

```
geoscope-geohazard-workshop (/github/anangsahroni/geoscope-geohazard-workshop/tree/main)
/
1_Instalasi_Python_dalam_Miniconda.ipynb (/github/anangsahroni/geoscope-geohazard-workshop/tree/main/1_Instalasi_Python_dalam_Miniconda.ipynb)
```

Geoscope Geohazard Workshop HMGF UGM

Instalasi Miniconda dan Jupyter Notebook

- Tujuan: Memperkenalkan Conda dan Jupyter Notebook
- Keluaran: Peserta dapat menjalankan Python dalam berbagai antarmuka terutama Jupyter Notebook
- Sesi: 1
- Waktu/Tempat: Sabtu, 13 Februari 2021/ Zoom Meeting

Instalasi Miniconda pada Windows, Linux, ataupun MacOS

Python merupakan bahasa pemrograman yang akan kita pelajari, lantas mengapa kita harus menginstall Miniconda? Miniconda adalah versi ringan dari Anaconda yang dikembangkan oleh Anaconda Distribution. Miniconda (atau lebih umum disebut conda) merupakan perangkat lunak untuk manajemen paket-paket (*packages*) dalam Python. Apakah yang dimaksud paket (*package*) dalam Python? Mari kita install terlebih dahulu Miniconda untuk mempelajarinya.

Instalasi Miniconda diawali dengan mengunduh installer Miniconda pada [link \(<https://docs.conda.io/en/latest/miniconda.html>\)](https://docs.conda.io/en/latest/miniconda.html) berikut ini, pilih untuk versi Python 3.8 baik untuk Windows, MacOS, maupun Linux. Contoh dalam gambar di bawah ini adalah untuk Windows 64bit.

The screenshot shows the official Miniconda download page. On the left, there's a sidebar with links like Conda, Conda-build, and Miniconda. Under Miniconda, there are links for Windows installers, Mac OSX installers, Linux installers, and other resources. Below that are links for Help and support, Contributing, and Conda license. At the bottom, there's a "Read the Docs" link and a dropdown menu set to "v: latest". A large orange arrow points from the "Windows installers" heading towards the download progress bar at the bottom. The progress bar shows "Miniconda3-late...exe" and "13.5/57.0 MB, 3 mins left".

Miniconda

Miniconda is a free minimal installer for conda. It is a small, boots other useful packages, including pip, zlib and a few others. Use it

[See if Miniconda is right for you.](#)

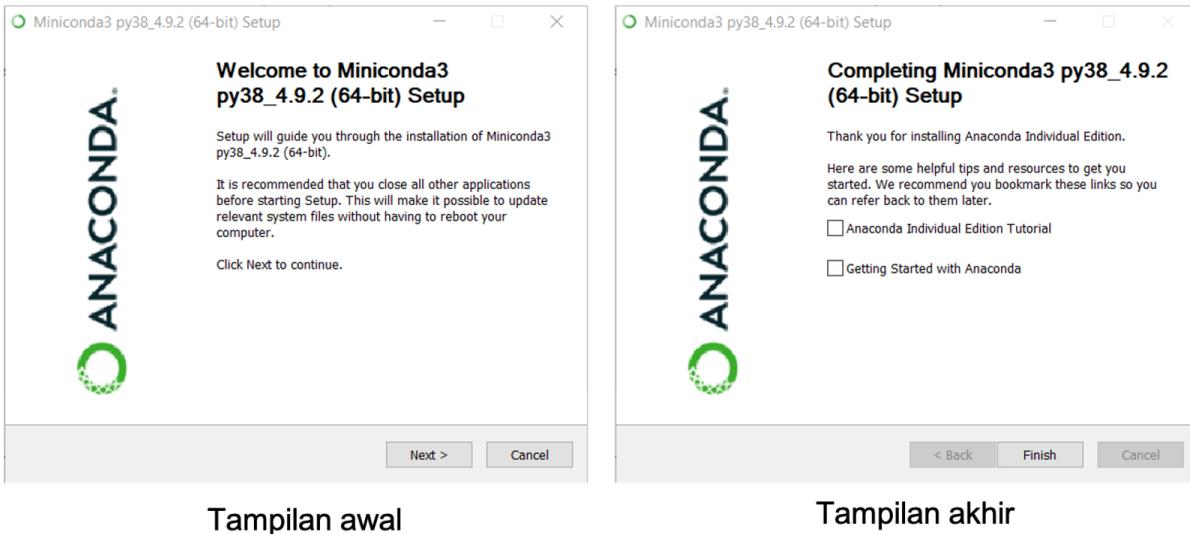
Windows installers

Python version	Name	Size
Python 3.8	Miniconda3 Windows 64-bit	57.0 MiB
	Miniconda3 Windows 32-bit	54.2 MiB
Python 2.7	Miniconda2 Windows 64-bit	54.1 MiB
	Miniconda2 Windows 32-bit	47.7 MiB

MacOSX installers

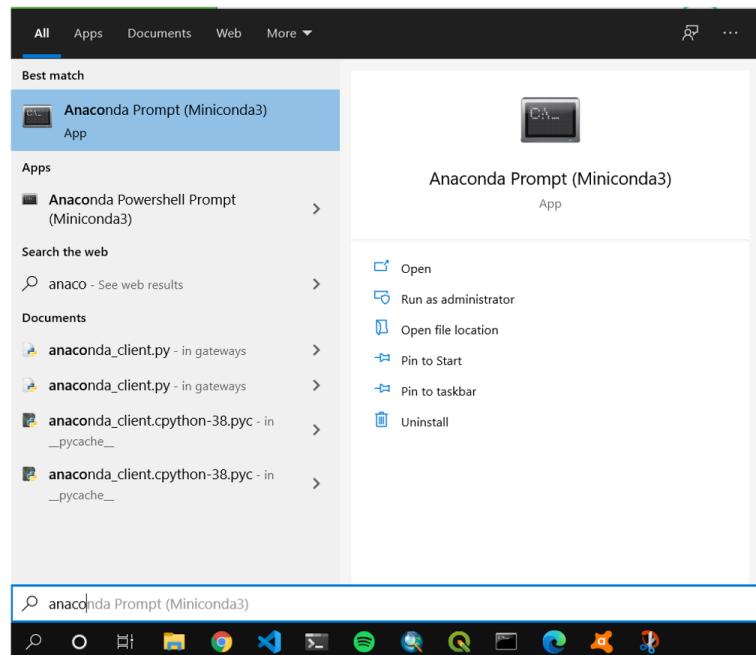
Python version	Name	Size
Python 3.8	Miniconda3 Mac OSX 64-bit hash	54.5 MiB

Setelah *installer* berhasil diunduh maka langkah selanjutnya adalah menjalankan installer `Miniconda3-latest-Windows-x86_64.exe`. Jendela *installer* akan muncul, ikuti petunjuknya dengan menekan `Next` sampai instalasi berhasil, seperti perangkat lunak pada umumnya.



Menjalankan Python pada Anaconda Prompt

Setelah **Miniconda** berhasil diinstall pada perangkat kita maka akan muncul beberapa aplikasi baru, salah satunya adalah **Anaconda Prompt**. **Anaconda Prompt** ini adalah aplikasi utama yang akan sering kita gunakan. Tampilan pada **Anaconda Prompt** ini tidak ada bedanya dengan Command Prompt pada Windows atau Terminal pada MacOS dan Linux. Hal paling utama yang membedakan adalah pada **Anaconda Prompt** ini Python sudah terintegrasi dan bisa kita jalankan. Buka **Anaconda Prompt** melalui menu:



Untuk menjalankan Python maka kita dapat mengetikkan `python` dan menekan tombol enter pada **Anaconda Prompt** sehingga akan muncul antarmuka untuk Python. Antarmuka ini adalah tampilan paling sederhana dari Python, dapat kita sebut sebagai **Python Console**, kok tampilannya rumit ya? Tenang, nanti kita akan bermain kode dengan tampilan yang lebih *friendly* 😊 tapi sekarang kita coba dulu dari yang paling sederhana.

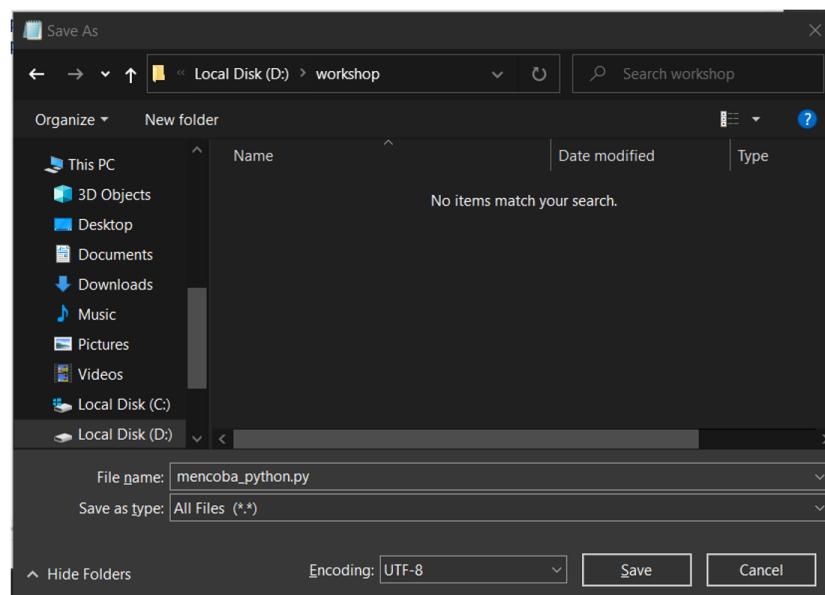
```
(base) C:\Users\NITRON-5>python
Python 3.8.5 (default, Sep 3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 1+1
2
>>> print("Hello World")
Hello World
>>>
```

Apa yang terjadi ketika kita mengetikkan `1+1` pada tampilan tersebut? Akan muncul hasil dari penjumlahan kedua angka tersebut yaitu `2`. Selamat anda sudah berhasil menjalankan Python! Kita bisa mencoba bermacam-macam ekspresi matematik yang sederhana sekadar untuk belajar. Bagaimana kalau yang diketik adalah `print("Hello World")` ?

Menuliskan Kode dalam Editor Teks dan Menjalankan Kode tersebut di Anaconda Prompt

Oke mari kita naik level, setelah menjalankan kode langsung melalui Python Console sekarang kita akan menuliskan kode di editor teks dan menjalankannya di **Anaconda Prompt**. Teman-teman dapat membuka editor teks masing-masing (seperti **Notepad**) dan coba ketikan kode berikut:

```
print("Mencoba menjalankan kode yang diketik di editor")
print("Berhasil!")
```



Kemudian simpan sebagai file dengan nama `mencoba_python.py` di lokasi yang teman-teman inginkan, misalnya disini saya simpan di `D:\workshop`, jangan lupa pilih `All files` pada pilihan `Save as type`. Setelah berhasil disimpan saatnya kita menjalankan kode Python yang sudah kita tuliskan di editor kode di dalam Anaconda Prompt. Saat kita membuka tampilan Anaconda Prompt maka biasanya kita akan berada

di drive C , karena file saya simpan di D maka saya akan mengetikkan:

```
D:  
cd workshop
```

maka kita akan masuk ke dalam folder tersebut, langkah ini bisa disesuaikan tergantung lokasi tempat teman-teman menyimpan file mencoba_python.py yang sebelumnya telah kita buat. Setelah masuk ke dalam folder tempat kita menyimpan maka selanjutnya kita dapat menjalankan kode kita dengan:

```
python mencoba_python.py
```

The screenshot shows a terminal window titled "Anaconda Prompt (Miniconda3)". The command history and output are as follows:

```
(base) C:\Users\NITRON-5>D:  
(base) D:\>cd workshop  
(base) D:\workshop>python mencoba_python.py  
Mencoba menjalankan kode yang diketik di editor  
Berhasil  
(base) D:\workshop>
```

Apakah yang terjadi?

Setelah sampai di level ini teman-teman sudah berhasil naik ke Level 2. Mari kita lanjutkan ke level berikutnya :D

Pengenalan IDE (*Integrated Development Environment*) dan Beberapa Contohnya

Dimana biasanya kita menuliskan kode kita sebelum kita jalankan? Ada banyak alternatif tempat kita menuliskan kode, salah satu yang paling sederhana adalah menuliskan kode kita di editor teks seperti Notepad apabila di Windows. Ada banyak aplikasi yang memang dikhawasukan untuk menuliskan kode dimana terdapat berbagai macam fitur untuk mendukung produktivitas seperti pewarnaan kode (syntax highlighting), indentasi otomatis, dan debugging. Aplikasi-aplikasi tersebut bisa kita sebut sebagai IDE, beberapa IDE yang cukup umum digunakan untuk Python adalah:

1. PyCharm
2. Spyder
3. Visual Studio Code

Mana yang akan kita gunakan? Jawabannya adalah tidak ketiganya

Kita pakai apa dong? Kita akan menggunakan **Jupyter Notebook**, aplikasi ini hampir mendekati IDE tetapi tidak memiliki fitur yang lengkap untuk disebut sebagai IDE,

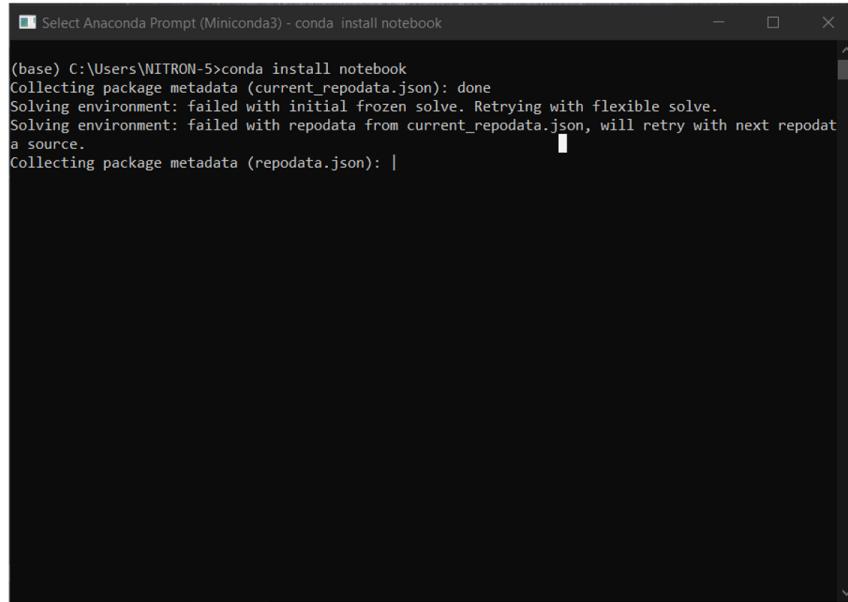
Mengapa kita memakai **Jupyter Notebook**? Aplikasi ini memungkinkan kita mencatat sekaligus menuliskan kode dan menjalankannya. Hal ini mempermudah dalam rangka belajar karena penjelasan akan mempermudah dalam memahami kode.

Agar tidak penasaran kita dapat menginstall **Jupyter Notebook** sekarang :D

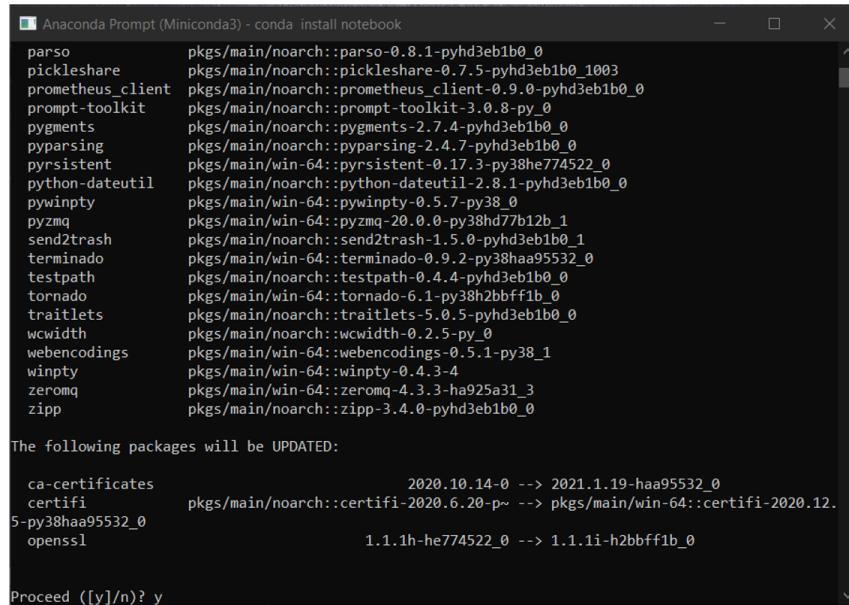
Menginstall dan Menjalankan Jupyter Notebook

Jupyter Notebook merupakan salah satu paket (*package*) yang dapat diinstall dan memiliki fungsi untuk memberikan antarmuka pengguna grafis dalam penulisan kode dan membuat catatan bersama dengan kode tersebut. Sebuah paket bisa kita bayangkan sebagai kumpulan kode Python yang memiliki fungsi yang spesifik. Karena sudah dikemas dalam sebuah paket, maka kita hanya perlu memanggil paket tersebut saat akan menggunakan, salah satunya **Jupyter Notebook** ini. Anaconda mempermudah kita dalam menginstall sebuah paket atau modul tertentu, untuk menginstall paket **Jupyter Notebook** kita dapat mengetikkan perintah sebagai berikut di **Anaconda Prompt**:

```
conda install -c conda-forge notebook
```

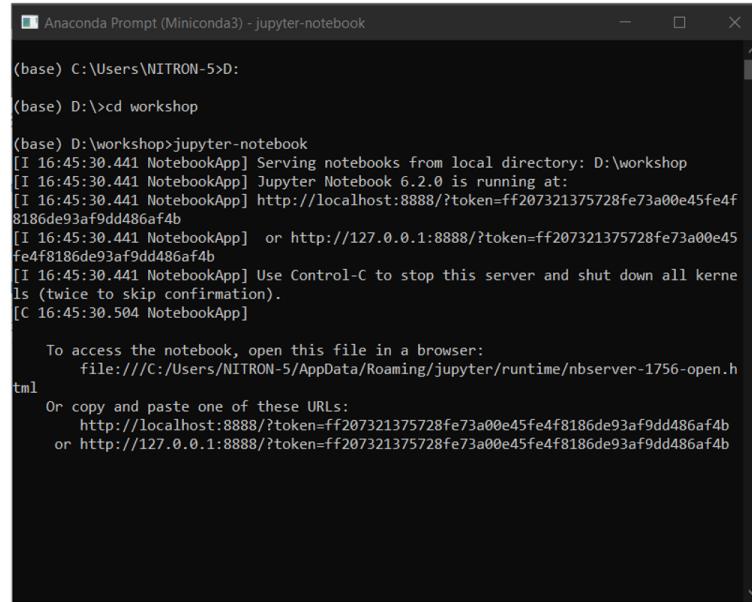


Setelah kita mengetikkan dan menekan enter maka akan muncul keterangan atau log bahwa paket **Jupyter Notebook** sedang diinstall. Saat ada pertanyaan seperti gambar di bawah ini, ketik `y` kemudian tekan enter.



Setelah proses selesai kita dapat pindah ke folder tempat kita akan menyimpan *notebook* kemudian menjalankan **Jupyter Notebook** dengan mengetikkan perintah di Anaconda Prompt kemudian tekan enter:

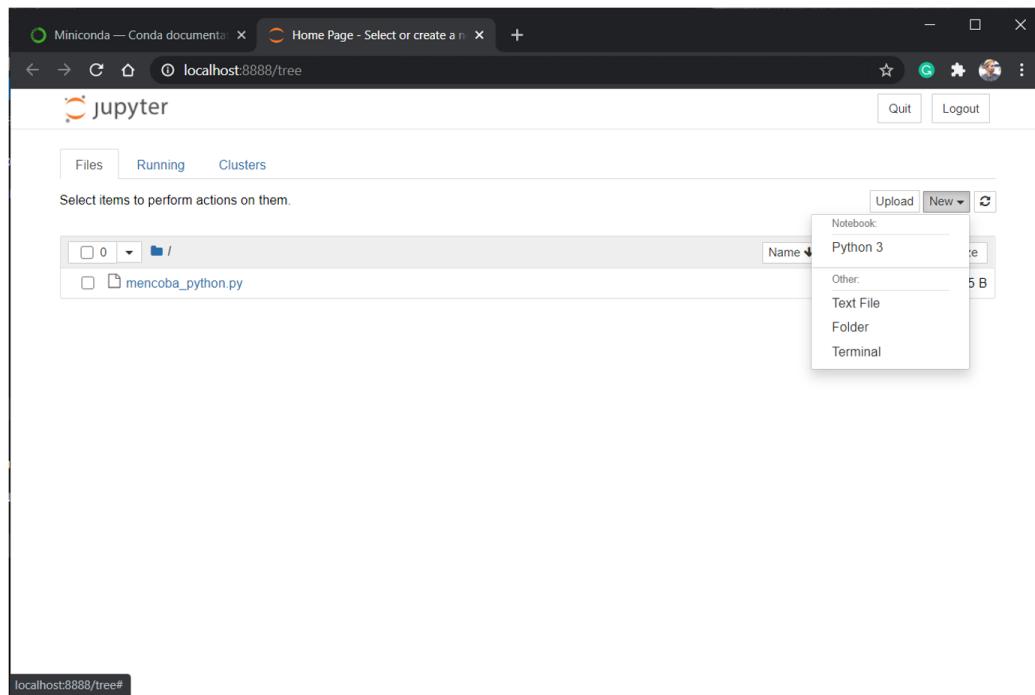
```
D:
cd workshop
jupyter-notebook
```



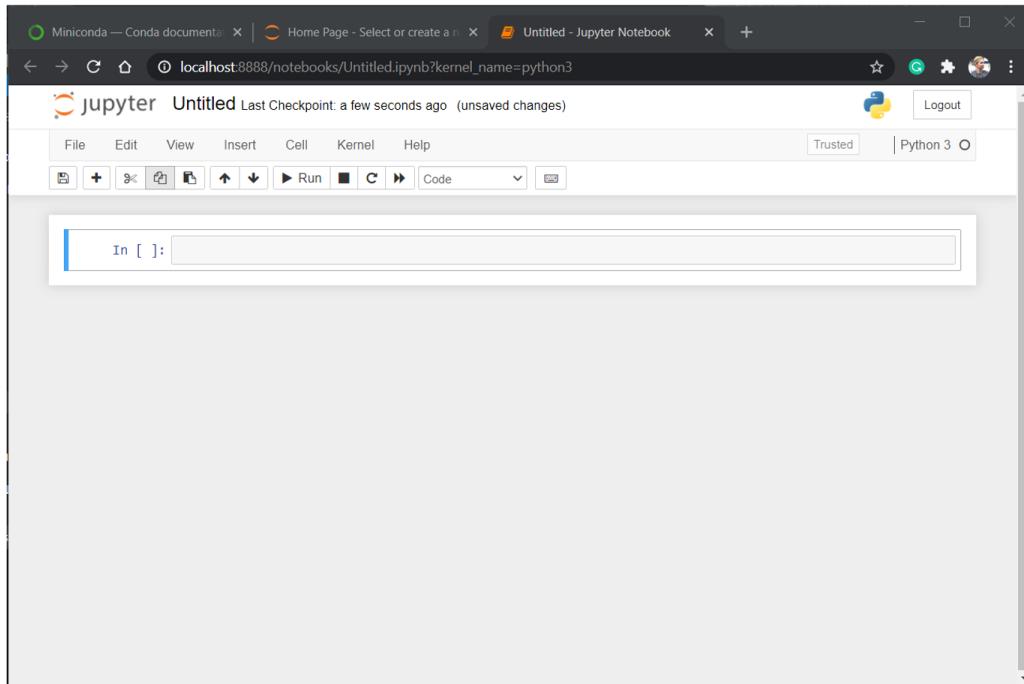
```
(base) C:\Users\NITRON-5>D:
(base) D:\>cd workshop
(base) D:\workshop>jupyter-notebook
[I 16:45:30.441 NotebookApp] Serving notebooks from local directory: D:\workshop
[I 16:45:30.441 NotebookApp] Jupyter Notebook 6.2.0 is running at:
[I 16:45:30.441 NotebookApp] http://localhost:8888/?token=ff207321375728fe73a00e45fe4f
8186de93af9dd486af4b
[I 16:45:30.441 NotebookApp] or http://127.0.0.1:8888/?token=ff207321375728fe73a00e45
fe4f8186de93af9dd486af4b
[I 16:45:30.441 NotebookApp] Use Control-C to stop this server and shut down all kerne
ls (twice to skip confirmation).
[C 16:45:30.504 NotebookApp]

To access the notebook, open this file in a browser:
  file:///C:/Users/NITRON-5/AppData/Roaming/jupyter/runtime/nbserver-1756-open.h
tml
Or copy and paste one of these URLs:
  http://localhost:8888/?token=ff207321375728fe73a00e45fe4f8186de93af9dd486af4b
or http://127.0.0.1:8888/?token=ff207321375728fe73a00e45fe4f8186de93af9dd486af4b
```

Selanjutnya antarmuka **Jupyter Notebook** akan muncul pada browser seperti Chrome, Firefox, atau Safari seperti pada gambar di bawah ini:



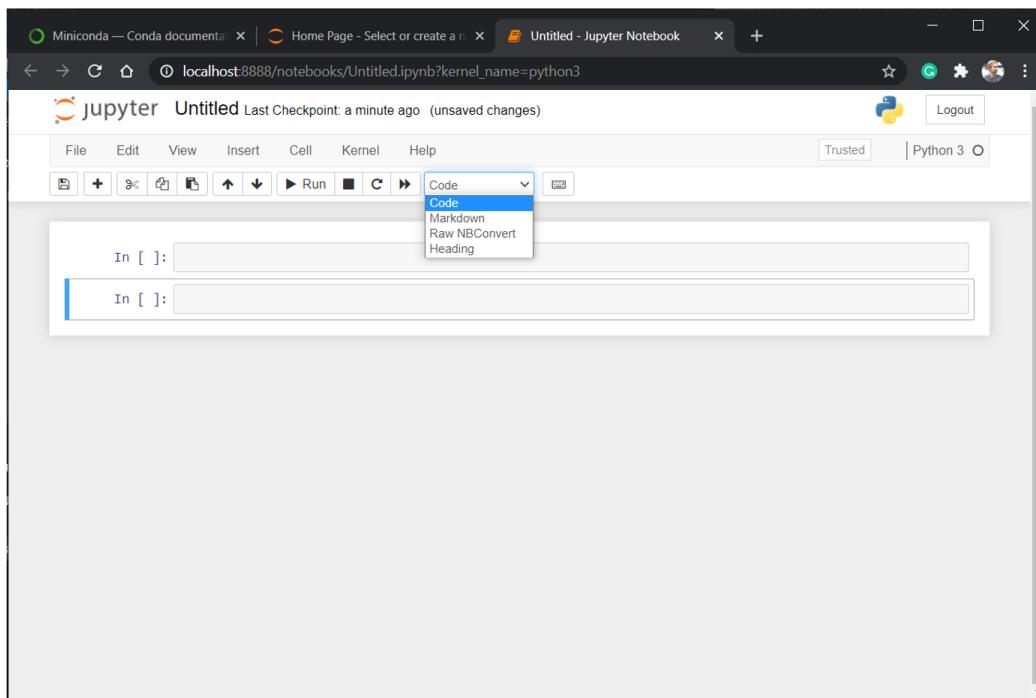
Untuk mulai membuat *notebook* baru kita dapat menekan tombol `New` kemudian pilih `Python` maka *notebook* akan muncul sebagai tab baru:



Menjalankan kode sederhana di Jupyter Notebook

Kita dapat mengetikkan kode pada setiap `cell` yang ada di notebook, jenis `cell` dapat dibagi menjadi:

1. `Code`, jenis `cell` ini berguna untuk menuliskan kode Python yang akan kita eksekusi
2. `Markdown`, markdown berguna untuk menambahkan berbagai macam catatan pada notebook, markdown memiliki banyak fitur dan formatting yang dapat kita sesuaikan sesuai dengan kebutuhan catatan kita seperti tabel, gambar, list, dsb



Karena kita akan menuliskan kode Python maka kita akan memilih jenis `cell` `Code` dan coba kita ketikkan kode berikut:

```
print("Mencoba menulis kode di Jupyter Notebook")
print("Sukses~")
```

Kita dapat menjalankan kode tersebut dengan menekan tombol segitiga atau tombol `play` (▶) di bagian atas. Kode akan berjalan dan keluaran output akan keluar, setelah ini berhasil berarti teman-teman telah naik ke level 3 😊 . Untuk penulisan kode pada materi-materi selanjutnya akan menggunakan **Jupyter Notebook** ini, sekaligus agar teman-teman bisa sambil mencatat materi yang diberikan. Beberapa percobaan lain:

In [15]:

```
print("Hari ini kita akan menjumlahkan dua bilangan")
bilangan1 = 10
bilangan2 = 30
jumlah = bilangan1 + bilangan2
print("Hasil penjumlahan", bilangan1, "dan", bilangan2, "adalah", jumlah)
```

Hari ini kita akan menjumlahkan dua bilangan
Hasil penjumlahan 10 dan 30 adalah 40

Bagaimana jika bilangan yang akan dijumlahkan kita ketikkan sendiri?

In [5]:

```
print("Hari ini kita akan menjumlahkan dua bilangan")
bilangan1 = input("Masukkan bilangan pertama")
bilangan2 = input("masukkan bilangan kedua")
jumlah = bilangan1 + bilangan2
print("Hasil penjumlahan", bilangan1, "dan", bilangan2, "adalah", jumlah)
```

Hari ini kita akan menjumlahkan dua bilangan
Hasil penjumlahan 10 dan 20 adalah 1020

Jika hasilnya aneh dan tidak sesuai dengan yang teman-teman harapkan berarti Pythonnya bekerja dengan baik. Hasil dari fungsi `input` adalah data dengan format `String` atau sederhananya kita sebut sebagai bukan angka, sedangkan yang akan kita jumlahkan seharusnya adalah angka, bisa berupa bilangan bulat atau `Integer` atau bilangan desimal atau `Float`. Apa yang terjadi ketika kita menjumlahkan dua `String` ?

In [6]:

```
print("Pada cell ini kita akan menjumlahkan dua buah")
bilangan1 = "1"
bilangan2 = "20"
jumlah = bilangan1 + bilangan2
print("Hasil penjumlahan string adalah", jumlah)
```

Pada cell ini kita akan menjumlahkan dua buah
Hasil penjumlahan string adalah 120

Hasilnya sama seperti saat menjumlahkan hasil dari fungsi `input` bukan? Lalu bagaimana cara agar hasil `input` merupakan angka dan bukan `String` ?

In [7]:

```
print("Hari ini kita akan menjumlahkan dua bilangan bulat")
bilangan1 = input("Masukkan bilangan pertama")
bilangan2 = input("masukkan bilangan kedua")
jumlah = int(bilangan1) + int(bilangan2)
print("Hasil penjumlahan bilangan bulat", bilangan1, "dan", bilangan2, "adalah", jumlah)
```

Hari ini kita akan menjumlahkan dua bilangan bulat
Hasil penjumlahan bilangan bulat 10 dan 30 adalah 40

Hasilnya sudah seperti yang kita harapkan, ini karena hasil `input` atau yang kita masukkan sudah diubah menjadi bilangan bulat `Integer` saat kita memberikan fungsi `int` pada:

```
jumlah = int(bilangan1) + int(bilangan2)
```

Karena fungsi yang digunakan adalah `int` yang mengubah ke bilangan bulat maka ketika bilangan yang kita masukkan adalah bilangan desimal hasilnya akan error

In [8]:

```
print("Hari ini kita akan menjumlahkan dua bilangan bulat")
bilangan1 = input("Masukkan bilangan pertama")
bilangan2 = input("masukkan bilangan kedua")
jumlah = int(bilangan1) + int(bilangan2)
print("Hasil penjumlahan bilangan bulat", bilangan1, "dan", bilangan2, "adalah", jumlah)
```

Hari ini kita akan menjumlahkan dua bilangan bulat

```
-----
ValueError                                                 Traceback (most recent call last)
<ipython-input-8-656b47b3bfbd> in <module>
      2 bilangan1 = input("Masukkan bilangan pertama")
      3 bilangan2 = input("masukkan bilangan kedua")
----> 4 jumlah = int(bilangan1) + int(bilangan2)
      5 print("Hasil penjumlahan bilangan bulat", bilangan1, "dan", bilangan2, "adalah", jumlah)

ValueError: invalid literal for int() with base 10: '10.3'
```

Oleh karena itu kita harus ubah menjadi bilangan desimal terlebih dahulu menggunakan fungsi `float` :

In [9]:

```
print("Hari ini kita akan menjumlahkan dua bilangan")
bilangan1 = input("Masukkan bilangan pertama")
bilangan2 = input("masukkan bilangan kedua")
jumlah = float(bilangan1) + float(bilangan2)
print("Hasil penjumlahan bilangan bulat", bilangan1, "dan", bilangan2, "adalah", jumlah)
```

Hari ini kita akan menjumlahkan dua bilangan

Hasil penjumlahan bilangan bulat 0.55 dan 1.43 adalah 1.98

Dari contoh-contoh di atas kita sudah mempelajari tiga macam tipe data yaitu `String`, `Integer`, dan `Float`.

Memanggil Paket/Modul Bawaan Python (seperti `math`), Mencoba, dan Memanggil bantuan (`help`) untuk Masing-Masing Fungsi

Pada saat akan menginstall **Jupyter Notebook** kita sudah sempat membahas bahwa terdapat berbagai macam paket atau modul Python yang berisikan kumpulan kode untuk fungsi yang spesifik dimana kita tinggal memanggil modul tersebut. Sebagai contoh kita akan memanggil modul bawaan yang sudah terinstall bersamaan dengan Python yaitu paket `math`. Seperti namanya, paket ini berfungsi untuk melakukan berbagai macam operasi matematika. Bagaimana cara memanggil modul ini?

Untuk memanggil model `math` kita dapat menggunakan sintaks `import` seperti di bawah ini.

In [10]:

```
import math
```

Setelah kita jalankan kodennya kita tidak mendapat keluaran atau output apapun, karena proses ini hanya memanggil saja, untuk menggunakannya kita dapat memanggil fungsi-fungsi spesifik dari modul `math` ini. Apa saja fungsi yang dapat kita panggil? Kita dapat melihat apa saja yang tersedia dengan menggunakan fungsi `help`, mari kita coba

```
In [11]:
```

```
help(math)
```

```
Help on module math:
```

```
NAME
```

```
math
```

```
MODULE REFERENCE
```

```
https://docs.python.org/3.7/library/math
```

The following documentation is automatically generated from the Python source files. It may be incomplete, incorrect or include features that are considered implementation detail and may vary between Python implementations. When in doubt, consult the module reference at the location listed above.

```
DESCRIPTION
```

This module provides access to the mathematical functions defined by the C standard.

```
FUNCTIONS
```

```
acos(x, /)
    Return the arc cosine (measured in radians) of x.
```

```
acosh(x, /)
    Return the inverse hyperbolic cosine of x.
```

```
asin(x, /)
    Return the arc sine (measured in radians) of x.
```

```
asinh(x, /)
    Return the inverse hyperbolic sine of x.
```

```
atan(x, /)
    Return the arc tangent (measured in radians) of x.
```

```
atan2(y, x, /)
    Return the arc tangent (measured in radians) of y/x.
```

Unlike atan(y/x), the signs of both x and y are considered.

```
atanh(x, /)
    Return the inverse hyperbolic tangent of x.
```

```
ceil(x, /)
    Return the ceiling of x as an Integral.
```

This is the smallest integer $\geq x$.

```
copysign(x, y, /)
    Return a float with the magnitude (absolute value) of x but the sign of y.
```

On platforms that support signed zeros, copysign(1.0, -0.0) returns -1.0.

```
cos(x, /)
    Return the cosine of x (measured in radians).
```

```
cosh(x, /)
    Return the hyperbolic cosine of x.
```

```
degrees(x, /)
    Convert angle x from radians to degrees.
```

```
erf(x, /)
    Error function at x.
```

```
erfc(x, /)
    Complementary error function at x.
```

```
exp(x, /)
    Return e raised to the power of x.
```

```
expm1(x, /)
    Return  $\exp(x)-1$ .
```

This function avoids the loss of precision involved in the direct evaluation of $\exp(x)-1$

```
fabs(x, /)
    Return the absolute value of the float x.
```

```
factorial(x, /)
```

```
Find x!.
```

Raise a ValueError if x is negative or non-integral.

```
floor(x, /)
    Return the floor of x as an Integral.
```

This is the largest integer <= x.

```
fmod(x, y, /)
    Return fmod(x, y), according to platform C.
```

x % y may differ.

```
frexp(x, /)
    Return the mantissa and exponent of x, as pair (m, e).
```

m is a float and e is an int, such that x = m * 2.**e.
If x is 0, m and e are both 0. Else 0.5 <= abs(m) < 1.0.

```
fsum(seq, /)
    Return an accurate floating point sum of values in the iterable seq.
```

Assumes IEEE-754 floating point arithmetic.

```
gamma(x, /)
    Gamma function at x.
```

```
gcd(x, y, /)
    greatest common divisor of x and y
```

```
hypot(x, y, /)
    Return the Euclidean distance, sqrt(x*x + y*y).
```

```
isclose(a, b, *, rel_tol=1e-09, abs_tol=0.0)
    Determine whether two floating point numbers are close in value.
```

rel_tol
maximum difference for being considered "close", relative to the magnitude of the input values

abs_tol
maximum difference for being considered "close", regardless of the magnitude of the input values

Return True if a is close in value to b, and False otherwise.

For the values to be considered close, the difference between them must be smaller than at least one of the tolerances.

-inf, inf and NaN behave similarly to the IEEE 754 Standard. That is, NaN is not close to anything, even itself. inf and -inf are only close to themselves.

```
isfinite(x, /)
    Return True if x is neither an infinity nor a NaN, and False otherwise.
```

```
isinf(x, /)
    Return True if x is a positive or negative infinity, and False otherwise.
```

```
isnan(x, /)
    Return True if x is a NaN (not a number), and False otherwise.
```

```
ldexp(x, i, /)
    Return x * (2**i).
```

This is essentially the inverse of frexp().

```
lgamma(x, /)
    Natural logarithm of absolute value of Gamma function at x.
```

```
log(...)
    log(x, [base=math.e])
    Return the logarithm of x to the given base.
```

If the base not specified, returns the natural logarithm (base e) of x.

```
log10(x, /)
    Return the base 10 logarithm of x.
```

```
log1p(x, /)
    Return the natural logarithm of 1+x (base e).
```

The result is computed in a way which is accurate for x near zero.

```
log2(x, /)
```

```

Return the base 2 logarithm of x.

modf(x, /)
    Return the fractional and integer parts of x.

Both results carry the sign of x and are floats.

pow(x, y, /)
    Return x**y (x to the power of y).

radians(x, /)
    Convert angle x from degrees to radians.

remainder(x, y, /)
    Difference between x and the closest integer multiple of y.

    Return x - n*y where n*y is the closest integer multiple of y.
    In the case where x is exactly halfway between two multiples of
    y, the nearest even value of n is used. The result is always exact.

sin(x, /)
    Return the sine of x (measured in radians).

sinh(x, /)
    Return the hyperbolic sine of x.

sqrt(x, /)
    Return the square root of x.

tan(x, /)
    Return the tangent of x (measured in radians).

tanh(x, /)
    Return the hyperbolic tangent of x.

trunc(x, /)
    Truncates the Real x to the nearest Integral toward 0.

    Uses the __trunc__ magic method.

```

DATA

```

e = 2.718281828459045
inf = inf
nan = nan
pi = 3.141592653589793
tau = 6.283185307179586

```

FILE
`/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/lib-dynload/math.cpython-37m`

Seperti yang kita lihat di atas, fungsi `help` akan menampilkan petunjuk untuk paket, modul, atau fungsi yang kita masukkan ke dalam fungsi `help` tersebut. Pada modul `math` ditampilkan fungsi-fungsi yang dapat dipanggil, selain fungsi juga terdapat DATA yang berupa nilai-nilai konstanta seperti `e`, `inf`, `pi`. Mari kita coba panggil beberapa fungsi dan konstanta dari modul `math`:

In [12]:

```
math.e
```

Out[12]:

```
2.718281828459045
```

In [13]:

```
math.log(100)
```

Out[13]:

```
4.605170185988092
```

Mengapa nilai dari fungsi `log(100)` dari modul `math` tidak menunjukkan angka 2? Jadi apa yang dilakukan oleh fungsi `log` tersebut? Kita bisa mencari tau dengan memanfaatkan fungsi `help` lagi tetapi lebih spesifik untuk fungsi `log`:

In [14]:

help(math.log)

Help on built-in function log in module math:

```
log(...)
    log(x, [base=math.e])
    Return the logarithm of x to the given base.
```

If the base not specified, returns the natural logarithm (base e) of x.

Menurut petunjuk di atas, ternyata fungsi `log` dapat diisi dengan berapapun basis yang kita inginkan ditandai dengan tulisan `Return the logarithm of x to the given base.`. Nah apabila tidak diketikkan basisnya di dalam fungsi tersebut maka basis akan **secara default** terisi sebagai bilangan natural (`If the base not specified, returns the natural logarithm (base e) of x`) atau `ln`. Jika kita ingin menggunakan `log` dengan basis 10 maka kita dapat mengetikkan:

In [13]:

math.log(100,10)

Out[13]:

2.0

Hasilnya adalah nilai `log(100)` dengan basis 10, seperti yang kita inginkan. Selain fungsi `log` yang lebih umum dimana kita bisa memasukkan berapapun nilai basis yang kita inginkan, ternyata di modul `math` sudah disediakan `log` khusus dengan basis 10 yaitu `log10`:

In [15]:

math.log10(100)

Out[15]:

2.0

In [16]:

angka_saya = 1000

In [17]:

math.log10(angka_saya)

Out[17]:

3.0

In [18]:

```
nilai = 5
nilai_kuadrat = math.pow(nilai,2)
nilai_kuadrat
```

Out[18]:

25.0

Pada fungsi `math.sin` kita dapat memasukkan nilai sudut dalam **radian** seperti pada petunjuk ini:

In [19]:

help(math.sin)

Help on built-in function sin in module math:

```
sin(x, /)
    Return the sine of x (measured in radians).
```

Nilai $\sin(\pi/2)$ adalah:

In [20]:

```
math.sin(math.pi/2)
```

Out[20]:

1.0

Berapakah nilai $\pi/2$ dalam derajat?

In [21]:

```
math.degrees(math.pi/2)
```

Out[21]:

90.0

Berapakah nilai 90 derajat dalam radian?

In [22]:

```
math.radians(90)
```

Out[22]:

1.5707963267948966

atau $\pi/2$.

Pada contoh di atas kita dapat mengubah sudut dari derajat ke radian menggunakan fungsi `math.radians`, dengan fungsi ini kita dapat langsung menghitung nilai $\sin(90)$ dengan 90 dalam derajat

In [32]:

```
sudut_derajat = 90
sudut_radian = math.radians(sudut_derajat)
nilai_sinus = math.sin(sudut_radian)
print(nilai_sinus)
```

1.0

Contoh-contoh di atas hanya membahas beberapa fungsi dari modul `math`, teman-teman bisa mencoba berbagai macam fungsi seperti yang tertulis di `help(math)`.

Memberikan Catatan dan Gambar dalam Bentuk Markdown di Jupyter Notebook

Hal yang spesial dari **Jupyter Notebook** seperti yang sudah kita sempat singgung di awal yaitu kita bisa menambahkan catatan sekaligus menjalankan kode. Di bawah ini merupakan petunjuk-petunjuk untuk menambahkan berbagai macam jenis catatan.

Menambah **heading** atau **judul bab/subbab**

Menambah **heading** pada `cell` dengan jenis `markdown` dapat dilakukan dengan menggunakan simbol `#` sebelum nama judul, jumlah tanda `#` menunjukkan tingkat dari subbab.

```
markdown
# Merupakan heading level 1
## Merupakan heading level 2
### Merupakan heading level 3
```

akan menjadi:

Merupakan heading level 1

Merupakan heading level 2

Merupakan heading level 3

Menambahkan huruf tebal ataupun *miring*

Untuk menambahkan huruf tebal kita dapat menambah tanda asteriks (**) sebelum dan sesudah kata atau kalimat yang akan dicetak tebal, ****contoh**** akan menjadi **contoh**. Apabila kita mengurangi jumlah tanda asteriks menjadi hanya satu maka huruf akan tercetak miring, ***contoh*** akan menjadi *contoh*.

Membuat List

List baik yang berurutan atau tidak dapat ditulis menggunakan markdown, contohnya adalah:

```
markdown
* Pepaya
* Jambu
* Apel
```

akan menjadi:

- Pepaya
- Jambu
- Apel

Apabila pada list berurutan maka penulisan didahului angka yang menunjukkan urutan dari komponen list:

```
markdown
1. Menginstall Python
2. Menginstall Anaconda
3. Menjalankan Notebook
```

akan menjadi:

1. Menginstall Python
2. Menginstall Anaconda
3. Menjalankan Notebook

Menambah Tabel

```
markdown
Kolom 1 | Kolom 2
----- | -----
Konten 1 Kolom 1 | Konten 1 Kolom 2
Konten 2 Kolom 1 | Konten 2 Kolom 2
```

akan menjadi:

Kolom 1	Kolom 2
Konten 1 Kolom 1	Konten 1 Kolom 2
Konten 2 Kolom 1	Konten 2 Kolom 2

Menambahkan Gambar

Menambahkan gambar dapat dilakukan dengan sintaks `![keterangan](link/lokasi gambar)`, seperti contoh dibawah ini:

```
markdown
![GithubOctocat]({https://github.githubassets.com/images/modules/logos_page/Octocat.png})
```

akan menjadi:



sekarang kita menaruh gambar di dalam folder `figures`, cara memanggil gambar tersebut adalah:

```
markdown
![GithubMark](figures/Github-Mark.png)
```

akan menjadi:



Menambahkan Link

Link menuju suatu halaman web atau file dapat ditambahkan pada catatan di **notebook** kita dengan menggunakan sintaks `[Tulisan Link](lokasi link)`. Contoh, sintaks berikut ini `[Modul Workshop](https://github.com/anangsahroni/geoscope-geohazard-workshop)` akan menjadi link [Modul Workshop](https://github.com/anangsahroni/geoscope-geohazard-workshop).

Menambahkan Kode

Kita juga dapat menambahkan kode sebagai tulisan dalam markdown, contohnya adalah kode Python berikut ini:

```
```python
def pangkat(angka,pangkat):
 hasil = angka**pangkat
 return hasil
```
```

akan menjadi:

```
def pangkat(angka,pangkat):
    hasil = angka**pangkat
    return hasil
```

Dengan menulis di `cell` bertipe `Markdown` dan ditulis seperti di atas maka kode tidak tereksekusi melainkan menjadi sebuah catatan. Banyak bahasa pemrograman lain, kita tinggal mengganti tulisan `python` dengan bahasa pemrograman yang kita inginkan.

```
```javascript
document.getElementById("logo").style.display="block";
```

```

akan menjadi:

```
document.getElementById("logo").style.display="block";
```

Menambahkan Rumus Matematika

Yang menarik dari cell bertipe Markdown adalah kita juga dapat menuliskan perumusan matematika menggunakan sintaks dari MathJax , contohnya adalah untuk:

```
markdown
$$
e^{i\pi} + 1 = 0
$$
```

yang akan menjadi:

$$e^{i\pi} + 1 = 0$$

Untuk menulis secara inline atau berada di dalam baris kita dapat menggunakan sintaks `$ e^{i\pi} + 1 = 0 $` yang akan menjadi $e^{i\pi} + 1 = 0$. Contoh-contoh lain:

```
mathjax
$$
\begin{matrix}
1 & x & x^2 \\
1 & y & y^2 \\
1 & z & z^2
\end{matrix}
$$
```

| | | |
|---|---|-------|
| 1 | x | x^2 |
| 1 | y | y^2 |
| 1 | z | z^2 |

Lebih kompleks lagi:

```
mathjax
$$ \bbox[5px, border:2px solid red]
\begin{aligned}
e^x = \lim_{n \rightarrow \infty} \left( 1 + \frac{x}{n} \right)^n
\end{aligned}
$$
```

akan menjadi:

$$e^x = \lim_{n \rightarrow \infty} \left(1 + \frac{x}{n} \right)^n \quad (2)$$

Referensi secara lebih lengkap dapat dibuka di [sini \(<https://math.meta.stackexchange.com/questions/5020/mathjax-basic-tutorial-and-quic-reference>\)](https://math.meta.stackexchange.com/questions/5020/mathjax-basic-tutorial-and-quic-reference) atau dengan kata kunci "MathJax Tutorial".

Membuat Check List

Berikut ini adalah contoh sintaks untuk membuat *checklist*:

- [x] Menginstall Jupyter Notebook
- [] Menjalankan Jupyter Notebook
- [] Mempelajari sintaks markdown

yang akan menjadi:

- [x] Menginstall Jupyter Notebook
- [] Menjalankan Jupyter Notebook
- [] Mempelajari sintaks markdown

Membuat quote

Bagi yang suka sastra dan ingin mengutip sastrawan atau ilmuan, sintaksnya adalah seperti ini

markdown

- > Ini adalah quote yang bagus
- > - Oleh: Saya sendiri

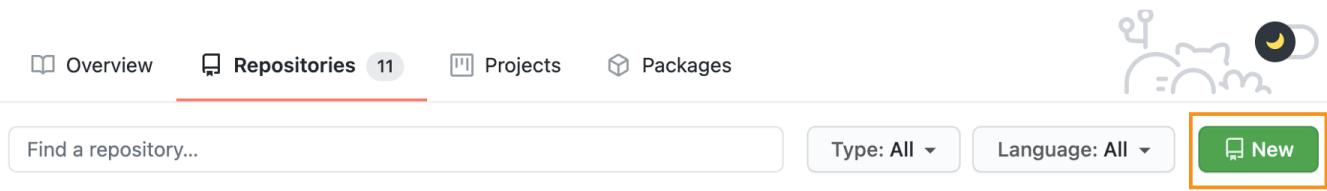
Ini adalah quote yang bagus

- Oleh: Saya sendiri

Menyimpan notebook pada repositori Github dan menambahkan ke Binder

Notebook dapat kita simpan melalui menu `File - Save notebook` atau mengganti nama di bagian atas kemudian menekan `Enter`, kemudian file dengan ekstensi `.ipynb` akan muncul di lokasi tempat kita pertama kali menjalankan `jupyter-notebook`. File tersebut dapat dibuka lagi dengan menggunakan `jupyter-notebook` lagi suatu saat. Selain menyimpan secara lokal di komputer, kita juga akan menyimpan *notebook* kita secara daring di sarana repositori GitHub. GitHub adalah sarana daring untuk pengembangan software dan pengontrol versi (*version control*). Kita akan coba melakukan kontrol versi paling sederhana pada *notebook* kita, tapi sebelumnya kita harus membuat akun lebih dahulu melalui [\(https://github.com\)](https://github.com).

Setelah membuat akun kita mulai dapat membuat repositori pertama kita dengan menekan tombol `New` pada halaman [\(https://github.com/\[namaunder\]?tab=repositories\)](https://github.com/[namaunder]?tab=repositories) (`namaunder` dapat diisi username teman-teman, **tanpa** tanda `[]`).



Kita selanjutnya diharuskan mengisi informasi repositori yang akan kita buat sebelum menekan tombol `Create repository`, seperti ini contohnya:

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 * |
| anangsahroni | / workshop-python |

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

Description (optional)

Repositori ini digunakan untuk menyimpan notebook saat workshop Python.

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](#).

This will set `main` as the default branch. Change the default name in your [settings](#).

Create repository

Selelah repositori dibuat kita dapat mengunggah *notebook* yang sudah kita buat dengan memilih opsi `Upload files` seperti pada gambar di bawah ini.

The screenshot shows a GitHub repository interface. At the top, there are buttons for 'main', '1 branch', '0 tags', 'Go to file', 'Add file', and a 'Code' dropdown. The 'Code' dropdown is open, showing 'Create new file' and 'Upload files' (which is highlighted in blue). Below this, a commit card for 'anangsahroni Initial commit' is shown, containing a file named 'README.md'. The main content area displays the contents of 'README.md', which includes the title 'workshop-python' and a descriptive message about the repository's purpose.

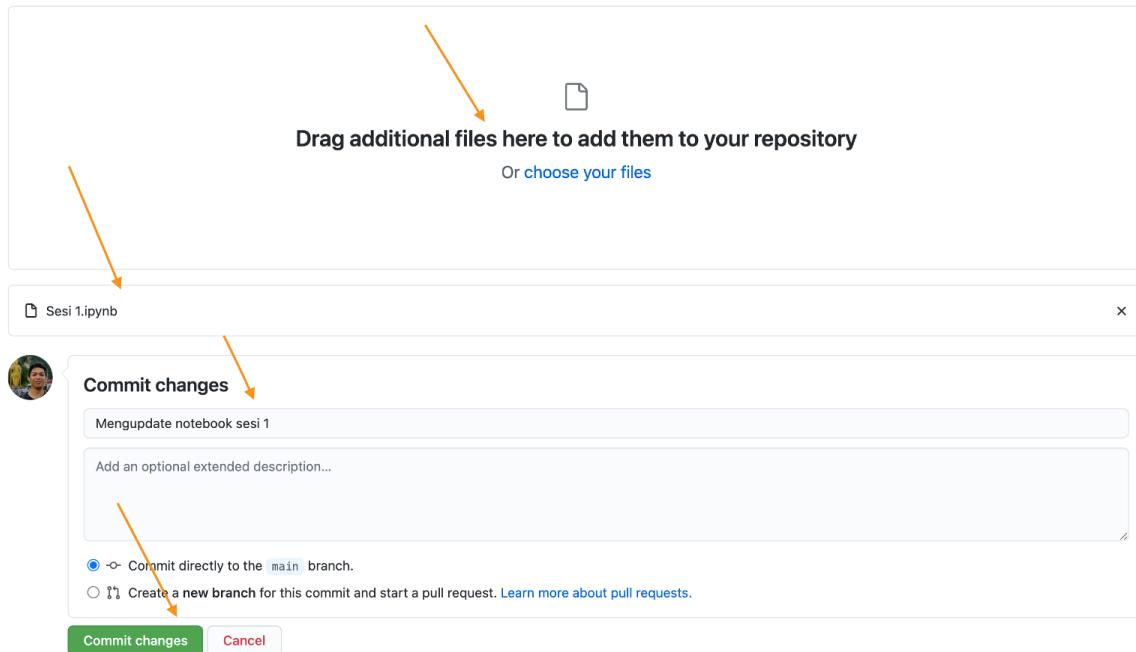
Notebook dapat kita tarik atau *drag* ke tempat *upload* seperti pada gambar di bawah ini, untuk setiap file yang ditambahkan, diedit, ataupun dihapus kita harus menambahkan pesan *commit*. Pesan ini yang menjelaskan perubahan apa yang kita lakukan dan menjadi penanda untuk *history* file.

The screenshot shows the GitHub 'Commit changes' dialog. It features a large orange box with a file icon and the text 'Drag additional files here to add them to your repository' and 'Or choose your files'. Below this, a file named 'Sesi 1.ipynb' is listed with a delete 'x' button. At the bottom, there is a 'Commit changes' button and a 'Cancel' button. The 'Commit changes' input field contains the text 'Menambah notebook untuk sesi 1'. The 'Commit directly to the main branch' radio button is selected. There is also an option to 'Create a new branch for this commit and start a pull request'. Orange arrows point from the 'Upload files' button in the previous screenshot to the 'choose your files' link in this dialog, and from the 'choose your files' link to the 'Commit changes' input field.

Untuk menyimpan perubahan kita dapat menekan tombol `Commit`.

Mengupdate Notebook di Github dan Melihat *History* Perubahannya

Apabila kita mengubah notebook kita dan ingin mengupdate notebook di repositori, kita dapat mengulang proses upload notebook. Contoh pada kasus ini saya menambahkannya 4 gambar di notebook. Untuk mengupdate kita dapat mengupload lagi dan memberikan pesan untuk `Commit` kita, pada kasus ini pesan `commit` adalah "Mengupdate notebook sesi 1". Pesan `commit` tersebut bukan contoh yang baik sebenarnya, karena tidak jelas, lebih baik kita dapat menuliskan "Menambah gambar [nam gambar1], [nama gambar2], [nama gambar3], dan [nama gambar4]".



Kita dapat melihat riwayat perubahan dari notebook kita dengan memilih notebook kita pada tampilan repositori:

The screenshot shows a GitHub repository page for 'anangsahroni / workshop-python'. The repository has 1 branch and 0 tags. The commit history shows three commits by 'anangsahroni': 'Initial commit' (README.md) at 37 minutes ago and 'Mengupdate notebook sesi 1' (Sesi 1.ipynb) at 4 minutes ago. The README.md file is highlighted with an orange arrow pointing to its name. The repository description is 'Repositori ini digunakan untuk menyimpan notebook saat workshop Python.'

Selanjutnya kita dapat menekan tombol History :

The screenshot shows a Jupyter Notebook viewer interface. At the top, there's a navigation bar with a dropdown menu labeled 'main', a link to 'workshop-python / Sesi 1.ipynb', and buttons for 'Go to file' and '...'. Below the navigation is a header bar with a user profile picture, the name 'anangsahroni', and the message 'Mengupdate notebook sesi 1'. It also shows '1 contributor' and 'Latest commit 5865b20 20 minutes ago'. A blue arrow points from the text 'inilah alasan mengapa kita harus menuliskan pesan commit yang jelas:' to the 'History' button in the header.

Below the header, there's a summary: '1321 lines (1321 sloc) | 45.1 KB' with buttons for 'Raw' and 'Blame'. The main content area contains a section titled 'Geoscope Geohazard Workshop HMGF UGM' and two bold sections: 'Instalasi Miniconda dan Jupyter Notebook' and 'Instalasi Miniconda pada Windows, Linux, ataupun MacOS'.

Pilih salah satu perubahan yang dapat kita lihat, berdasarkan pesan *commit*, inilah alasan mengapa kita harus menuliskan pesan *commit* yang jelas:

The screenshot shows the GitHub commit history for the repository 'workshop-python / Sesi 1.ipynb'. It lists two commits:

- Mengupdate notebook sesi 1** (anangsahroni committed 22 minutes ago)
- Menambah notebook untuk sesi 1** (anangsahroni committed 1 hour ago)

Buttons for 'Verified', 'Raw', '5865b20', and a copy icon are shown next to each commit. A blue arrow points from the text 'inilah alasan mengapa kita harus menuliskan pesan commit yang jelas:' to the first commit message.

Disini dapat dilihat bahwa terdapat perubahan dimana kita menambah beberapa baris yang isinya untuk menambah gambar:

The screenshot shows a diff view of a code file. The changes are highlighted in green (additions) and red (deletions). The additions are related to adding image files to the repository:

```

1280 1280      "![isi_info repositori baru](gambar/fill_repo_info.png)\n",
1281 1281      "\n",
1282 -     "![upload_notebook](gambar/upload_notebook.png)"
1282 +     "![upload_notebook](gambar/upload_notebook.png)\n",
1283 +     "\n",
1284 +     "![memulai_commit](gambar/add_commit.png)\n",
1285 +     "\n",
1286 +     "![copy_url](gambar/copy_repo_url.png)\n",
1287 +     "\n",
1288 +     "![binder](gambar/binder.png)\n"
1283 1289 ]

```

Menambah Notebook ke Binder

Seperti untuk modul kuliah ini yang dapat diakses secara daring, teman-teman juga bisa mengunggah **notebook** yang teman-teman gunakan. Langkah setelah membuat repositori di GitHub adalah mengunjungi laman web [MyBinder](http://mybinder.org) (<http://mybinder.org>). Salin URL repositori teman-teman, seperti pada gambar di bawah ini:

The screenshot shows a GitHub repository page for 'anangsahroni / workshop-python'. At the top, there's a navigation bar with links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below the navigation bar, the repository name 'anangsahroni / workshop-python' is displayed. The main content area shows a commit history for the 'main' branch. The first commit is by 'anangsahroni' titled 'Menambah notebook untuk sesi 1' made 7 minutes ago. The second commit is 'Initial commit' made 13 minutes ago. The third commit is 'Sesi 1.ipynb' made 7 minutes ago. Below the commit history, there's a preview of the 'README.md' file which contains the text 'workshop-python' and a note about its purpose.

Kemudian tempel salinan tadi ke form isian dari MyBinder kemudian tekan **Launch**.

The screenshot shows the 'Build and launch a repository' interface. It has fields for 'GitHub repository name or URL' (set to 'https://github.com/anangsahroni/workshop-python'), 'Git ref (branch, tag, or commit)' (set to 'HEAD'), and 'Path to a notebook file (optional)'. There's a large orange 'launch' button. Below these fields, there's a section to copy the URL for sharing: 'Copy the URL below and share your Binder with others:' followed by the URL 'https://mybinder.org/v2/gh/anangsahroni/workshop-python/HEAD'. At the bottom, there are sections for copying text to show a binder badge: 'Copy the text below, then paste into your README to show a binder badge:' followed by two examples: one for Markdown ('m') and one for ReStructuredText ('rst').

Selesai