**Welcome!**

Enable fullscreen

Hacktiv8-PTP Python For Data Science // Welcome

Title: Hacktiv8-PTP Python For Data Science Starter Notebook  
Last Updated: September 20, 2020  
Author: Hacktiv8 Ai Curriculum Team

**Hi! Welcome to Hacktiv8-PTP Python for Data Science materials landing page.**

**Program Overview**

Hacktiv8-PTP Python for Data Science Program has everything you need to become a Data Scientist. This program is perfect for those who want to learn statistics, machine learning, and visualizing information. You will also learn various Phyton toolkits such as pandas, matplotlib, scikit-learning, seaborn, and many more. Besides that, your Data Science knowledge will be supported by various Machine Learning theory. After finishing this program, you are expected to become a competent Data Scientist.

**Educational Objectives**

As a student of Hacktiv8-PTP Python for Data Science Program, you are expected to meet certain targets which are:

* Using functionality and feature of Phyton, applying it for data science needs.
* Understanding Practical Statistics used for data science
* Query DataFrame structure to clean and process dataset
* Creating basic to advanced visualization to represent data
* Creating ML features to fullfill data analysis needs
* Implementing Supervised and Unsupervised Learning to solve real problems

This Program, Hacktiv8-PTP Python for Data Science, has 16 Sections and 1 Final Project. The final project will determine the student's level of understanding. They are expected to implement every material included in this program. Aside from that, the project will act as a demo tool to see the student's practical ability.

**Prerequisite**

To enroll this Online Program, you should have basic understanding of computer (web browsing, app installation, and file navigating). Experience in any programming language(s) is optional.

Minimum device requirements: Processor Core i3 gen 6 / Ryzen gen 1 (recommended: Core i5 gen 6 / Ryzen gen 2), RAM minimum 4GB (recommended: 8GB) and at least 100GB storage available. Preferably using Unix or Linux as an Operating System, or Windows 10 with WSL2 Ubuntu 18.04/20.04 Kernel.

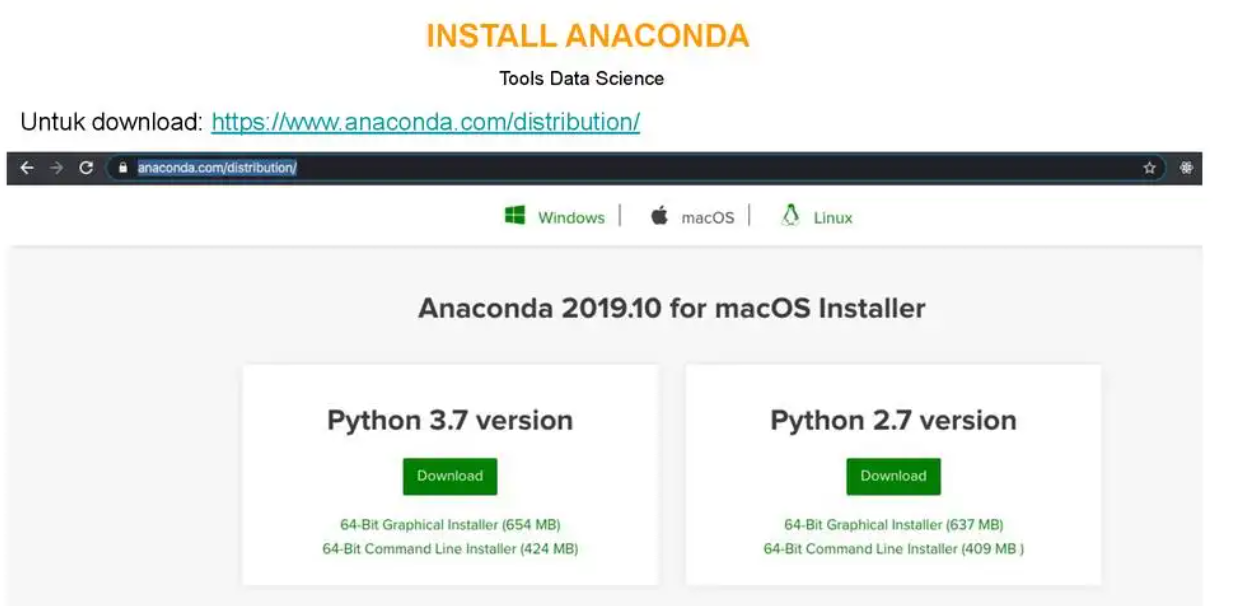
**Contents**

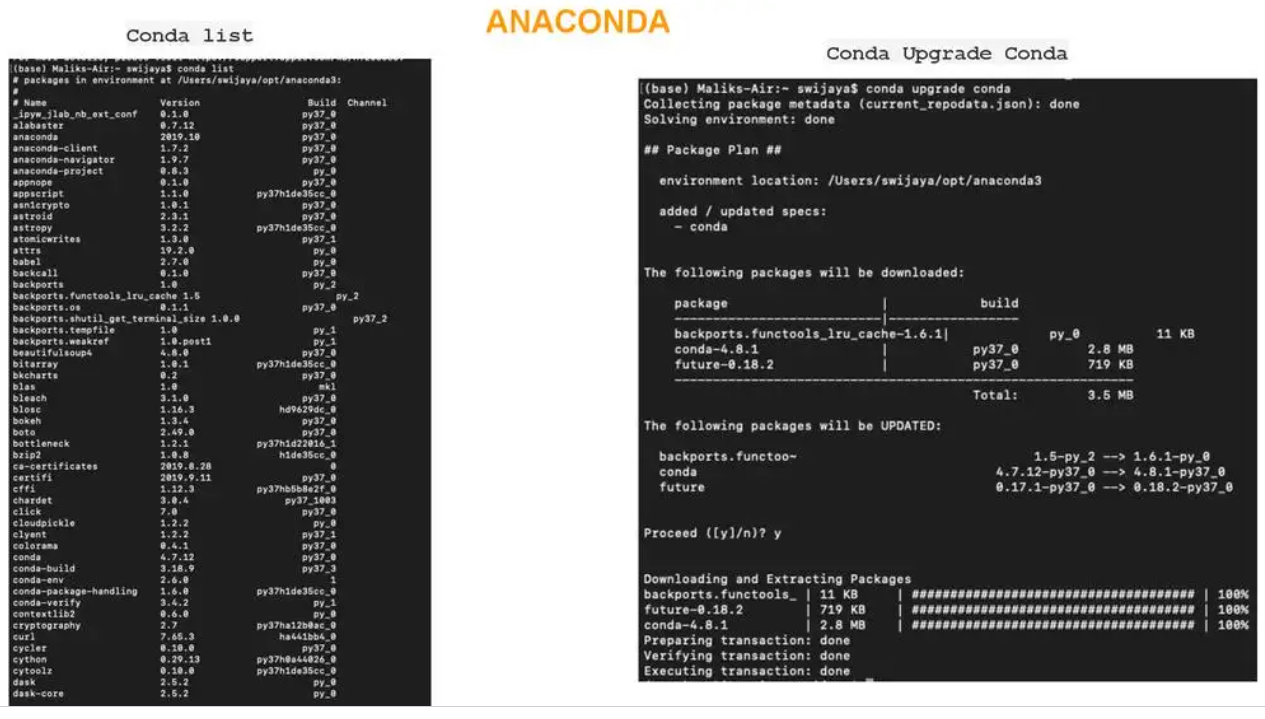
This section covers what you'll learn in this labs.

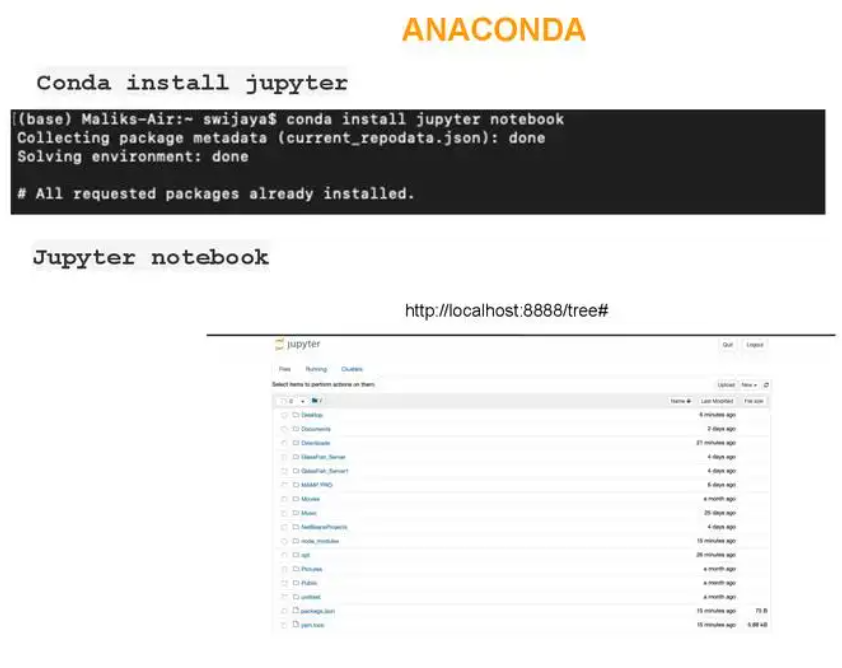
1. Introduction: Basic Syntax, Data Type, and Variable
2. Conditions, Control Flow & Looping
3. Function, Basic Module, and Package
4. Numpy
5. Pandas Introduction
6. Data Cleaning with Pandas
7. Basic Visualization
8. Advanced Visualization
9. Descriptive Statistics
10. Inferential Statistics
11. Regression Models
12. Classification 1
13. Classification 2
14. Unsupervised
15. Model Deployment
16. Final Projects

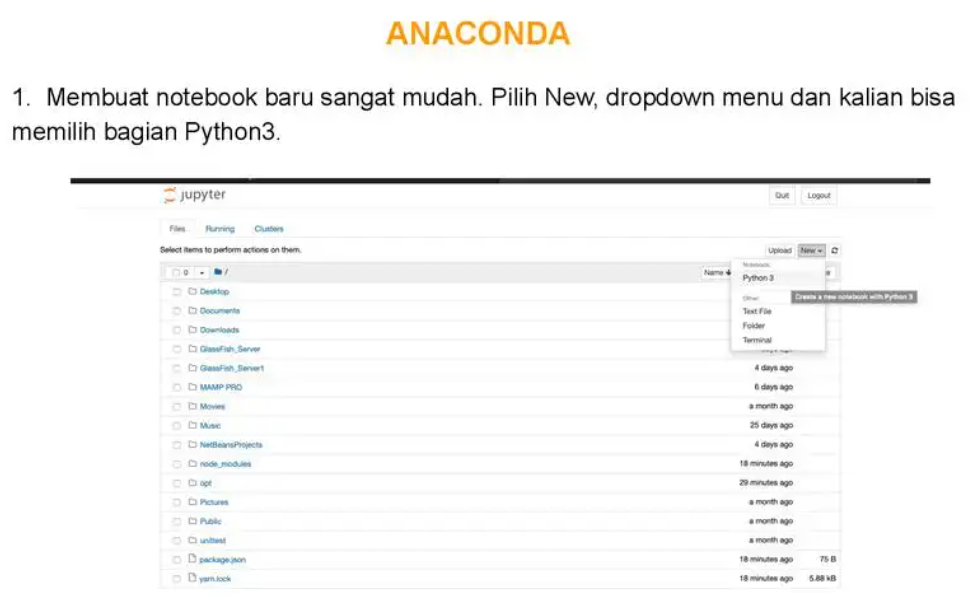
**Anaconda Installation**

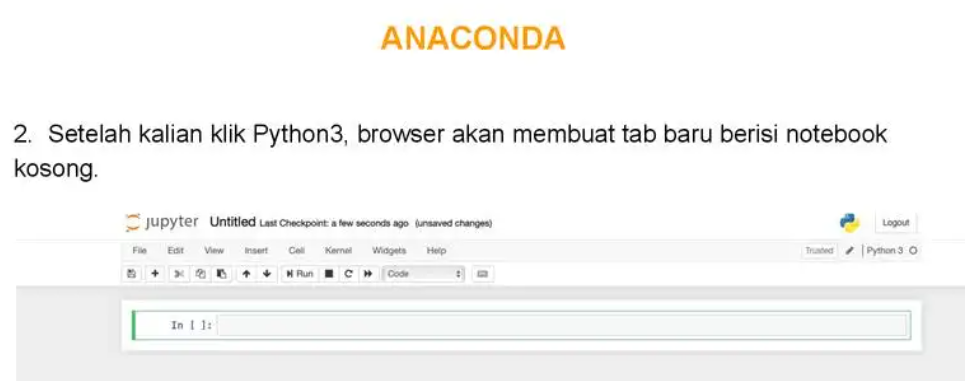
Enable fullscreen













## Pengenalan Python

Enable fullscreen

# Hacktiv8-PTP Python For Data Science // S.1 // Introduction, Data Types, and Variables

Python adalah high-level, interpreted scripting language yang dikembangkan pada akhir 1980-an oleh Guido van Rossum di National Research Institute for Mathematics and Computer Science di Belanda. Versi awal diterbitkan di the alt.sources newsgroup pada 1991, dan version 1.0 dirilis pada tahun 1994.

Python 2.0 dirilis pada tahun 2000, dan versi 2.x adalah prevalent releases hinggaDecember 2008. Pada saat itu, development team membuat keputusan untuk merilis versi 3.0, yang berisi beberapa perubahan yang relatif kecil tetapi signifikan yang tidak kompatibel dengan versi sebelumnya. dengan versi 2.x. Python 2 dan 3 sangat mirip, dan beberapa fitur Python 3 telah di-backport ke Python 2. Tetapi secara umum, fitur-fitur tersebut tetap kurang kompatibel.

Baik Python 2 dan 3 terus di-maintaine and developed, dikembangkan, dengan pembaruan rilis berkala untuk keduanya.

Nama Python tidak berasal dari ular, tetapi dari rombongan komedi Inggris, Monty Python's Flying Circus, di mana Guido dulunya, dan mungkin sekarangpun masih, menjadi penggemar.

## Mengapa Memilih Python?

Jika kalian akan menulis program, ada banyak bahasa yang umum digunakan untuk dipilih. Mengapa memilih Python? Berikut adalah beberapa fitur yang membuat Python menjadi pilihan yang menarik.

### **Python is Popular**

Python semakin populer selama beberapa tahun terakhir. Survei dari Stack Overflow 2018 menempatkan Python sebagai teknologi terpopuler ke-7 dan teknologi paling dicari nomor satu tahun ini. Perusahaan pengembangan perangkat lunak kelas dunia di seluruh dunia menggunakan Python setiap hari.

Menurut penelitian oleh Dice, Python juga merupakan salah satu skill terpopuler yang dimiliki dan bahasa pemrograman paling populer di dunia berdasarkan Popularity of Programming Language Index.

### **Python is Interpreted**

Banyak bahasa yang di-complie, artinya source code yang kalian buat perlu diterjemahkan ke dalam kode mesin, bahasa prosesor komputer kalian, sebelum dapat dijalankan. Program yang ditulis dalam interpreted language diteruskan langsung ke interpreter yang menjalankannya secara langsung.

Ini membuat siklus pengembangan aplikasi lebih cepat karena kalian cukup mengetikkan kode kalian dan menjalankannya, tanpa langkah kompilasi.

### **Python is Free**

Python interpreter dikembangkan di bawah lisensi open source dari OSI-approved open-source license, membuat Python gratis untuk dipasang, digunakan, dan didistribusikan, bahkan untuk tujuan komersial.

### **Python is Simple**

Python 3 memiliki 33 keywords, and Python 2 memliki 31 keywords. Sebagai perbandingan, C++ punya 62, Java punya 53, dan Visual Basic punya lebih dari 120, meskipun contoh yang terakhir ini mungkin agak berbeda berdasarkan implementasi atau dialek.

Kode Python memiliki struktur yang sederhana dan bersih yang mudah dipelajari dan dibaca.

### **But It’s Not That Simple**

Untuk semua kesederhanaan sintaksisnya, Python mendukung sebagian besar konstruksi yang diharapkan dalam bahasa tingkat tinggi, termasuk tipe data dinamis yang kompleks, pemrograman terstruktur dan fungsional, dan pemrograman berorientasi objek.

Selain itu, perpustakaan class dan function yang sangat luas tersedia yang menyediakan kemampuan melebihi apa yang dibangun ke dalam bahasa, seperti manipulasi database atau pemrograman GUI.

Python menyelesaikan apa yang tidak dimiliki oleh banyak bahasa pemrograman: bahasanya sendiri dirancang dengan sederhana, tetapi sangat fleksibel dalam hal apa yang dapat kalian capai dengannya.

## Integers

Pada Python3, secara efektif tidak ada batasan berapa banyak nilai integer. Tentu saja, itu dibatasi oleh jumlah memori yang dimiliki sistem kalian, seperti halnya semua hal, tetapi di luar itu integer bisa sepanjang apapun selama kalian membutuhkannya:

print(123123123123123123123123123123123123123123123123 + 1)

123123123123123123123123123123123123123123123124

# Python interprets a sequence of decimal digits without any prefix to be a decimal number:

print(10)

print(type(10))

10

<class 'int'>

## Floating-Point Numbers

Tipe float di Python menunjukkan bilangan floating-point. Nilai float ditentukan dengan titik desimal. Secara opsional, karakter e atau E diikuti dengan bilangan bulat positif atau negatif dapat ditambahkan untuk menentukan notasi ilmiah:

print(4.2)

print(type(4.2))

print(4.)

print(.2)

print(.4e7)

print(4.2e-4)

4.2

<class 'float'>

4.0

0.2

4000000.0

0.00042

## Strings

String adalah urutan data karakter. Jenis string dalam Python disebut str.  
  
String literals dapat dipisahkan menggunakan tanda kutip tunggal atau ganda. Semua karakter antara pembatas pembuka dan pembatas penutup yang 'match' adalah bagian dari string:

print("Hacktiv8")

print(type("Hacktiv8"))

Hacktiv8

<class 'str'>

Sebuah string dalam Python dapat berisi karakter sebanyak yang kalian inginkan. Satu-satunya batasan adalah resource memori kalian. String juga boleh kosong:

print('')

print("This string contains a single quote (') character.")

print('This string contains a double quote (") character.')

This string contains a single quote (') character.

This string contains a double quote (") character.

## Boolean Type, Boolean Context, and “Truthiness”

Python 3 menyediakan tipe data Boolean. Objek tipe Boolean mungkin memiliki salah satu dari dua nilai, True atau False:

print(type(True))

print(type(False))

<class 'bool'>

<class 'bool'>

## Variable Assignment

Bayangkan variabel sebagai nama yang dilampirkan ke objek tertentu. Di Python, variabel tidak perlu dideklarasikan atau didefinisikan sebelumnya, seperti halnya di banyak bahasa pemrograman lainnya. Untuk membuat variabel, kaliancukup menetapkan nilai dan bsia digunakan. Variable assignment dilakukan dengan satu tanda sama dengan (=):

n = 300

print(n)

300

n

300

# Later, if you change the value of n and use it again, the new value will be substituted instead:

n = 1000

print(n)

n

1000

1000

Python juga memungkinkan chained assignment, yang memungkinkan untuk menetapkan nilai yang sama ke beberapa variabel secara bersamaan:

a = b = c = 300

print(a, b, c)

300 300 300

## Variable Types in Python

Dalam banyak bahasa pemrograman, variabel diketik secara statis. Artinya, variabel yang dideklarasikan diawal memiliki tipe data tertentu, dan nilai apa pun yang ditetapkan pada variable tersebut selama masa pakainya harus selalu memiliki tipe itu.  
  
Variabel dalam Python berbeda dengan batasan ini. Di Python, variabel dapat diberi value dari satu jenis dan kemudian diberi value dari jenis yang berbeda:

var = 23.5

print(var)

var = "Now I'm a string"

print(var)

23.5

Now I'm a string

## Variable Names

Secara resmi, nama variabel di Python bisa berapa pun panjangnya dan bisa terdiri dari huruf besar dan huruf kecil (A-Z, a-z), digit (0-9), dan karakter garis bawah (\_). Batasan tambahan adalah bahwa, meskipun nama variabel dapat berisi angka, karakter pertama dari nama variabel tidak boleh berupa digit.

name = "Hacktiv8"

Age = 54

has\_laptops = True

print(name, Age, has\_laptops)

Hacktiv8 54 True

Nama variabel tidak boleh dimulai dengan angka:

9\_kepala\_naga = True

File "<ipython-input-23-737512de54da>", line 1

9\_kepala\_naga = True

^

SyntaxError: invalid token

Perhatikan bahwa case disini sangat berperan. Huruf kecil dan huruf besar tidak sama. Penggunaan karakter garis bawah juga penting. Masing-masing dari berikut ini mendefinisikan variabel yang berbeda:

age = 1

Age = 2

aGe = 3

AGE = 4

a\_g\_e = 5

\_age = 6

age\_ = 7

\_AGE\_ = 8

print(age, Age, aGe, AGE, a\_g\_e, \_age, age\_, \_AGE\_)

1 2 3 4 5 6 7 8

Di sisi lain, belum tentu semuanya dapat terbaca. Penamaan variable adalah masalah preferensi pribadi, tetapi kebanyakan orang akan menganggap dua contoh pertama, di mana semua huruf disatukan, lebih sulit untuk dibaca, terutama yang menggunakan huruf kapital semua. Metode yang paling umum digunakan untuk membuat nama variabel multi-kata adalah tiga contoh dibawah ini:

* Camel Case: Second and subsequent words are capitalized, to make word boundaries easier to see. (Presumably, it struck someone at some point that the capital letters strewn throughout the variable name vaguely resemble camel humps.)  
  Example: numberOfCollegeGraduates
* Pascal Case: Identical to Camel Case, except the first word is also capitalized.  
  Example: NumberOfCollegeGraduates
* Snake Case: Words are separated by underscores.  
  Example: number\_of\_college\_graduates

Gunakan salah satu dari ketiganya yang paling menarik secara visual bagi kalian. Pilih satu dan gunakan secara konsisten.

## Operators and Expressions in Python

Di Python, operators adalah simbol khusus yang menunjukkan bahwa suatu jenis komputasi harus dilakukan. Nilai yang ditindaklanjuti oleh operators disebut operands.  
  
Berikut ini contohnya:

a = 10

b = 20

a + b

30

Dalam kasus ini, operators + menjumlahkan operands a dan b. Operand dapat berupa nilai literal atau variabel yang mereferensikan objek:

a = 10

b = 20

a + b - 5

25

Urutan operan dan operator, seperti a + b - 5, disebut expression. Python mendukung banyak operator untuk menggabungkan objek data menjadi expression.

### **Arithmetic Operators**

# Here are some examples of these operators in use:

a = 4

b = 3

print(a + b)

print(a - b)

print(a \* b)

print(a / b)

print(a % b)

print(a \*\* b)

7

1

12

1.3333333333333333

1

64

# The result of standard division (/) is always a float, even if the dividend is evenly divisible by the divisor:

10 / 5

2.0

### **Comparison Operators**

# Here are examples of the comparison operators in use:

a = 10

b = 20

print(a == b)

print(a != b)

print(a <= b)

print(a >= b)

a = 30

b = 30

print(a == b)

print(a <= b)

print(a >= b)

False

True

True

False

True

True

True

## String Manipulation

# + Operators

s = 'foo'

t = 'bar'

u = 'baz'

print(s + t)

print(s + t + u)

print('Hacktiv8 ' + 'PTP')

foobar

foobarbaz

Hacktiv8 PTP

# \* Operators

s = 'foo.'

s \* 4

'foo.foo.foo.foo.'

# in Operators

s = 'foo'

print(s in 'That food for us')

print(s in 'That good for us')

True

False

# Case Conversion

s = 'HackTIV8'

# Capitalize

print(s.capitalize())

# Lower

print(s.lower())

# Swapcase

print(s.swapcase())

# Title

print(s.title())

# Uppercase

print(s.upper())

Hacktiv8

hacktiv8

hACKtiv8

Hacktiv8

HACKTIV8

## Python Lists

Singkatnya, list adalah kumpulan objek yang arbitrary, agak mirip dengan array dalam banyak bahasa pemrograman lain tetapi lebih fleksibel. List didefinisikan dalam Python dengan menyertakan sequence object yang dipisahkan koma dalam tanda kurung siku ([]), seperti yang ditunjukkan di bawah ini:

a = ['foo', 'bar', 'baz', 'qux']

print(a)

['foo', 'bar', 'baz', 'qux']

Karakteristik penting dari List Python adalah sebagai berikut:

* Lists are ordered.
* Lists can contain any arbitrary objects.
* List elements can be accessed by index.
* Lists can be nested to arbitrary depth.
* Lists are mutable.
* Lists are dynamic.

### **Lists Are Ordered**

List bukan hanya kumpulan object. List adalah kumpulan dari object yang berurutan. Urutan elemen ketika kalian membuat list akan bertahan selama List tersebut ada.

List yang memiliki elemen yang sama dalam urutan yang berbeda, tidak sama:

a = ['foo', 'bar', 'baz', 'qux']

b = ['baz', 'qux', 'bar', 'foo']

a == b

False

### **Lists Can Contain Arbitrary Objects**

Sebuah List dapat berisi bermacam-macam objek. Element dari List semuanya bisa berjenis sama:

a = [21.42, 'foobar', 3, 4, 'bark', False, 3.14159]

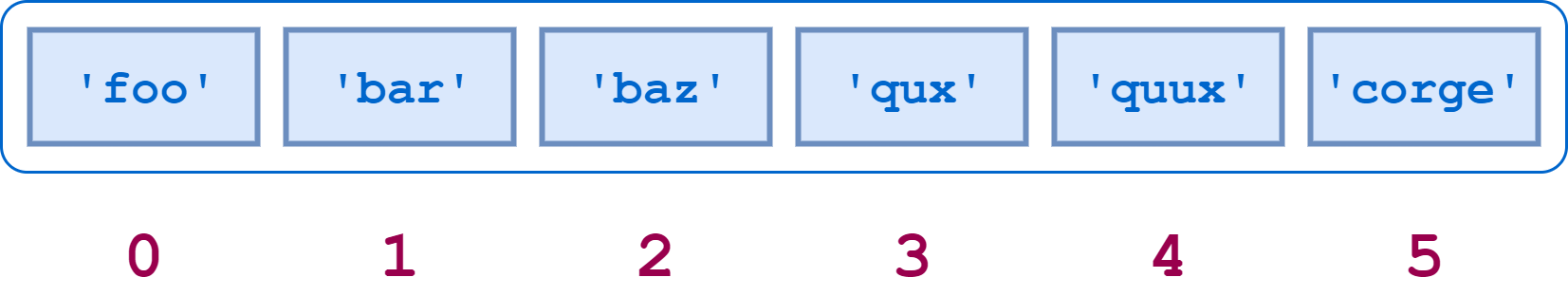
print(a)

[21.42, 'foobar', 3, 4, 'bark', False, 3.14159]

### **List Elements Can Be Accessed by Index**

Elemen individual dalam List dapat diakses menggunakan indeks dalam tanda kurung siku. Ini sama persis dengan mengakses karakter individu dalam sebuah string. Pengindeksan List berbasis nol seperti halnya dengan string.  
  
Perhatikan List berikut ini:

a = ['foo', 'bar', 'baz', 'qux', 'quux', 'corge']



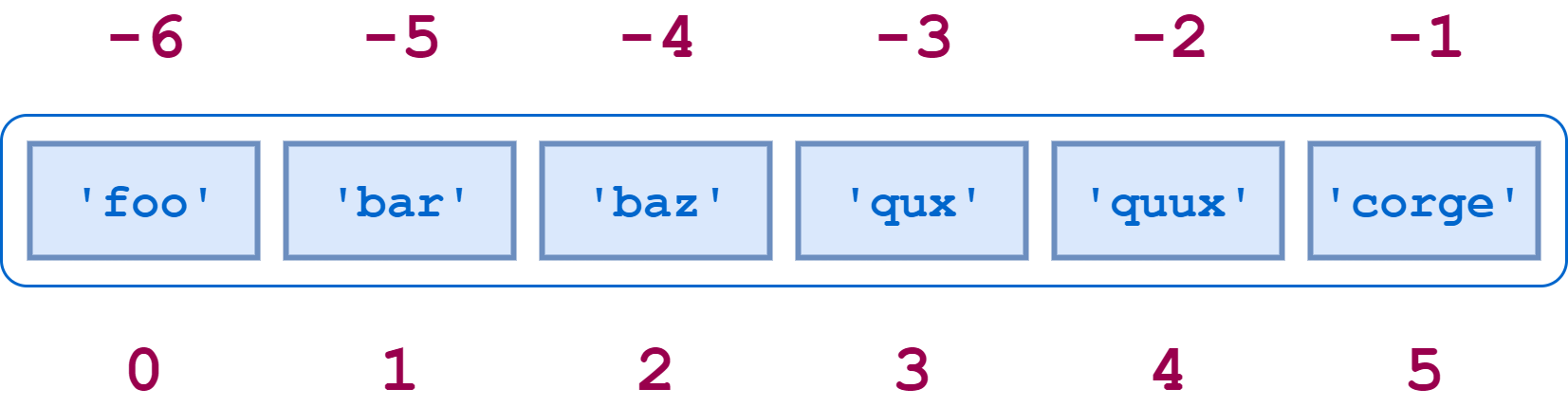
print(a[0])

print(a[5])

foo

corge

Hampir semua hal tentang pengindeksan string berfungsi serupa untuk list. Misalnya, indeks listnegatif dihitung dari akhir list:



print(a[-1])

print(a[-6])

corge

foo

Slicing juga bisa digunakan. Jika a adalah list, ekspresi a[m:n] mengembalikan porsi a dari indeks m ke n, tetapi tidak termasuk, indeks n:

a = ['foo', 'bar', 'baz', 'qux', 'quux', 'corge']

a[2:5]

['baz', 'qux', 'quux']

# The concatenation (+) and replication (\*) operators:

print(a)

print(a + ['grault', 'garply'])

print(a \* 2)

['foo', 'bar', 'baz', 'qux', 'quux', 'corge']

['foo', 'bar', 'baz', 'qux', 'quux', 'corge', 'grault', 'garply']

['foo', 'bar', 'baz', 'qux', 'quux', 'corge', 'foo', 'bar', 'baz', 'qux', 'quux', 'corge']

# len(), min(), max()

print(a)

print(len(a))

print(min(a))

print(max(a))

['foo', 'bar', 'baz', 'qux', 'quux', 'corge']

6

bar

qux

### **Modifying a Single List Value**

Satu nilai dalam list dapat diganti dengan pengindeksan dan assignment sederhana:

a = ['foo', 'bar', 'baz', 'qux', 'quux', 'corge']

print(a)

a[2] = 10

a[-1] = 20

print(a)

['foo', 'bar', 'baz', 'qux', 'quux', 'corge']

['foo', 'bar', 10, 'qux', 'quux', 20]

# A list item can be deleted with the del command:

del a[3]

print(a)

['foo', 'bar', 10, 'quux', 20]

### **Modifying Multiple List Values**

Bagaimana jika kalian ingin mengubah beberapa elemen yang bersebelahan dalam list sekaligus? Python memungkinkan hal ini dengan slice, yang memiliki sintaks berikut:

a = ['foo', 'bar', 'baz', 'qux', 'quux', 'corge']

print(a[1:4])

a[1:4] = [1.1, 2.2, 3.3, 4.4, 5.5]

print(a)

['bar', 'baz', 'qux']

['foo', 1.1, 2.2, 3.3, 4.4, 5.5, 'quux', 'corge']

## Python Tuples

Python menyediakan tipe lain dari kumpulan objek yang berurutan, disebut tuple.

### **Defining and Using Tuples**

Tuples identik dengan list dalam segala hal, kecuali untuk properti berikut:  
  
Tuples didefinisikan dengan mengapit elemen dalam tanda kurung (()) daripada tanda kurung siku ([]). Tuples tidak bisa diubah. Berikut adalah contoh singkat yang menunjukkan pendeklarasian tuple, pengindeksan, dan slicing:

t = ('foo', 'bar', 'baz', 'qux', 'quux', 'corge')

print(t)

('foo', 'bar', 'baz', 'qux', 'quux', 'corge')

print(t[0])

print(t[-1])

foo

corge

# packing and unpacking

(s1, s2, s3, s4) = ('foo', 'bar', 'baz', 'qux')

s1

'foo'

## Python Dictionary

ictionaries and lists share the following characteristics:

* Both are mutable.
* Both are dynamic. They can grow and shrink as needed.
* Both can be nested. A list can contain another list. A dictionary can contain another dictionary. A dictionary can also contain a list, and vice versa.

Dictionaries differ from lists primarily in how elements are accessed:

* List elements are accessed by their position in the list, via indexing.
* Dictionary elements are accessed via keys.

### **Defining a Dictionary**

Dictionaries adalah implementasi Python dari struktur data yang lebih dikenal sebagai associative array. Dictionaries terdiri dari key-value pairs. Setiap key-value pairs memetakan key ke value yang terkait.

Kalian bisa mendefinisikan Dictionaries dengan mengapit daftar key-value pairs yang dipisahkan koma dalam tanda kurung kurawal ({}). Titik dua (:) memisahkan setiap kunci dari nilai yang terkait.

MLB\_team = {

'Colorado': 'Rockies',

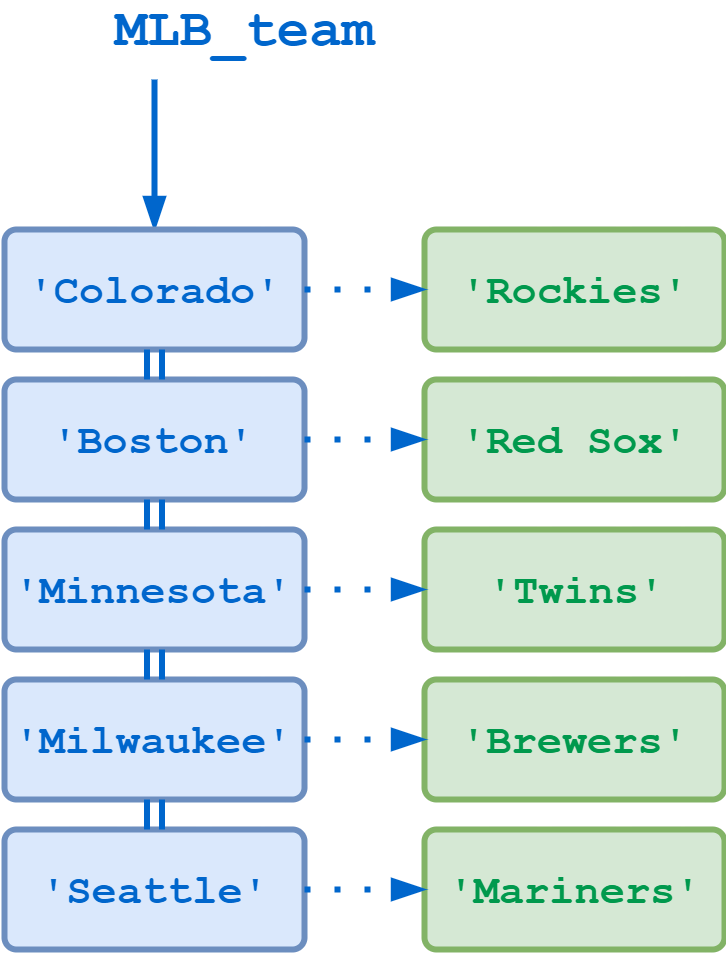
'Boston': 'Red Sox',

'Minnesota': 'Twins',

'Milwaukee': 'Brewers',

'Seattle': 'Mariners'

}



### **Accessing Dictionary Values**

Sebuah nilai diambil dari dictionary dengan menentukan kunci yang sesuai dalam tanda kurung siku ([]):

print(MLB\_team['Minnesota'])

print(MLB\_team['Colorado'])

# error

# MLB\_team['Toronto']

Twins

Rockies

#Adding an entry to an existing dictionary is simply a matter of assigning a new key and value:

MLB\_team['Kansas City'] = 'Royals'

MLB\_team

{'Colorado': 'Rockies',

'Boston': 'Red Sox',

'Minnesota': 'Twins',

'Milwaukee': 'Brewers',

'Seattle': 'Mariners',

'Kansas City': 'Royals'}

# If you want to update an entry, you can just assign a new value to an existing key:

MLB\_team['Seattle'] = 'Seahawks'

MLB\_team

{'Colorado': 'Rockies',

'Boston': 'Red Sox',

'Minnesota': 'Twins',

'Milwaukee': 'Brewers',

'Seattle': 'Seahawks',

'Kansas City': 'Royals'}

# To delete an entry, use the del statement, specifying the key to delete:

del MLB\_team['Seattle']

MLB\_team

{'Colorado': 'Rockies',

'Boston': 'Red Sox',

'Minnesota': 'Twins',

'Milwaukee': 'Brewers',

'Kansas City': 'Royals'}

### **Building a Dictionary Incrementally**

Mendefinisikan dictionary menggunakan tanda kurung kurawal dan key-value pairs, seperti yang ditunjukkan di atas, tidak masalah jika kalian mengetahui semua kunci dan nilai sebelumnya. Tetapi bagaimana jika kalian ingin membuat dictionary sambil jalan?  
  
Kalian bisa mulai dengan membuat dictionary kosong, yang ditentukan dengan tanda kurung kurawal kosong. Kemudian kalian dapat menambahkan kunci dan nilai baru satu per satu:

person = {}

type(person)

person['fname'] = 'Hack'

person['lname'] = 'PTP'

person['age'] = 51

person['spouse'] = 'Edna'

person['children'] = ['Ralph', 'Betty', 'Joey']

person['pets'] = {'dog': 'Fido', 'cat': 'Sox'}

person

{'fname': 'Hack',

 'lname': 'PTP',

'age': 51,

'spouse': 'Edna',

'children': ['Ralph', 'Betty', 'Joey'],

'pets': {'dog': 'Fido', 'cat': 'Sox'}}

print(person['fname'])

print(person['lname'])

Hack

PTP

print(person['children'])

print(person['children'][1])

['Ralph', 'Betty', 'Joey']

Betty

print(person['pets'])

print(person['pets']['cat'])

{'dog': 'Fido', 'cat': 'Sox'}

Sox

# Built-in Methods

d = {'a': 10, 'b': 20, 'c': 30}

# items

print(d.items())

# keys

print(d.keys())

# values

print(d.values())

dict\_items([('a', 10), ('b', 20), ('c', 30)])

dict\_keys(['a', 'b', 'c'])

dict\_values([10, 20, 30])

## Line Continuation

Misalkan satu pernyataan dalam kode Python kalian sangat panjang. Misalnya, kalian mungkin memiliki assignment statement dengan banyak terms:

person1\_age = 42

person2\_age = 16

person3\_age = 71

someone\_is\_of\_working\_age = (person1\_age >= 18 and person1\_age <= 65) or (person2\_age >= 18 and person2\_age <= 65) or (person3\_age >= 18 and person3\_age <= 65)

someone\_is\_of\_working\_age

True

someone\_is\_of\_working\_age = (

(person1\_age >= 18 and person1\_age <= 65)

or (person2\_age >= 18 and person2\_age <= 65)

or (person3\_age >= 18 and person3\_age <= 65)

)

someone\_is\_of\_working\_age

True

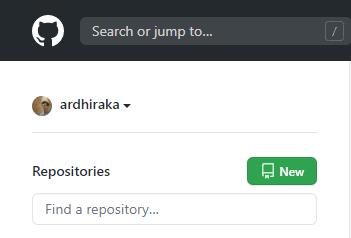
## gitToday

Enable fullscreen

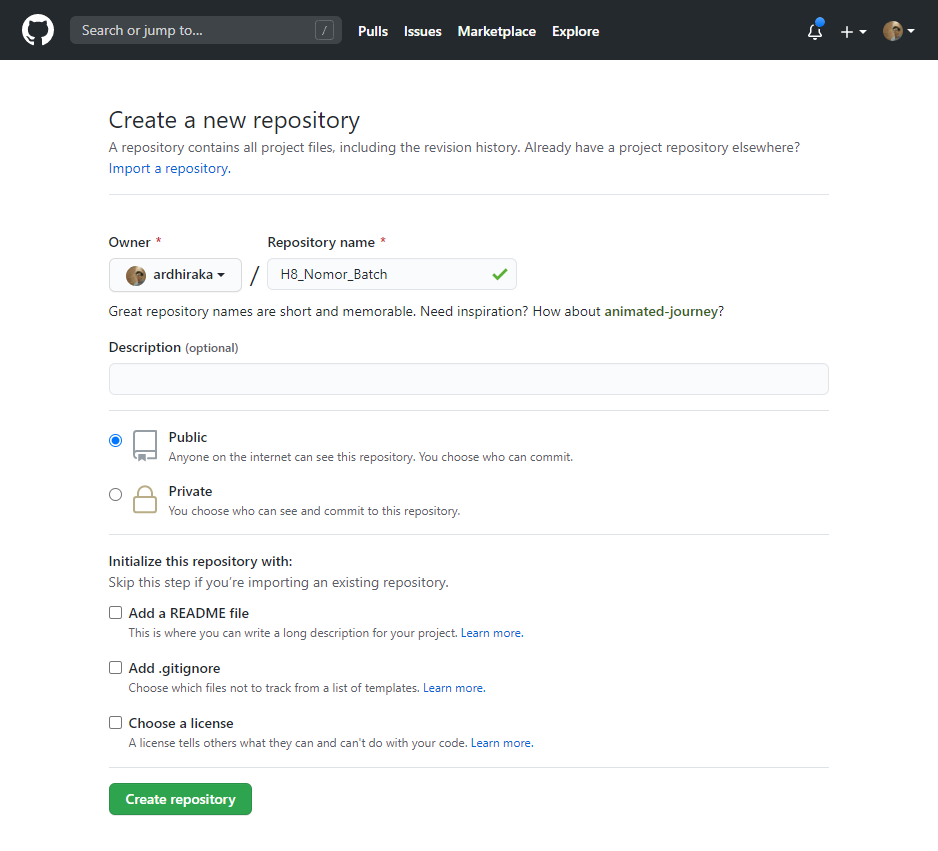
# git Today

Terlebih dahulu, download git dari [sini](https://git-scm.com/downloads)! Pilih sesuai OS kalian masing-masing lalu install. Sambil menunggu installasi, registrasi akun github dari [sini](https://github.com/). Setelahnya buat repository baru.

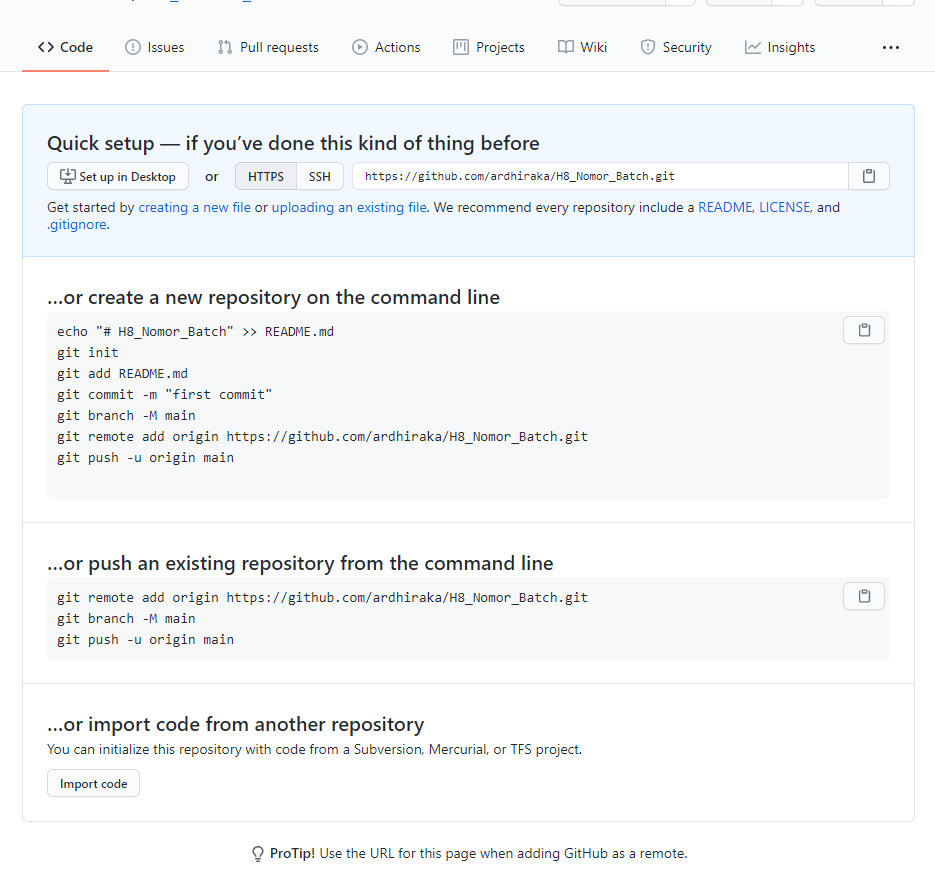
* Pada halaman home dari Github, pilih New.



* Masukan nama repository berupa H8\_<Batch\_Number>, hanya isi pada Repository name, tidak perlu mengisi deskripsi atau centang apapun pada checkbox dibawahnya. Pilih Create Repository.



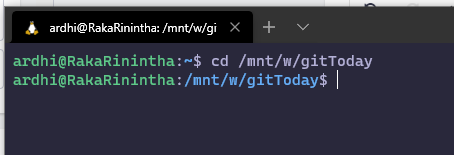
* Kalian akan diarahkan ke halaman dibawah ini, jangan tutup halaman tersebut.



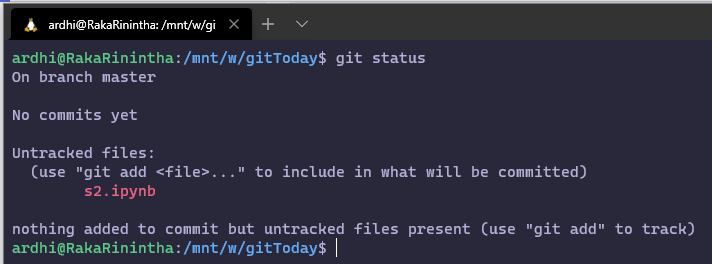
* Lihat gambar diatas, perhatikan pada line git remote add origin https://github.com/ardhiraka/H8\_Nomor\_Batch.git kalian akan membutuhkannya nanti.

Setelah installasi dan create repo berhasil, lakukan langkah berikut:

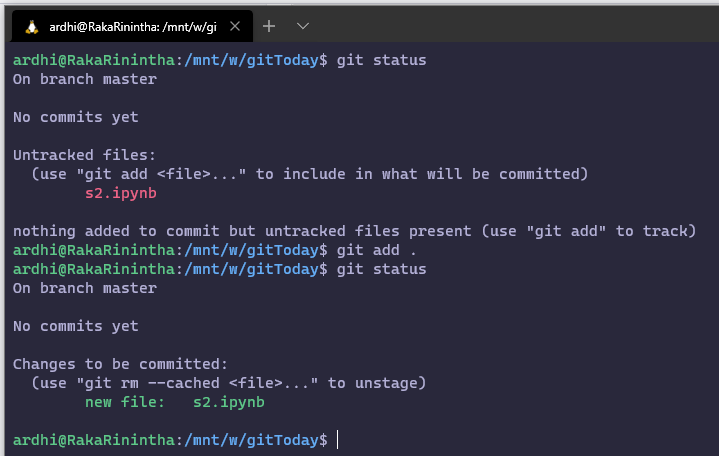
* (Opsional) Masukan untuk command git --version untuk memastikan bahwa git sudah terinstall dengan benar pada environment kalian.
* Masuk ke terminal, dan arahkan ke directory file Jupyter kalian berada.



* Masukan command git init untuk menginisialisasi git di awal. Hal ini akan menyebabkan folder tersebut akan ditandai sebagai git project dan membuat sebuah folder bernama .git. Ini hanyalah **first-time setup**.
* Masukan command git config --global user.name "John Doe" dan git config --global user.email johndoe@example.com, sesuaikan dengan username dan email yang kalian pakai untuk akun github. Ini hanyalah **first-time setup**.
* Masukan command git remote add origin https://github.com/ardhiraka/H8\_Nomor\_Batch.git, (lihat gambar 3. Copy dari command di halaman repo kalian, link masing-masing mungkin berbeda). Ini hanyalah **first-time setup**.
* Pada terminal, ketik command git status untuk melihat status local repository kalian.



* Masukan command git add . untuk menambahkan file Jupyter di sesi ini ke dalam stagging area.



* Jika sudah tidak ada perubahan lagi, masukan command git commit -m "Initial Commit add Sesi 1" kemudian enter.
* Jalankan command git push origin master untuk melakukan push file yang telah dicommit ke dalam remote repository. Mungkin akan diminta untuk memasukan password untuk pertama kali push repository.
* Jangan lupa untuk melihat repository kalian pada website github untuk memastikan bahwa file kalian sudah dipush dengan benar.

## TipsPy

Enable fullscreen

# TipsPy Day 1

Tahukan kalian ada beberapa tools lain yang populer untuk membuat Notebook selain di Jupyter Notebook?

1. **Google Colab**

Kalian dapat membuat Notebook di cloud dengan komputasi yang sangat baik dengan menggunakan Google Colab yang bisa diakses di alamat <https://colab.research.google.com/> . File notebook yang kalian buat akan langsung masuk ke Google Drive kalian dan bisa pula langsung dikoneksikan dengan github!

1. **Visual Studio Code**

Dengan VSCode, kalian dapat membuat dan bekerja dengan file notebook langsung di dalam kode editor ini. Tapi tetap pastikan kalau Jupyter server sudah terinstall di environment kalian ya.

1. **Amazon Sagemaker**

Dengan AWS Sagemaker kita dapat membuat dan mengelola notebook Jupyter yang selanjutnya dapat digunakan untuk memproses data dan melatih dan menerapkan model ML lebih lanjut. Untuk pelatihan dan penerapan model, Sagemaker menyediakan API. Amazon SageMaker menyediakan konsol yang memungkinkan kita menggunakan konsol UI untuk memulai pelatihan model atau menerapkan model. Sagemaker memungkinkan integrasi model ML dengan mudah dalam aplikasi dan dapat diakses di <https://aws.amazon.com/sagemaker/>

1. **Azure Notebooks**

Microsoft Azure sangat mirip dengan Colab dalam hal fungsionalitas. Kedua platform ini memiliki fungsionalitas berbagi cloud yang tersedia secara gratis. Azure Notebooks unggul dalam hal kecepatan dan jauh lebih baik daripada Colab dan dapat diakses di <https://azure.microsoft.com/>