasi data, serta untuk membangun hubungan antar tabel.

Beberapa Jenis Key :

- ◎ **Primary key:** Candidate key yang paling tepat untuk dijadikan kunci utama.
- ◎ **Candidate key:** Sekumpulan kolom minimal yang dapat secara unik mengidentifikasi setiap record dalam sebuah tabel.
- ◎ **Composite Key:** Key yang terdiri dari dua atau lebih atribut yang secara unik mengidentifikasi setiap record dalam sebuah tabel. Individual Key yang bersama-sama membentuk Composite Key tidak dapat bertindak sebagai Key secara individual/mandiri.
- ◎ **Foreign Key:** Kolom yang membuat hubungan antara dua tabel.

The background features several overlapping circles in orange, yellow, green, and blue. Some circles have dashed outlines in matching colors. A large, light blue dashed circle is centered in the upper half of the slide.

3

Data Normalization

The process of minimizing redundancy from a relations.

Proses meminimalkan redundansi dari suatu relasi.

Rules Data Normalization:

First Normal Form (1NF)

- ⦿ Bernilai tunggal: tiap kolom tidak boleh berisi beberapa nilai.
- ⦿ Domain tidak berubah: Nilai yang disimpan jenisnya sama.
- ⦿ Nama unik.
- ⦿ Urutan tidak penting.

Second Normal Form (2NF)

- ⦿ Tabel harus sudah dalam 1NF.
- ⦿ Tidak ada ketergantungan parsial: Tidak ada subset yang tepat dari Candidate Key yang menentukan atribut bukan prima.

Third Normal Form (3NF)

- ⦿ Tabel harus sudah dalam 2NF.
- ⦿ Tidak ada ketergantungan transitif: Tidak ada atribut non-primary yang bergantung pada atribut bukan prima.



4

Join and Merge

Extract result
from more than 1 table.

Digunakan untuk mengekstrak hasil dari lebih dari 1 tabel.
Bisa menggabungkan kemiripan (*merge*) atau memiliki hubungan (*join*)

Contoh:

Terdapat 2 Tabel, yaitu *customer list* dan *transaction list*, fungsi JOIN digunakan untuk menghubungkan antara *customer* dengan kondisi tertentu dengan jumlah transaksinya.

Input.

```
select
  count(c.id) as potential_customer,
  c.city,
  c.email,
  c.gender
from datasource_sql_ds11.customer c
inner join
(
  select customer_id, count(customer_id)
  from datasource_sql_ds11."transaction" t
  group by customer_id
  having count(customer_id) >= 10
) as t
on c.id = t.customer_id
where
  city = 'Jakarta' and
  email = 'Gmail' and
  gender = 'Female'
group by c.city, c.email, c.gender
```

Output.

potential_customer	city	email	gender
395	Jakarta	Gmail	Female

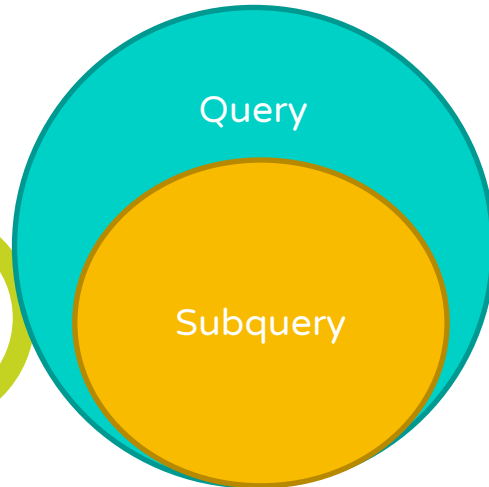


5

Subquery

The next level of SQL queries is to put query inside query.

Subquery berperan sebagai konstanta untuk evaluasi variabel dari Query dan dapat digunakan untuk mengekstrak hasil dari beberapa table seperti fungsi JOIN.



Subquery dapat diletakkan di:

- ⦿ dalam fungsi SELECT
- ⦿ dalam fungsi FROM
- ⦿ dalam fungsi WHERE
- ⦿ dalam fungsi JOIN
- ⦿ dalam fungsi *subquery*
- ⦿ Tiap *subquery* wajib hanya memiliki 1 buah *value*, tidak boleh lebih.

Contoh:

Dari tabel *transaction*, kita diminta untuk mencari 5 produk dengan transaksi terbanyak.

Input:

```
select
  product_id,
  sum(quantity) as total_transactions
from datasource_sql_ds11."transaction" t
where created_at between
  '2018-10-01' and
  '2018-12-31' and
  store_id = '2'
group by product_id
having sum(quantity) > avg(quantity)
order by total_transactions desc
limit 5
```

Output:

product_id	total_transactions
49	892,462
39	834,085
38	816,780
50	758,775
58	370,041

The background is white and decorated with various colorful circles and dashed lines. In the top left, there is a large orange circle with a dashed red outline, overlapping a yellow circle. Below it is a small pink circle. In the top center, a large red number '2' is centered within a large dashed light blue circle. In the top right, there is a green circle with a white dot inside, a small yellow circle, and a lime green circle with a dashed yellow outline. In the bottom left, there is a green circle with a dashed green outline, a small cyan circle, and a large lime green circle. In the bottom right, there is a large cyan circle with a white dot inside, and a cyan circle with a dashed blue outline.

2

Versioning / Version Control



Version Control System (VCS)

Software yang dapat mengelola dan mengatasi perubahan-perubahan dalam kode sumber dari waktu ke waktu.

Perangkat lunak VCS melacak setiap detail modifikasi yang dibuat dalam kode sumber.

Kenapa penting digunakan?

1. Terciptanya kolaborasi yang baik antara developer terutama dalam proyek yang lebih besar .
2. Dapat memahami perubahan yg terjadi pada sumber kode.
3. Terhindar dari beberapa versi yang tidak diinginkan.
4. Menjadi sumber cadangan.

GIT

Sebuah sistem kontrol *version* yang terdistribusi, bersifat gratis dan terbuka yang dapat menangani segala hal, dari proyek kecil hingga besar dengan tingkat efisiensi yang tinggi.

Fitur :

1. History of files
2. Traceability
3. Branching
4. Merging
5. Distributed system



GIT Features : History of Every Files

Dalam memulai sebuah proyek, GIT akan melacak semua hal, termasuk kembali ke versi proyek yang lama.

1. File dibuat
2. File dihapus
3. File dimodifikasi
 - a. Tambahkan lebih banyak baris ke dalam file
 - b. File berganti nama
 - c. File dipindahkan ke folder lain



GIT Features : Traceability

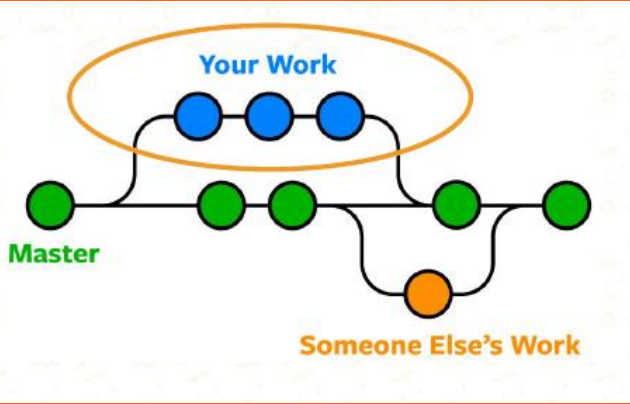
Setiap perubahan dilacak bersamaan.

Hal ini bermanfaat untuk memahami alur dari perkembangan data.

Hal yang dapat dilihat antara lain :

1. Author
2. Reason
3. Time
4. Changes

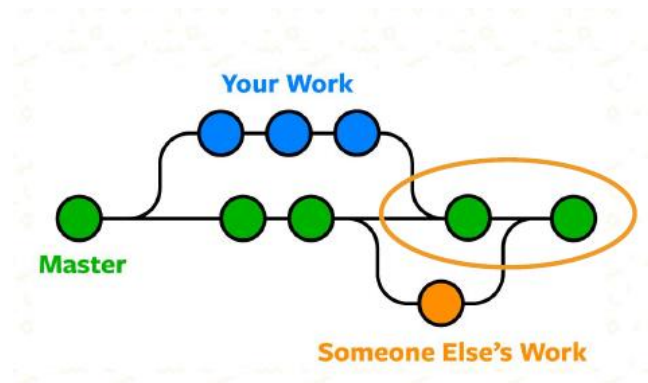
GIT Features : Branching



GIT dapat membuat percabangan dari sumber data master. Hal ini membuat pekerjaan terkolaborasi dengan baik menggunakan GIT dengan mudah.

GIT Features : Merging

Setelah pekerjaan di cabang selesai, GIT dapat menggabungkan cabang ke proyek utama dengan menggabungkannya.

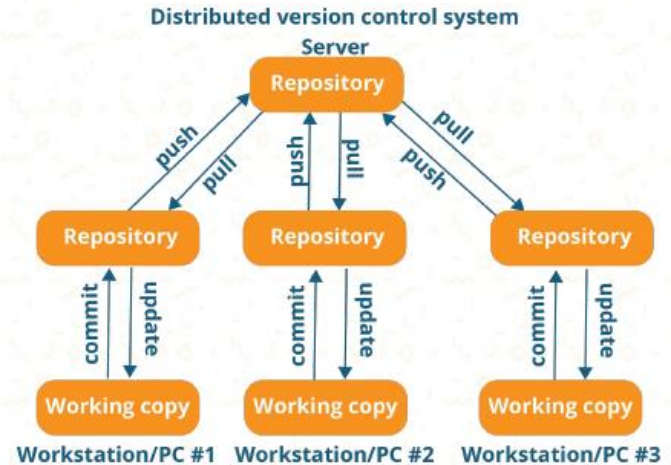


GIT Features : Distributed System

Setiap *developer* memiliki *repository*-nya masing-masing.

Hal ini dapat mengurangi koneksi jaringan dan pemblokiran pada kesalahan, tidak seperti sistem yg terpusat.

Sistem terdistribusi juga memungkinkan pencadangan lebih mudah.





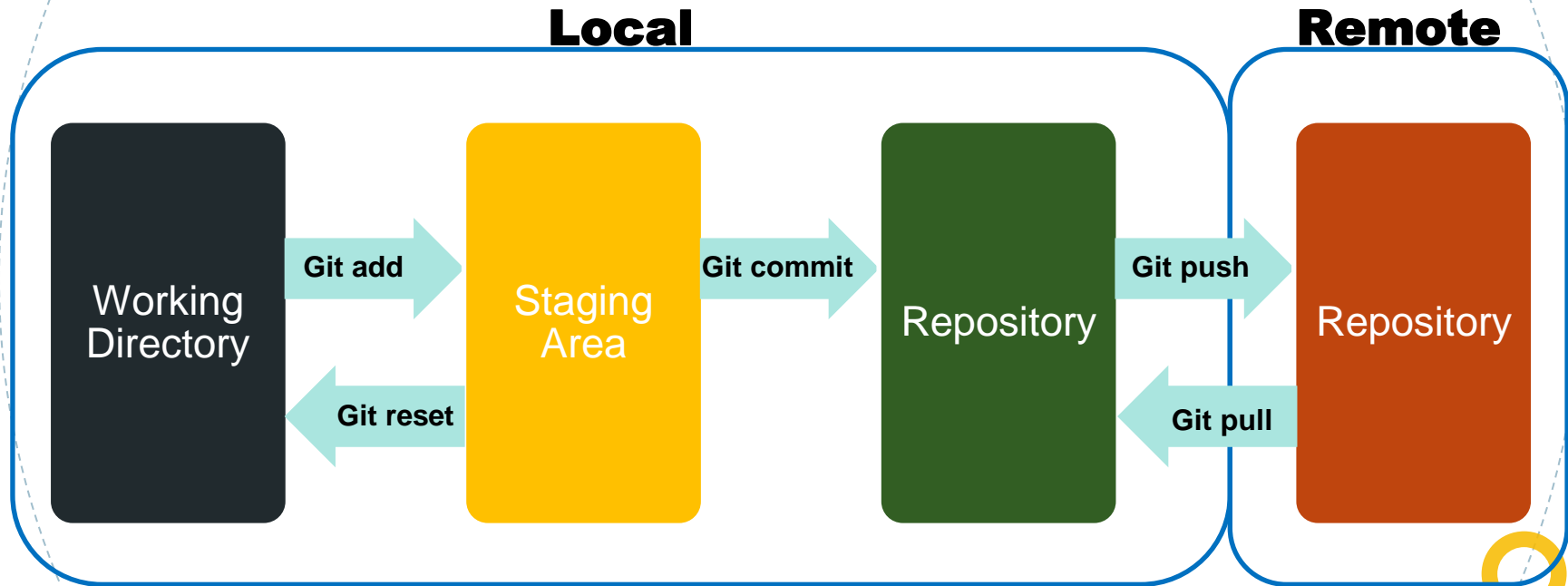
GIT Hub

GitHub akan menjadi repositori atau server jarak jauh. Tidak seperti repositori lokal, semua file yang disimpan di GitHub dapat diakses dari mana saja selama kita memiliki akses tersebut.

Mengapa GitHub ?

1. Banyak digunakan oleh pengembang dan perusahaan membangun, mengirimkan, dan memelihara perangkat lunak mereka
2. Pengodean kolaboratif. Banyak komunitas dan pengembang berkolaborasi di GitHub
3. Banyak *add-on* di pasar GitHub tersedia untuk diintegrasikan
4. Gratis! Untuk penyimpanan individu. Berguna untuk menampung portofolio Ilmu Data

GIT Environment



Git add : memasukkan data ke area *staging*

Git commit : memasukkan data dari area sementara ke *repository local*

Git push : mengunggah data ke *repository remote*

Git pull : menarik data dari *repository remote*

Git reset : mereset data ke folder lokal

Tools Yang Akan Digunakan

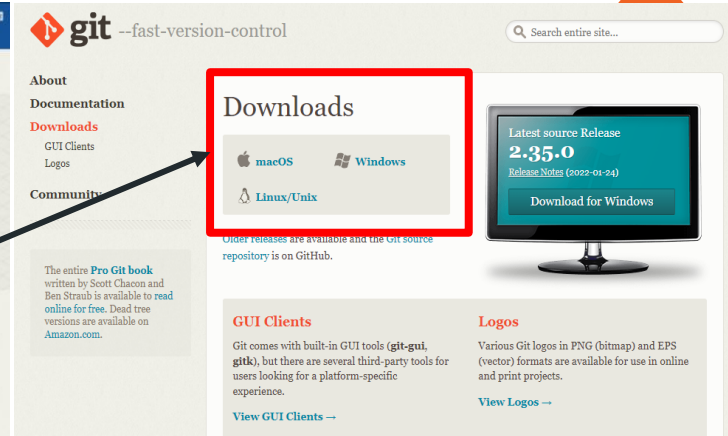
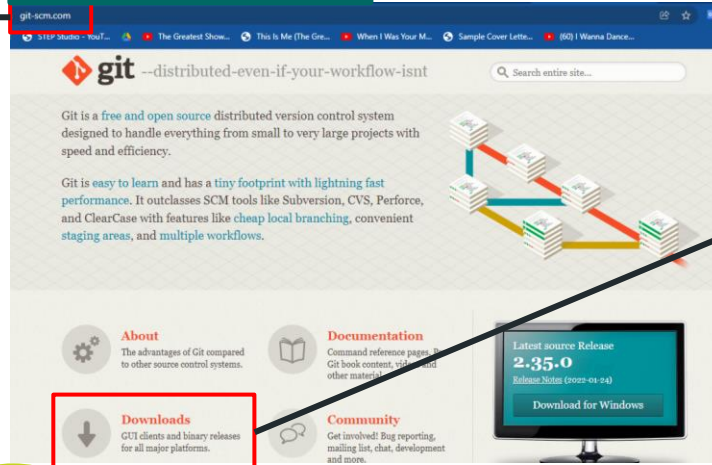


git



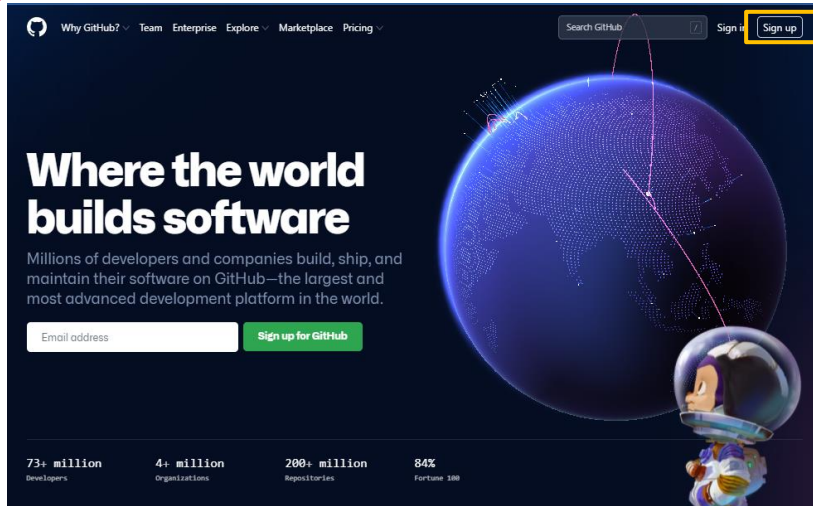
Instalasi GIT

www.git-scm.com



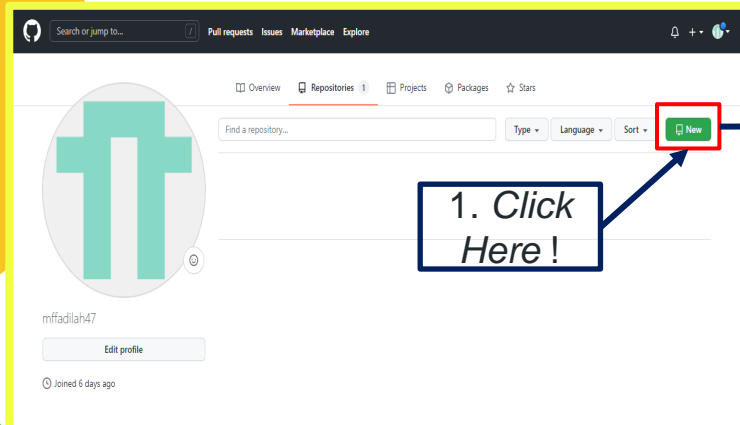
1. *Download* dan *install* GIT sesuai dengan OS komputer.
2. Buka GIT *installation files* dan ikuti petunjuk instalasinya.
3. Untuk OS windows, jangan lupa centang ***"install GIT bash"*** pada jendela instalasi.

Sign Up GitHub

A detailed view of the GitHub sign-up form. It starts with 'Welcome to GitHub! Let's begin the adventure'. The first section is 'Enter your email' with a green checkmark and the email 'fadilxoo@gmail.com'. The second section is 'Create a password' with a red 'x' and '****'. The third section is 'Enter a username' with a red 'x' and 'mffadilah47'. The fourth section asks 'Would you like to receive product updates and announcements via email?' with a red 'x' and 'y', and a 'Continue' button. The final section is 'Verify your account' with the text 'Please solve this puzzle to verify that you are human' and 'Click "Start puzzle" to continue', followed by a 'Start puzzle' button. At the bottom are two buttons: a refresh icon and a back icon.

1. Kunjungi Laman situs <https://github.com/>
2. Mulai proses *sign up* dan masukkan informasi yang diperlukan sesuai dengan petunjuk.
3. Lakukan verifikasi email.
4. Sign in ke akun GitHub.

Membuat Remote Repository di GitHub




Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *

Repository name *

 mffadilah47

/

Great repository names are short and memorable. Need inspiration? How about [ubiquitous-octo-invention?](#)

Description (optional)

☒ **Public**
 Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
 You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ **Add a README file**
 This is where you can write a long description for your project. [Learn more.](#)

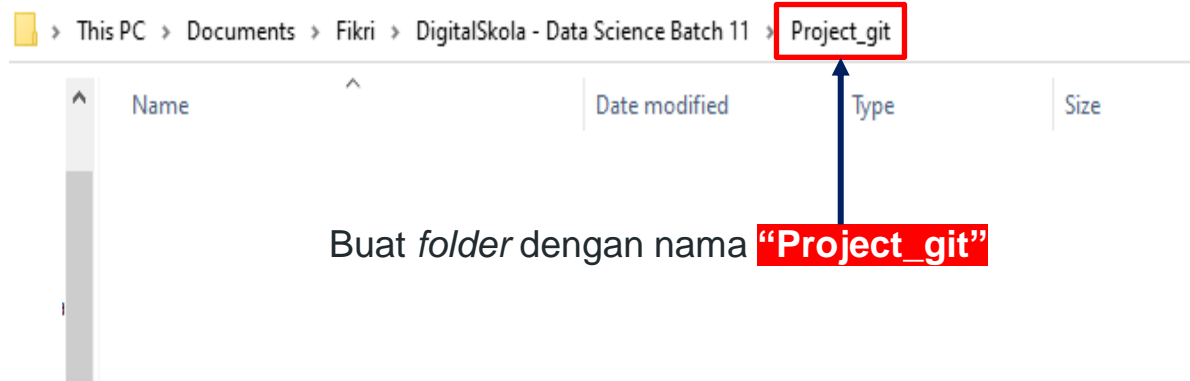
☐ **Add .gitignore**
 Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**
 A license tells others what they can and can't do with your code. [Learn more.](#)



Selamat ! Kamu berhasil membuat repository.

Buat *Folder* Untuk GIT Di Komputer



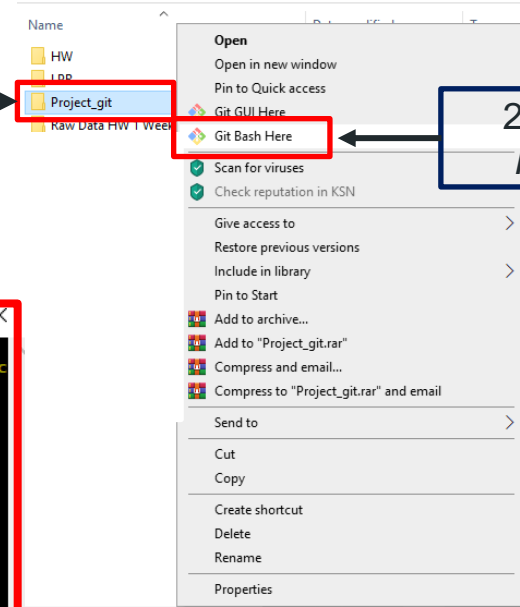
Running GIT

1. Right
Click Here !

2. Click
Here !

```
MINGW64:/c/Users/Fikri/Documents/Fikri/DigitalSkola - Data Science Batch ...  
Fikri@LAPTOP-17STP50H MINGW64 ~/Documents/Fikri/DigitalSkola - Data Science Batch ...  
h 11/Project_git (master)  
$ |
```

3. GIT
window is
opened.



GIT Configuration Commands

```
Fikri@LAPTOP-17STP50H MINGW64 ~/Documents/Fikri/DigitalSkola - Data Science Batc
h 11/Project_git (master)
$ git --version
git version 2.34.1.windows.1
```

Git --version
#Verify GIT version installed

```
Fikri@LAPTOP-17STP50H MINGW64 ~/Documents/Fikri/DigitalSkola - Data Science Batc
h 11/Project_git (master)
$ git config --global user.name "Muhammad Fikri Fadila"
```

Git config --global user.name "username"
#Configure GIT Username

```
Fikri@LAPTOP-17STP50H MINGW64 ~/Documents/Fikri/DigitalSkola - Data Science Batc
h 11/Project_git (master)
$ git config --global user.email "muh.fikrifadila@gmail.com"
```

Git config --global user.email "nama_email"
#Configure GIT email

```
Fikri@LAPTOP-17STP50H MINGW64 ~/Documents/Fikri/DigitalSkola - Data Science Batc
h 11/Project_git (master)
$ git config user.name
Muhammad Fikri Fadila
```

Git config user.name
#Verify GIT username

```
Fikri@LAPTOP-17STP50H MINGW64 ~/Documents/Fikri/DigitalSkola - Data Science Batc
h 11/Project_git (master)
$ git config user.email
muh.fikrifadila@gmail.com
```

Git config user.email
Verify GIT email

GIT Repository Commands

```
Fikri@LAPTOP-17STP5OH MINGW64 ~/Documents/Fikri/DigitalSkola - Data Science Batch 11/Project_git (master)
$ git init
Reinitialized existing Git repository in C:/Users/Fikri/Documents/Fikri/DigitalSkola - Data Science Batch 11/Project_git/.git/
```

Git init
#Create repository (local)

```
Fikri@LAPTOP-17STP5OH MINGW64 ~/Documents/Fikri/DigitalSkola - Data Science Batch 11/Project_git (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.
```

Git status
#Check status changes
(untuk memperlihatkan perubahan apa saja yang sedang/harus kita lakukan)

```
Fikri@LAPTOP-17STP5OH MINGW64 ~/Documents/Fikri/DigitalSkola - Data Science Batch 11/Project_git (master)
$ git log
commit 621caafa2629774b9599937608388ec8b97cab98 (HEAD -> master, origin/master)
Author: Muhammad Fikri Fadila <muh.fikrifadila@gmail.com>
Date: Mon Jan 24 18:54:59 2022 +0700

    Advanced SQL

commit 317a428e92db6669d97ebaf157bf05d064d0e739
Author: Muhammad Fikri Fadila <muh.fikrifadila@gmail.com>
Date: Mon Jan 24 18:53:31 2022 +0700

    Basic SQL

commit 78e181f5e531fc1bfff658b507cb438e6b82ab67c
Author: Muhammad Fikri Fadila <muh.fikrifadila@gmail.com>
Date: Mon Jan 24 06:02:01 2022 +0700

    coba coba
```

Git log
#Commit historical log
(untuk memperlihatkan perubahan apa saja yang telah kita lakukan. Atribut yang dapat diperlihatkan adalah **author**, **date**, **commit message**).

GIT Branching Commands

```
Fikri@LAPTOP-17STP50H MINGW64 ~/Documents/Fikri/DigitalSkola - Data Science Batch 11/Project_git (master)
$ git branch "branch_name"
```

Git branch "branch_name"
#Creating a new branch

```
Fikri@LAPTOP-17STP50H MINGW64 ~/Documents/Fikri/DigitalSkola - Data Science Batch 11/Project_git (master)
$ git branch -d "branch_name"
```

Git branch -d "branch_name"
#Delete branch (not yet being pushed to remote repository)

```
Fikri@LAPTOP-17STP50H MINGW64 ~/Documents/Fikri/DigitalSkola - Data Science Batch 11/Project_git (master)
$ git branch -D "branch_name"
```

Git branch -D "branch_name"
#Delete branch (either already or not yet being pushed to remote repository)

```
Fikri@LAPTOP-17STP50H MINGW64 ~/Documents/Fikri/DigitalSkola - Data Science Batch 11/Project_git (master)
$ git checkout master
Already on 'master'
M   Advanced SQL - DS DigitalSkola batch11.sql
Your branch is up to date with 'origin/master'.
```

Git checkout "branch_name"
#Go inside the branch

```
Fikri@LAPTOP-17STP50H MINGW64 ~/Documents/Fikri/DigitalSkola - Data Science Batch 11/Project_git (master)
$ git branch
* master
```

Git branch
#List down all active branch

GIT Execution Commands

```
Fikri@LAPTOP-17STP50H MINGW64 ~/Documents/Fikri/DigitalSkola - Data Science Batch
h 11/Project_git (master)
$ git add .
Fikri@LAPTOP-17STP50H MINGW64 ~/Documents/Fikri/DigitalSkola - Data Science Batch
h 11/Project_git (master)
$ git add "file_name"
```

Git add .
Git add "file_name"
#Adding file to staging area

```
Fikri@LAPTOP-17STP50H MINGW64 ~/Documents/Fikri/DigitalSkola - Data Science Batch
h 11/Project_git (master)
$ git rm --cached .
Fikri@LAPTOP-17STP50H MINGW64 ~/Documents/Fikri/DigitalSkola - Data Science Batch
h 11/Project_git (master)
$ git rm --cached "file_name"
```

Git rm --cached .
Git rm --cached "file_name"
#Remove file in staging area

```
Fikri@LAPTOP-17STP50H MINGW64 ~/Documents/Fikri/DigitalSkola - Data Science Batch
h 11/Project_git (master)
$ git commit -m "commit_message"
```

Git commit -m "commit_message"
#Commit to changes

```
Fikri@LAPTOP-17STP50H MINGW64 ~/Documents/Fikri/DigitalSkola - Data Science Batch
h 11/Project_git (master)
$ git commit -a -m "commit_message"
```

Git commit -a -m "commit_message"
#Add and commit all files changed

GIT Remote Commands

```
Fikri@LAPTOP-17STP5OH MINGW64 ~/Documents/Fikri/DigitalSkola - Data Science Batch 11/Project_git (master)
$ git remote add origin "HTTPS link from GitHub"
```

Git remote add origin
"HTTPS link from GitHub"
*#Integrating remote repository
to local repository*

```
Fikri@LAPTOP-17STP5OH MINGW64 ~/Documents/Fikri/DigitalSkola - Data Science Batch 11/Project_git (master)
$ git clone "HTTPS link from GitHub"
```

Git clone "HTTPS link from GitHub"
#Clone repository from GitHub

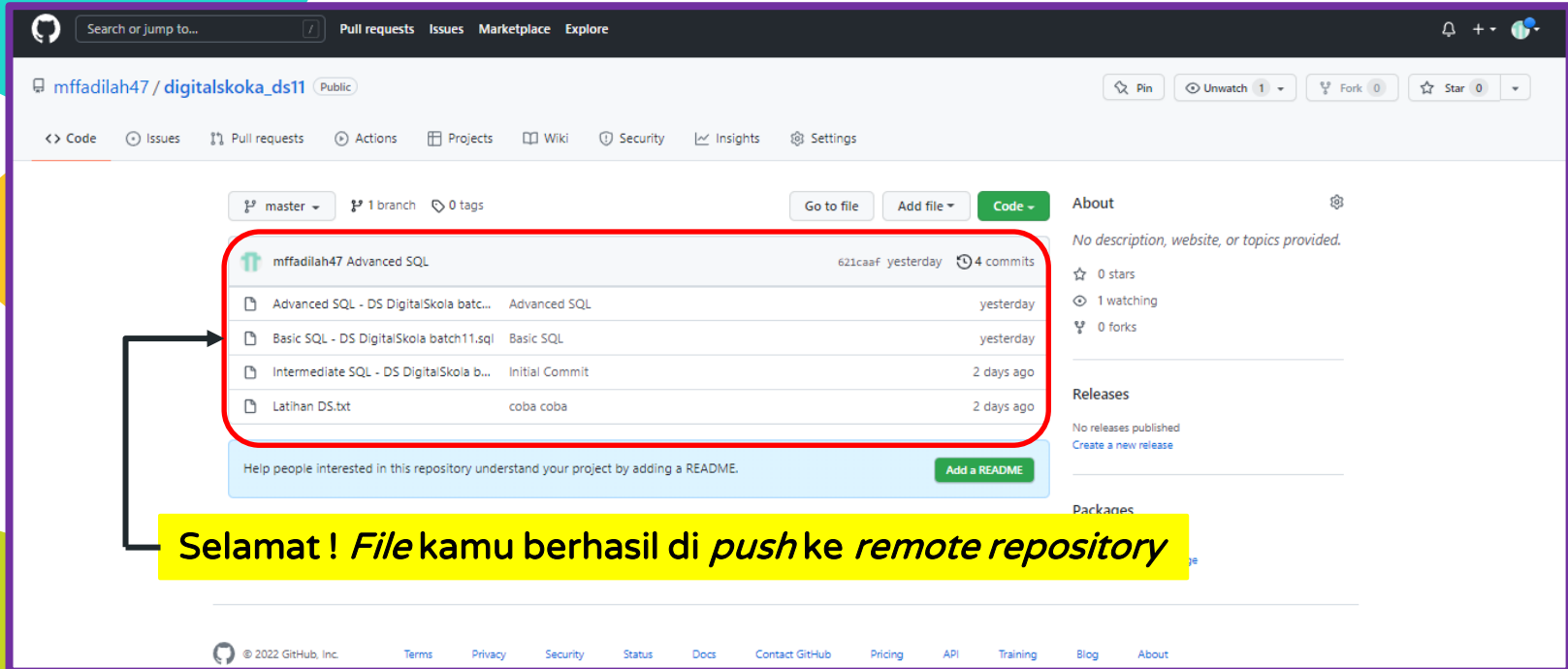
```
Fikri@LAPTOP-17STP5OH MINGW64 ~/Documents/Fikri/DigitalSkola - Data Science Batch 11/Project_git (master)
$ git push -u origin master
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 729 bytes | 104.00 KiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/mffadilah47/digitalskoka_ds11.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

Git push -u origin "branch_name"
Git push origin "branch_name"
#Push all commits to GitHub repository

```
Fikri@LAPTOP-17STP5OH MINGW64 ~/Documents/Fikri/DigitalSkola - Data Science Batch 11/Project_git (master)
$ git pull origin "branch_name"
```

Git pull origin "branch_name"
#Pull all remote repository to local repository

Contoh Tampilan GitHub Saat Berhasil Melakukan git push



The screenshot shows the GitHub interface for the repository 'mffadilah47 / digitalskoka_ds11'. The commit history table is highlighted with a red box, and a yellow box contains the following text:

Selamat ! *File* kamu berhasil di *push* ke *remote repository*

Commit Message	Author	Time
Advanced SQL - DS DigitalSkola batc...	Advanced SQL	yesterday
Basic SQL - DS DigitalSkola batch11.sql	Basic SQL	yesterday
Intermediate SQL - DS DigitalSkola b...	Initial Commit	2 days ago
Latihan DS.txt	coba coba	2 days ago

The background is white and decorated with various colorful circles and dashed lines. In the top left, there is a large orange circle with a dashed red outline, overlapping a yellow circle. Below them is a small pink circle. In the top center, a large red number '3' is centered within a large dashed light blue circle. In the top right, there is a green circle with a white dot inside, a small yellow circle, and a lime green circle with a dashed yellow outline. In the bottom left, there is a green circle with a dashed green outline, a large lime green circle, and a small cyan circle. In the bottom right, there is a large cyan circle with a white dot inside, and a small cyan circle with a dashed blue outline.

3

Introduction to Python

OUTLINE

PENGENALAN
PYTHON

1

2

PYTHON *BASIC*
STYLE GUIDE

3

PYTHON
OPERATORS

4

TIPE DATA
PADA PYTHON

1

PENGENALAN PYTHON

1. Apa itu Python?
2. Kegunaan Python
3. Mengapa Python?
4. Penggunaan Python dalam *Data Science*
5. Popular Python Notebook

APA ITU PYTHON?

- ◎ Python adalah bahasa pemrograman multifungsi yang dibuat oleh Guido van Rossum dan dirilis pada tahun 1991.
- ◎ Python menjadi Bahasa pemrograman yang populer karena sangat mudah dipahami dan fleksibel.

KEGUNAAN PYTHON

Dapat digunakan untuk:

- Mengakomodasi berbagai gaya pemrograman, termasuk structured, prosedural, berorientasi-objek, maupun fungsional.
- Dapat berjalan pada berbagai sistem operasi yang tersedia. Beberapa pemanfaatan bahasa Python di antaranya:

1. *Web development (server-side),*
2. *Software development,*
3. *Mathematics & data science,*
4. *Machine learning,*
5. *System scripting.*
6. *Internet of Things (IoT) development.*

Dapat digunakan untuk:

- Pembuatan aplikasi web di server.
- Membuat alur kerja bersama perangkat lunak.
- Membaca dan memodifikasi file dengan terhubung ke sistem database. .
- Menangani data besar dan melakukan matematika yang kompleks.
- Pembuatan prototipe cepat, atau untuk pengembangan perangkat lunak siap produksi.

MENGAPA PYTHON?

- ⦿ Dapat bekerja pada *platform* yang berbeda (Windows, Mac, Linux, dll).
- ⦿ Sempel sintaks (seperti bahasa Inggris).

Java	Python
<pre>public class Main { public static void main(String[] args) { System.out.println("Hello World"); } }</pre>	<pre>print('Hello World')</pre>
Output:	
<pre>Hello World</pre>	<pre>Hello World</pre>

- ⦿ Sistem Interpreter
 - ⦿ Kode dapat dieksekusi segera setelah ditulis.
 - ⦿ Pembuatan prototipe bisa sangat cepat.

PENGUNAAN PYTHON DALAM DATA SCIENCE

- ① *Data Collection & Cleaning*
CSV, TSV, JSON, dan *import* tabel dari SQL LANGSUNG ke *web code scrapping*.
- ② *Data exploration*
Untuk mengidentifikasi dan memisahkan data berdasarkan tipenya.
- ③ *Data Visualization & Interpretation*
Penggunaan *library Plotly* untuk *men-generate* grafik dasar dan diagram.
- ④ *Data Modelling*
Membantu proses *machine learning* untuk melaksanakan perintah yang terkait *data modelling*.
- ⑤ *Deploying*
Mengubah model pada bahasa yang bermakna dan dapat dihapami oleh sistem maupun end user menggunakan Flask.

POPULAR PYTHON NOTEBOOK



Jupyter
Notebook



Google
Colaboratory

2

PYTHON BASIC STYLE GUIDE

1. Indentasi
2. Mengganti baris: Sebelum Operator Binary
3. Whitespace pada Expressions dan Statements
4. Komentar

Indentasi

- ◎ Gunakan 4 spasi/ 1 Tab pada setiap tingkatan indentasi.
- ◎ Statement yang memiliki indentasi yang sama dan diletakkan secara berurutan dikenali sebagai blok statement oleh Python dan akan dijalankan secara berurutan.

1. **Statement** tingkat 1:
2. **Statement** tingkat 2()
3. **Statement** tingkat 2 yang kedua()

Mengganti baris: Sebelum Operator Binary

Mempermudah pembaca kode mengerti operasi yang dilakukan terhadap variabel berikutnya.

```
1. income = (gross_wages
2.         + taxable_interest
3.         + (dividends - qualified_dividends)
4.         - ira_deduction
5.         - student_loan_interest)
```

Whitespace pada Expressions dan Statements

- Wajib dihindari penambahan whitespace yang tidak perlu.
- Antara kurung, kurawal, kurung siku.

- Yes:** `spam(ham[1], {eggs: 2})`
- No:** `spam(ham[1], { eggs: 2 })`

- Setelah koma, tanpa argumen lain setelahnya.

- Yes:** `foo = (0,)`
- No:** `bar = (0,)`

- Sebelum koma, titik dua, atau titik koma.

- Yes:** `if x == 4: print x, y; x, y = y, x`
- No:** `if x == 4 : print x , y ; x , y = y , x`

Whitespace pada Expressions dan Statements

⦿ Penggunaan titik dua/colon sebagai slice (sub-list), pastikan memiliki spasi/whitespace yang

1. **Yes:**

2. `ham[1:9], ham[1:9:3], ham[:9:3], ham[1::3], ham[1:9:]`
3. `ham[lower:upper], ham[lower:upper:], ham[lower::step]`
4. `ham[lower+offset : upper+offset]`
5. `ham[: upper_fn(x) : step_fn(x)], ham[: : step_fn(x)]`
6. `ham[lower + offset : upper + offset]`
- 7.

8. **No:**

9. `ham[lower + offset:upper + offset]`
10. `ham[1 :9], ham[1 :9], ham[1:9 :3]`
11. `ham[lower : : upper]`
12. `ham[: upper]`

⦿ Saat memberikan parameter pada fungsi, sebelum kurung tidak boleh

1. **Yes:** `spam(1)`
2. **No:** `spam (1)`

⦿ Saat memberikan parameter/index pada list, sebelum kurung siku tidak boleh ada spasi.

1. **Yes:** `dct["key"] = lst[index]`
2. **No:** `dct ["key"] = lst [index]`

Whitespace pada Expressions dan Statements



Saat membuat assignment pada variabel, sebaiknya tidak menambahkan whitespace yang tidak perlu.

1. **Yes:**

2. `x = 1`

3. `y = 2`

4. `long_variable = 3`

5.

6. **No:**

7. `x = 1`

8. `y = 2`

9. `long_variable = 3`

1. **Yes:**

2. `i = i + 1`

3. `submitted += 1`

4. `x = x*2 - 1`

5. `hypot2 = x*x + y*y`

6. `c = (a+b) * (a-b)`

7.

8. **No:**

9. `i=i+1`

10. `submitted +=1`

11. `x = x * 2 - 1`

12. `hypot2 = x * x + y * y`

13. `c = (a + b) * (a - b)`

Komentar



Blok Komentar

Untuk menjelaskan fungsi utuh atau sub-fungsi yang mengikuti/berada di bawahnya. Blok komentar diindentasi setara dengan kode yang dijelaskan. Setiap barisnya diawali dengan # dan sebuah spasi serta setiap paragrafnya dimulai pada baris baru.



Komentar Inline



- Diletakkan pada baris yang sama dengan kode.



- Dipisahkan dan dirapikan dengan jarak dua spasi dari kode yang dimaksud, diawali # dan sebuah spasi.



- Komentar inline dapat juga digunakan di atas baris yang ingin diberikan komentar, agar tidak mengurangi jumlah karakter yang dapat dituliskan dalam sebuah baris.

Komentar

- ⦿ Untuk semua jenis komentar, jangan menuliskan komentar untuk hal yang sudah langsung dapat dibaca dari kodenya, seperti contoh berikut:

- ⦿ Tidak disarankan:

1. $x = x + 1$	# Tambahkan x

- ⦿ Disarankan (kontekstual):

1. $x = x + 1$	# Mengakomodasi layar ukuran Z



3

PYTHON *OPERATORS*

1. Python *Arithmetic Operators*
2. Python *Assignment Operators*

Python *Arithmetic Operators*

Operator	Nama	Contoh
+	Penambahan	$x + y$
-	Pengurangan	$x - y$
*	Perkalian	$x * y$
/	Pembagian	x / y
%	Modulus	$x \% y$
**	Eksponen	$x ** y$
//	Floor division	$x // y$

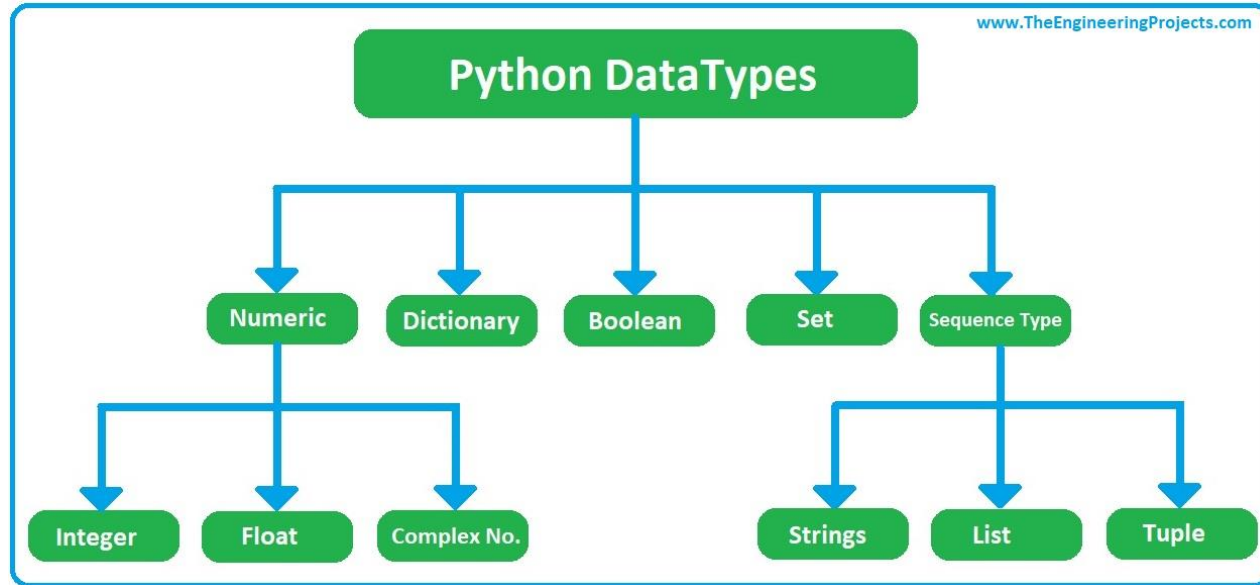
Python *Assignment Operators*

Operator	Contoh	Penulisan Lain
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
=	x = 3	x = x 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

4

TIPE DATA PADA PYTHON

1. Numbers
2. String
3. Bool / Boolean
4. List []



Numbers

1. Tipe numerik pada Python dibagi menjadi 3:
2. Int (*Integer*)
3. Float

```
1. a = 10
2. print(a, "bertipe", type(a))
3. b = 1.7
4. print(b, "bertipe", type(b))
5. c = 1+3j
6. print(c, " Bertipe bilangan kompleks? ", isinstance(1+3j,complex))
```

Output:

10 bertipe <class 'int'>

1.7 bertipe <class 'float'>

(1+2j) Bertipe bilangan kompleks? True

Strings

- ◎ String adalah urutan dari karakter unicode yang dideklarasikan dengan petik tunggal atau ganda.
- ◎ String > 1 baris dapat ditandai dengan tiga petik tunggal atau ganda `'''` atau `"""`.

```
1. s = "Ini adalah string baris tunggal"
```

```
1. s = '''Ini adalah string  
2. yang memiliki baris pertama  
3. dan selanjutnya baris kedua'''
```

Bool / Boolean

- ⦿ Tipe data bool atau Boolean merupakan turunan dari bilangan bulat (integer atau int) yang hanya punya dua nilai konstanta: True dan False.
- ⦿ **Nilai Boolean**
Nilai konstanta False dan True merepresentasikan nilai kebenaran (truth values), meskipun ada nilai-nilai lain yang juga dianggap benar atau salah.
- ⦿ Berikut adalah objek bawaan yang didefinisikan bernilai salah dalam pengujian nilai kebenaran:
 - ⦿ Konstanta yang sudah didefinisikan bernilai salah: None dan False.
 - ⦿ Angka nol dari semua tipe numeric: 0, 0.0, 0j, Decimal(0), Fraction(0, 1).
 - ⦿ Urutan (sequence) dan koleksi (collection) yang kosong: "", (), {}, set(), range(0).

List []

- List adalah jenis kumpulan data terurut (*ordered sequence*).
- Setiap data di dalamnya dapat diakses dengan **indeks** yang dimulai dari 0.

1.	<code>x = [5,10,15,20,25,30,35,40]</code>	Output:
2.	<code>print(x[5])</code>	30
3.	<code>print(x[-1])</code>	40
4.	<code>print(x[3:5])</code>	[20, 25]
5.	<code>print(x[:5])</code>	[5, 10, 15, 20, 25]
6.	<code>print(x[-3:])</code>	[30, 35, 40]
7.	<code>print(x[1:7:2])</code>	[10, 20, 30]

List []

Mengubah elemen pada List

1. `x = [1,2,3]`
2. `x[2]=4`
3. `print(x)`

Output:

[1, 2, 4]

Menambahkan elemen pada List

1. `x = [1,2,3]`
2. `x[2]=4`
3. `x.append(5)`
4. `print(x)`

Output:

[1, 2, 4, 5]

Menghapus elemen pada List

1. `binatang = ['kucing', 'rusa', 'badak', 'gajah']`
2. `del binatang[2]`
3. `print(binatang)`

Output:

['kucing', 'rusa', 'gajah']

Tuple ()

- ◎ Tuple adalah jenis dari list yang tidak dapat diubah elemennya. Umumnya tuple digunakan untuk data yang bersifat sekali tulis, dan dapat dieksekusi lebih cepat.
- ◎ Seperti list, kita dapat melakukan *slicing*, namun pada tuple kita tidak dapat melakukan perubahan:

```
1. t = (5,'program', 1+3j)
2. print(t[1])
3. print(t[0:3])
4. print(t[0]=10)
```

Output:

'program'

(5, 'program', (1+3j))

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: 'tuple' object does not support item assignment

Set { }

- Set adalah kumpulan item bersifat unik dan tanpa urutan (*unordered collection*).
- Dapat melakukan *union* dan *intersection*, sekaligus otomatis melakukan penghapusan data duplikat.

- Karena set bersifat *unordered*, maka kita tidak bisa mengambil sebagian data / elemen datanya menggunakan proses *slicing*.

- `a = {1,2,3}`
- `print(a[1])`

Output:

Traceback (most recent call last):

File "<string>", line 301, in runcode

File "<interactive input>", line 1, in <module>

TypeError: 'set' object does not support indexing

- `a = {1,2,2,3,3,3}`
- `print(a)`

Output:

{1, 2, 3}

Dictionary { }

Kumpulan pasangan kunci-nilai (*pair of key-value*) yang bersifat tidak berurutan. Dictionary dapat digunakan untuk menyimpan data kecil hingga besar. Untuk mengakses datanya, harus mengetahui kuncinya (*key*). Berikut tambahan definisi berikut:

- Setiap elemen *pair key-value* dipisahkan dengan koma (,).
- key* dan *value* dipisahkan dengan titik dua (:).
- key* dan *value* dapat berupa tipe variabel/obyek apapun.

Dictionary bukan termasuk dalam implementasi urutan (*sequences*), sehingga tidak bisa dipanggil dengan urutan indeks.

```
1. d = {'value':'key':2}
2. print(type(d))
3. print("d[1] = ", d[1]);
4. print("d['key'] = ", d['key']);
5.
6. # Generates error
7. print("d[2] = ", d[2]);
```

Output:

```
<class 'dict'>
d[1] = value
d['key'] = 2
```

```
-----
KeyError Traceback (most recent call last)
<ipython-input-7-4b566e677ca2> in <module>()
1 d = {'value':'key':2}
----> 2 print("d[2] = ", d[2]);
```

```
KeyError: 2
```

Terima Kasih !

