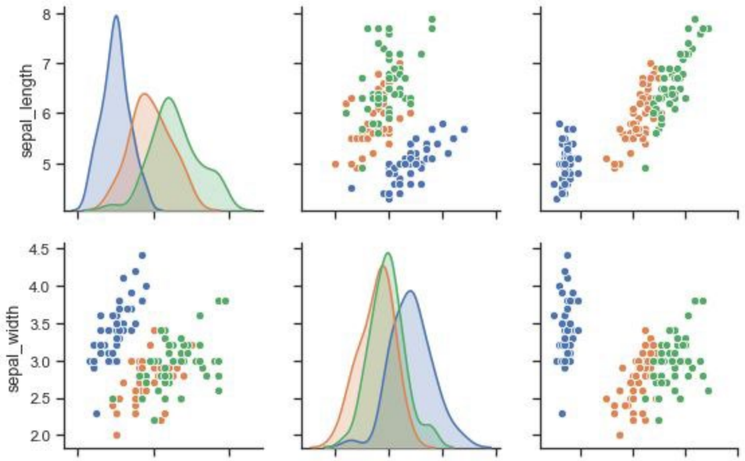
Basic Python Programming and Visualization (+ Data Analysis)



xx - xx Oct 2023

Mr. Pipatpon Mitasit

วิศวกรระดับ 7
แผนกวางแผนโครงการพลังงานหมุนเวียน (หวคม-พ.)
กองแผนงานและศึกษาความเหมาะสมพลังงานหมุนเวียน (กผศม-พ.)
ฝ่ายพัฒนาและแผนงานโรงไฟฟ้า (อผร.)
สายงานรองผู้พัฒนาโรงไฟฟ้า และ พลังงานหมุนเวียน (รวพ.)

Rules

OK

About

Be in B.Eng. (Electrical [Control] Engineering) KMUTNB (2nd class honors)
M.Eng. (Energy Engineering) KMUTNB

Mr. Pipatpon Mitasit
(Mc) – 33 Yrs

University Electrical

Work Before EGAT

Work EGAT

Year 1-2

Year 3-4

Toyota

Scneider Electric Thailand

EGAT

2008

- First meet C/C++ , Java, PLC Programing Language

2010

- Embedded System / Micro Controller
- Thesis : Power Electronics

2012 - 1 Year

- First meet Powerful function Excel/VBA
- Lean/Short Time Working
- MATLAB Simulink Test/Design Software Module (Adaptive Cruise Control)

2013 - 3 Month

- Designed Power Monitoring Software/Tools work with Scneider Power Meter Products via Database (PM-700)

2013 -2015

- Designed and Feasibility Study Solar / FPV / RE Power Plants

2014 - 2021

- Planning Power Plant Projects
- RE in PDP2018, PDP2022
- Interest : Excel, Data Analytics , Data Analysis and Visualization
- ASP.Net Web App Develop

Present

- AI Specialist : Data Management Platform (PMO)
- AI Lecturer in EGAT

PUBLICATIONS: " Development of a Dual Axis Sun Tracking System with Astronomical Equation Program on Arduino Via GPS Module" ACSEE 2017 in Japan (ISSN:2186-2311 The Asian Conference on Sustainability, Energy & the Environment 2017)

Proceeding Presentation VDO

Energy Engineering

2015

- Thesis : Development of a Dual Axis Sun Tracking System with Astronomical Equation Program on Arduino Via GPS Module

Super AI Eng. SEASON 2

2021-2022

- Machine Learning / Deep Learning
- Data Science / IoT
- Natural Language Processing (NLP)
- Image Processing / Signal Processing
- Mathematics for AI
- Programming and Tools in AI
- Bronze Medal

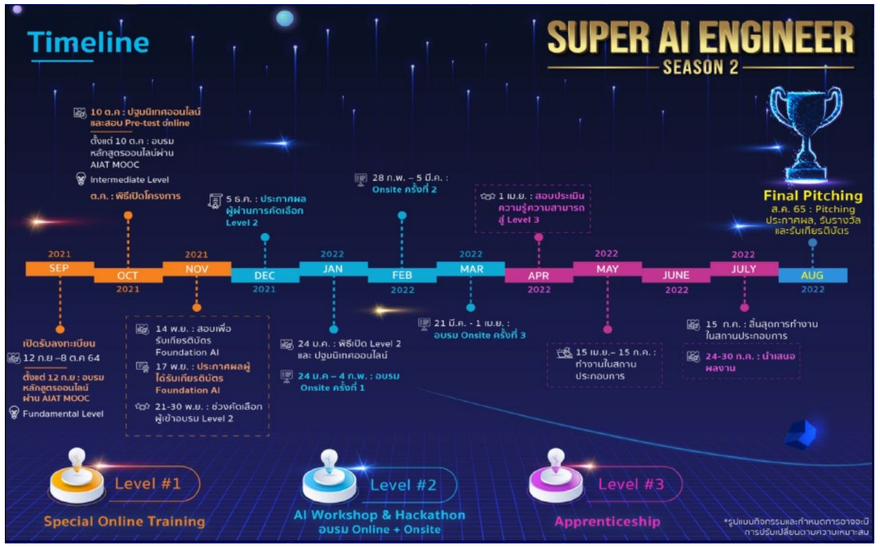
To be continue:

CV

Super AI Season2 Shared

Linkedin Github Upwork Kaggle Medium

Super AI Engineer (Season 2)



5,600 Applicants **120 Applicants**

Chatbot Command with IoT
Link : Medium

Measuring Data with ESP32 built dashboard on Google sheet and new blynk dashboard
Link : Medium

hand recognition with mediapipe
Link : Medium

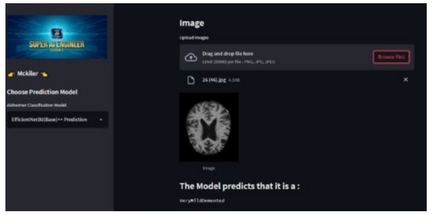
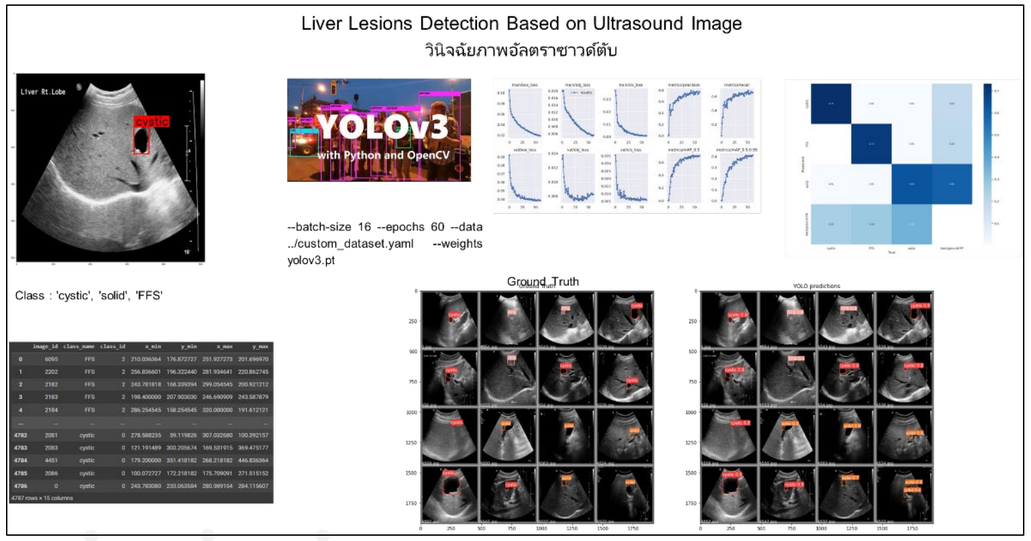


Image Processing (Alzheimer MRI Scan) Classification
Link : Medium



Cert.

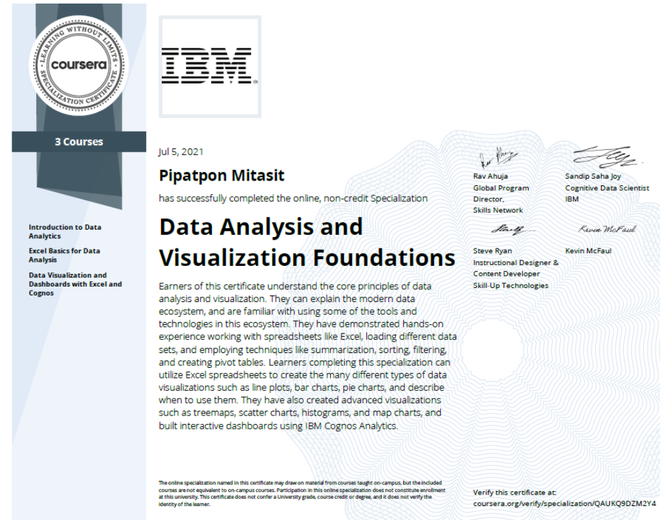

SUPER AI ENGINEER 2021
Artificial Intelligence Association of Thailand
This is to certify that
Pipatpon Mitasit
has completed enrichment program in Super AI Engineer Season 2
and obtained SUPER AI ENGINEER (Bronze Medal)
on the 19th August 2022
Dr. Thepchai Supnithi
Vice President, AIAT
Prof Dr. Thanarux Theeramunkong
President, AIAT
Dr. Sanparith Marukatat
Executive Chairman
of Academic Committee, AIAT


CHULA MOOC
achieve
PATHWAY FORWARD
IT / DATA / TECHNOLOGY
GOLD
PIPATPON MITASIT
has successfully completed the open online non-credit courses
with gold award
DATA SCIENCE PATHWAY
provided by Chulalongkorn University
on January 22, 2023
Parichart Sthapitanonda, Ph.D
Chairperson
CHULA MOOC Achieve
Natawut Nupairoj, Ph.D
Director
Data Science Pathway Committee
Verify at: cert.chulamoocachieve.com/LMJVB.pdf

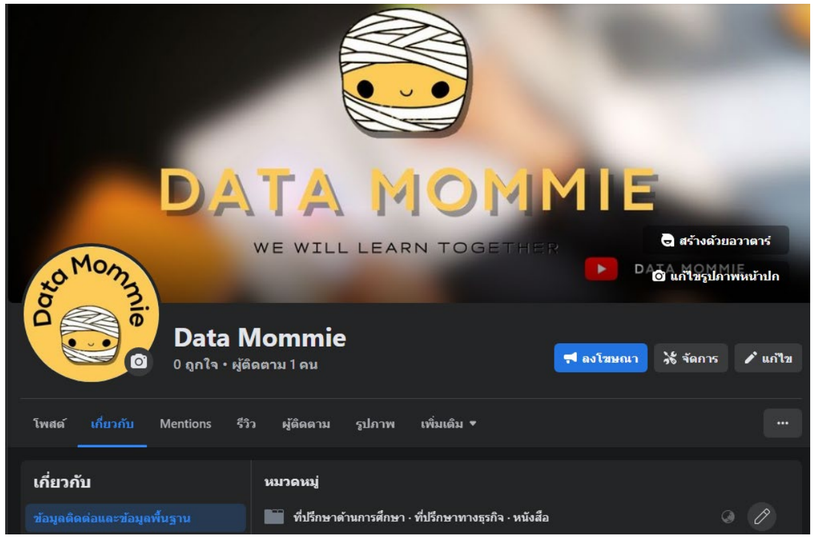

พิพัฒน์พล มิตะสิทธิ์
Achievement Profile
successfully completed, received a passing grade, and was awarded this MUx Honor Code
Certificate of Completion in
Introduction to Data Analytics
Dr. Siripen Pongpaichet
Instructor of ICT Faculty
ICT Faculty
Miss Siriporn Rojanakosol
Computer Technical Officer of ICT Faculty
ICT Faculty
http://mux.mahidol.ac.th/edxphp/getcer/d40eff1fca4f5a490d3e98d62c251c85edd22a4b
29 มิถุนายน 2564

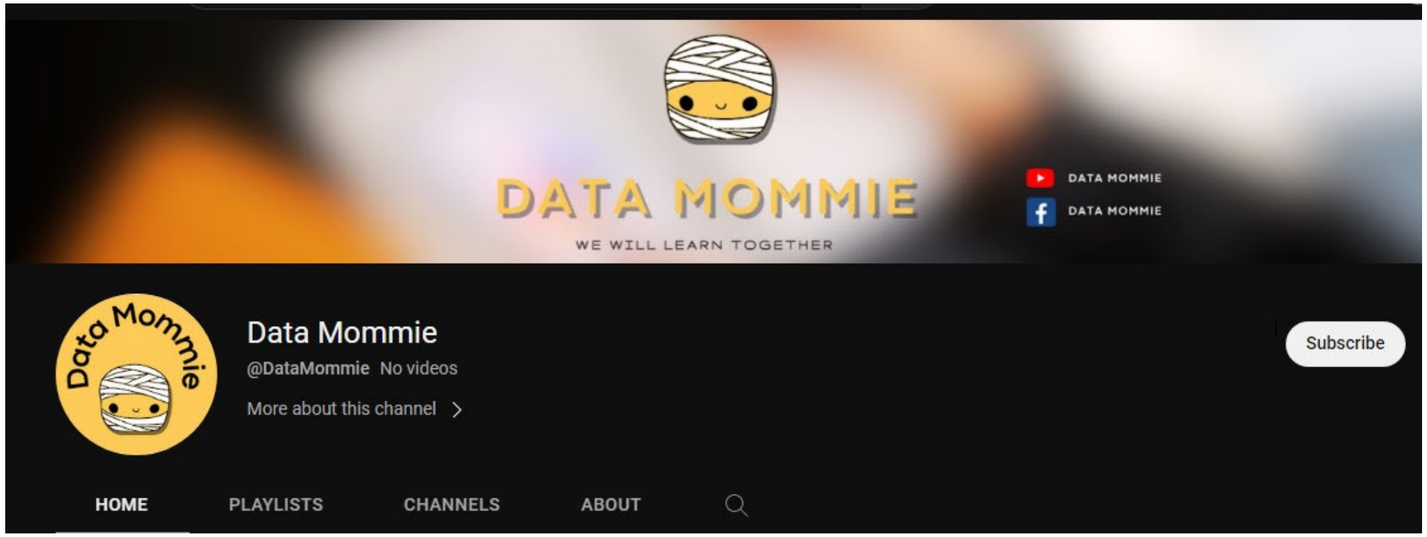

coursera
7 Courses
Foundations of Data Science
Get Started with Python
Go Beyond the Numbers: Translate Data into Insights
The Power of Statistics
Regression Analysis: Simplify Complex Data Relationships
The Nuts and Bolts of Machine Learning
Google Advanced Data Analytics Capstone
May 17, 2023
Mr. Pipatpon Mitasit
has successfully completed the online, non-credit Professional Certificate
Google Advanced Data Analytics
Those who earn the Google Advanced Data Analytics Certificate have completed seven courses that include hands-on, practice-based assessments and are designed to prepare them for advanced roles in data analytics and entry-level roles in data science. They are competent in exploring large datasets, applying data analysis techniques, and building models to extract insights. They are also competent in machine learning, predictive modeling, and statistics.
Amanda Brophy
Global Director of Google Career Certificates
Verify this certificate at:
https://coursera.org/verify/professional-cert/ZZX59P5XNN4C


coursera
3 Courses
Foundations of Business Intelligence
The Path to Insights: Data Models and Pipelines
Decisions, Decisions: Dashboards and Reports
May 17, 2023
Mr. Pipatpon Mitasit
has successfully completed the online, non-credit Professional Certificate
Google Business Intelligence
Those who earn the Google Business Intelligence Certificate have completed three courses that include hands-on, practice-based assessments and are designed to prepare them for entry-level roles in business intelligence. They are competent in creating processes that transform data into actionable insights to inform business decisions, developing data visualizations, and building dashboards that employ design best practices to meet stakeholder needs.
Amanda Brophy
Global Director of Google Career Certificates
Verify this certificate at:
https://coursera.org/verify/professional-cert/YF2BW9TZEDLW


coursera
3 Courses
Introduction to Data Analytics
Excel Basics for Data Analysis
Data Visualization and Dashboards with Excel and Cognos
IBM
Jul 5, 2021
Pipatpon Mitasit
has successfully completed the online, non-credit Specialization
Data Analysis and Visualization Foundations
Earners of this certificate understand the core principles of data analysis and visualization. They can explain the modern data ecosystem, and are familiar with using some of the tools and technologies in this ecosystem. They have demonstrated hands-on experience working with spreadsheets like Excel, loading different data sets, and employing techniques like summarization, sorting, filtering, and creating pivot tables. Learners completing this specialization can utilize Excel spreadsheets to create the many different types of data visualizations such as line plots, bar charts, pie charts, and describe when to use them. They have also created advanced visualizations such as treemaps, scatter charts, histograms, and map charts, and built interactive dashboards using IBM Cognos Analytics.
Rav Ahuja
Global Program Director, Skills Network
Sandip Saha Joy
Cognitive Data Scientist
IBM
Steve Ryan
Instructional Designer & Content Developer
Skill-Up Technologies
Kevin McFaul
Verify this certificate at:
coursera.org/verify/specialization/QAUKQ9DZM2Y4

Social Media





Data Mommie

Data Mommie

Agenda

DAY 1

9:00 – 12:00

1. Introduction to Data Science
2. Introduction to Python and Google Co-Lab
3. First for Use Google Colab
4. Python Operators and Variables
5. Function
6. Data Type

13:00 – 16:00

1. Conditional Statements
2. Data Type (List)
3. Loops in Python
4. Data Type (Dictionary)
5. Modules and library functions

DAY 2

9:00 – 16:00

1. Numpy
2. Pandas

13:00 – 16:00

1. Visualization with Matplotlib and Seaborn

DAY 3

9:00 – 16:00

1. Visualization with Matplotlib and Seaborn

13:00 – 16:00

1. Visualization with Matplotlib and Seaborn
2. Web Scraping with BeautifulSoup

Material

- https://colab.research.google.com/drive/1xcejrBMy9aaJTRm2nEfIf0_xT8jLmHQO?usp=sharing

- https://github.com/DataMommie/2023_Basic_Python_Programming_Visualization_EGAT

Introduction to Data Science

"DATA IS THE NEW OIL."

Coined in 2006 by Clive Humby, a British data commercialization entrepreneur, this now famous phrase was embraced by the World Economic Forum in a 2011 report, which considered data to be an economic asset, like oil.

From the beginning of recorded time until 2003, we created **5 exabytes** (5 billion gigabytes) **of data.**

In 2011 the same amount was created every two days.

By 2013, it's expected that the time will shrink to 10 minutes.

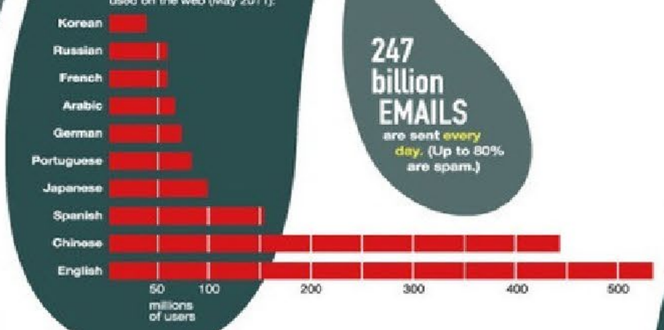
There are nearly as many bits of information in the digital universe as there are stars in our actual universe.

There are **133 million BLOGS** on the web.

As of August 2012, there were just over **4 million** articles in the English Wikipedia.

80% of all humans own a mobile phone of some sort. Out of 5 billion mobiles, 1 billion are smartphones. (In Singapore, 54% of citizens are smartphone users.)

English is the dominant language of the web. But by 2014 it will be **Chinese,** if its current rate of increase continues.



247 billion EMAILS are sent every day. (Up to 80% are spam.)

Every hour, we create enough internet traffic to fill **7 billion DVDs.** Side by side, that's that's seven times the height of Everest.

Just as a study of activity on Twitter gave residents, family members, and journalists advance warning of details about the devastating earthquake and tsunami in Japan, **high-frequency traders,** with the help of computer algorithms, use Big Data to follow trends and to act quickly on their findings.

These specialized algorithms make split-second decisions to buy or sell a commodity. New cable being laid under the Atlantic will shave **5 milliseconds** from the current 65 milliseconds it takes for trading instructions to travel between New York City and London.

With new fiber-optic cable, the round-trip time between New York and London will be 59.6 milliseconds.

This 5-millisecond saving is worth many millions of dollars to the trading firms who use the cable (and who will pay millions to do so).

60% of all humans (5.4 billion people) are active texters. In 2010, 193,000 text messages were sent every second.

10% of all photos ever taken were taken in 2011.

50% of 5-year-old kids in the U.S. are given access to a smartphone.

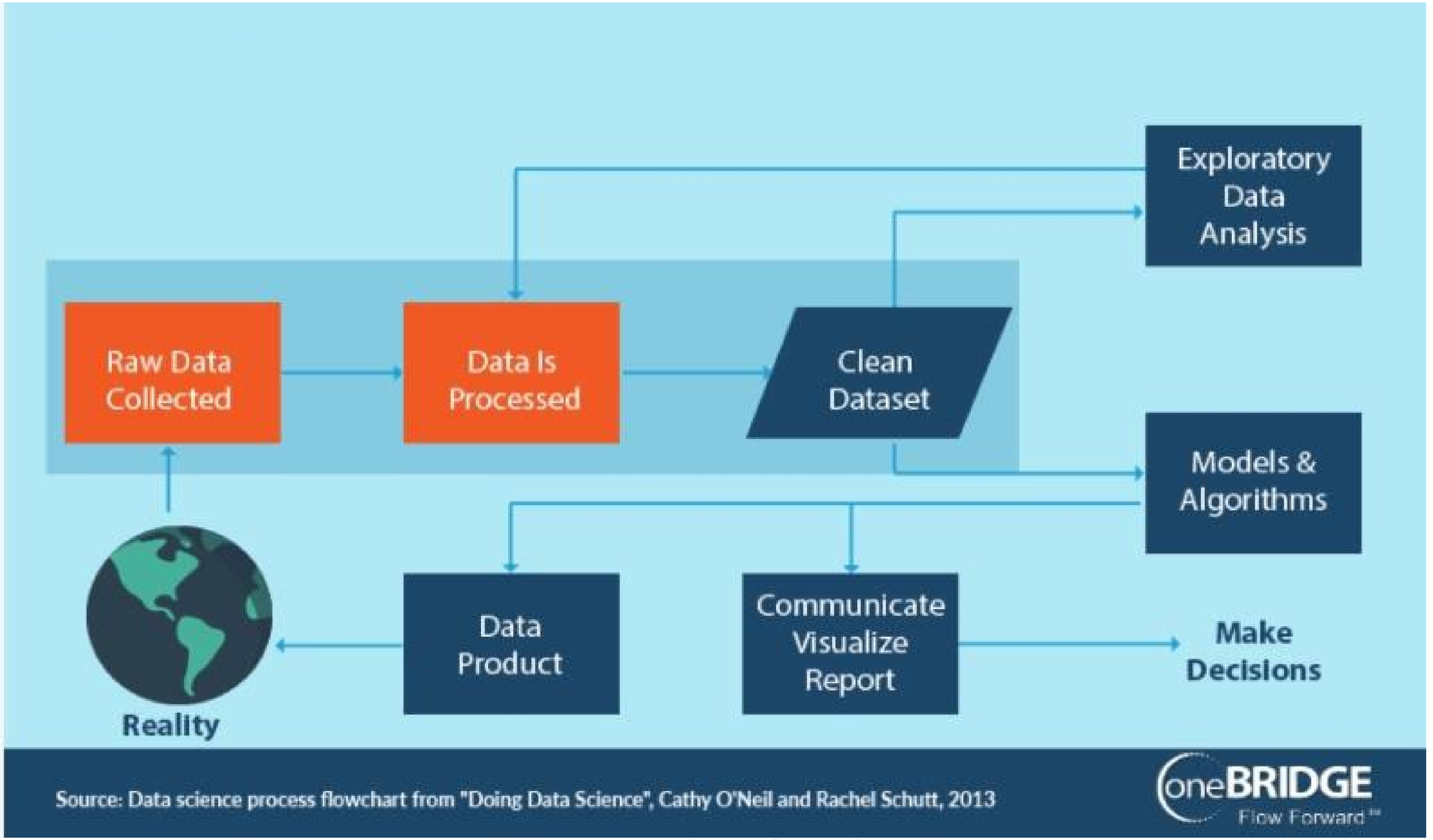
How they save 5 milliseconds

The depth of the Atlantic Ocean varies.

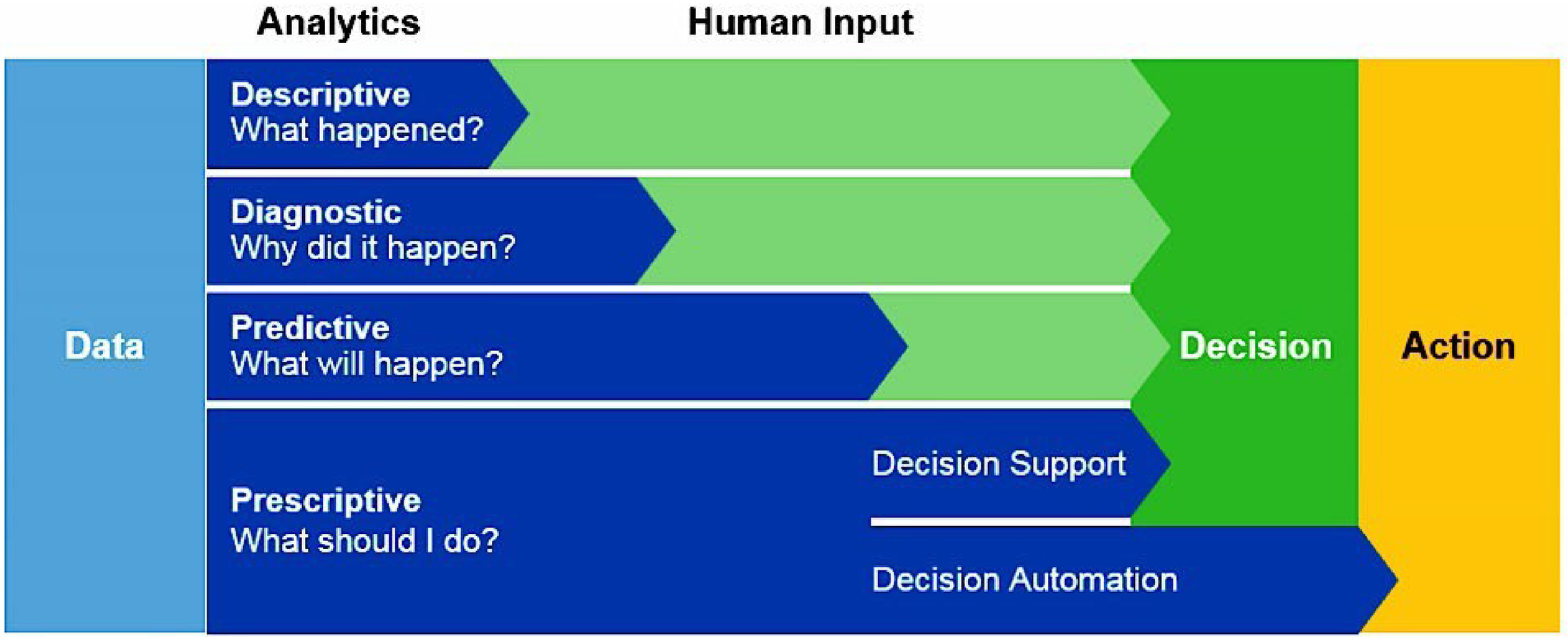
The new cable will lie on areas of the ocean floor that are up to 1,000 feet shallower than the current fastest cable. By taking a different route, the new cable is shorter, meaning that the time it takes for messages to travel along it is shortened.

Information Graphics by Nigel Holmes

DATA -> DECISION



DATA -> DECISION



Source: Gartner (May 2015)

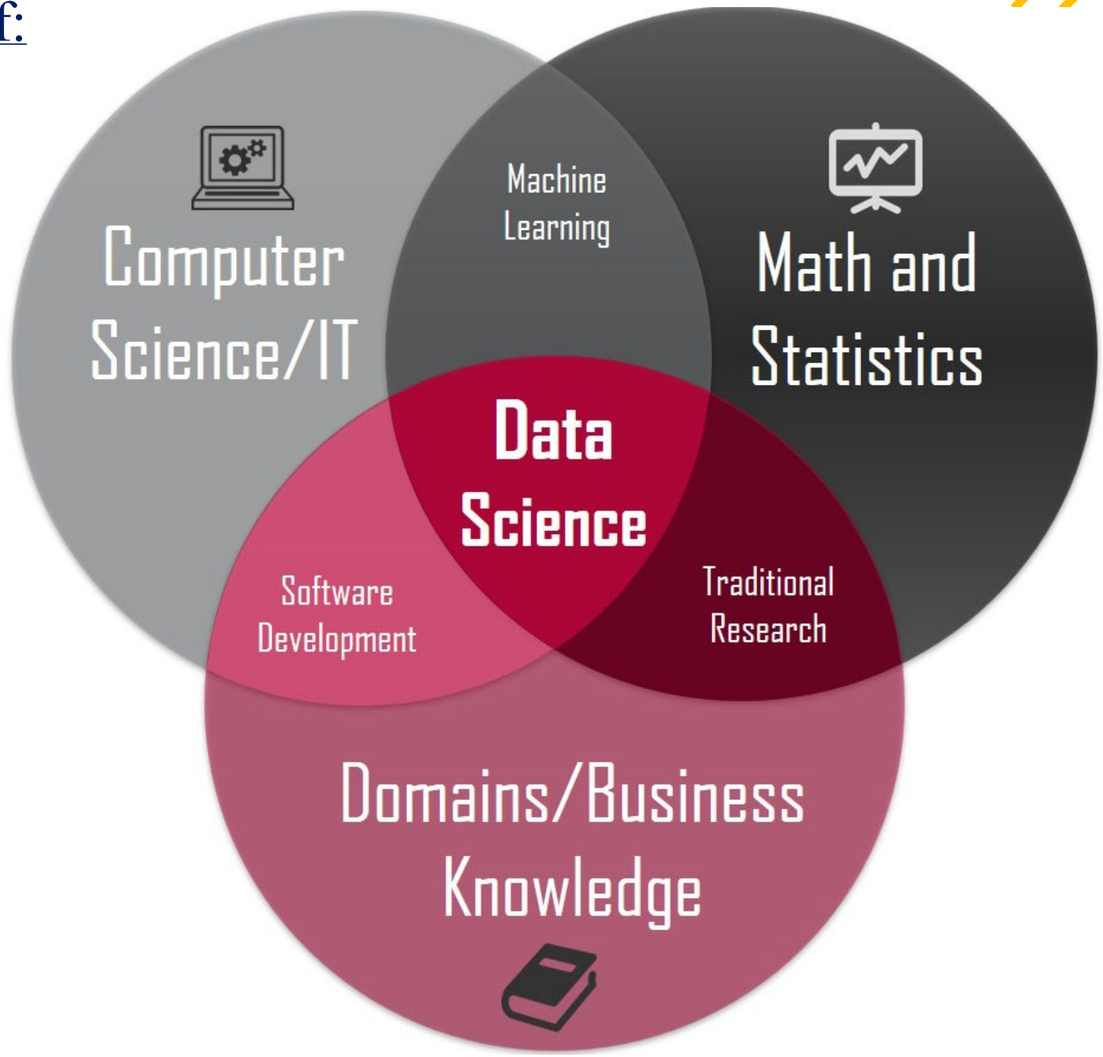
What is Data Science?

Data science is basically the newly developed blend of:

- Math and Statistics
  - Mathematical Modeling
  - Statistical and Stochastic Modeling
  - Probability, …

- Computer Science/ IT
  - Artificial Intelligence (AI)
  - Pattern Recognition
  - Visualization
  - Data warehousing
  - Databases, …

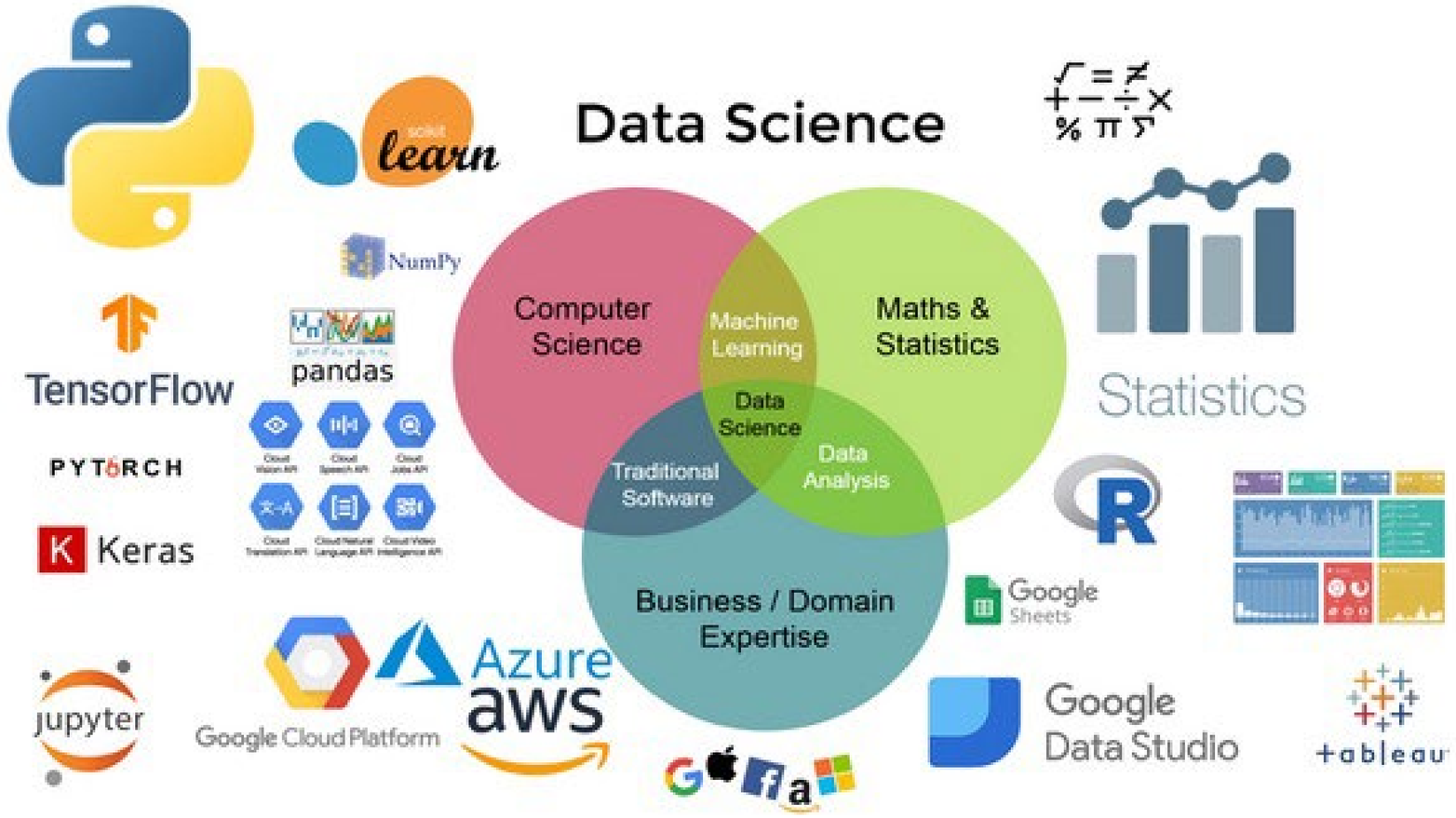
- Business Intelligence



Credit : Drew Conway's Venn diagram of data science from 2010

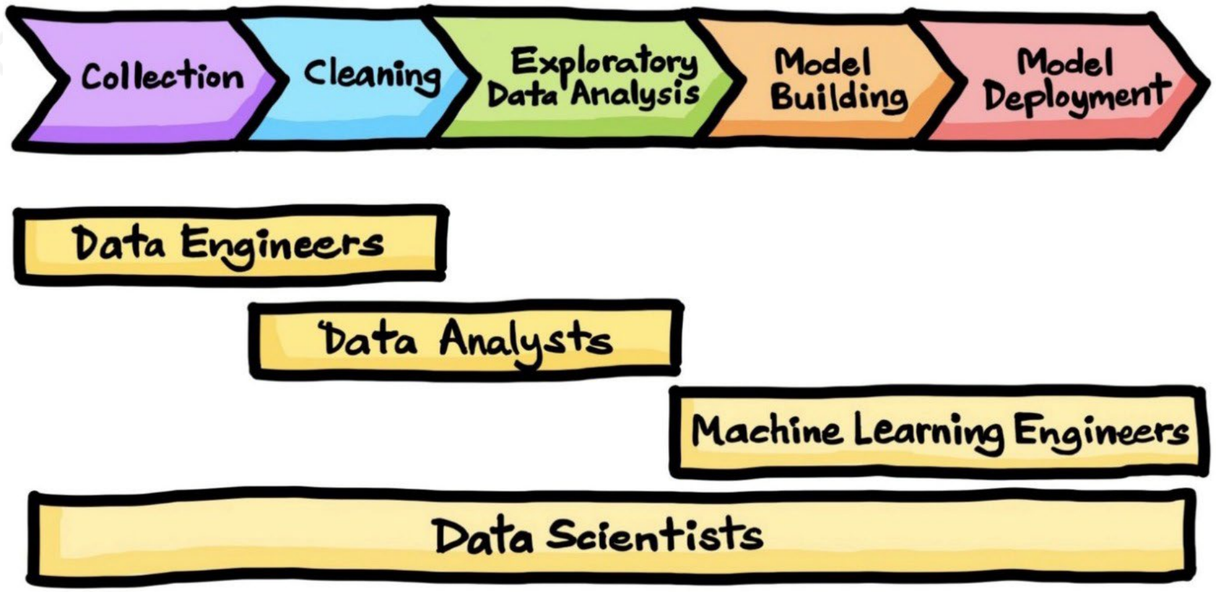
What is Data Science?

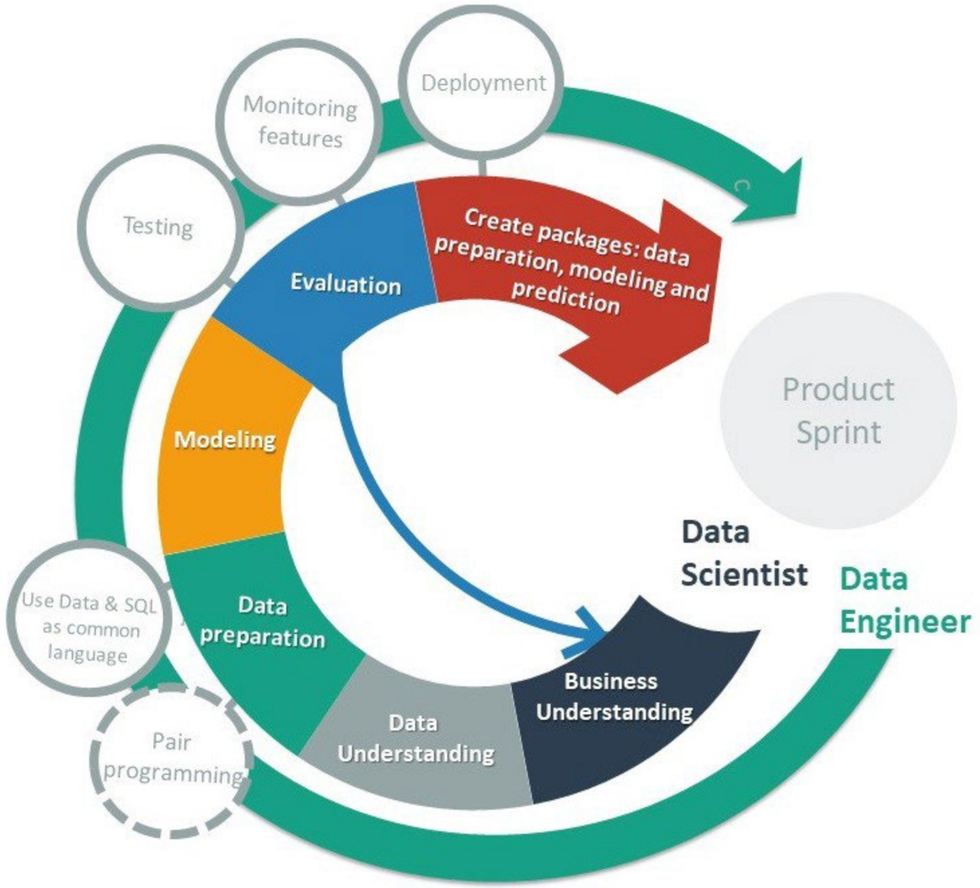
- **With the help of constantly upgrading tools and technologies, we get to understand the social, commercial, as well as industrial processes.**



Credit : https://www.meetup.com/ja-JP/Tokyo-Data-Science-Engineering-Business-Meetup-Group/

Data Science Life Cycle

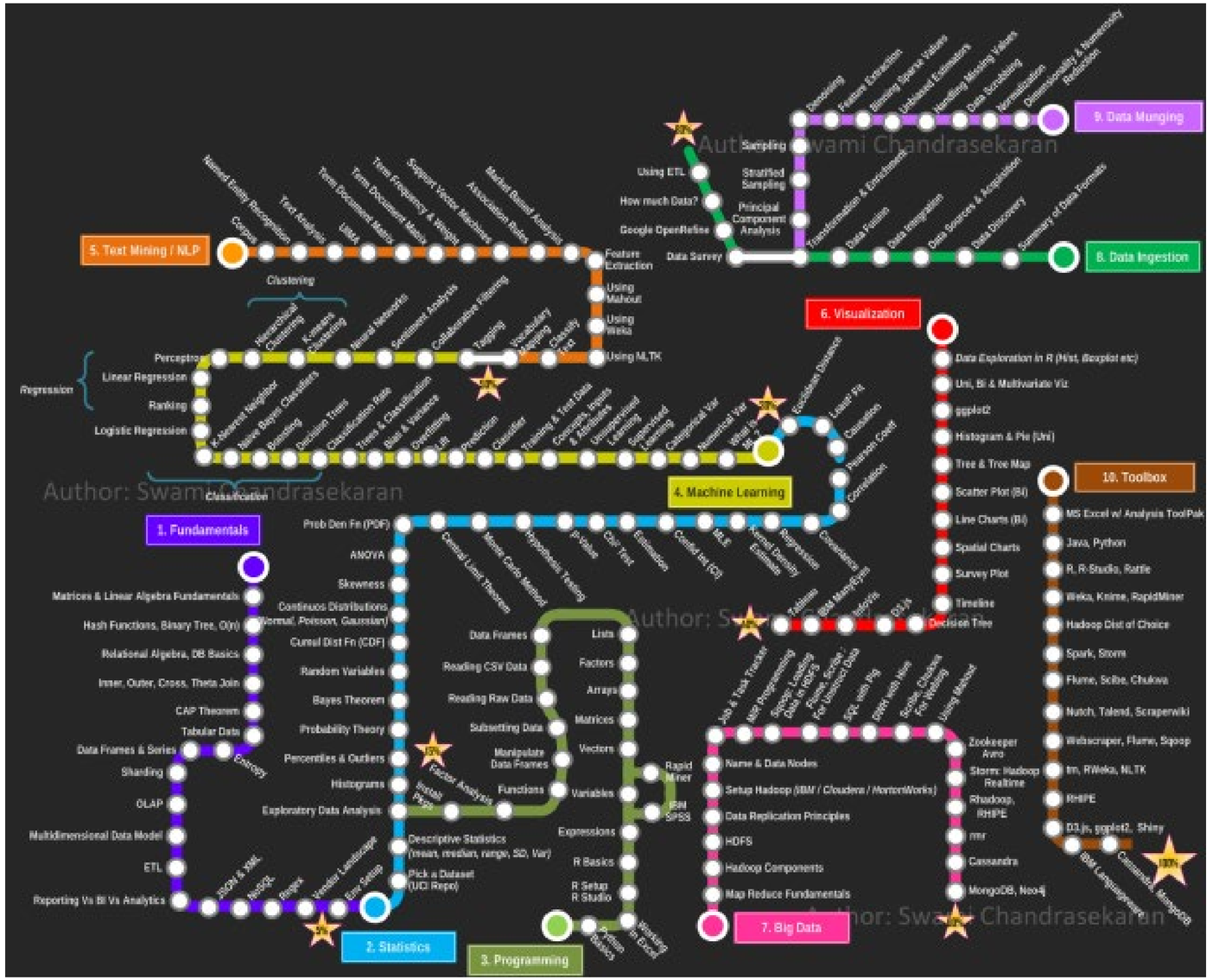




Credit: [The Data Science Process. A Visual Guide to Standard Procedures... | by Chanin Nantasenamat | Towards Data Science]

Credit: https://miro.medium.com/max/2400/1*-8sSf4po2pm-fzhBU7CUmQ.jpeg

Data Scientist Roadmap



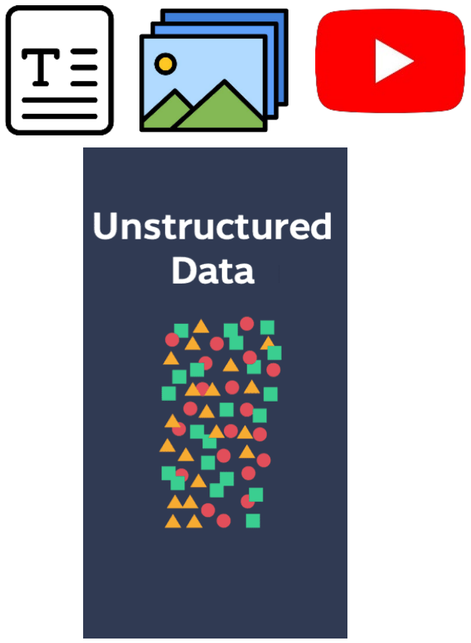
Credit: https://whatsthebigdata.com/2013/07/12/visualising-the-road-to-becoming-a-data-scientist/

What is Data?

Structured data

เช่น ข้อมูลใน Excel, จำนวนการซื้อขายกับลูกค้า, เปอร์เซ็นต์ความเคลื่อนไหวภายในตลาดหุ้น ฯลฯ

Unstructured data



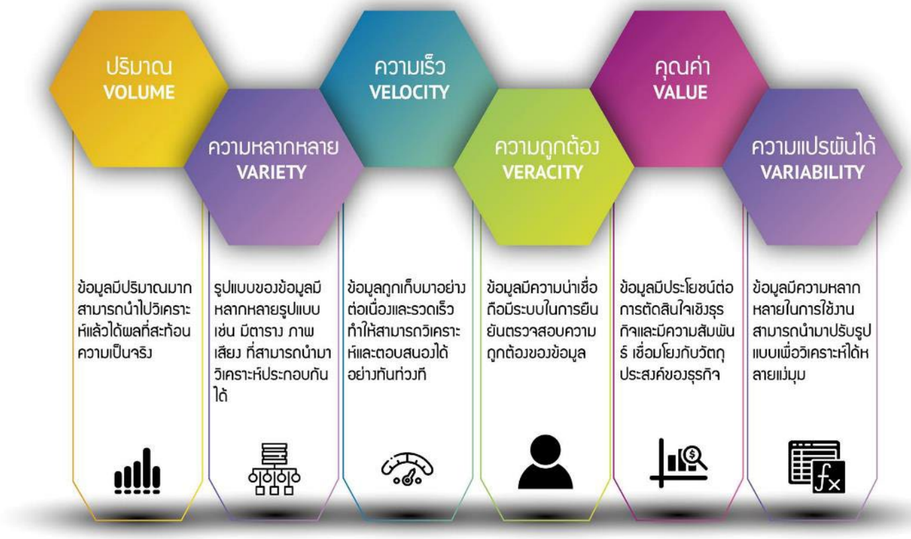
เช่น รูปแบบของรูปภาพ คลิปวิดีโอ ไฟล์เสียง หรือบทสนทนาโต้ตอบกับลูกค้าทาง Social Media

Semi structured data

เช่น สเตตัส ใน Social Media เป็นข้อมูลที่ไม่มีโครงสร้าง แต่ในกรณีที่มี Hashtag (#) เข้ามาช่วยในการจัดหมวดหมู่ จะทำให้ข้อมูลมีความเป็นระเบียบขึ้นมาเล็กน้อย

The Six Vs of Big Data

Big data ที่มีคุณภาพสูงควรมีลักษณะพื้นฐานอยู่ 6 ประการ (6 Vs) ดังนี้



Introduction to Python

- บิดา ภาษา Python ชื่อว่า Guido van Rossum
- Born 31 January 1956
- was born and raised in the Netherlands
- Master's degree in mathematics and computer science from the University of Amsterdam in 1982.
- Dutch
- ภาษา Python เกิดขึ้นปี 1990

Introduction to Python

V. 3.X เป็น Version ในปัจจุบัน
V. 2.X ยังคงมีการใช้งานอยู่แต่หยุดการพัฒนาแล้ว

คู่มือสามารถหาอ่านเพิ่มเติมได้
https://docs.python.org/3/tutorial/index.html

Active Python Releases

For more information visit the Python Developer's Guide.

Python version	Maintenance status	First released	End of support
3.11	bugfix	2022-10-24	2027-10
3.10	bugfix	2021-10-04	2026-10
3.9	security	2020-10-05	2025-10
3.8	security	2019-10-14	2024-10
3.7	security	2018-06-27	2023-06-27

https://www.python.org/downloads/

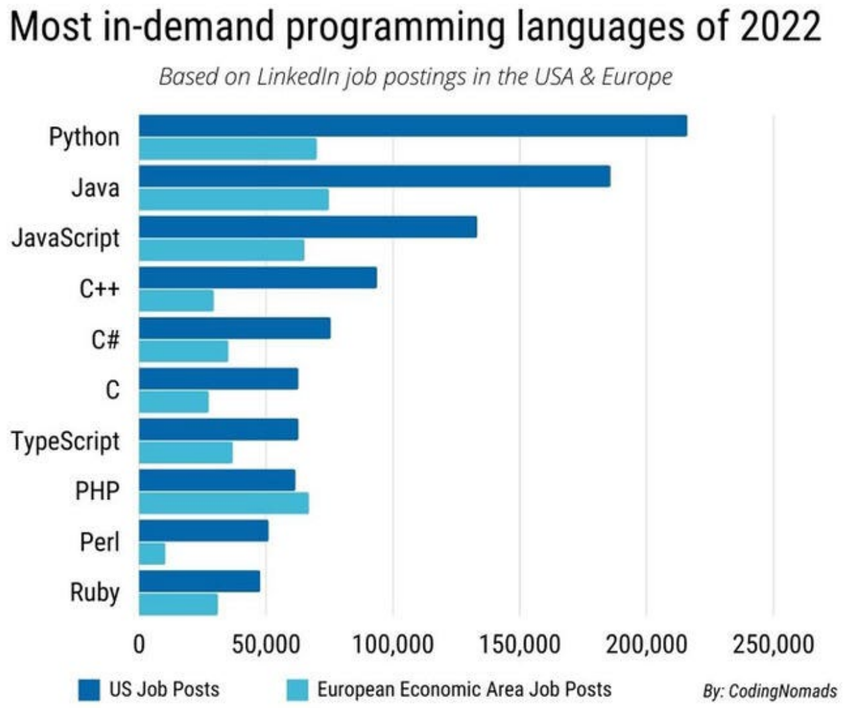
Why Python?

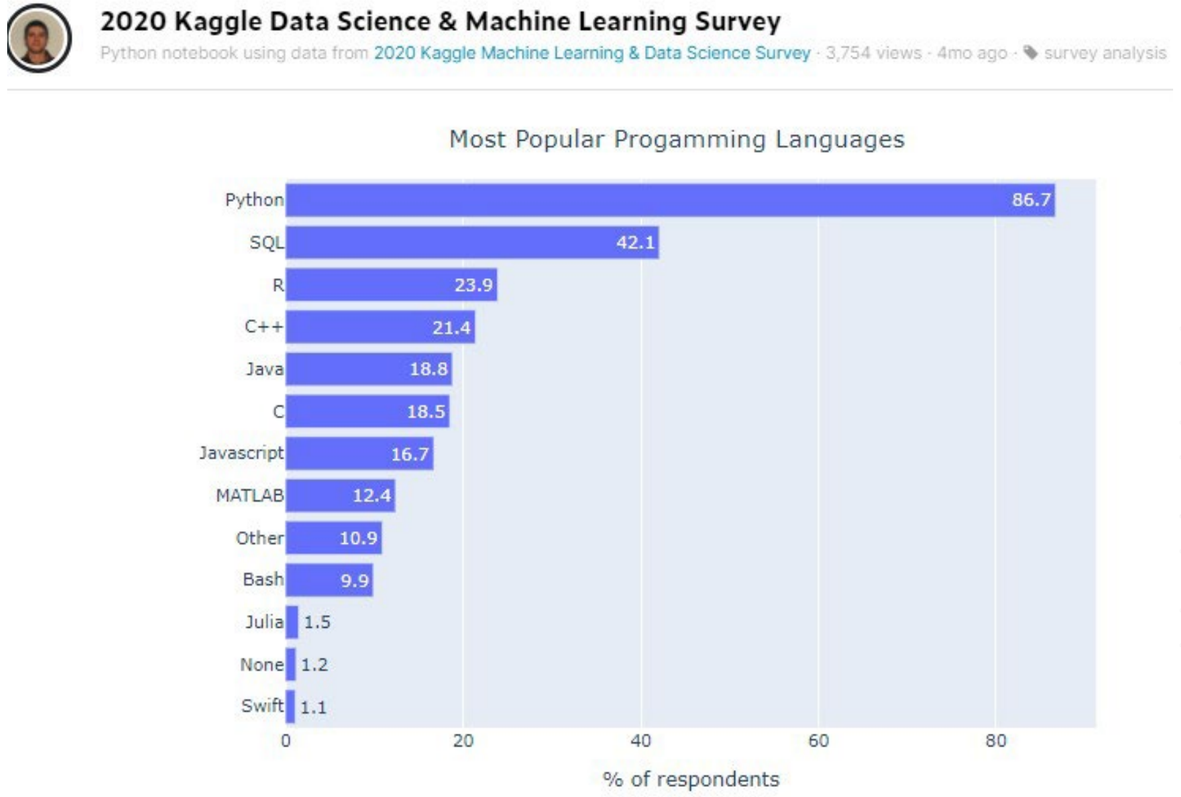
Python is an *open-source, general-purpose high-level* programming language.

Term	Definition
Open-source software (OSS)	Open-source means it is free. Python has a large and active scientific community with access to the software's source code and contributes to its continuous development and upgrading, depending on users' needs.
General-purpose	There is a broad set of fields where Python could be applied – web programming, analysis of financial data, analysis of big data, and more.
High-level	High-level languages employ syntax a lot closer to human logic, which makes the language easier to learn and implement.

Python's popularity lies on two main pillars. One is that it is an easy-to-learn programming language designed to be highly readable, with a syntax quite clear and intuitive. And the second reason is its user-friendliness does not take away from its strength. Python can execute a variety of complex computations and is one of the most powerful programming languages preferred by specialists.

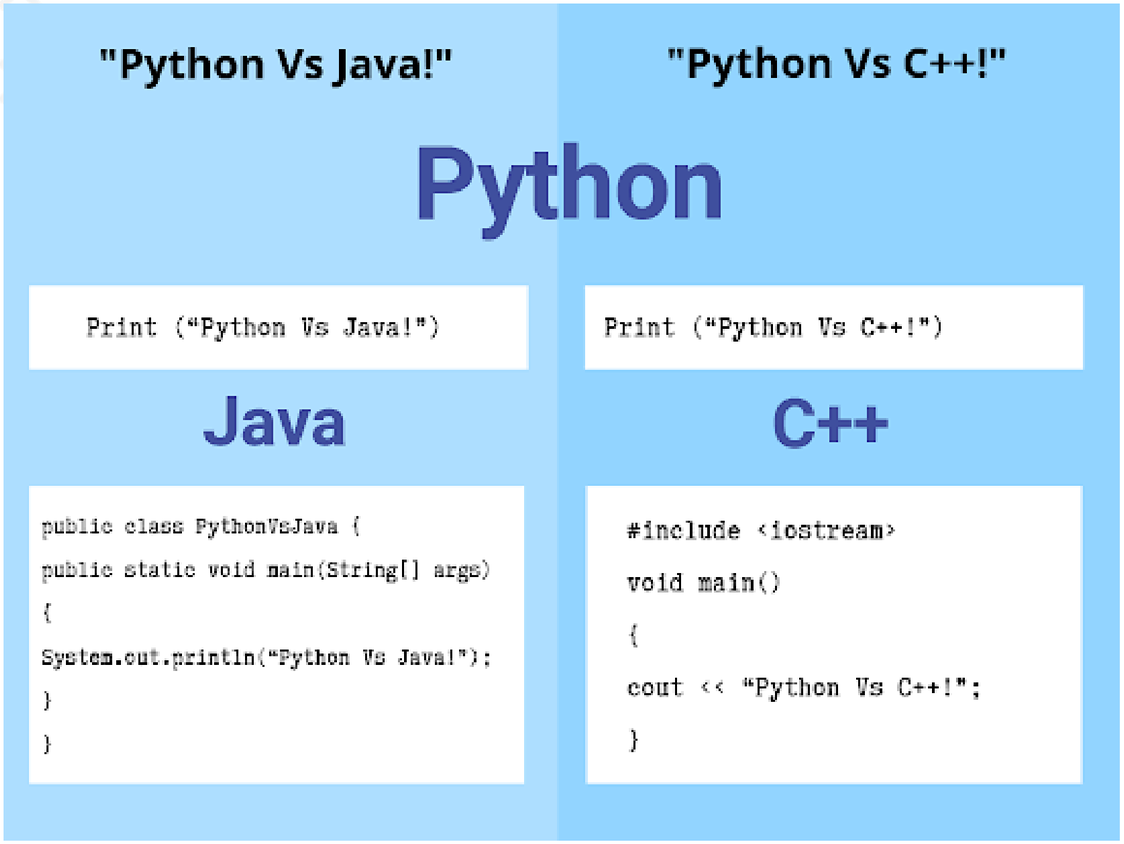
Programming Languages





Credit: https://www.kaggle.com/paultimothymooney/2020-kaggle-data-science-machine-learning-survey

Programming Languages



Java

```java
File dir = new File("."); // get current directory
File fin = new File(
    dir.getCanonicalPath() + File.separator + "Code.txt"
);

FileInputStream fis = new FileInputStream(fin);

// Construct the BufferedReader object
BufferedReader in = new BufferedReader(new InputStreamReader(fis));

String aLine = null;
while ((aLine = in.readLine()) != null) {
    // //Process each line, here we count empty lines
    if (aLine.trim().length() == 0) {}
}

// do not forget to close the buffer reader
in.close();
```

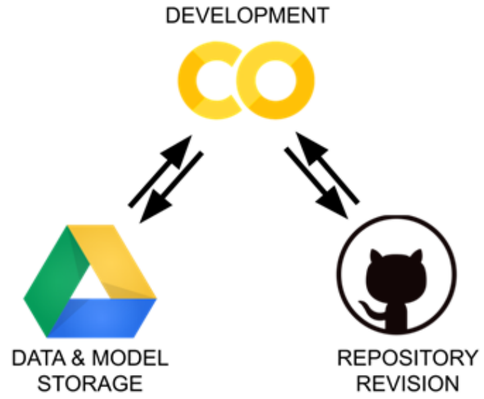
Python

```python
my_file = open("/home/xiaoran/Desktop/test.txt")

print(my_file.read())
my_file.close()
```

Introduction to Jupyter Notebook and Google Co-Lab

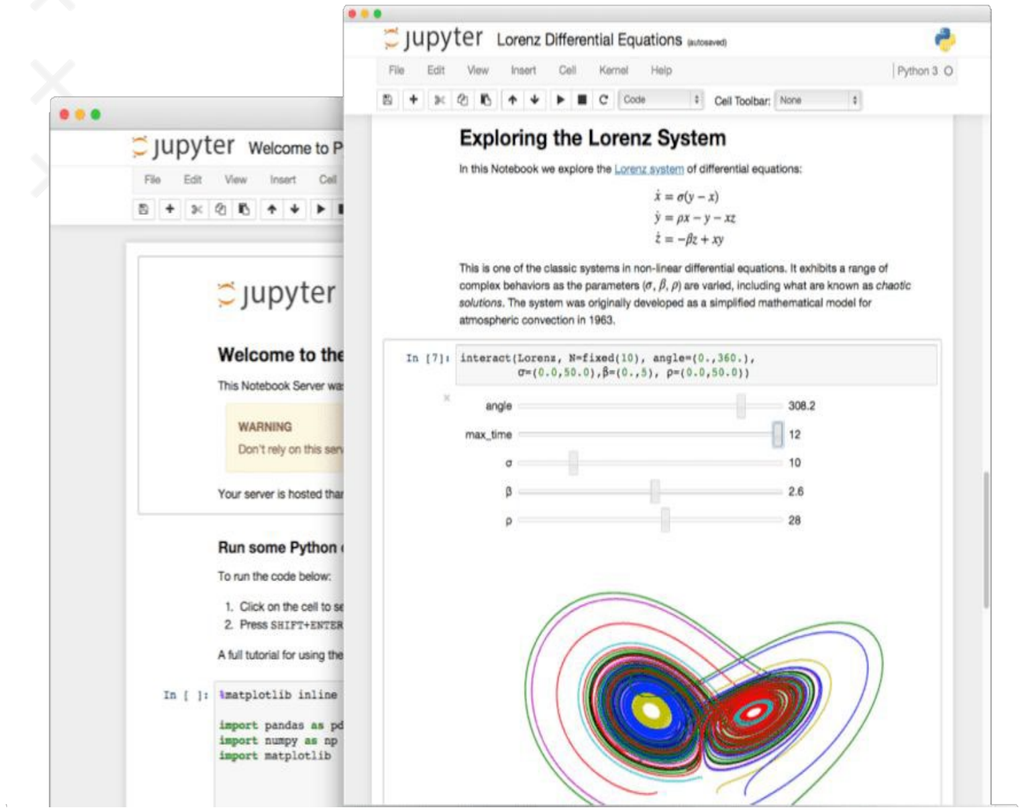
Python Development Environments

This Class use Google Colab



** Cloud + Free **

Jupyter Notebook



- a web-based interactive computing platform
- เรียกใช้งาน library พร้อมทั้งเขียน code และดูผล ได้เลย

</>

Language of choice

Jupyter supports over 40 programming languages, including Python, R, Julia, and Scala.

Share notebooks

Notebooks can be shared with others using email, Dropbox, GitHub and the Jupyter Notebook Viewer.

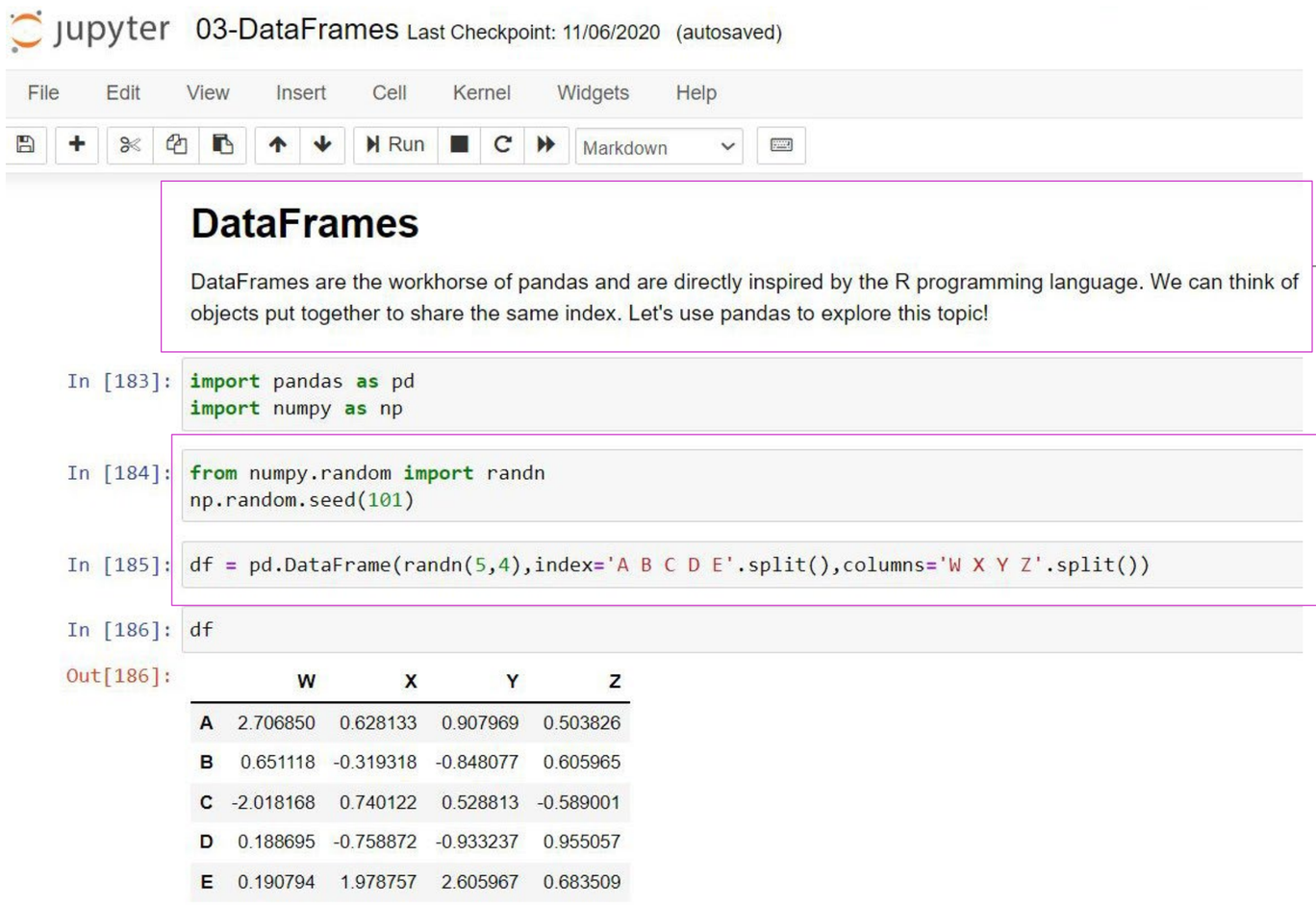
Interactive output

Your code can produce rich, interactive output: HTML, images, videos, LaTeX, and custom MIME types.

Big data integration

Leverage big data tools, such as Apache Spark, from Python, R, and Scala. Explore that same data with pandas, scikit-learn, ggplot2, and TensorFlow.

Jupyter Notebook



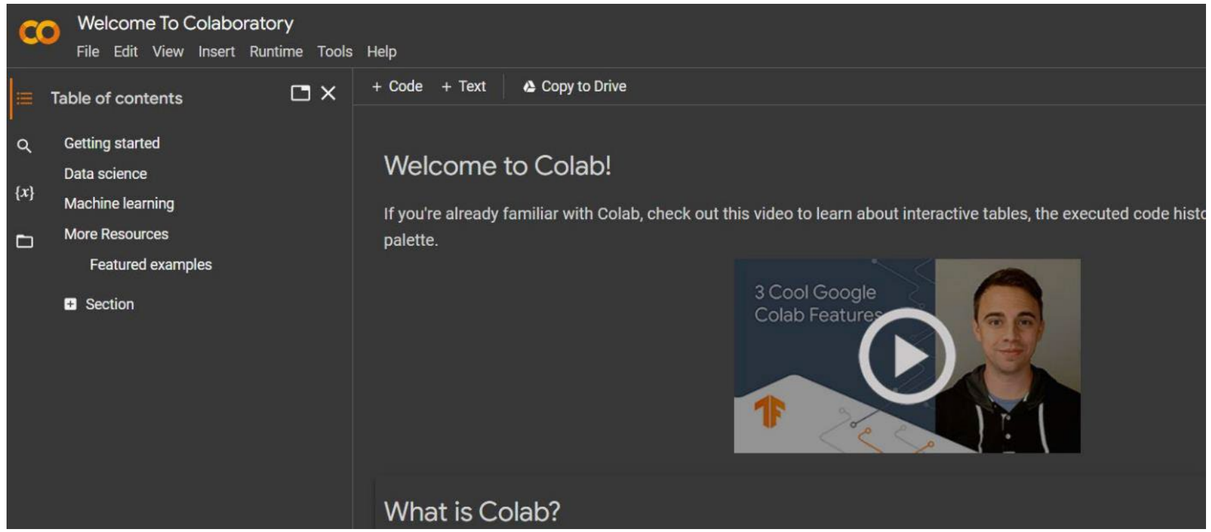
	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

Text Cell

Code Cell

Google Colab

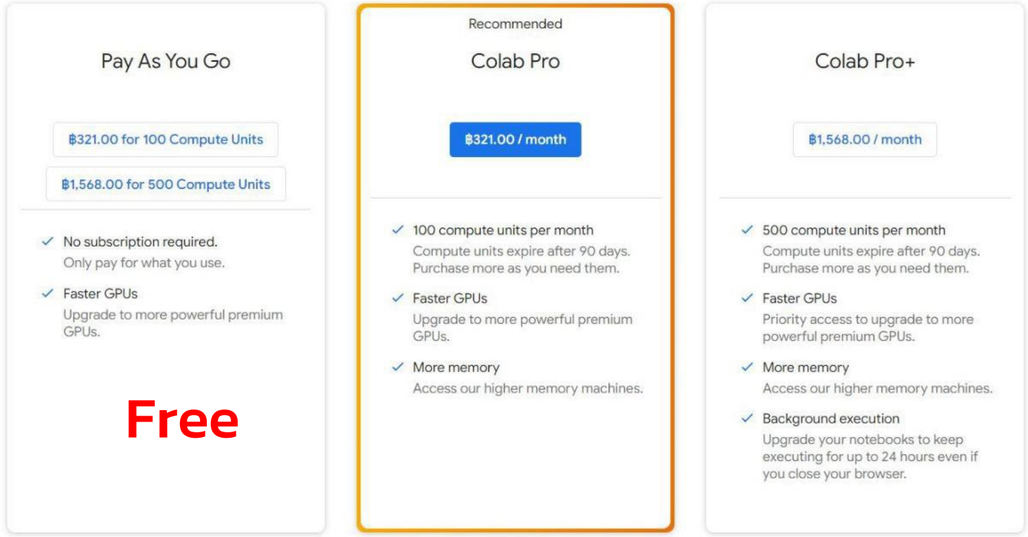
Free Jupyter notebook environment that runs entirely in the cloud.
write and execute Python in your browser



- Advantage
  - Zero configuration required
  - Access to GPUs free of charge
  - Easy sharing
- Disadvantage
  - Internet connection needed
  - Limit memory and runtime (free plan)

What Colab offer you?

- Write and execute code in Python
- Create/Upload/Share notebooks
- Import/Save notebooks from/to Google Drive
- Import/Publish notebooks from GitHub
- Import external datasets e.g. from Kaggle
- Integrate PyTorch, TensorFlow, Keras, OpenCV



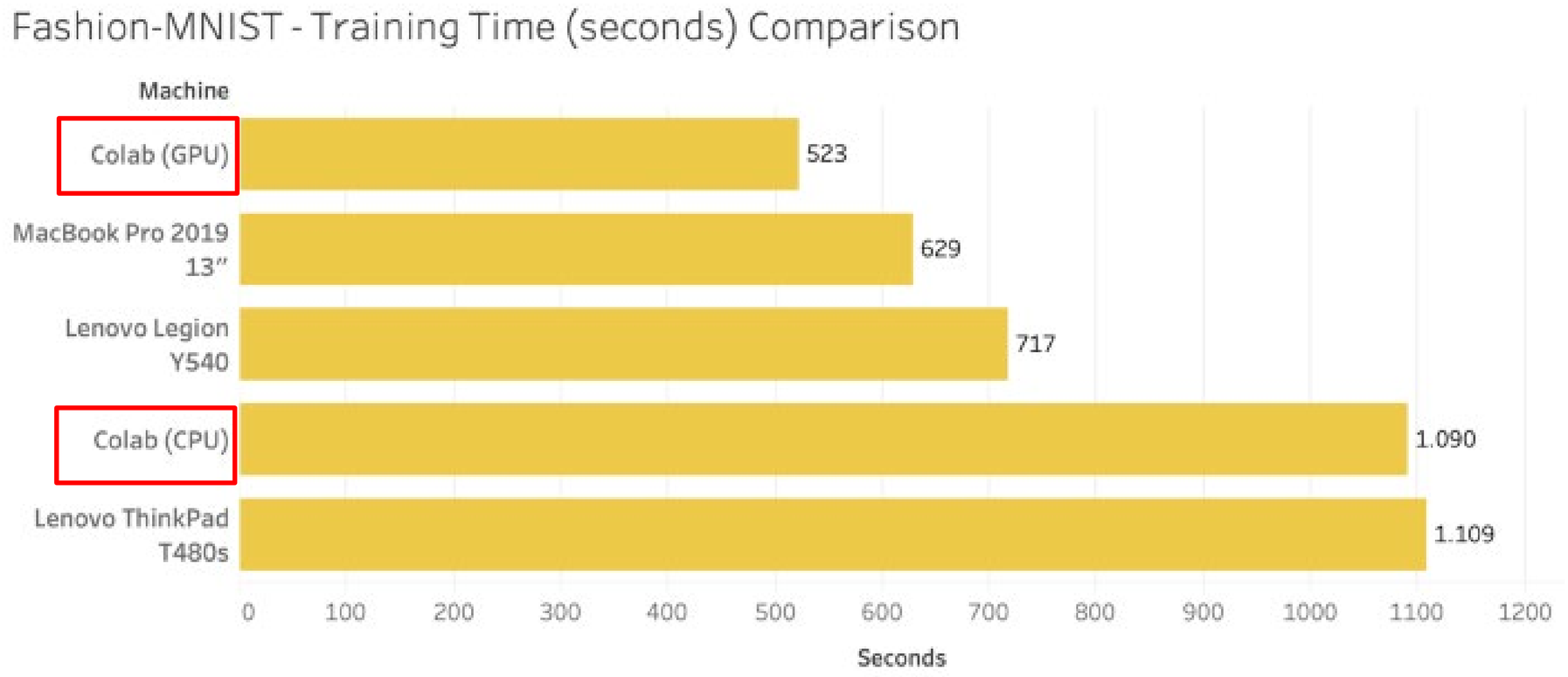
Google Colab

- What Colab offer you?
  - Free Cloud service with free GPU

	Colab Free	Colab Pro	Colab Pro +
Guarantee of resources	Low	High	Even Higher
GPU	K80	K80, T4 and P100	K80, T4 and P100
RAM	16 GB	32 GB	52 GB
Runtime	12 hours	24 hours	24 hours
Background execution	No	No	Yes
Costs	Free	9.99$ per month	49.99$ per month
Target group	Casual user	Regular user	Heavy user

For This Course ,
Free Version is enough !!

Google Colab

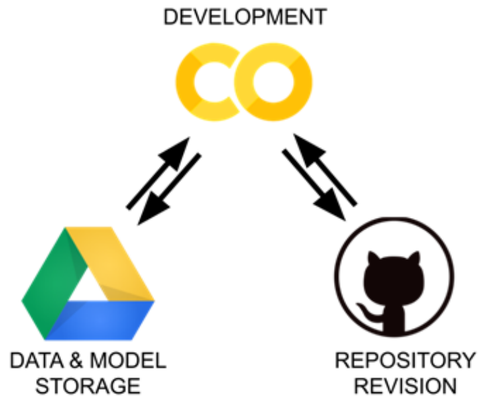


Sum of Seconds for each Machine.

Jupyter Notebook vs Google Colab

Feature	Jupyter Notebook	Google Colab
Cloud-based	No	Yes
File syncing	No	Yes
File sharing	No	Yes
Library install	Yes	No
File view without install	No	Yes

This Class use Google Colab

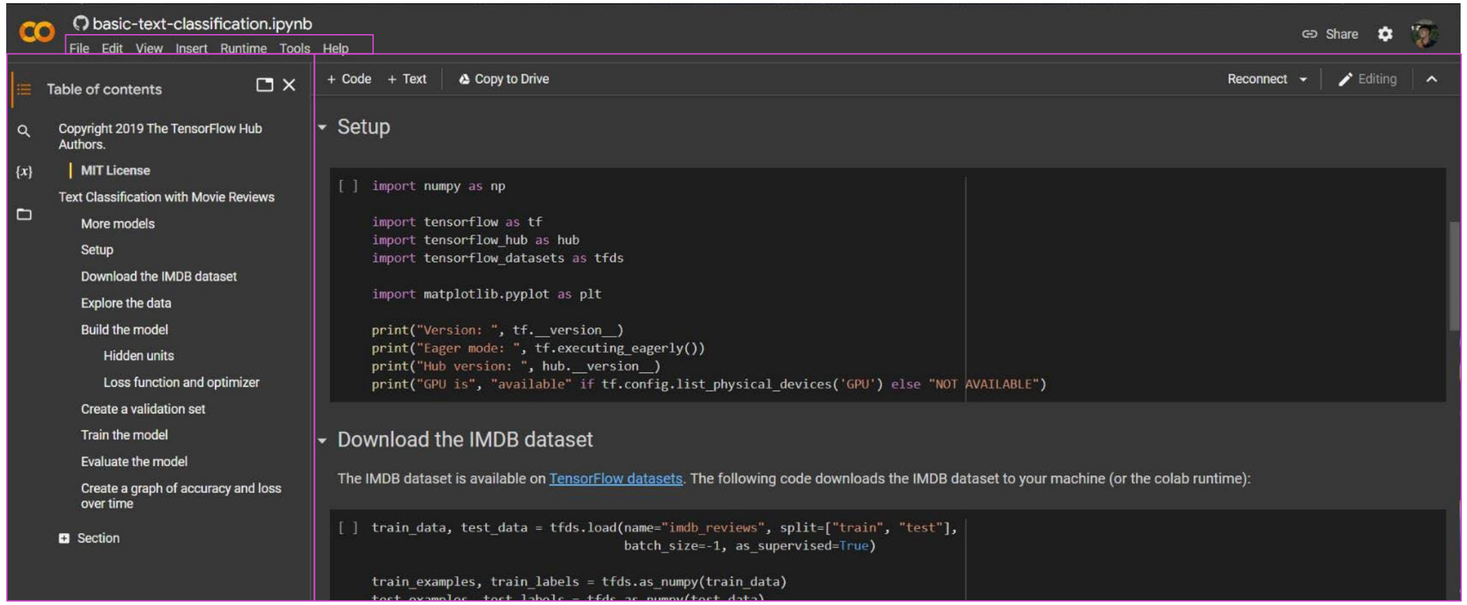


** Cloud + Free **

Google Colab Interface

Menu bar

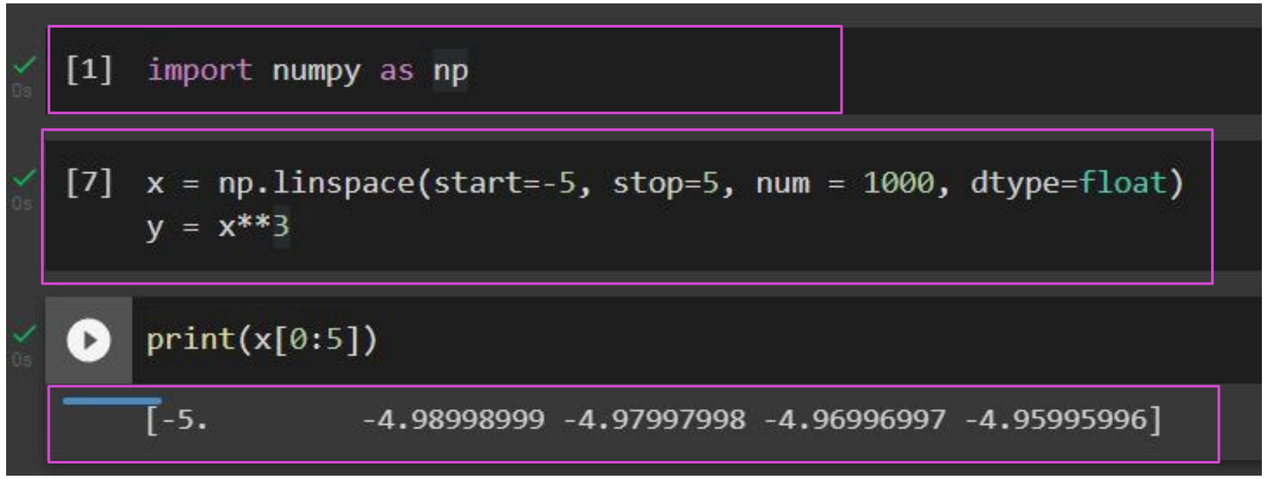
Side Bar



Working Space

Code

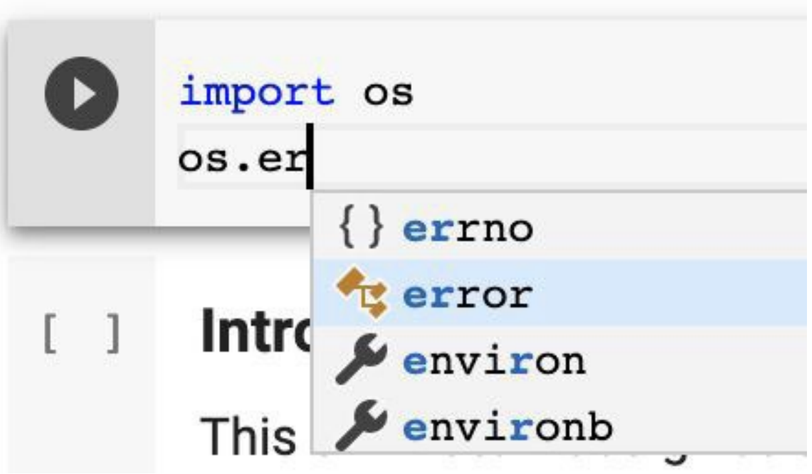
Output



https://colab.research.google.com/

GoogleColab

If you now insert your cursor after `imported library` and press Period(`.`), you will see the list of available completions within the `imported library` module. Completions can be opened again by using **Ctrl+Space**.

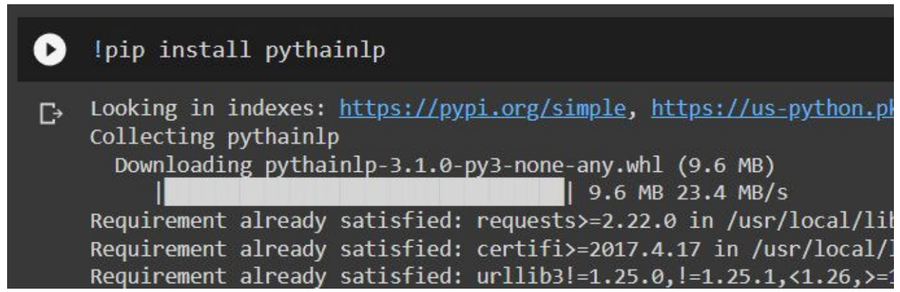


Google Colab

To import a library that's not in Colaboratory by default, you can use

!pip install ...<library name>... or

!apt-get install ...<library name>



Command Scripts

Any command that works at the command-line can be used in code cell by prefixing it with the ! or %

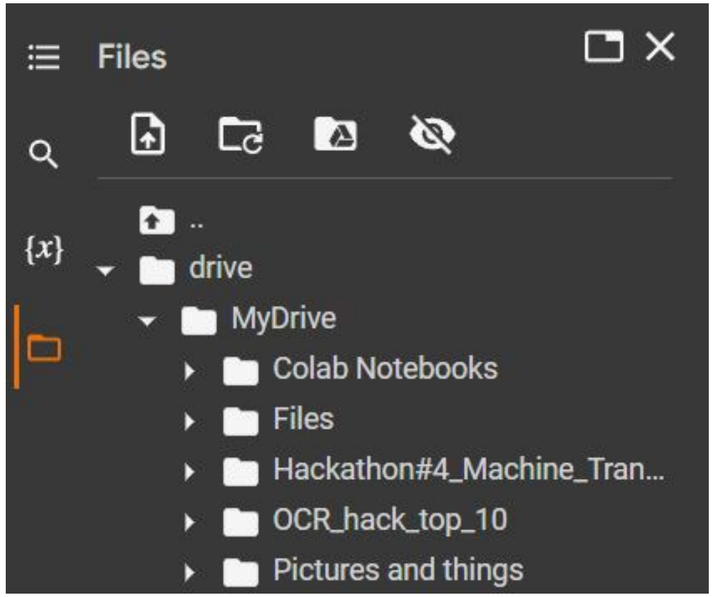
%cd, %cp, %cat, %mkdir, %pwd,

%ls, %rmdir, %dir, %rm, %md

Google Colab

Mounting your Google Drive in the runtime's virtual machine

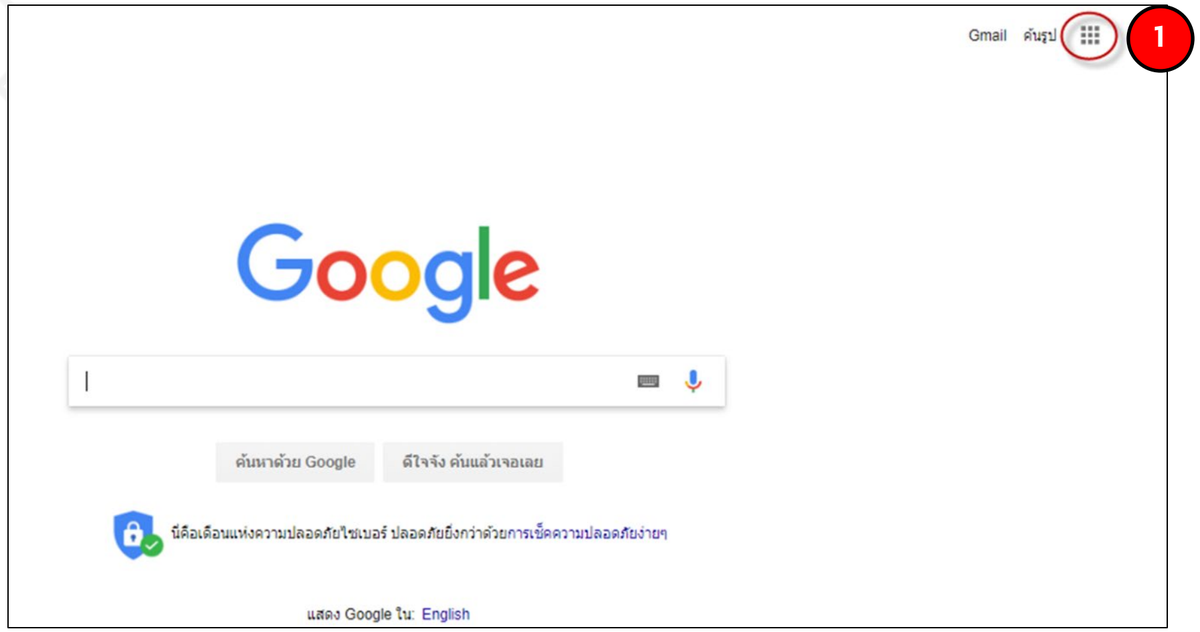
```
[18] from google.colab import drive
     drive.mount('/content/drive')

     Mounted at /content/drive
```

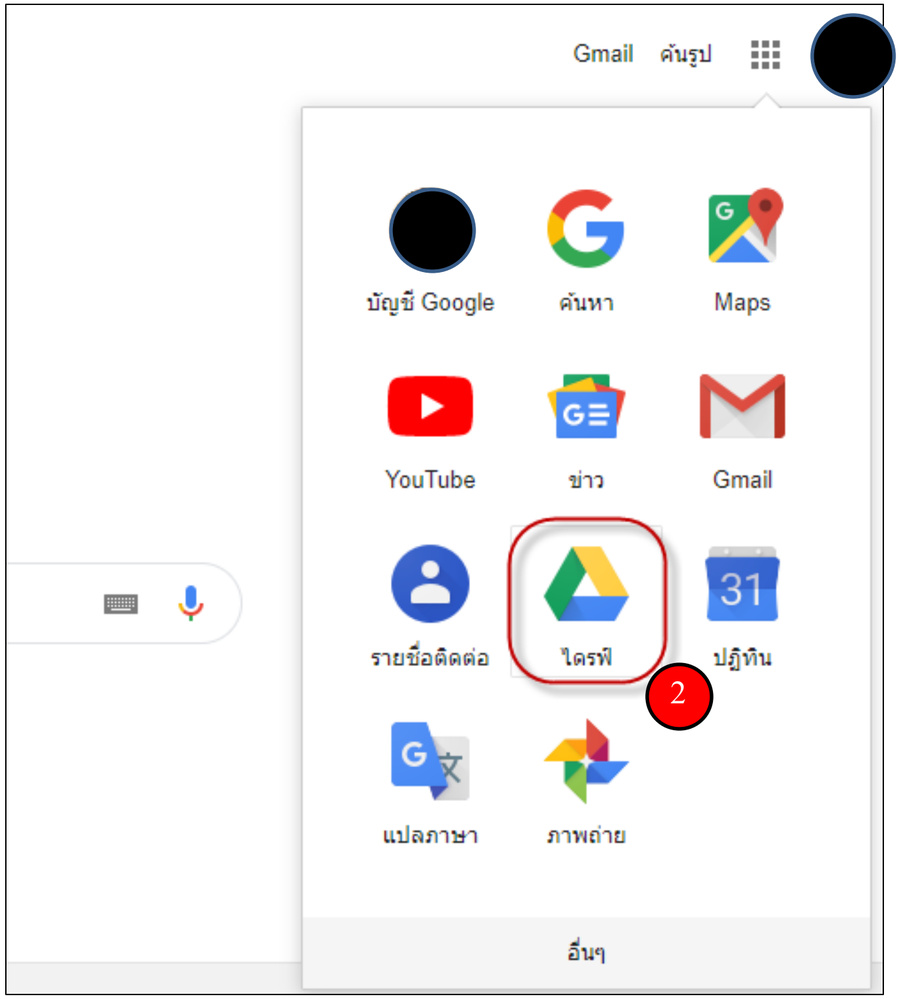


First for use Google Colab

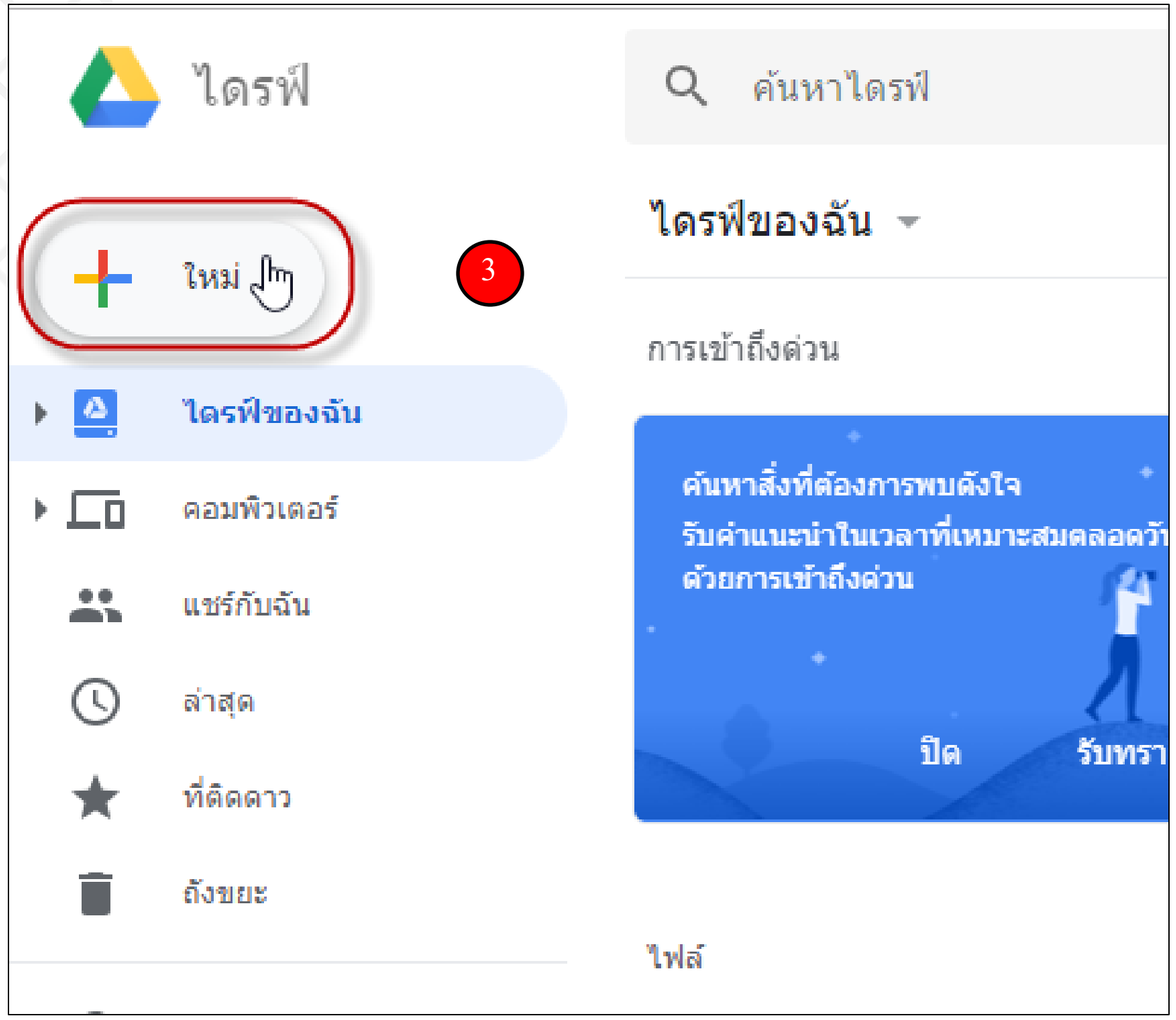
Google Colab

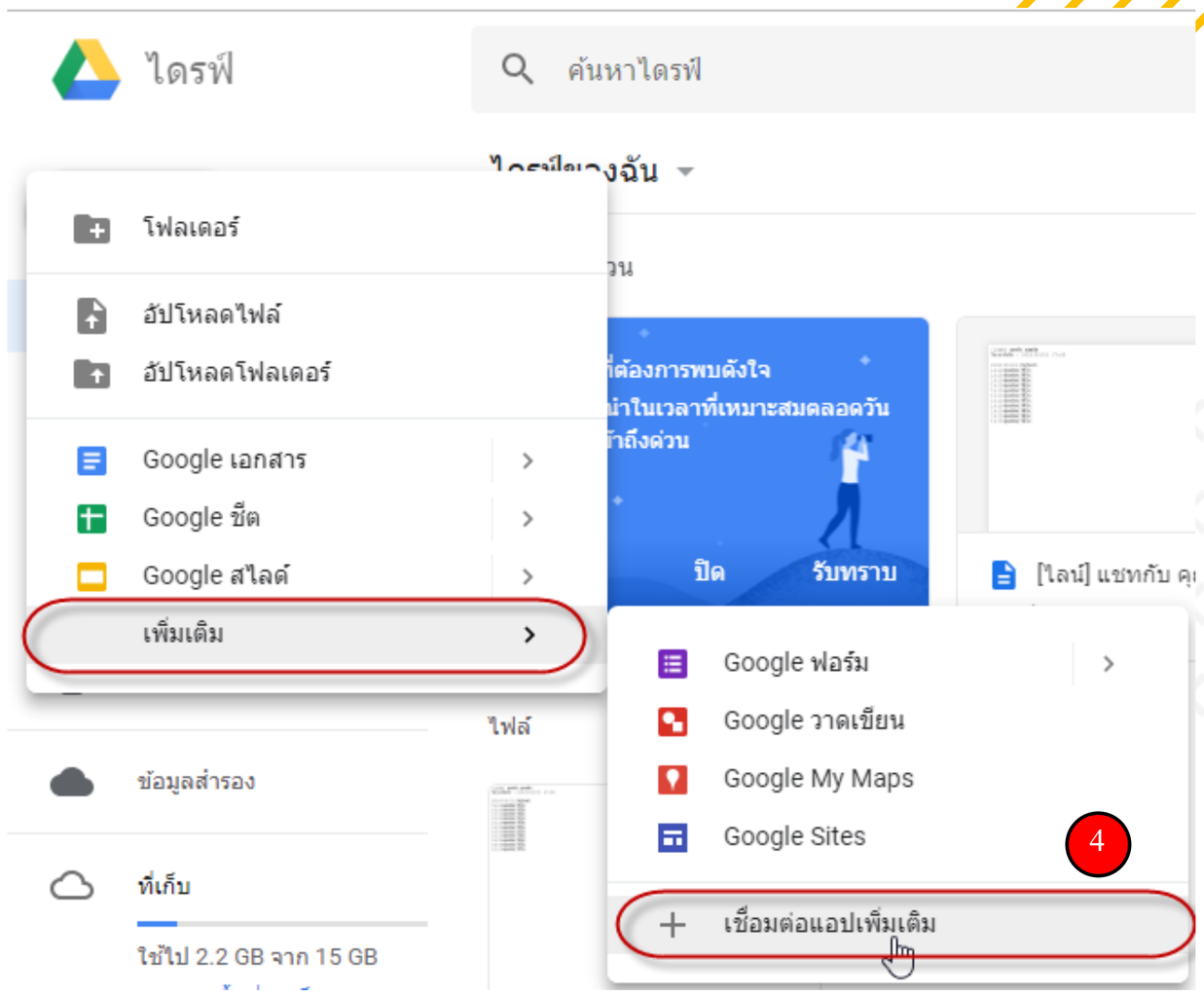
**** Must have Google Account**





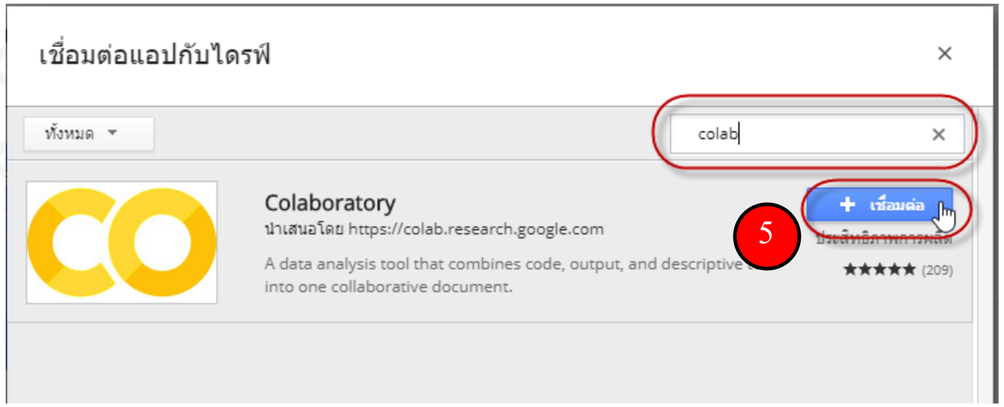
Google Colab

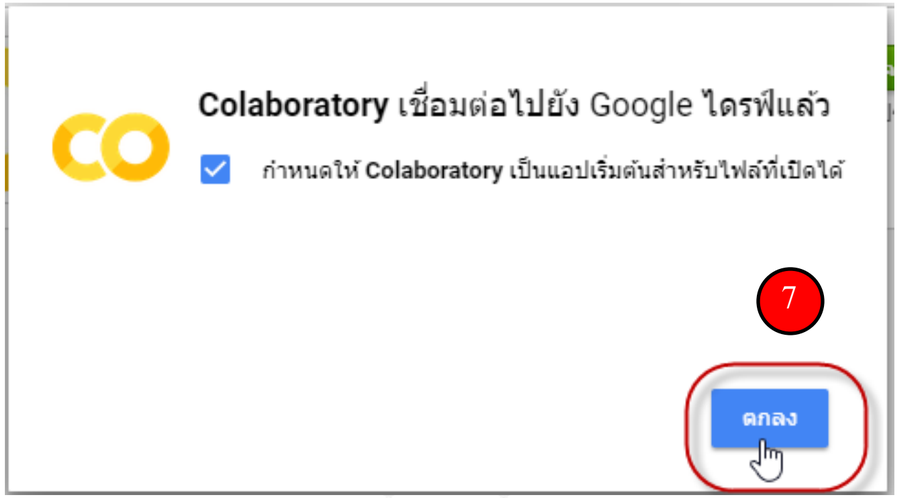




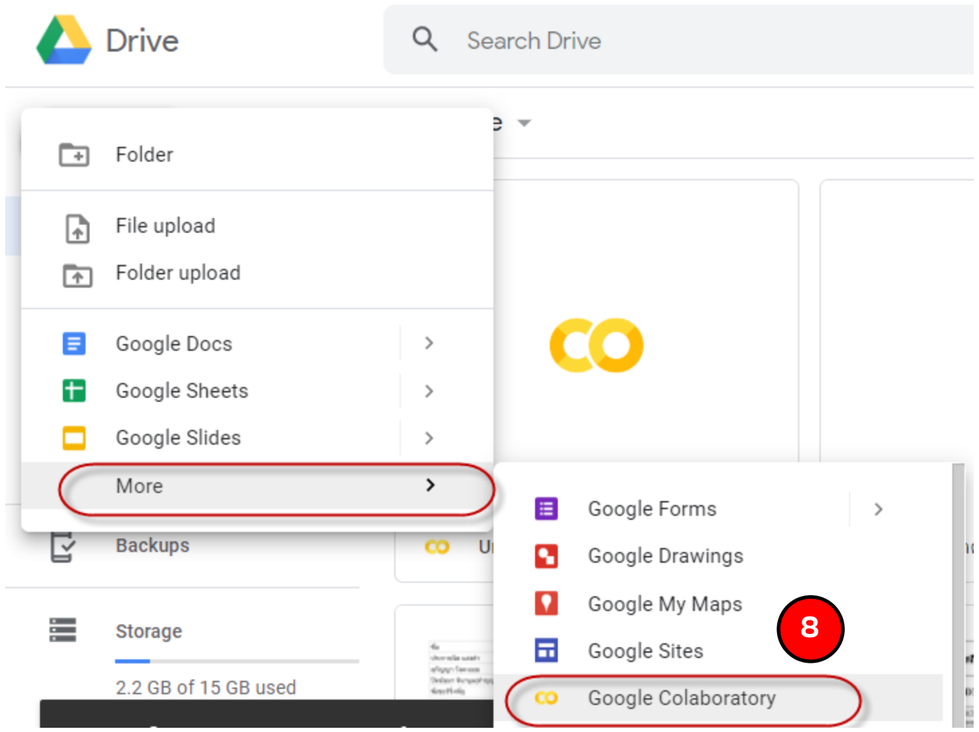
Credit:

Google Colab





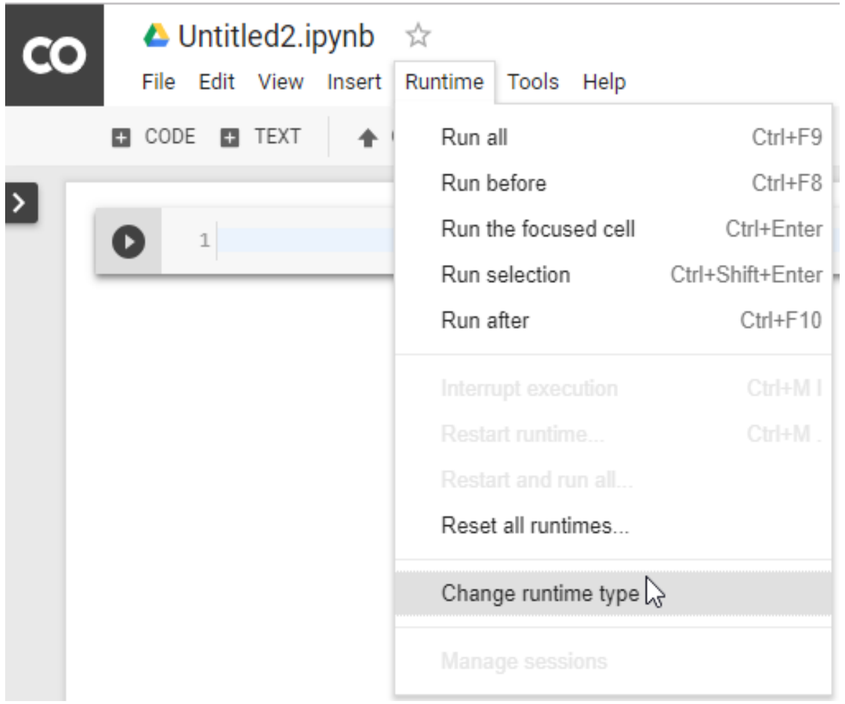
Google Colab

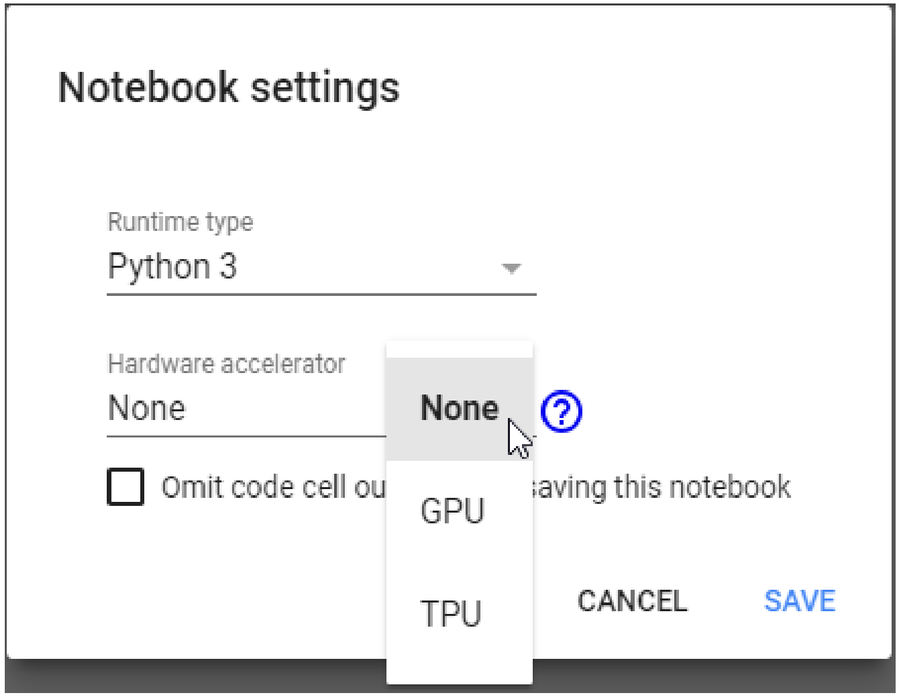


Ready ???

Or Click https://colab.research.google.com/

Google Colab





More : https://lengyi.medium.com/google-colab-tpu-e0ed8af776b3

Google Colab

```
1 from tensorflow.python.client import device_lib
2 device_lib.list_local_devices()
```

Tesla K80
For free!!


```
physical_device_desc: "device: 0, name: Tesla K80, pci bus id: 0000:00:04.0, compute capability: 3.7"]
```

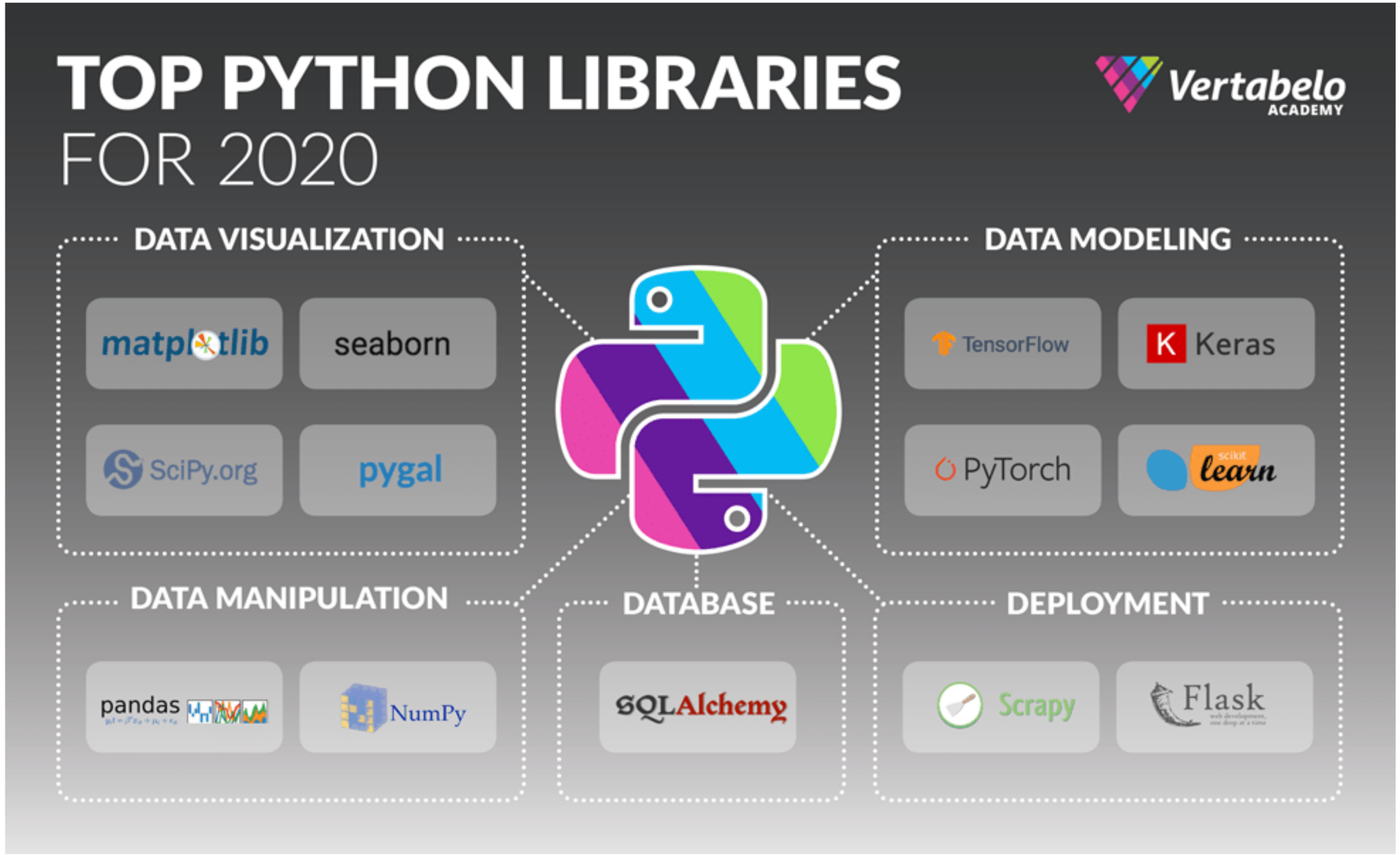
```
import tensorflow as tf
tf.test.gpu_device_name()
```

```
'/device:GPU:0'
```

```
!nvidia-smi
```

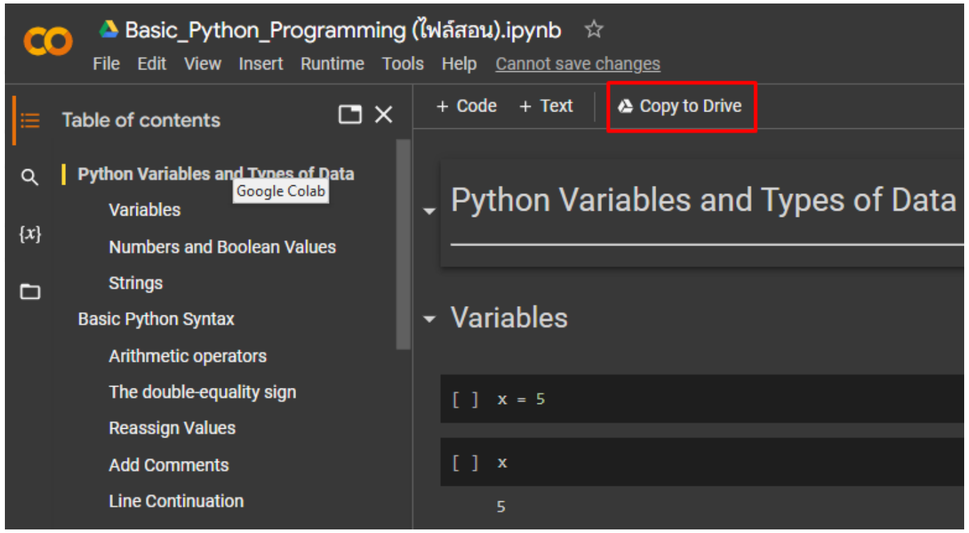
```
Fri Oct 21 02:43:44 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.32.03    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   60C    P0    29W /  70W |    312MiB / 15109MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

Top Python Libraries

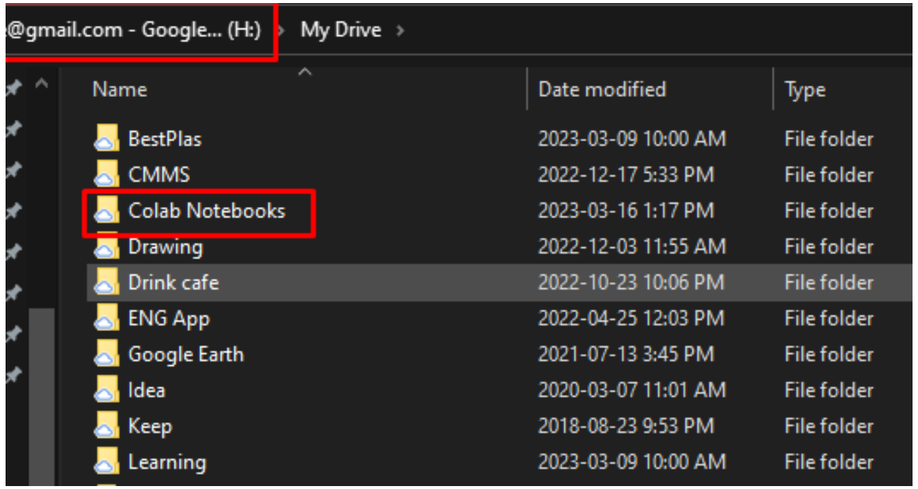


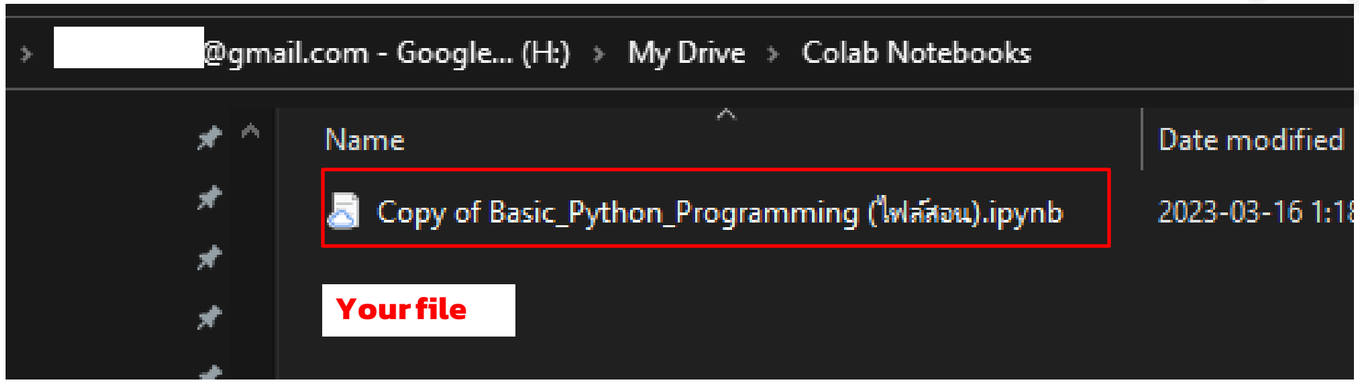
Credit: https://learnpython.com/blog/top-python-libraries-2020/

Copy to Your Drive!!



Your Drive

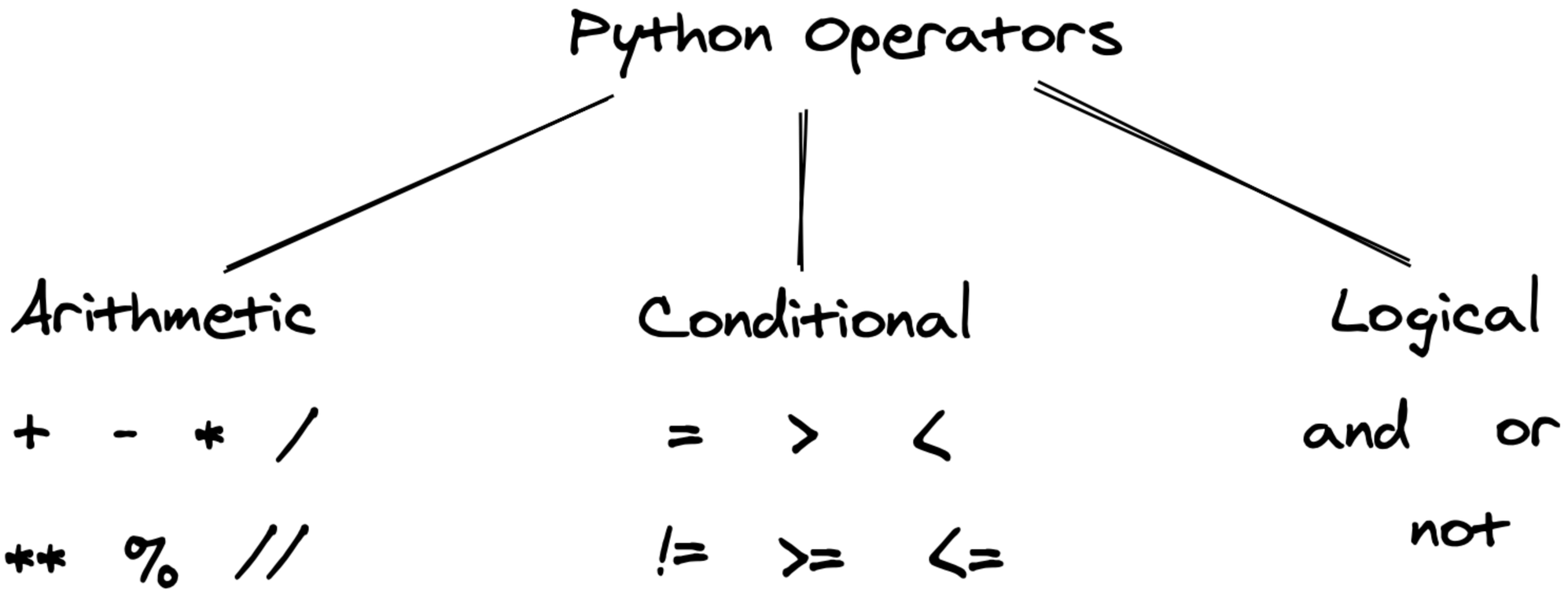




Always Copy to Your Drive , when you copy another code.

1. Python Operators and Variables

Python Operators



Python Operators

Operator	Purpose	Example	Result
+	Addition	2 + 3	5
-	Subtraction	3 - 2	1
*	Multiplication	8 * 12	96
/	Division	100 / 7	14.28..
//	Floor Division	100 // 7	14
%	Modulus/Remainder	100 % 7	2
**	Exponent	5 ** 3	125

Python Operators Precedence Rule – PEMDAS

Firstly, parentheses will be evaluated, then exponentiation and so on.

- P – Parentheses ( )
- E – Exponentiation **
- M – Multiplication *
- D – Division / // %
- A – Addition +
- S – Subtraction -

- Left -> Right

Example : (43+13-9/3*7)

Python Operators

Operator	Purpose	Example	Result
+	Addition	2 + 3	5
-	Subtraction	3 - 2	1
*	Multiplication	8 * 12	96
/	Division	100 / 7	14.28..
//	Floor Division	100 // 7	14
%	Modulus/Remainder	100 % 7	2
**	Exponent	5 ** 3	125

Python Operators Precedence Rule – PEMDAS

Firstly, parentheses will be evaluated, then exponentiation and so on.

- P – Parentheses ( )
- E – Exponentiation **
- M – Multiplication *
- D – Division / // %
- A – Addition +
- S – Subtraction -

- Left -> Right

Example : (43+13-9/3*7)

43 + 13 – 3 * 7

Python Operators

Operator	Purpose	Example	Result
+	Addition	2 + 3	5
-	Subtraction	3 - 2	1
*	Multiplication	8 * 12	96
/	Division	100 / 7	14.28..
//	Floor Division	100 // 7	14
%	Modulus/Remainder	100 % 7	2
**	Exponent	5 ** 3	125

Python Operators Precedence Rule – PEMDAS

Firstly, parentheses will be evaluated, then exponentiation and so on.

- P – Parentheses ( )
- E – Exponentiation **
- M – Multiplication *
- D – Division / // %
- A – Addition +
- S – Subtraction -

- Left -> Right

Example : (43+13-9/3*7)

43 + 13 – 3 * 7

43 + 13 – 21

Python Operators

Operator	Purpose	Example	Result
+	Addition	2 + 3	5
-	Subtraction	3 - 2	1
*	Multiplication	8 * 12	96
/	Division	100 / 7	14.28..
//	Floor Division	100 // 7	14
%	Modulus/Remainder	100 % 7	2
**	Exponent	5 ** 3	125

Python Operators Precedence Rule – PEMDAS

Firstly, parentheses will be evaluated, then exponentiation and so on.

- P – Parentheses ( )
- E – Exponentiation **
- M – Multiplication *
- D – Division / // %
- A – Addition +
- S – Subtraction -

- Left -> Right

Example : (43+13-9/3*7)

43 + 13 – 3 * 7

43 + 13 – 21

35

Logical operators

Operator	Description
`==`	Check if operands are equal
`!=`	Check if operands are not equal
`>`	Check if left operand is greater than right operand
`<`	Check if left operand is less than right operand
`>=`	Check if left operand is greater than or equal to right operand
`<=`	Check if left operand is less than or equal to right operand

AND	True	False
True	True	False
False	False	False

OR	True	False
True	True	True
False	True	False

Adding text styles using Markdown & Comment

Adding text styles using Markdown

Adding explanations using text cells (like this one) is a great way to make your notebook informative for other readers. It is also useful if you need to refer back to it in the future. You can double click on a text cell within Jupyter to edit it. In the edit mode, you'll notice that the text looks slightly different (for instance, the heading has a `##` prefix. This text is written using Markdown, a simple way to add styles to your text. Execute this cell to see the output without the special characters. You can switch back and forth between the source and the output to apply a specific style.

For instance, you can use one or more `#` characters at the start of a line to create headers of different sizes:

Header 1

Header 2

Header 3

Header 4

To create a bulleted or numbered list, simply start a line with `*` or `1.`.

A bulleted list:

- Item 1
- Item 2
- Item 3

A numbered list:

1. Apple
2. Banana
3. Pineapple

Inline and single-line comments start with `#`, whereas multi-line comments begin and end with three quotes, i.e. `"""`. Here are some examples of code comments:

```
my_favorite_number = 1 # an inline comment
```

```
# This comment gets its own line
my_least_favorite_number = 3
```

```
"""This is a multi-line comment.
Write as little or as much as you'd like.

Comments are really helpful for people reading
your code, but try to keep them short & to-the-point.

Also, if you use good variable names, then your code is
often self explanatory, and you may not even need comments!
"""
a_neutral_number = 5
```

Rules of Naming a Variable in Python

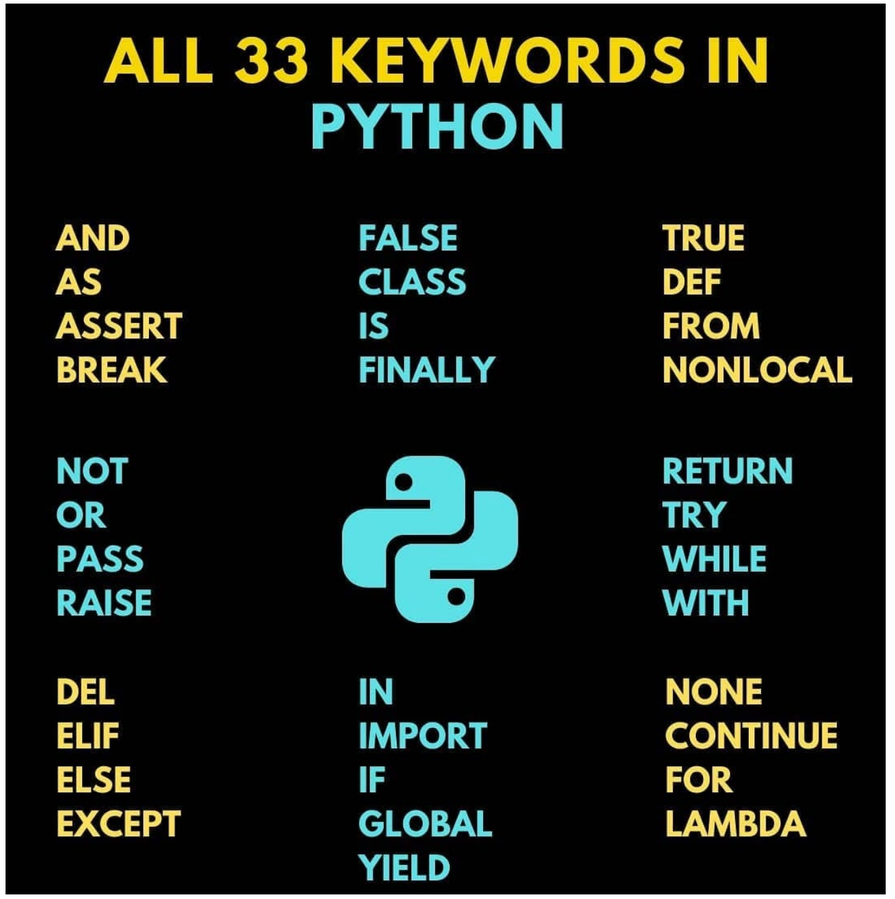
1. It must **start** with **a letter** or **the underscore (_)**.

2. It should not start with **a number.**

3. It should only contain alpha-numeric characters and underscores **(A-z, 0-9, and _).**
   It should not have any other characters or spaces.

4. The names are case-sensitive
   (i.e., DataMommie and datamommie are treated as different variables).

5. It should not be a keyword.



Variables

One of the main concepts in programming is **variables**. They are your best friends. You will deal with them all the time. You will use them to store information. They will represent your data input.

```
In [1]: x = 5

In [2]: x
Out[2]: 5
```

Let's say you want to have a variable *x* that is equal to the value of 5 and then ask the computer to tell you the value of that variable. Type *x* equals 5. Write *x* and then execute the cell. And here's the result: 5.

```
In [3]: y = 8

In [5]: print (y)
        8
```

An alternative way to execute the instruction that will provide the value we assigned to y would be to use the *print()* function.
If we say "print(y)", the machine will simply execute this command and provide the value of *y as a result*.

Break 15 mins

2. Functions

4. Functions

header - specifies name of function and its argument(s)

```
def add_three(input_var):
    output_var = input_var + 3
    return output_var
```

body - specifies the work that the function does

keyword that tells Python we are about to **def**ine a function

name you choose for your function

function's argument (name of variable the function will use)

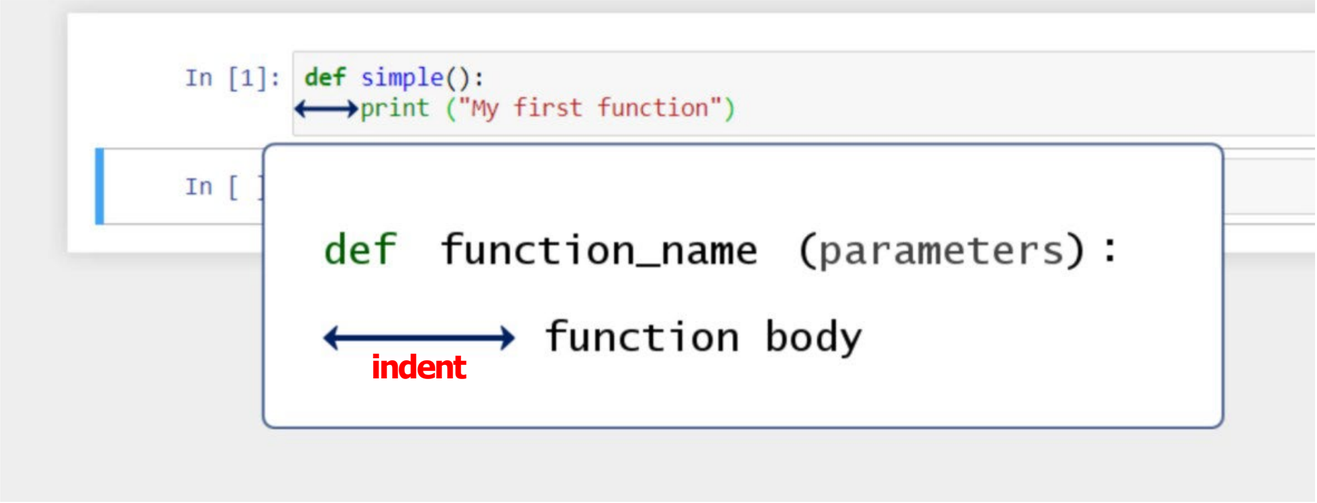
```
def add_three(input_var):
    output_var = input_var + 3
    return output_var
```

keyword that tells Python we are about to exit a function

value that is **return**ed when the function is exited

Defining a Function in Python

Python's **functions** are an invaluable tool for programmers.

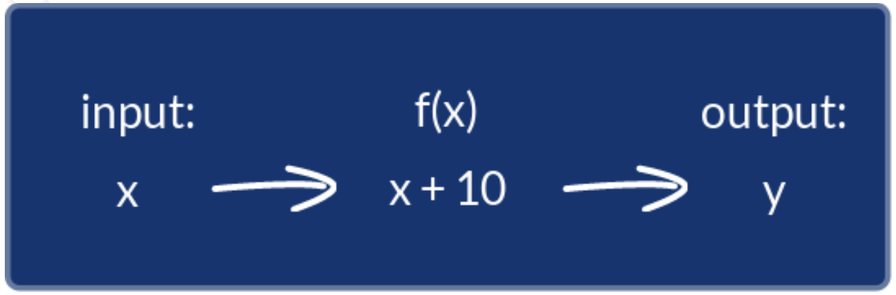


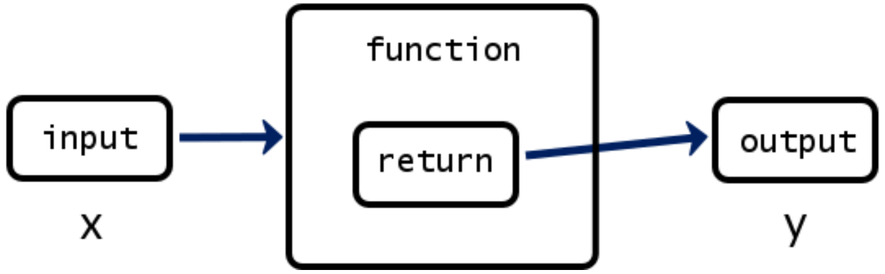
To tell the computer you are about to create a function, just write **def** at the beginning of the line. Def is neither a command nor a function. It is a **keyword**. To indicate this, Jupyter will automatically change its font color to green. Then, you can type the **name of the function** you will use. Then, you must add a pair of **parentheses**. Technically, within these parentheses, you could place the **parameters** of the function if it requires you to have any. It is no problem to have a function with zero parameters.

To proceed, don't miss to put a **colon** after the name of the function.

Since it is inconvenient to continue on the same line when the function becomes longer, it is much better to build the habit of laying the instructions on a new line, with an **indent** again. Good legibility counts for a good style of coding!

Creating a Function with a Parameter





In programming, **return** regards the value of y; it just says to the machine "after the operations executed by the function f, return to me the value of y". "Return" plays a connection between the second and the third step of the process. In other words, *a function can take an input of one or more variables and return a **single** output composed of one or more values*. This is why "return" can be used only once in a function.

People often confuse print and return, and the type of situations when we can apply them.

print vs. return

print	vs.	return
does not affect the calculation of the output		does not visualize the output
		it specifies what a certain function is supposed to give back

print() takes a statement or, better, an object, and provides its printed representation in the output cell. It just makes a certain statement visible to the programmer. Otherwise, **print()** does not affect the calculation of the output.

Differently, **return** does not visualize the output. It specifies what a certain function is supposed to give back.

It’s important you understand what each of the two keywords does. This will help you a great deal when working with functions.

Using a Function in Another Function

```
In [ ]: def wage(w_hours):
            return w_hours * 25

        def with_bonus(w_hours):
            return wage(w_hours) + 50
```

It isn't a secret we can have a function within the function.

In *with_bonus(w_hours)*, you can return directly the wage with working hours as an output, which would be the value obtained after the wage function has been run, plus 50.

Creating Functions Containing a Few Arguments

You can work with more than one parameter in a function. The way this is done in Python is by listing all the arguments within the parentheses, separated by a comma.

```
In [1]: def subtract_bc(a,b,c):
            result = a - b*c
            print ('Parameter a equals', a)
            print ('Parameter b equals', b)
            print ('Parameter c equals', c)
            return result
```

```
def function_name (parameter #1, parameter #2, ... ):
```

Creating Functions Containing a Few Arguments

```
In [2]: subtract_bc(10,3,2)

        Parameter a equals 10
        Parameter b equals 3
        Parameter c equals 2

Out[2]: 4
```

```
In [ ]:
```

You can call the function for, say, 10, 3, and 2. You will get 4.
Just be careful with the *order* in which you state the values. In this case, we assigned 10 to the variable a, 3 to b, and 2 to c.

```
In [3]: subtract_bc(b=3,a=10,c=2)

        Parameter a equals 10
        Parameter b equals 3
        Parameter c equals 2

Out[3]: 4
```

Otherwise, the order won't matter if you specify the names of the variables within the parentheses.

Notable Built-In Functions in Python

When you install Python on your computer, you are also installing some of its **built-in functions**. This means you won't need to type their code every time you use them – these functions are already on your computer and can be applied directly.

Function	Description
type()	obtains the type of variable you use as an argument
int()	transforms its argument in an *integer* data type
float()	transforms its argument in a *float* data type
str()	transforms its argument in a *string* data type
max()	Returns the highest value from a sequence of numbers
min()	Returns the lowest value from a sequence of numbers
abs()	Allows you to obtain the absolute value of its argument
sum()	Calculates the sum of all the elements in a list designated as an argument

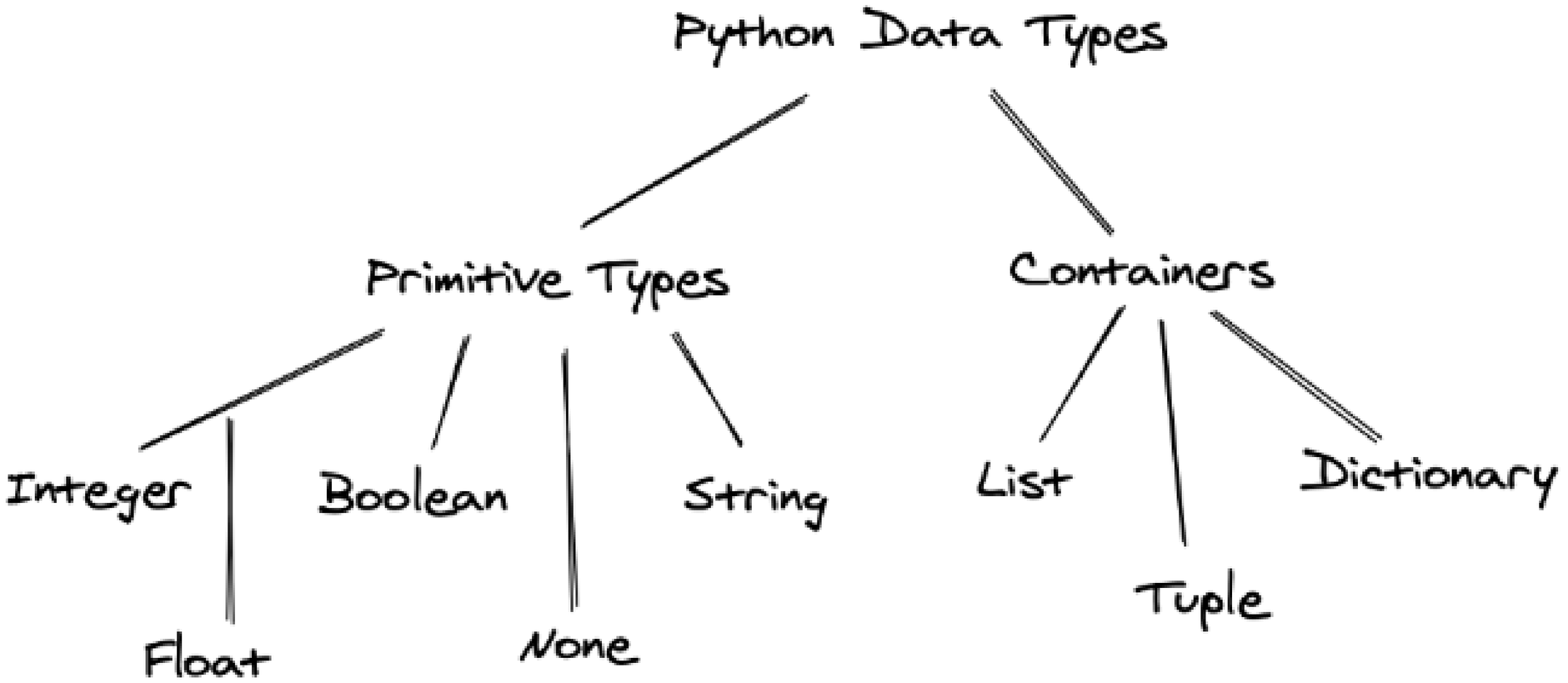
Notable Built-In Functions in Python

Function	Description
round(x,y)	returns the float of its argument (x), rounded to a specified number of digits (y) after the decimal point
pow(x,y)	returns x to the power of y
len()	returns the number of elements in an object

Break - Lunch

3. Data Types

Types of Variables



Numbers and Boolean Values

When programming, not only in Python, if you say that a variable has a numeric value, you are being ambiguous.

The reason is that numbers can be **integers** or floating points, also called **floats**, for instance.

Term	Definition
Integer	Positive or negative whole numbers without a decimal point *Example: **5, 10, -3, -15***
Floating point (float)	Real numbers. Hence, they have a decimal point *Example: **4.75, -5.50, 11.0***
Boolean value	a True or False value, corresponding to the machine's logic of understanding 1s and 0s, on or off, right or wrong, true or false. *Example: **True, False***

Strings

Strings are text values composed of a sequence of characters.

```
In [2]: 'George'
Out[2]: 'George'

In [3]: "George"
Out[3]: 'George'

In [4]: print ('George')
        George

In [5]: print ("George")
        George
```

Type **single** or **double** quotation marks around the name George. Python displays ‘George’ if you don’t use the *print()* function. Should you use *print()*, the output will be shown with no quotes – you’ll be able to see plain text.

Strings

Strings are text values composed of a sequence of characters.

```
In [11]: "I'm fine"
Out[11]: "I'm fine"

In [12]: 'I\'m fine'
Out[12]: "I'm fine"

In [13]: 'Press "Enter"'
Out[13]: 'Press "Enter"'
```

To type “I’m fine” you’ll need the apostrophe in the English syntax, not for the Pythonic one.
In such situations, you can distinguish between the two symbols – put the text within double quotes and leave the apostrophe, which technically coincides with the single quote between I and M.

An alternative way to do that would be to leave the quotes on the sides and place a back slash before the apostrophe within the phrase.

This backslash is called an **escape character**, as it changes the interpretation of characters immediately after it.

To state “press “Enter””, the outer symbols must differ from the inner ones. Put single quotes on the sides.

Strings

Strings are text values composed of a sequence of characters.

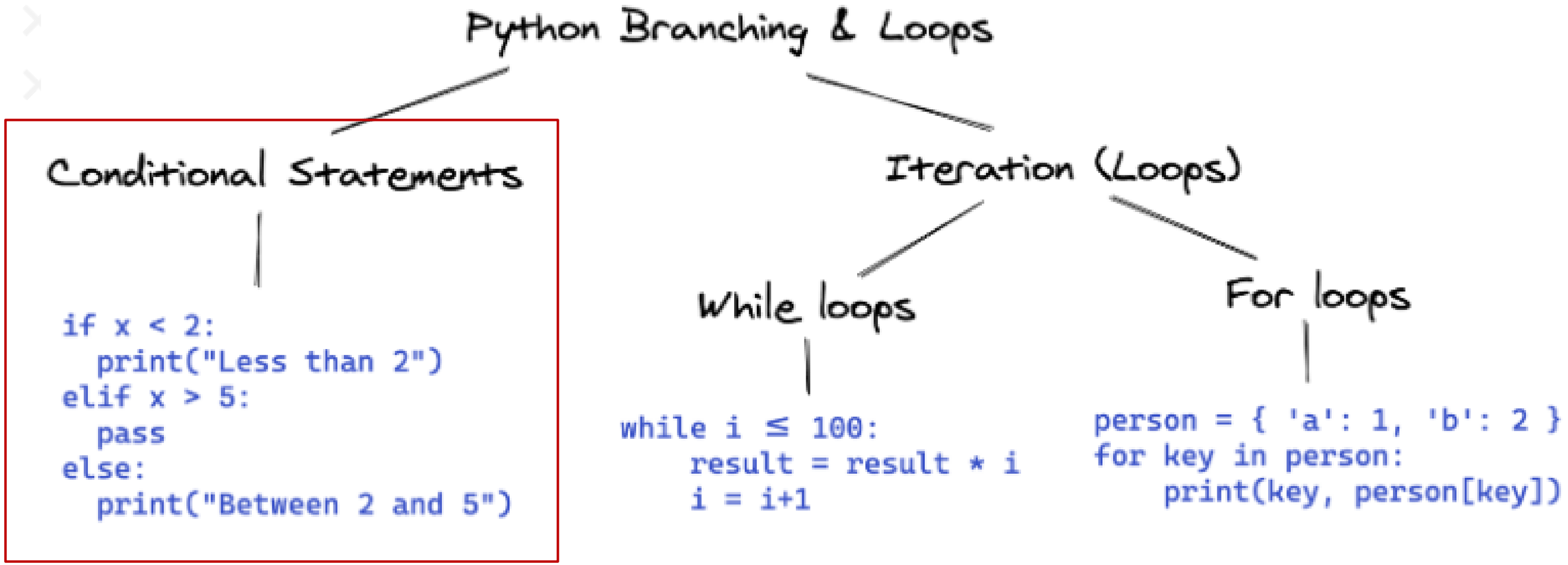
```
In [17]: print ('Red ' + 'car')

         Red car

In [18]: print ('Red', 'car')

         Red car
```

Say you wish to print "Red car" on the same line. "Add" one of the strings to the other by typing in a plus sign between the two. Put a blank space before the second apostrophe of the first word. (In [17])

Type *print('Red'*, and then put a comma, which is called a **trailing comma**, and Python will print the next word, 'car', on the same line, separating the two words with a blank space. (In [18])

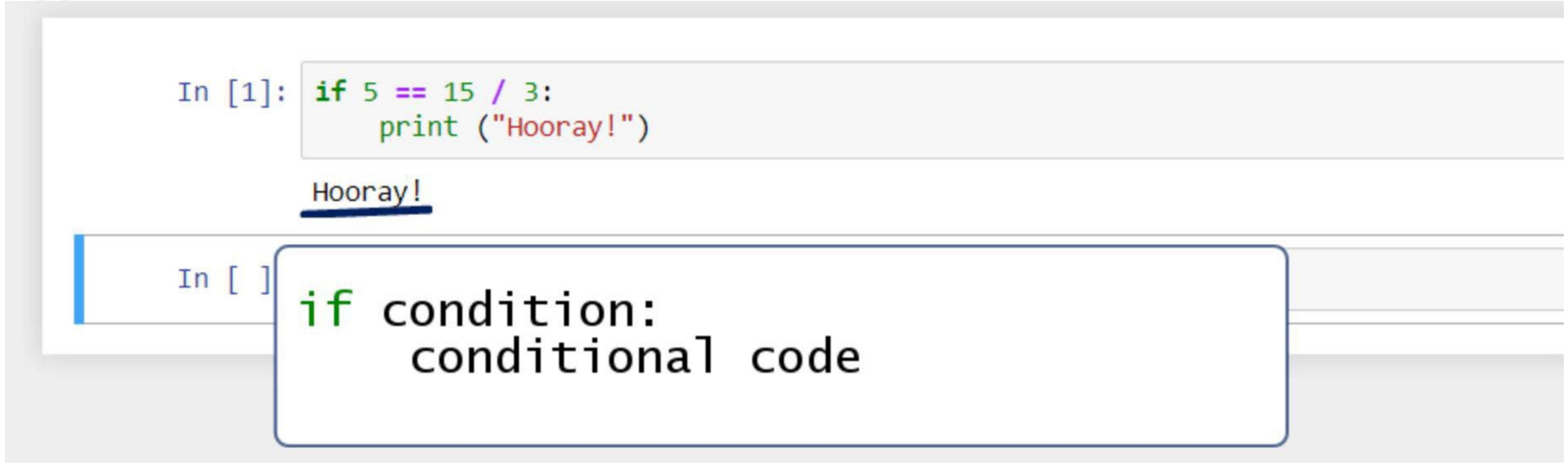
4. Conditional Statements

Conditional Statements

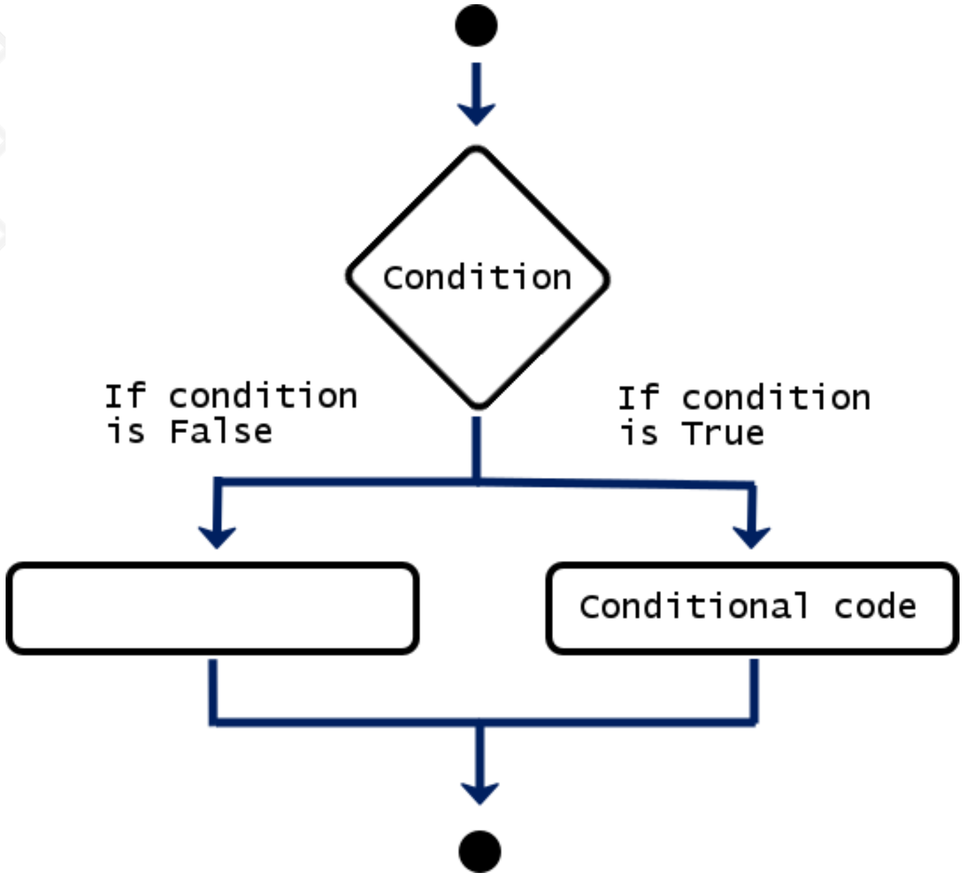


Introduction to the IF statement

A prominent example of **conditional statements** in Python is the **"If" statement**. What you can use it for is intuitive, but it is very important to learn the syntax.



Introduction to the IF statement



The graph could help you imagine the process of the conditionals. Before it displays the outcome of the operation, the machine follows these logical steps. If the conditional code is not to be executed because the if-condition is not true, the program will directly lead you to some other output or, as it is in this case, to nothing. After any of the two situations, the machine will go to the next black point and will progress from there on.

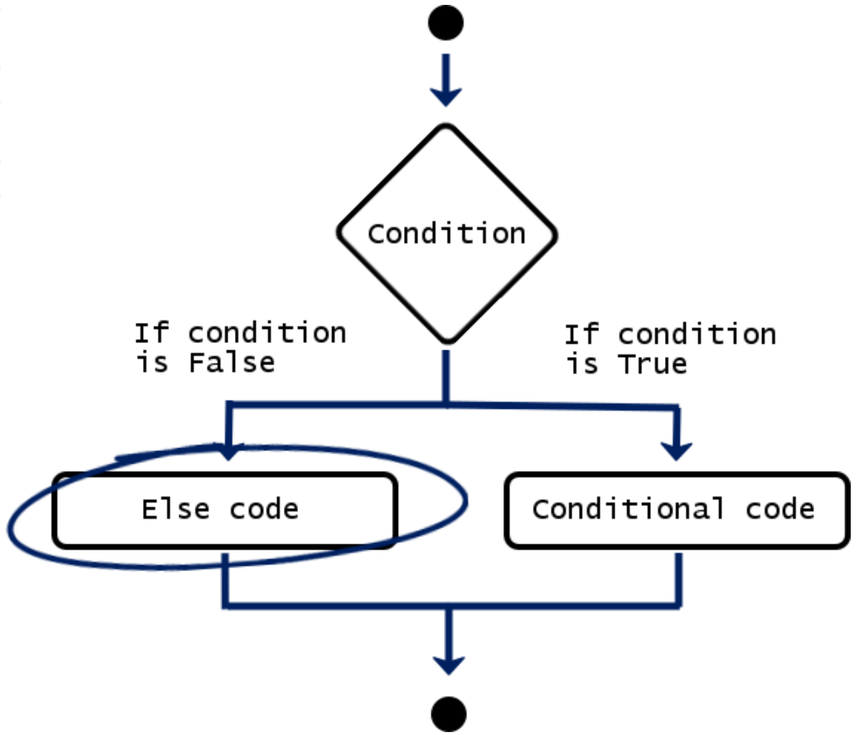
Add an ELSE statement

"Else" will tell the computer to execute the successive command in all other cases.

```
In [1]: x = 1

        if x > 3:
            print ("Case 1")
        else:
            print ("Case 2")

        Case 2
```

```
if condition:
    conditional code
else:
    else code
```

Add an ELSE statement



Instead of leading to no output, if the condition is false, we will get to an *else code*. Regardless whether the initial condition is satisfied, we will get to the end point, so the computer has concluded the entire operation and is ready to execute a new one.

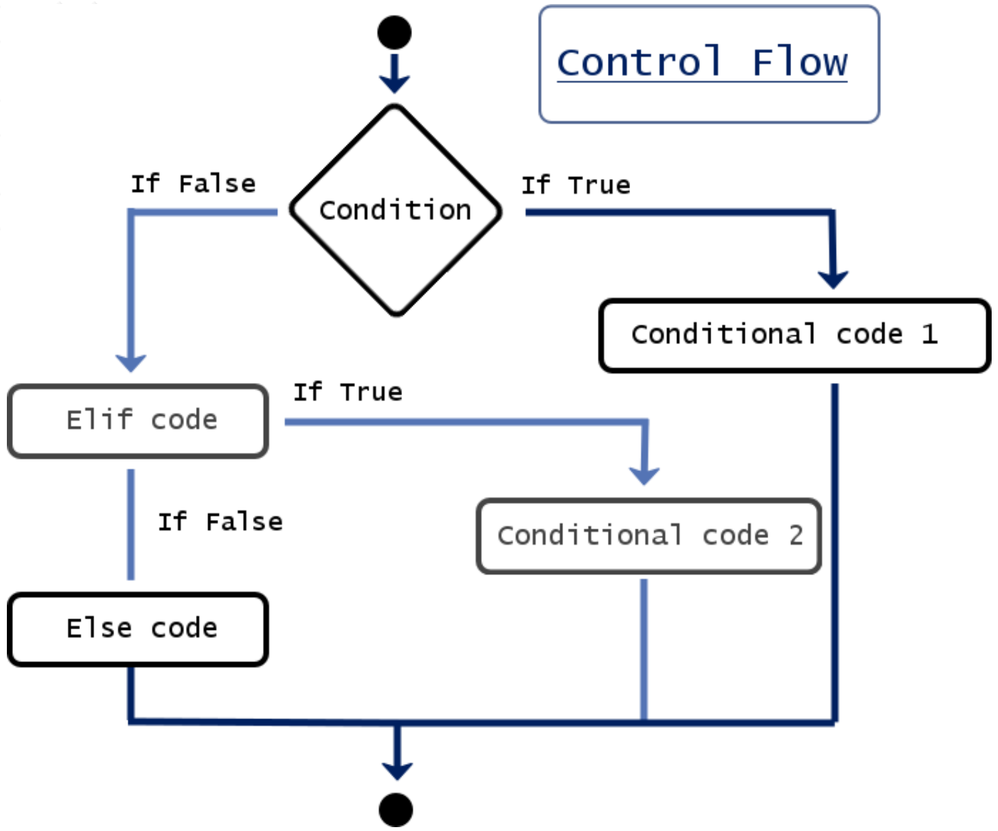
Else if, for Brief – ELIF

If y is not greater than 5, the computer will think: “else if y is less than 5”, written “**elif** y is less than 5”, then I will print out “Less”.

```
In [1]: def compare_to_five(y):
            if y > 5:
                return "Greater"
            elif y < 5:
                return "Less"
            else:
                return "Equal"
```

Know that you can add as many elif statements as you need.

Else if, for Brief – ELIF

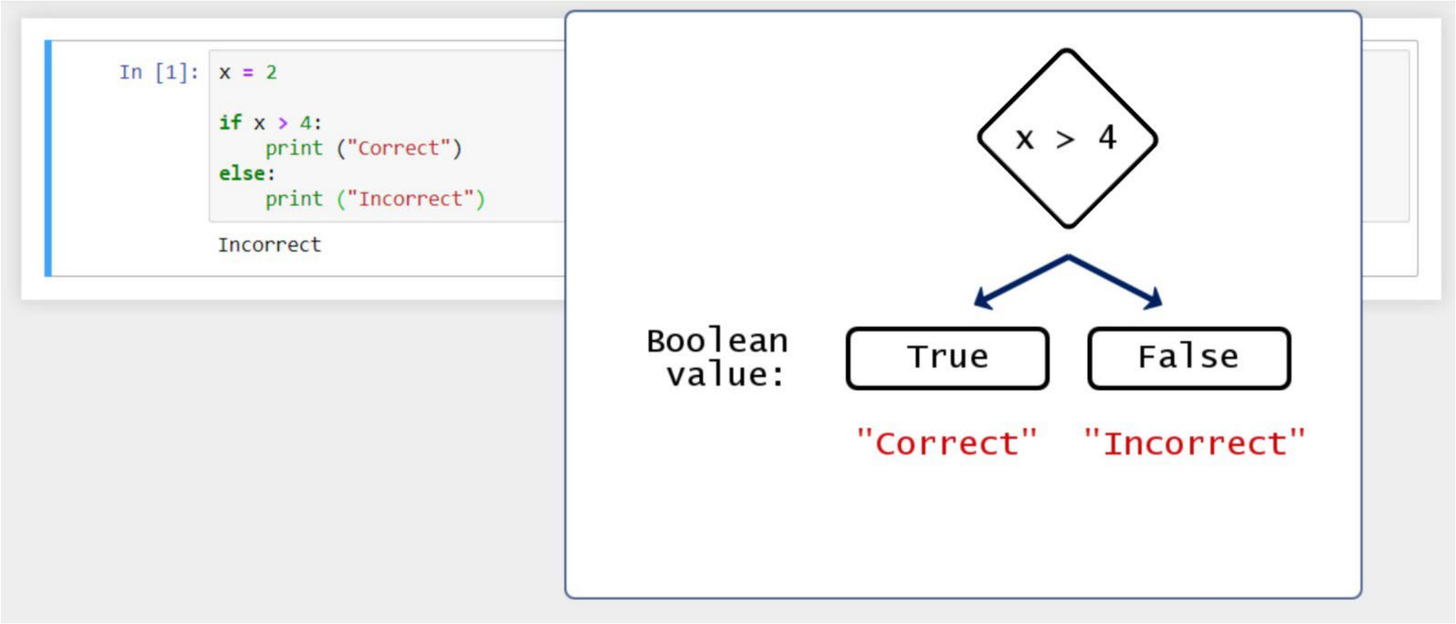


A very important detail you should try to remember is that the computer always reads your commands from top to bottom. Regardless of the speed at which it works, it executes only one command at a time. Scientifically speaking, the instructions we give to the machine are part of a control flow. This is something like the flow of the logical thought of the computer, the way the computer thinks – step by step, executing the steps in a rigid order.

When it works with a conditional statement, the computer's task will be to execute a specific command once a certain condition has been satisfied. It will read your commands from the if- statement at the top, through the elif-statements in the middle, to the else- statement at the end. The first moment the machine finds a satisfied condition, it will print the respective output and will execute no other part of the code from this conditional.

A Note on Boolean Values

From a certain perspective everything in a computer system is Boolean comprising sequences of 0s and 1s, “False” and “True”. This is why we are paying attention to the Boolean value. It helps us understand general computational logic and the way conditionals work in Python.



Basically, after you insert your if-statement, the computer will attach a Boolean value to it. Depending on the value of its outcome, “True” or “False”, it will produce one of the suggested outputs, “Correct” or “Incorrect”.

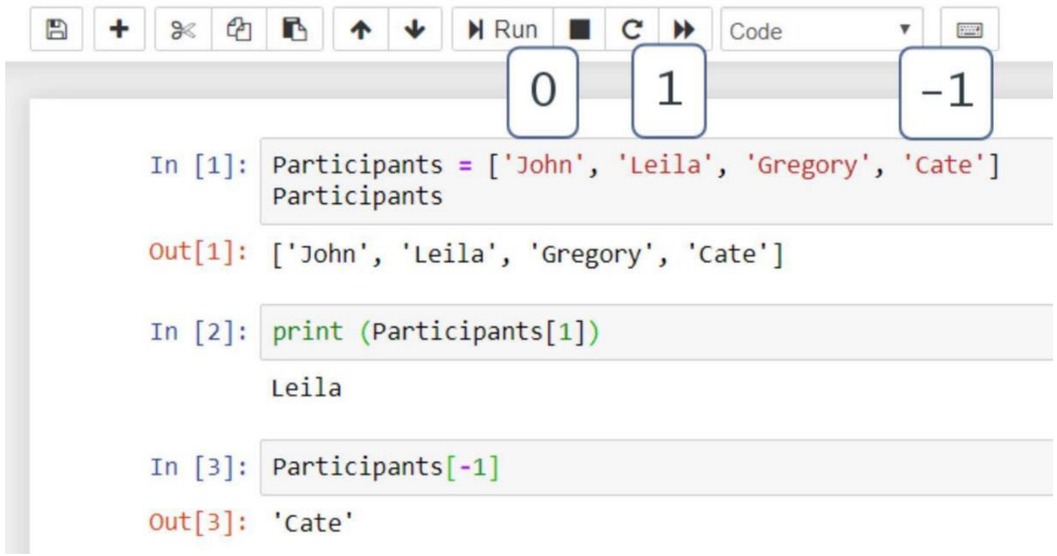
Container / Sequences

5. List

6.Tuple

Lists

A **list** is a type of sequence of data points such as floats, integers, or strings.

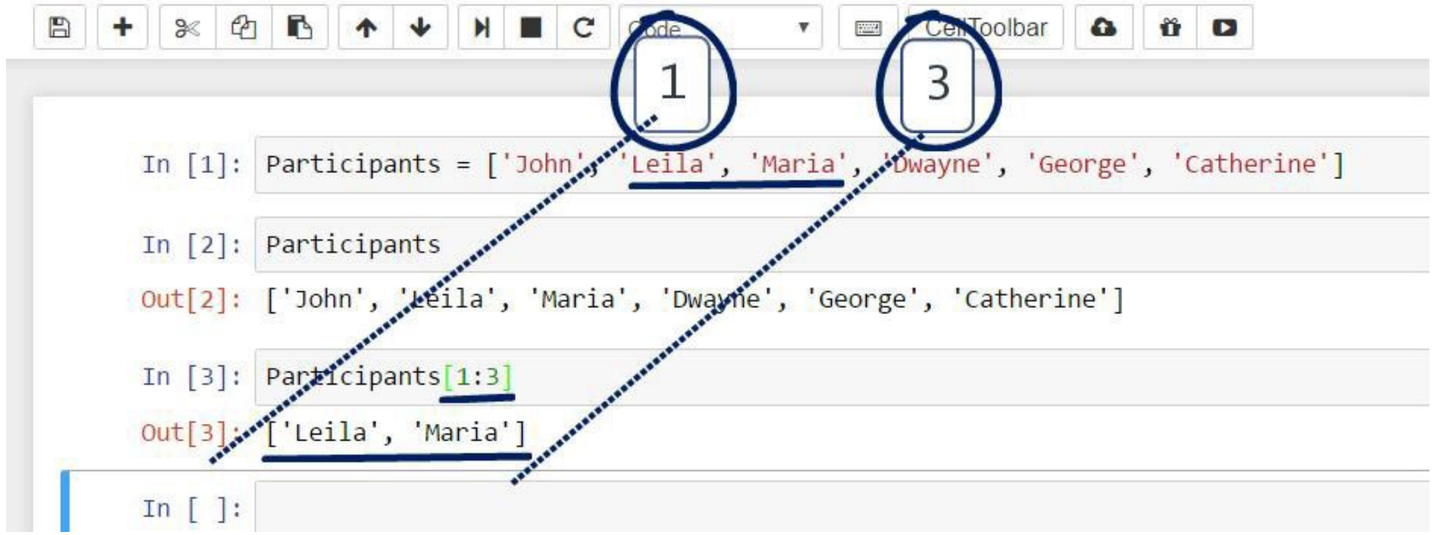


You can access the *Participants* list by indexing the value *1*. This means you have extracted the second of the elements in this list variable ['Leila'].

In addition, there is a way to get to the last element from your list. To do that, start counting from the end towards the beginning. Then, you'd need the minus sign before the digit.
Don't fall in the trap of thinking we begin enumerating from *0* again! To obtain "Cate", you must write *-1*.

List Slicing

Many of the problems that must be solved will regard just a portion of the data, and in such cases, you can apply slicing.



Imagine you want to use the "Participants" list to obtain a second much smaller list that contains only two names - Leila and Maria. In Pythonic, that would mean to extract the elements from the first and second position. To access these elements, we will open square brackets, just as we did with indexing, and write 1 colon 3. The first number corresponds precisely to the first position of interest, while the second number is one position above the last position we need.

Help Yourself with Methods

Here is the syntax that allows you to call ready-made **built-in methods** that you do not have to create on your own and can be used in Python directly.

After the name of the **object**, which in this case is the "Participants" list, you must put a dot called a **dot operator**. The dot operator allows you to **call** or **invoke** a certain method. To call the method "append", state its name, followed by **parentheses**.

```
object . method ()
```

```
In [8]: Participants.append("Dwayne")
        Participants

Out[8]: ['John', 'Leila', 'Maria', 'Dwayne']
```

To insert the name "Dwayne" in our list, you must put the string "Dwayne" in inverted commas between the parentheses.

Help Yourself with Methods

```
In [9]: Participants.extend(['George', 'Catherine'])
        Participants

Out[9]: ['John', 'Leila', 'Maria', 'Dwayne', 'George', 'Catherine']
```

Alternatively, the same result can be achieved by using the *.extend()* method. This time, within the parentheses, you'll have to add brackets, as you are going to extend the *Participants* list by adding a list specified precisely in these parentheses.

Tuples

Tuples are another type of data sequences, but differently to lists, they are ***immutable***. Tuples cannot be changed or modified; you cannot append or delete elements.

```
In [1]: x = (40,41,42)
        x

Out[1]: (40, 41, 42)

In [2]: y = 50,51,52
        y

Out[2]: (50, 51, 52)

In [3]: a,b,c = 1,4,6
        c

Out[3]: 6
```

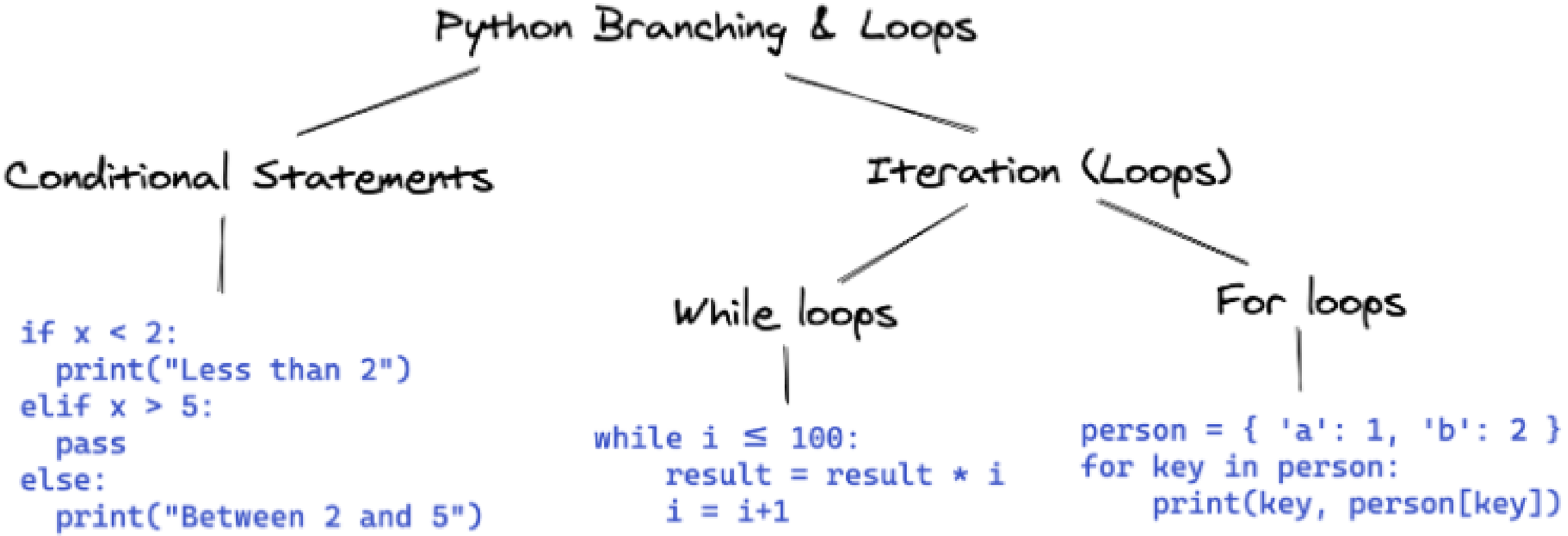
The syntax that indicates you are having a tuple and not a list is that the tuple's elements are placed within parentheses, not brackets.

The tuple is the default sequence type in Python, so if you list three values here, the computer will perceive the new variable as a tuple. We could also say the three values will be **packed** into a tuple.

For the same reason, you can assign a number of values to the same number of variables. On the left side of the equality sign, add a tuple of variables, and on the right, a tuple of values. That's why the relevant technical term for this feature is **tuple assignment**.

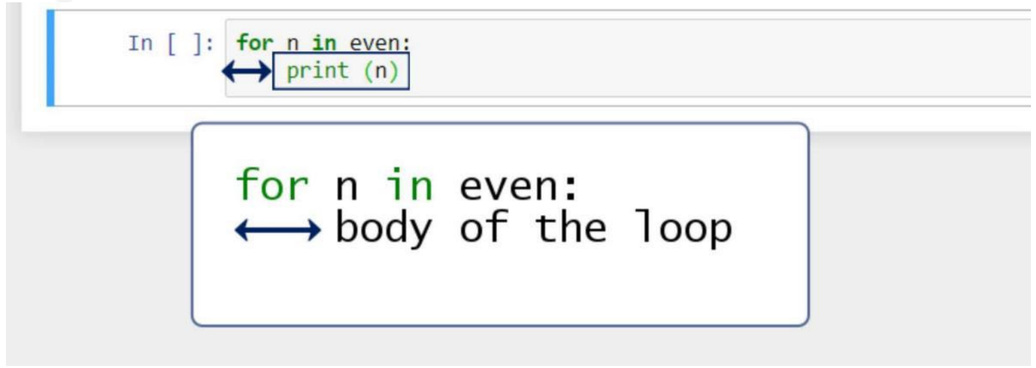
7. Loops in Python

Conditional Statements



For Loops

Iteration is a fundamental building block of all programs. It is the ability to execute a certain code repeatedly.



The list *even* contains all the even numbers from 0 to 20.
for n in even, colon,
which would mean *for* ***every*** **element *n* in the list *even*, do the following**:
print that element.

The command in the loop body is performed once *for* ***each*** element in the *even* list.

While Loops and Incrementing

The same output we obtained in the previous lesson can be achieved by using a while- loop, instead of a for- loop. However, the structure we will use will be slightly different.

```
In [1]: x = 0
        while x <= 20:
            print (x, end = " ")
            x = x + 2

0 2 4 6 8 10 12 14 16 18 20
```

```
In [2]: x = 0
        while x <= 20:
            print (x, end = " ")
            x += 2

0 2 4 6 8 10 12 14 16 18 20
```

Initially, we will set a variable x equal to zero. And we'll say: **while** this value is smaller than or equal to 20, print x.
We want to get the loop to end. What is supposed to succeed, the loop body in the "while" block, is a line of code that specifies a change in x or what has to happen to x after it is printed. In our case, we will tell the computer to bind x to a value equal to x + 2.

In programming terms, adding the same number on top of an existing variable during a loop is called **incrementing**. The amount being progressively added is called an **increment**. In our case, we have an increment of 2.

The Pythonic syntax offers a special way to indicate incrementing: **x += 2**

Create Lists with the range() Function

When you need to randomize data points and lists with data points, you can use Python's built-in *range()* function.

The syntax of the function is the following:

```
        0           1
range(start, stop, step)

start = the first number in the list

stop = the last value +1

step = the distance between each two consecutive values
```

The stop value is a required input, while the start and step values are optional. If not provided, the start value will be automatically replaced with a 0, and the step value would be assumed to be equal to 1.

Create Lists with the range() Function

```
In [1]: range(10)
Out[1]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

In [2]: range(3,7)
Out[2]: [3, 4, 5, 6]

In [3]: range(1,20,2)
Out[3]: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

range(10) will provide a list of 10 elements, starting from 0, implied after not indicating a start value, and ending at the tenth consecutive number – 9.

In another cell, if in the *range()* function we declare as arguments 3 and 7, for instance, Python will accept 3 as a start value, and 7 as a stop value of the range. So, we'll have 4 elements – 3, 4, 5, and 6.

To specify a step value in a range, the other two arguments must be chosen as well. *range(1,20,2)* creates a list including all the odd numbers from 1 to 19. It will start with the number 1, and the list will end with number 19 (which equals the stop value 20 minus 1), stating only the odd numbers.

Use Conditional Statements and Loops Together

```
In [2]: for x in range(20):
            if x % 2 == 0:
                print (x, end = " ")
            else:
                print ("Odd", end = " ")
```

```
0 Odd 2 Odd 4 Odd 6 Odd 8 Odd 10 Odd 12 Odd 14 Odd 16 Odd 18 Odd
```

You create an iteration that includes a conditional in the loop body. You can tell the computer to print all the even values between 0 and 19 and state "Odd" in the places where we have odd numbers.

Let's translate this into computational steps.

If *x* leaves a remainder of 0 when divided by 2, which is the same as to say "if x is even", then print *x* on the same line.
else, which means unless *x* is even, or if *x* is odd, print *"Odd"*.

All In – Conditional Statements, Functions, and Loops

We use iterations when we want to go through variables that are part of a list.

You can count the number of items whose value is less than 20 in a list. First, define a function that takes as an argument *numbers*, where *numbers* will be a certain list variable. The trick is to create a variable that, so to speak, "departs" from 0. Let's call it *total*.

```
In [1]: def count(numbers):
            total = 0
            for x in numbers:
                if x < 20:
                    total += 1
            return total
```

The idea is that, when certain conditions are verified, *total* will change its value. That's why, in such a situation, it is appropriate to call this variable a **rolling sum**.

More technically, when we consider *x* in the numbers list, if it is smaller than 20, we will increment the total by 1 and finally return the *total* value. This means that, if *x* is less than 20, *total* will grow by 1, and if *x* is greater than or equal to 20, *total* will not grow. So, for a given list, this *count()* function will return the amount of numbers smaller than 20.

Container / Sequences
8. Dictionary

Dictionaries

Dictionaries represent another way of storing data.

```
In [1]: dict = {'k1': "cat", 'k2': "dog", 'k3': "mouse", 'k4': "fish"}
        dict
Out[1]: {'k1': 'cat', 'k2': 'dog', 'k3': 'mouse', 'k4': 'fish'}

In [2]: dict['k1']
Out[2]: 'cat'
```

dict

key	k1	k2	k3	k4
value	cat	dog	mouse	fish

Each value is associated with a certain **key**. More precisely, a key and its respective value form a ***key-value pair***. After a certain dictionary has been created, a value can be accessed by its key, instead of its index!

```
In [ ]: dict['k5'] = 'parrot'
```

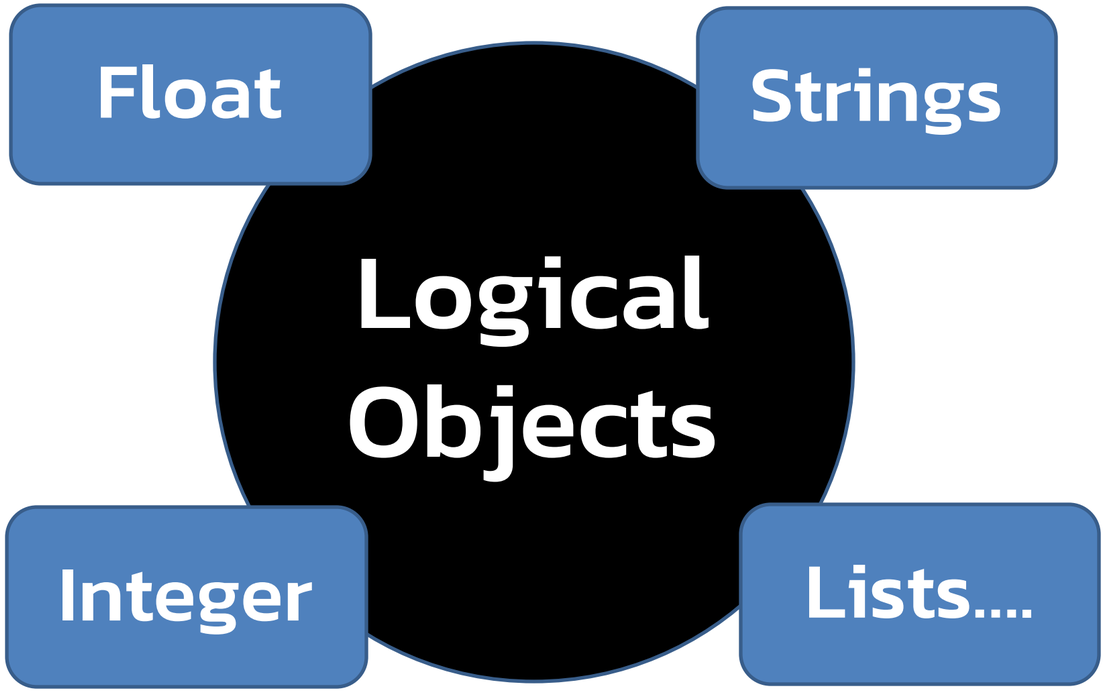
```
dictionary_name [new_key_name]  =  new_value_name
```

Similarly, as we could do with lists, we can add a new value to the dictionary in the following way: the structure to apply here is dictionary name, new key name within brackets, equality sign, and the name of the new value.

9. Modules and library functions

Object-Oriented programming (OOP)

Object-Oriented programming (OOP)



Object-Oriented programming (OOP)

List_temp_c	
Point 1	24
Point 2	26
Point 3	25
Point 4	24
Point 5	26
.....	.....
Point 24	32

List_temp_c = [24 , 26 , 25 , 24 , 26 ,... , 32]

24 data points

Dict_name_Gallery

Names :

‘National Gallery’ ,

‘Central Gallery’ ,

....

Dict_name_Gallery = { ‘Name’ : ‘National Gallery’ , ... }

Object-Oriented programming (OOP)

Object = Data + Manipulation Operation

"Bike" Class

Object :

A

Attributes :

Color Red, Black, Green

Size 12 inches, 16 inches

Type Touring, Mountain

Method :

.turn_left()

.turn_right()

.break()

.accelerate()

Object-Oriented programming (OOP)

Object = Data + Manipulation Operation

"List" Class

Object :

[15.9 , 12.4 , 64.0]

Attributes :

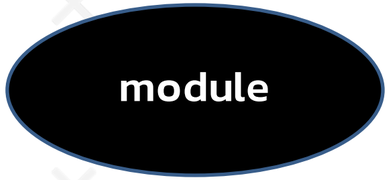
float

Method :

.extend()

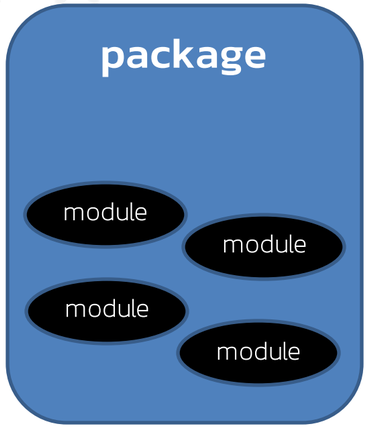
.index()

module vs package vs library



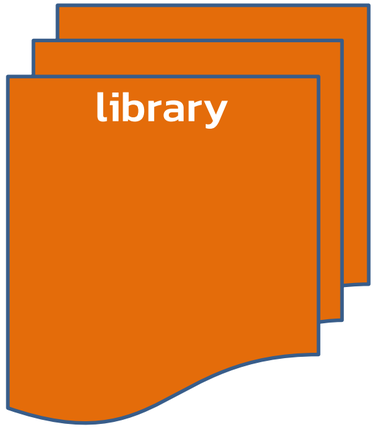
is a simple Python file that contains collections of functions and global variables.

Examples of modules:
Datetime
Regex
Random etc.



a simple directory having collections of modules.

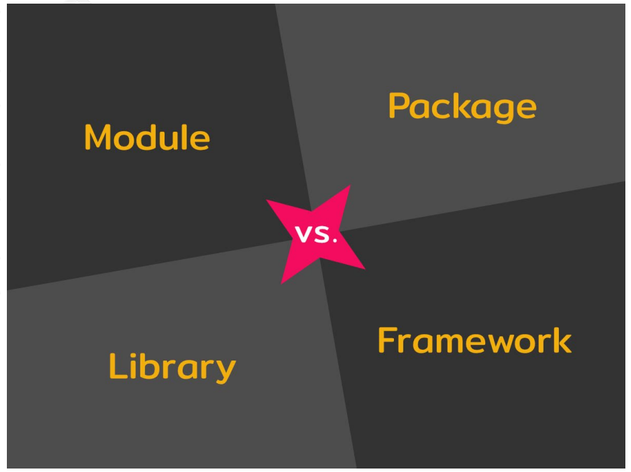
Examples of Packages:
Numpy
Pandas



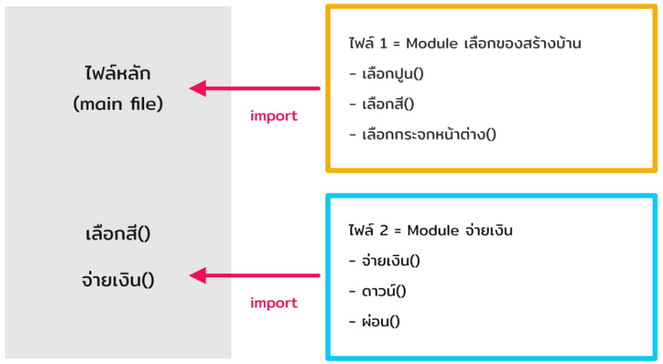
is having a collection of related functionality of codes
It is a reusable chunk of code that we can use by importing it into our program, we can just use it by importing that library and calling the method of that library with a period(.)

Examples of Libraries:
Matplotlib
Pytorch
Pygame
Seaborn etc.

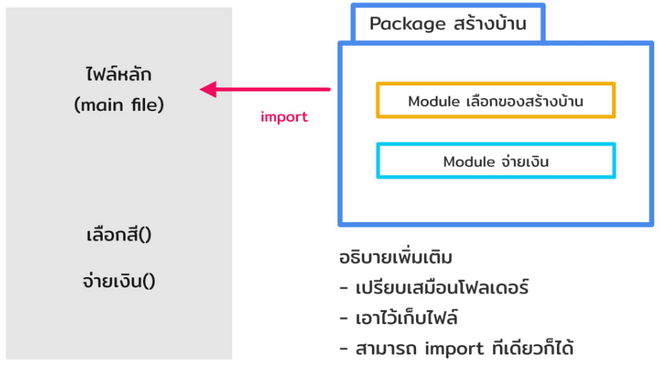
module vs package vs library



Module ~ ไฟล์ที่เอาไว้เก็บโค้ดตัวแปรและฟังก์ชั่น

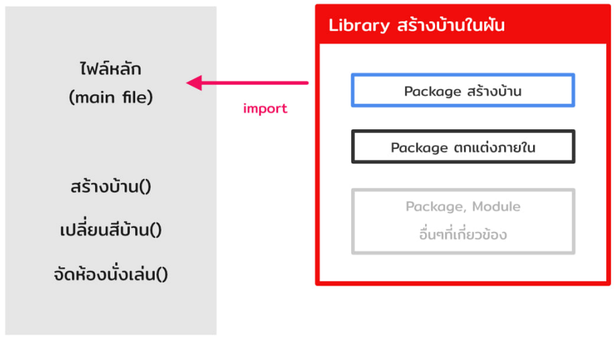


Package ~ การรวมกลุ่ม Module มาอยู่ด้วยกันในที่เดียว

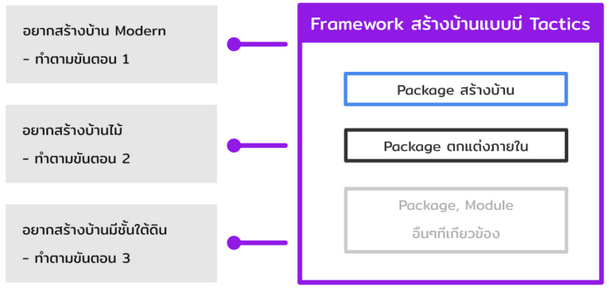


อธิบายเพิ่มเติม
- เปรียบเสมือนโฟลเดอร์
- เอาไว้เก็บไฟล์
- สามารถ import ทีเดียวก็ได้

Library ~ สิ่งที่เอา Package, Module มาอยู่รวมกัน
เพื่อสร้างผลงานเฉพาะเจาะจงได้รวดเร็วขึ้นแบบปรู๊ดปร๊าดดดดด



Framework ~ ใช้สร้างผลงานคล้ายกับ Library แตกต่างที่ว่า.....



Framework จะมีกรอบสำหรับกำหนดว่า เราต้องทำตามนี้ ถึงจะได้สิ่งที่ต้องการมา
หลายๆกรณี จะไม่ปล่อยอิสระเท่ากับ Library

Python Standard Library

- Data Compression and Archiving
  - `zlib` — Compression compatible with **gzip**
  - `gzip` — Support for **gzip** files
  - `bz2` — Support for **bzip2** compression
  - `lzma` — Compression using the LZMA algorithm
  - `zipfile` — Work with ZIP archives
  - `tarfile` — Read and write tar archive files
- File Formats
  - `csv` — CSV File Reading and Writing
  - `configparser` — Configuration file parser
  - `netrc` — netrc file processing
  - `xdrlib` — Encode and decode XDR data
  - `plistlib` — Generate and parse Mac OS X `.plist` files
- Cryptographic Services
  - `hashlib` — Secure hashes and message digests
  - `hmac` — Keyed-Hashing for Message Authentication
  - `secrets` — Generate secure random numbers for managing secre
- Generic Operating System Services
  - `os` — Miscellaneous operating system interfaces
  - `io` — Core tools for working with streams
  - `time` — Time access and conversions
  - `argparse` — Parser for command-line options, arguments and su
  - `getopt` — C-style parser for command line options
  - `logging` — Logging facility for Python
  - `logging.config` — Logging configuration
  - `logging.handlers` — Logging handlers
  - `getpass` — Portable password input
  - `curses` — Terminal handling for character-cell displays
  - `curses.textpad` — Text input widget for curses programs
  - `curses.ascii` — Utilities for ASCII characters
  - `curses.panel` — A panel stack extension for curses
  - `platform` — Access to underlying platform's identifying data
  - `errno` — Standard errno system symbols
  - `ctypes` — A foreign function library for Python

Importing Modules

```
[ ]  import math

[ ]  math.sqrt(16)

     4.0
```

```
[ ]  from math import sqrt

[ ]  sqrt(25)

     5.0
```

```
[ ]  import math as m

[ ]  m.sqrt(49)

     7.0
```

```
from math import *

sqrt(64)

8.0
```

ไม่นิยมใช้ * (all)
เพราะ ทำให้เราไม่รู้จริงๆ ว่า Library ที่ใช้จริงๆ คืออะไร
ถ้าเป็น Module ใหญ่ๆ จะเปลืองทรัพยากร

```
[ ]  help(math)

     Help on built-in module math:

     NAME
         math

     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.

             The result is between 0 and pi.

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

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

             The result is between -pi/2 and pi/2.
```

```
help(math.sqrt)

Help on built-in function sqrt in module math:

sqrt(...)
    sqrt(x)

    Return the square root of x.
```

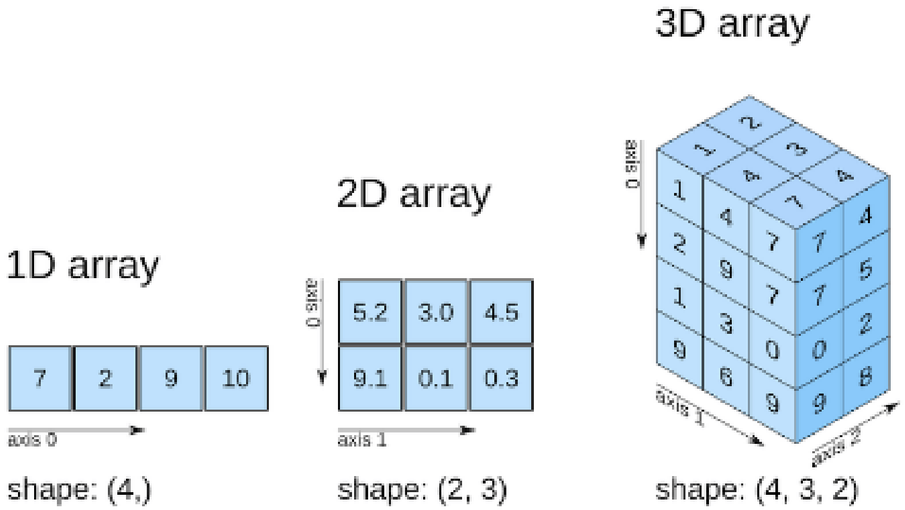
10. Introduction to Using NumPy

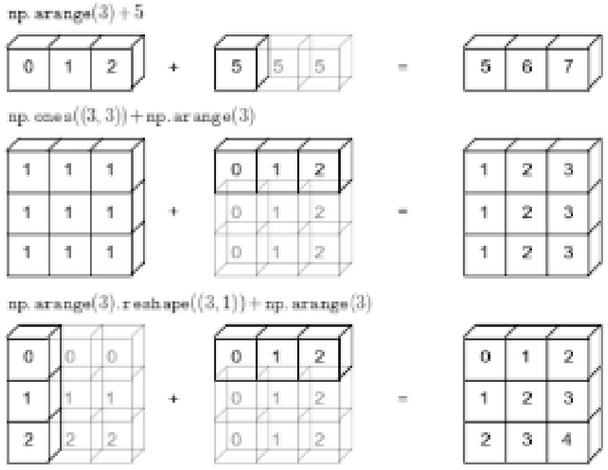
Introduction to Using NumPy and pandas

NumPy is the fundamental package for scientific computing in Python.

https://numpy.org/doc/stable/user/absolute_beginners.html

Numerical Computing with Python and Numpy





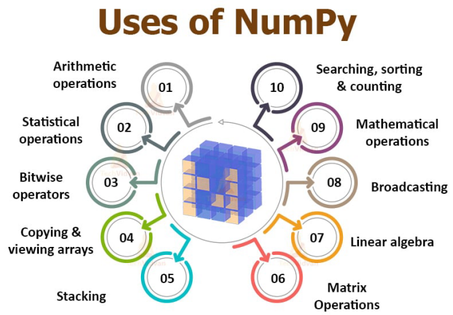
Numpy

- A Collection of pre-written functions, class and methods which are capable of handling and manipulating data and calculating results
- Main feature : Arrays

- ❑ NumPy is an open-source numerical Python library.
- ❑ NumPy contains a multi-dimensional array and matrix data structures.
- ❑ It can be utilized to perform a number of mathematical operations on arrays such as trigonometric, statistical, and algebraic routines. Therefore, the library contains a large number of mathematical, algebraic, and transformation functions.
- ❑ NumPy is an extension of Numeric and Numarray.
- ❑ Numpy also contains random number generators.
- ❑ NumPy is a wrapper around a library implemented in C.
- ❑ Pandas objects rely heavily on NumPy objects. Essentially, Pandas extends Numpy.

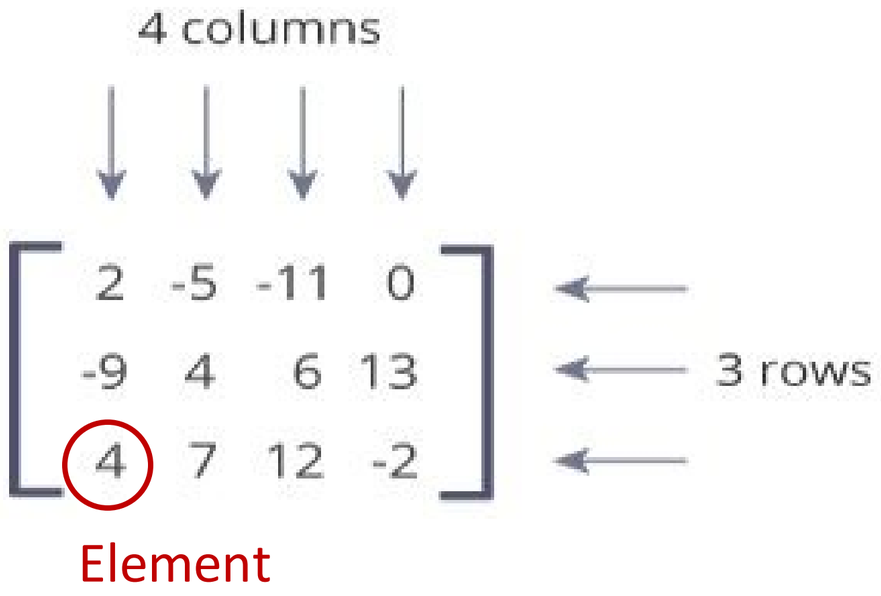
https://numpy.org/doc/stable/user/whatisnumpy.html

- Stores multiple types of data
- **Works in a lower level language => Shorter computation times**
- **Useful universal functions and methods designed specifically**



❑ Cheat sheet Numpy :
https://machinelearningmastery.com/wp-content/uploads/2022/04/cheatsheet.png

Matrices



Mathematics

$$\begin{matrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{31} \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{matrix}$$

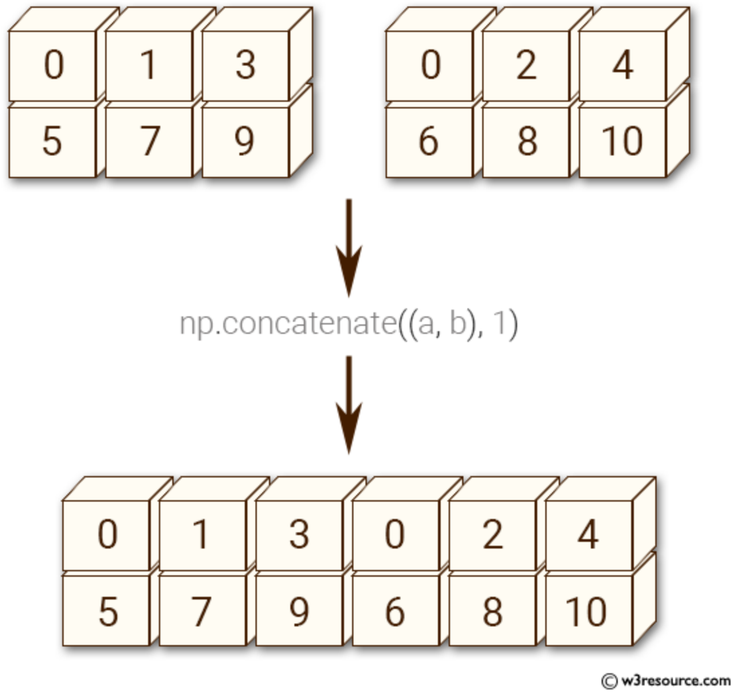
Operations

- ❑ **Addition**
- ❑ **Subtraction**
- ❑ **Multiplication**
- ❑ ....

Pragramming

$$\begin{matrix} a_{00} & a_{01} & a_{02} & \dots & a_{0(n-1)} \\ a_{01} & a_{11} & a_{12} & \dots & a_{1(n-1)} \\ a_{02} & a_{21} & a_{22} & \dots & a_{2(n-1)} \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ a_{(m-1)0} & a_{(m-1)1} & a_{(m-1)2} & \dots & a_{(m-1)(n-1)} \end{matrix}$$

Concatenate



Matrices

Scalar

$$\begin{bmatrix} 15 \end{bmatrix}$$

0D

Vector

$$\begin{bmatrix} 5 \\ -2 \\ 4 \end{bmatrix}$$

1D

Matrices

$$\begin{bmatrix} 2 & -5 & -11 & 0 \\ -9 & 4 & 6 & 13 \\ 4 & 7 & 12 & -2 \end{bmatrix}$$

2D

Addition / Subtraction

- Very Easy
- The 2 matrices must have to same dimensions

$$\begin{bmatrix} 8 & 5 \\ 2 & 3 \end{bmatrix} + \begin{bmatrix} 9 & 5 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 17 & 10 \\ 3 & 5 \end{bmatrix}$$

$$\begin{bmatrix} 8 & 5 \\ 2 & 3 \end{bmatrix} - \begin{bmatrix} 9 & 5 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 1 & 1 \end{bmatrix}$$

Transpose

$$A = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}_{2 \times 3}$$

$$A^T = \begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix}_{3 \times 2}$$

Dot product

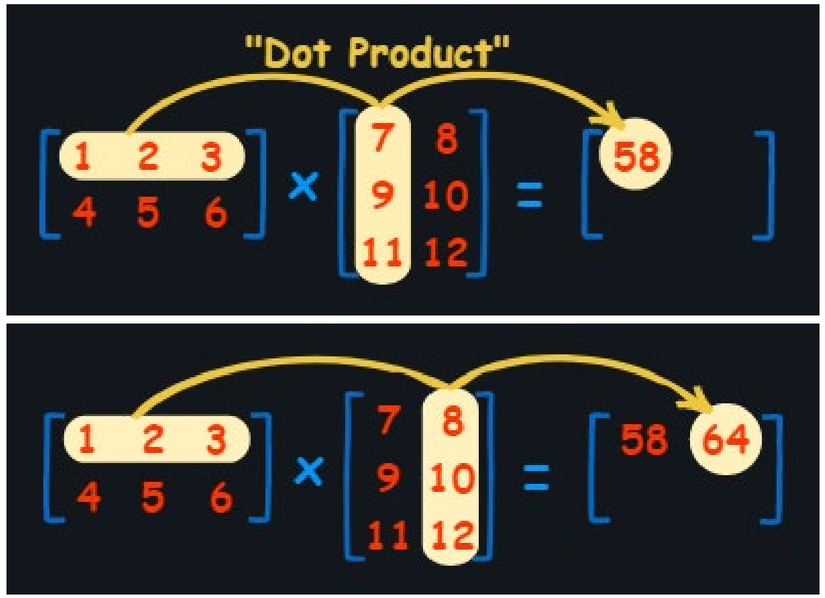
Vector

$$\begin{bmatrix} 2 \\ 7 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 8 \\ 2 \\ 8 \end{bmatrix} = 2 \cdot 8 + 7 \cdot 2 + 1 \cdot 8 = 38$$

Dot product

Dot product

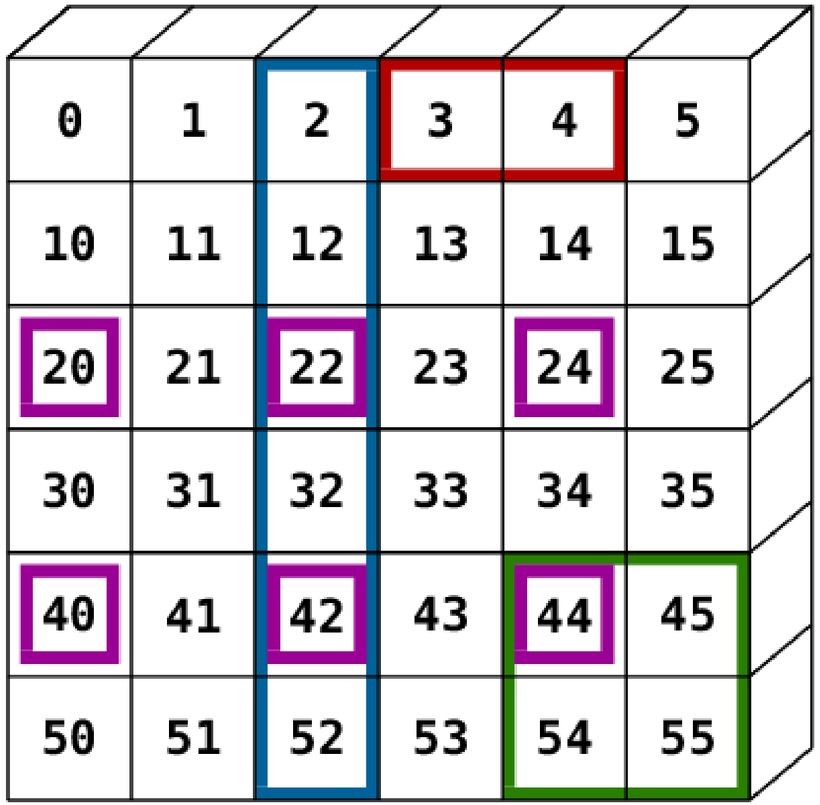
Matrices



$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix}$$

Array indexing and slicing

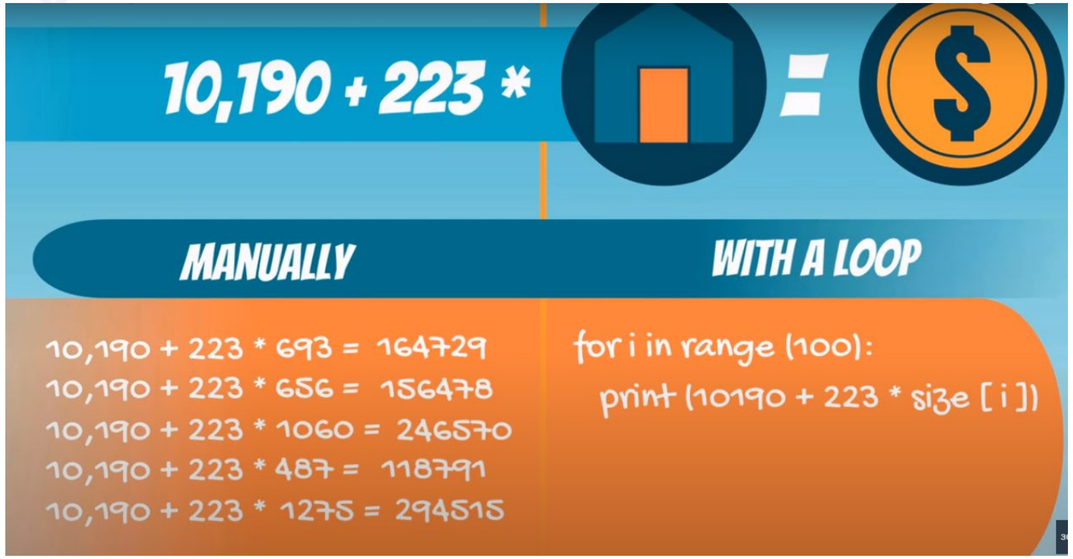
```
>>> a[0, 3:5]
array([3, 4])

>>> a[4:, 4:]
array([[44, 55],
       [54, 55]])

>>> a[:, 2]
a([2, 12, 22, 32, 42, 52])

>>> a[2::2, ::2]
array([[20, 22, 24],
       [40, 42, 44]])
```

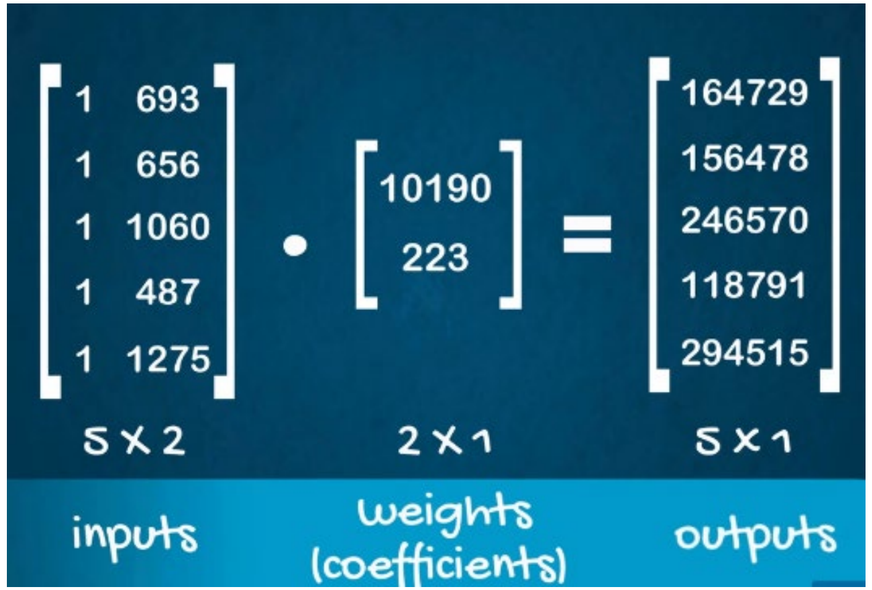


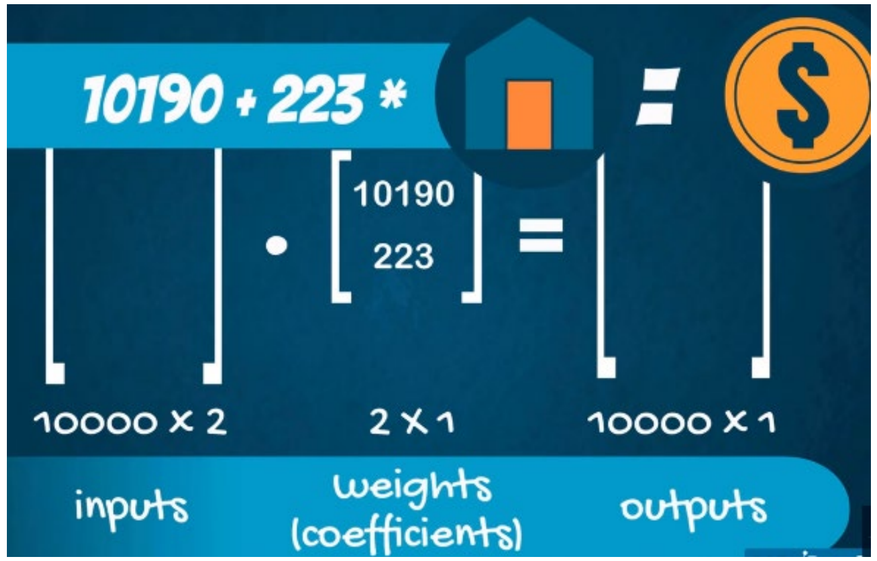
Why is Linear Algebra Useful?

Why is Linear Algebra Useful?

Vectorized code

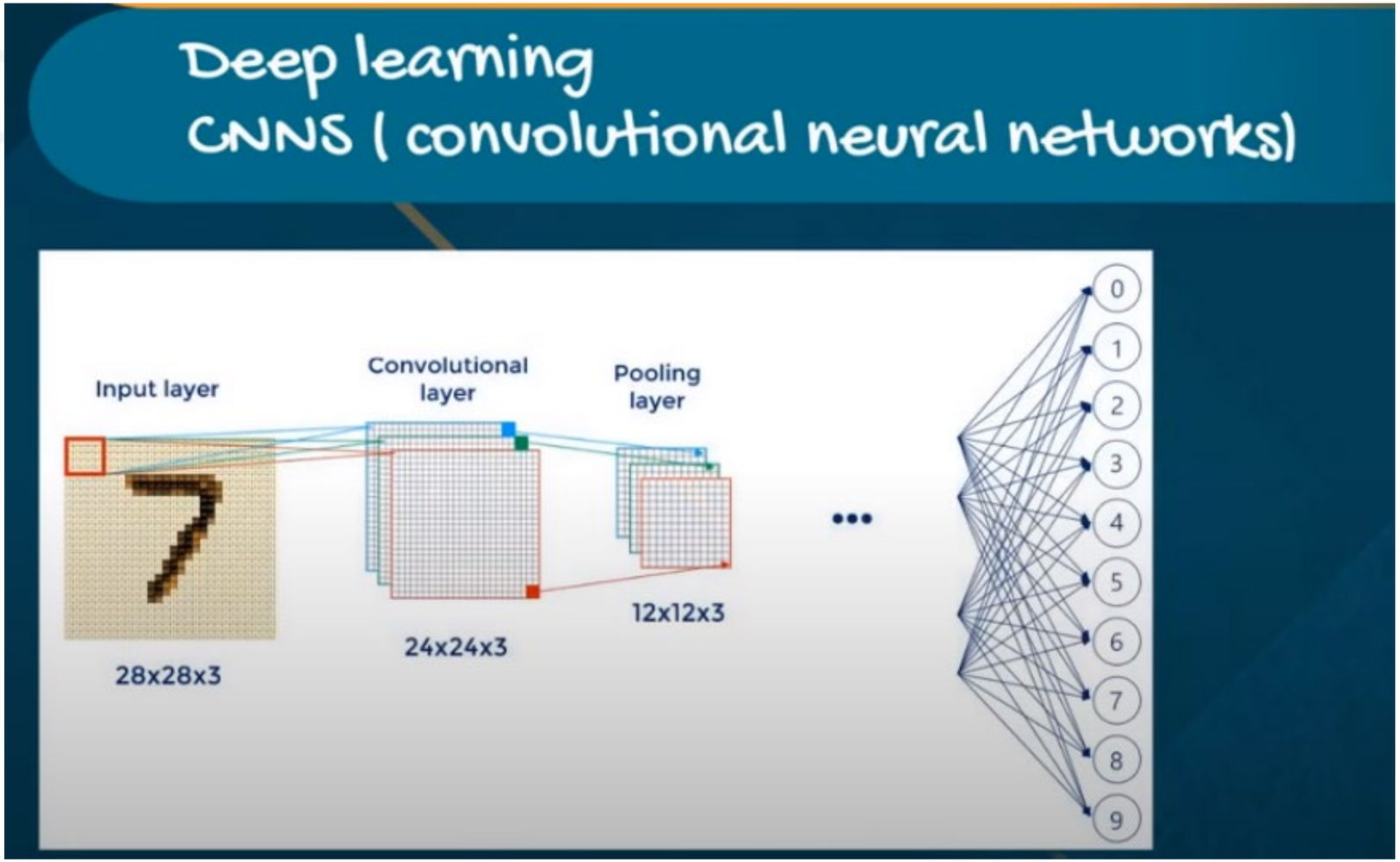






Why is Linear Algebra Useful?

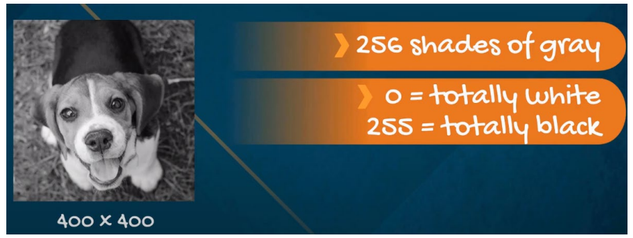
Image recognition

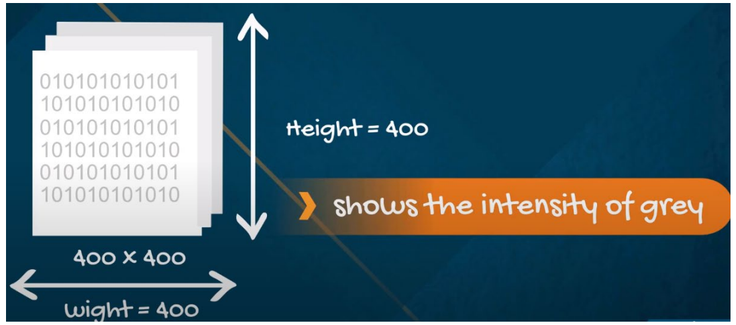


Why is Linear Algebra Useful?

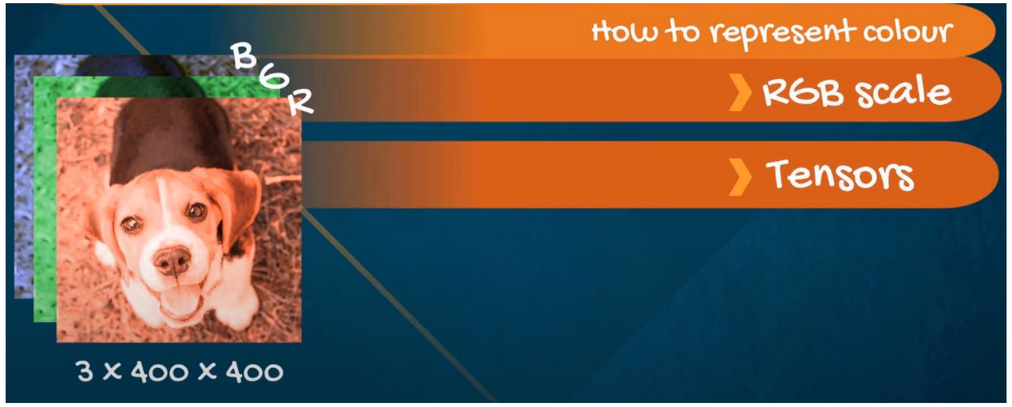
Image recognition

Dog ?
Cat ?
??





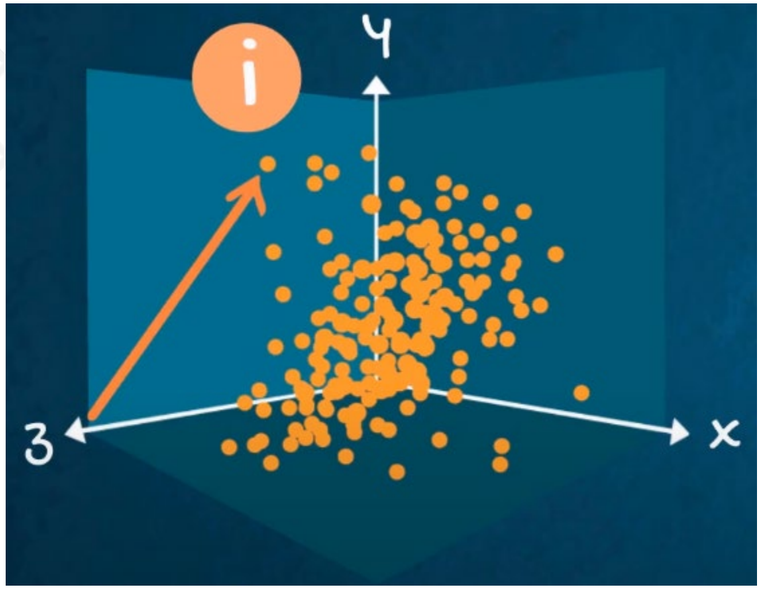
1D Matric

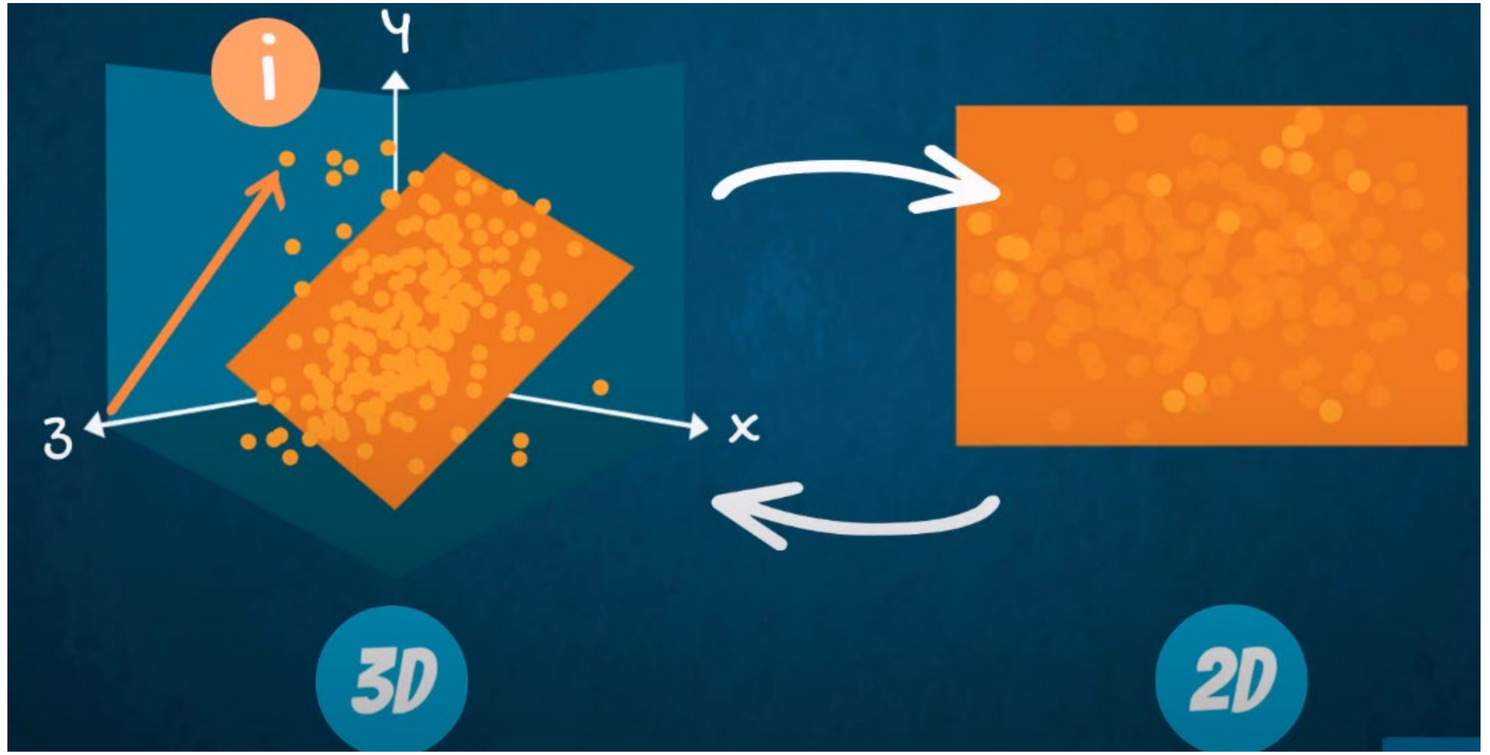


3D Matric

Why is Linear Algebra Useful?

Dimensionality reduction



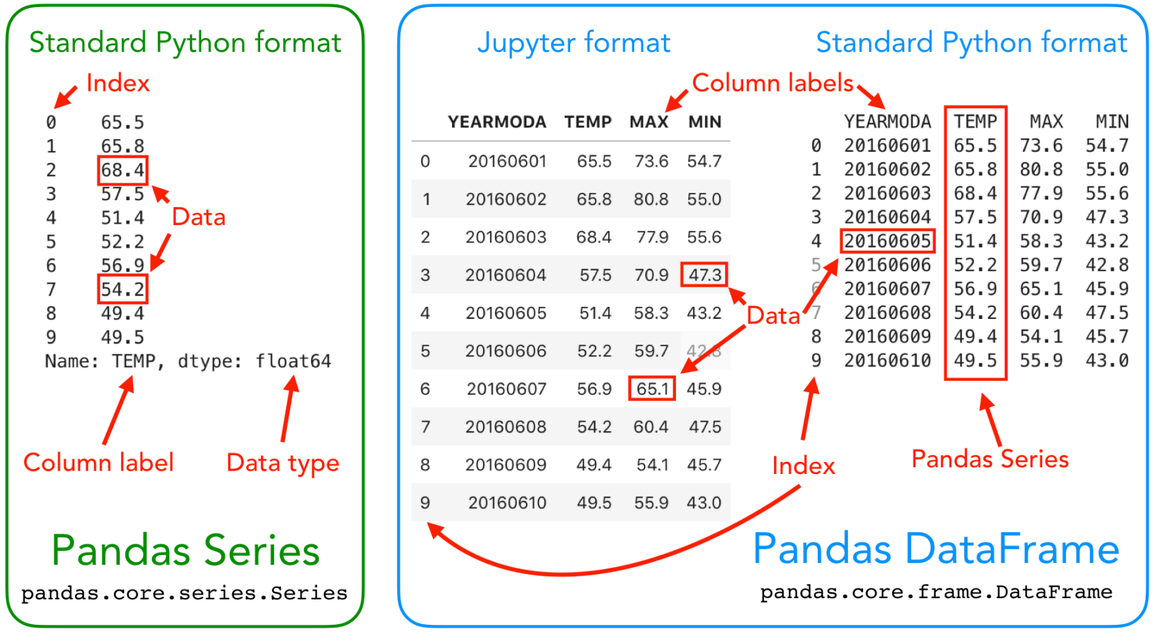


11. Analyzing Tabular Data with Pandas

is an open source, BSD-licensed library providing high-performance, easy-to-use data structures (json , xlsx , csv , sql , html, etc.) and data analysis tools for the Python programming language.

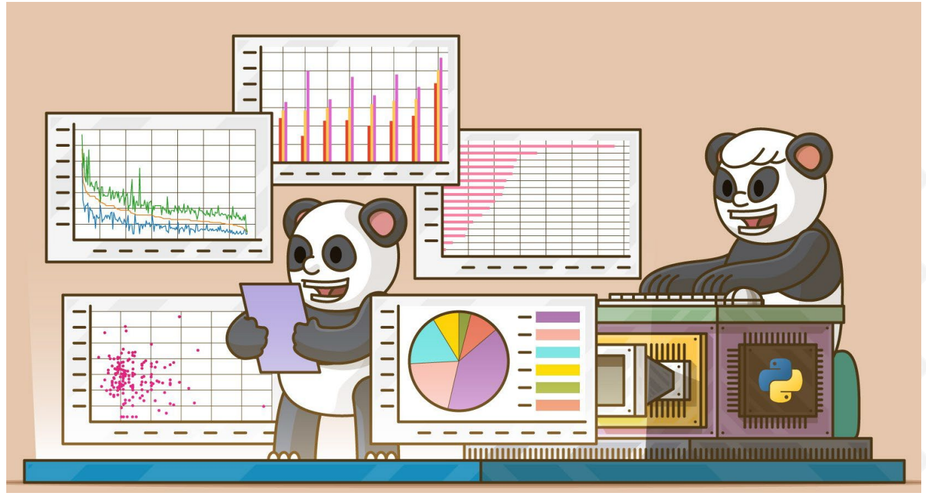
https://pandas.pydata.org/docs/getting_started/index.html

Introduction to Pandas



Documentation

Visualization

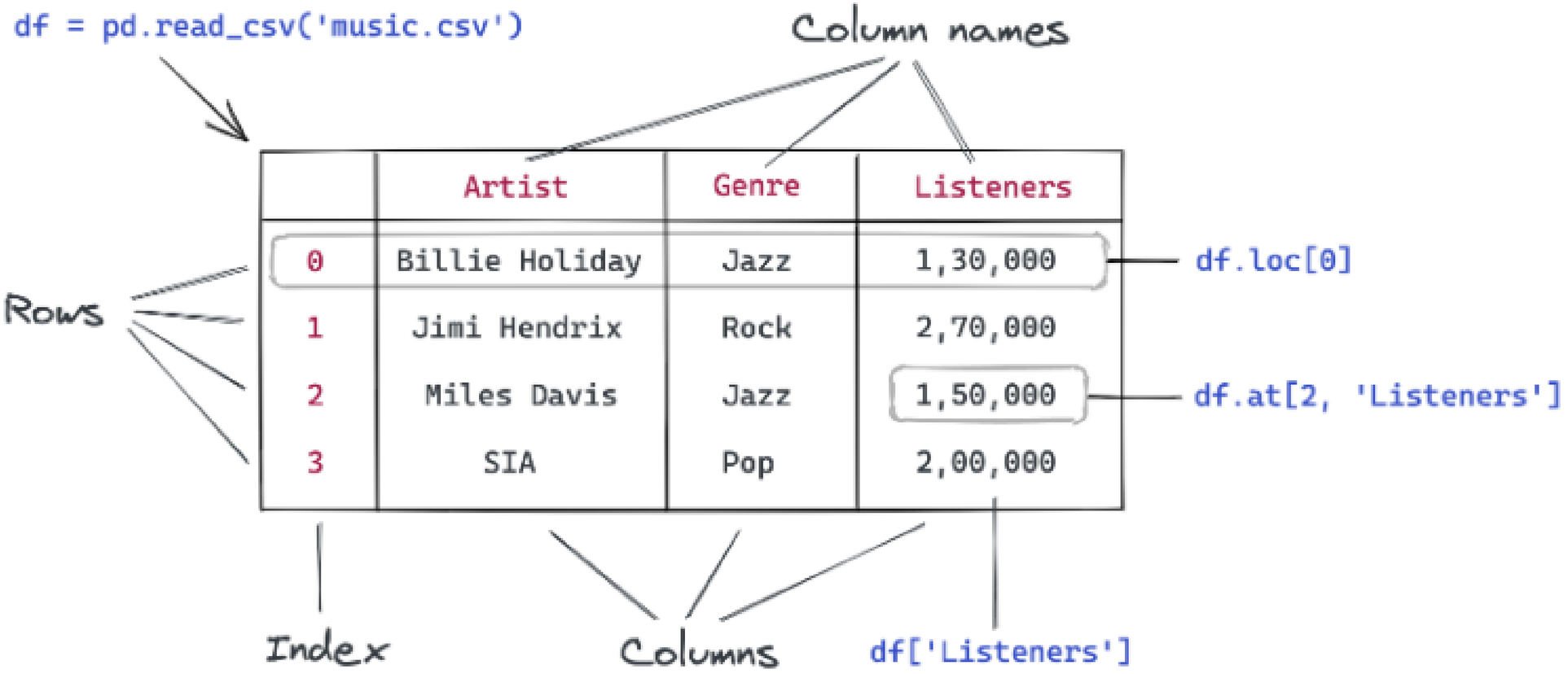


- **Makes out work much faster and easier**
- **Faster and easier data analysis and Visualization**
- **Import / Export .csv, .xls , .json , .npz**

❑ Cheat sheet Pandas :

https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf

Introduction to Pandas



Combining Dataframe

df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df2

	A	B	C	D
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7

df3

	A	B	C	D
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

Result

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df4

	B	D	F
2	B2	D2	F2
3	B3	D3	F3
6	B6	D6	F6
7	B7	D7	F7

Result

	A	B	C	D	B	D	F
0	A0	B0	C0	D0	NaN	NaN	NaN
1	A1	B1	C1	D1	NaN	NaN	NaN
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3
6	NaN	NaN	NaN	NaN	B6	D6	F6
7	NaN	NaN	NaN	NaN	B7	D7	F7

df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df4

	B	D	F
2	B2	D2	F2
3	B3	D3	F3
6	B6	D6	F6
7	B7	D7	F7

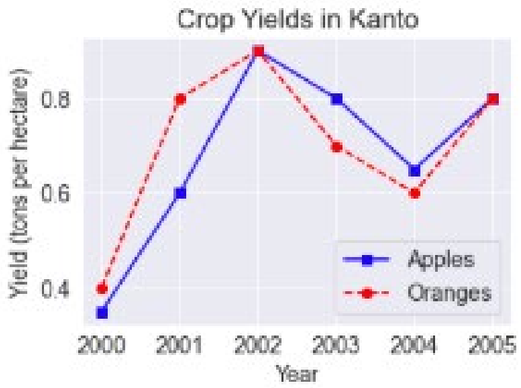
Result

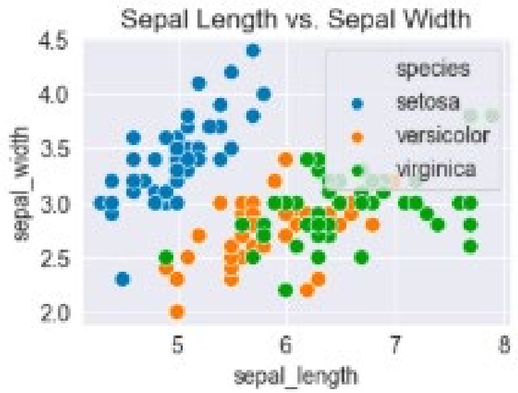
	A	B	C	D	B	D	F
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3

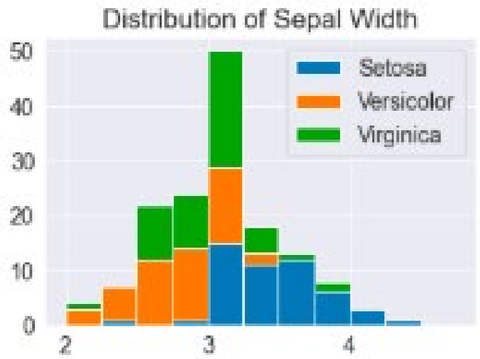
https://pandas.pydata.org/docs/user_guide/merging.html

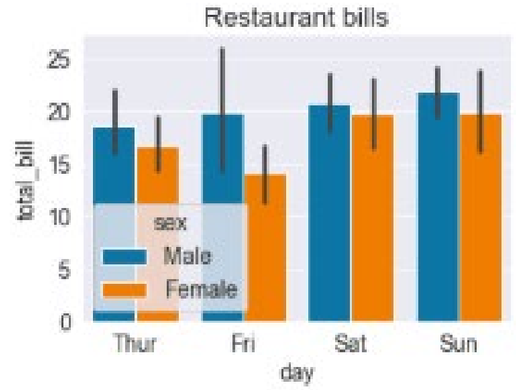
8. Visualization with Matplotlib and Seaborn

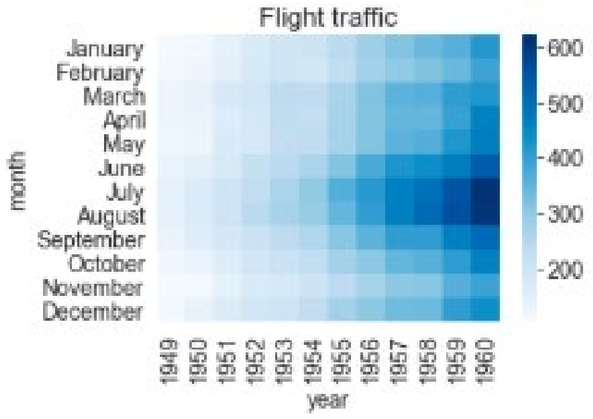
Visualization with Matplotlib and Seaborn

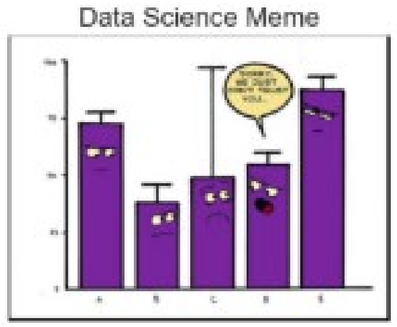




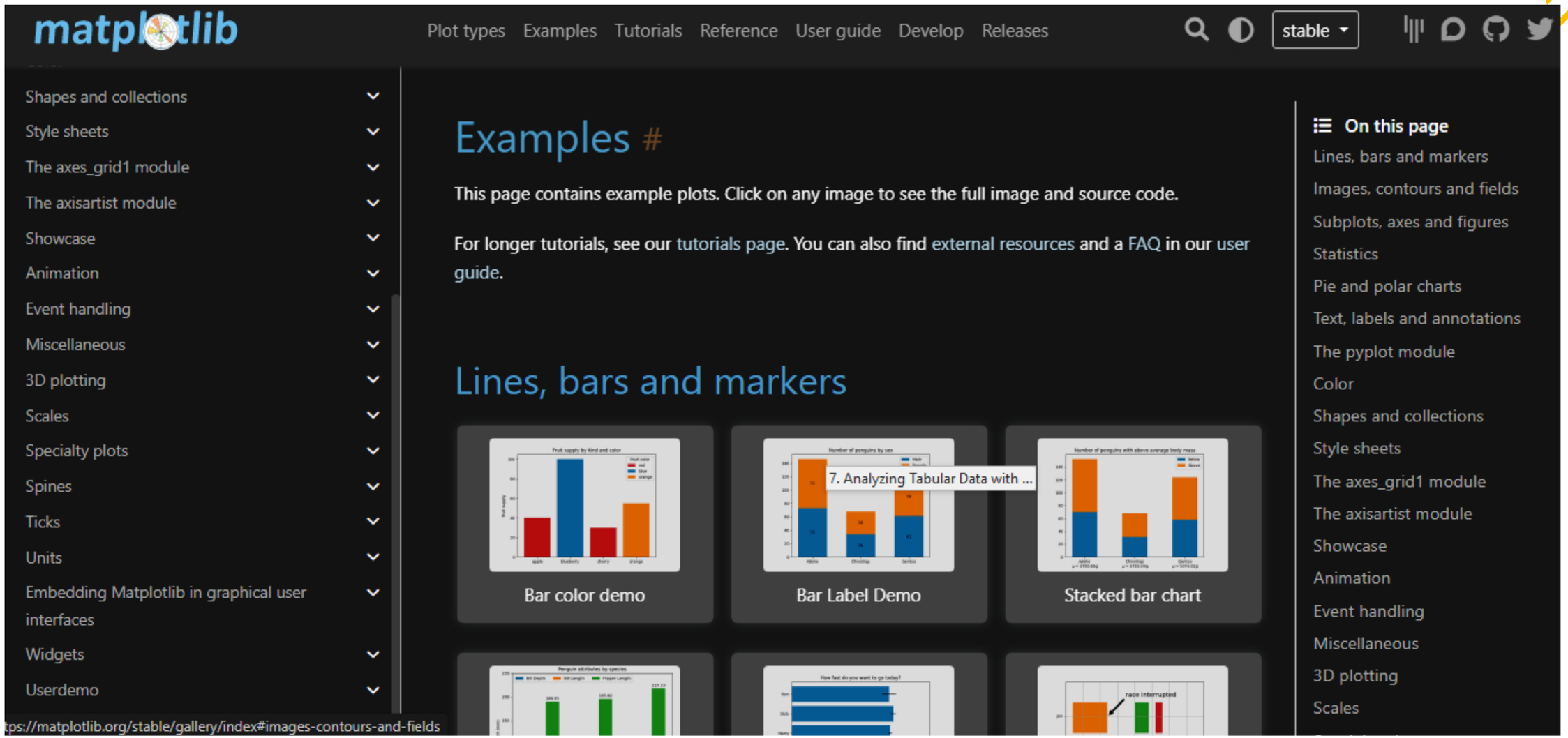








Visualization with Matplotlib and Seaborn



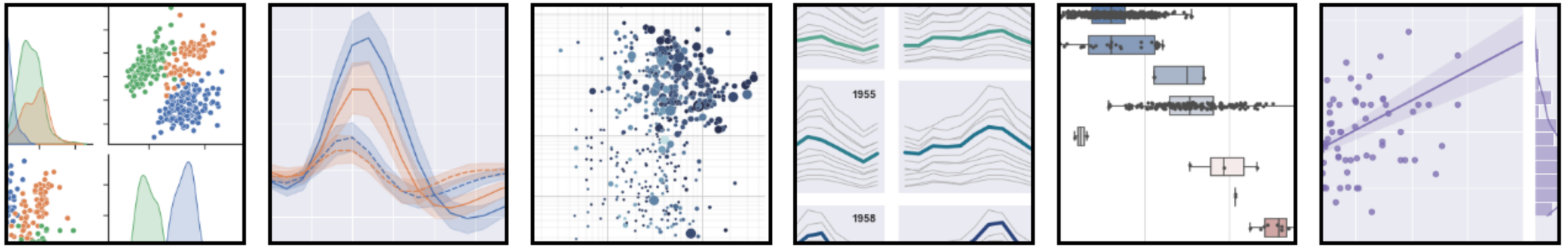
https://matplotlib.org/stable/gallery/index

https://matplotlib.org/cheatsheets/

Visualization with Matplotlib and Seaborn

seaborn Installing Gallery Tutorial API Releases Citing FAQ

seaborn: statistical data visualization



Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

For a brief introduction to the ideas behind the library, you can read the introductory notes or the paper. Visit the installation page to see how you can download the package and get started with it. You can browse the example gallery to see some of the things that you can do with seaborn, and then check out the tutorials or API reference to find out how.

To see the code or report a bug, please visit the GitHub repository. General support questions are most at home on stackoverflow, which has a dedicated channel for seaborn.

Contents

Installing
Gallery
Tutorial
API
Releases
Citing

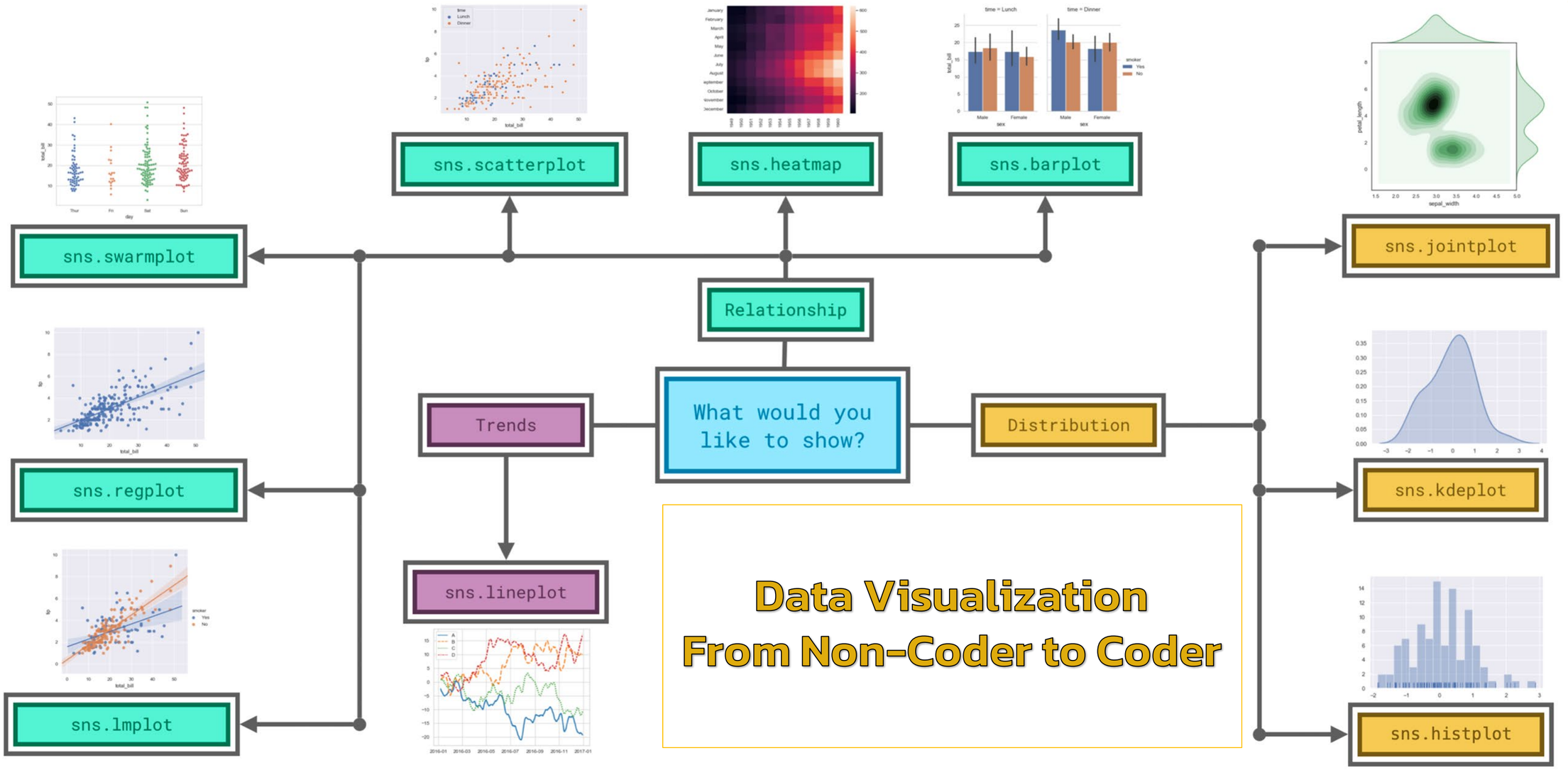
Features

- New Objects: API | Tutorial
- Relational plots: API | Tutorial
- Distribution plots: API | Tutorial
- Categorical plots: API | Tutorial
- Regression plots: API | Tutorial
- Multi-plot grids: API | Tutorial
- Figure theming: API | Tutorial

https://seaborn.pydata.org/

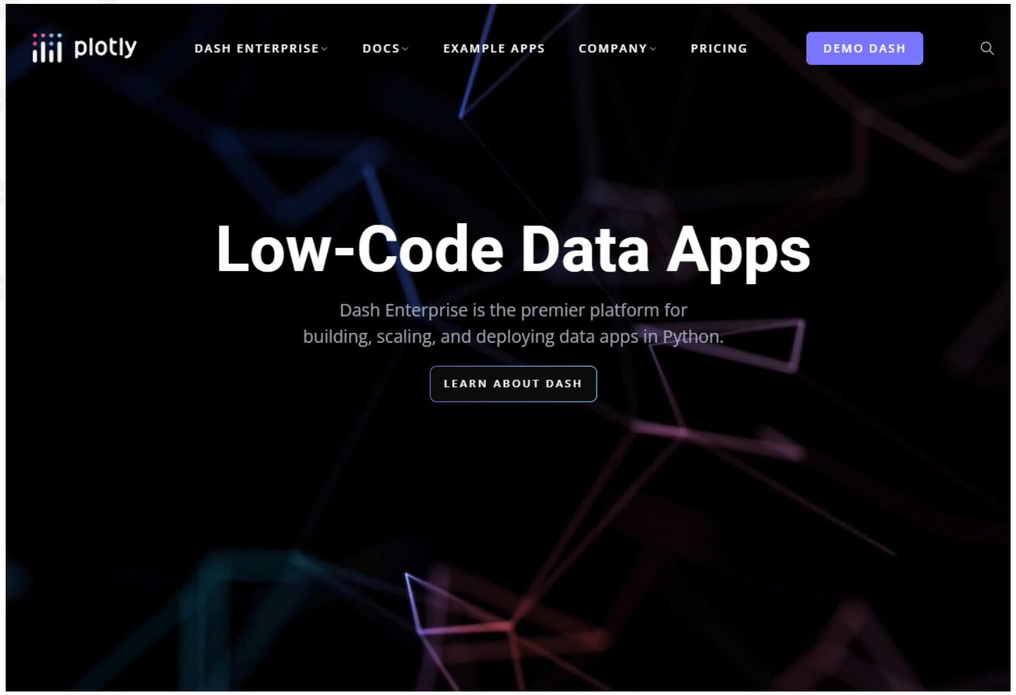
https://seaborn.pydata.org/tutorial/introduction.html

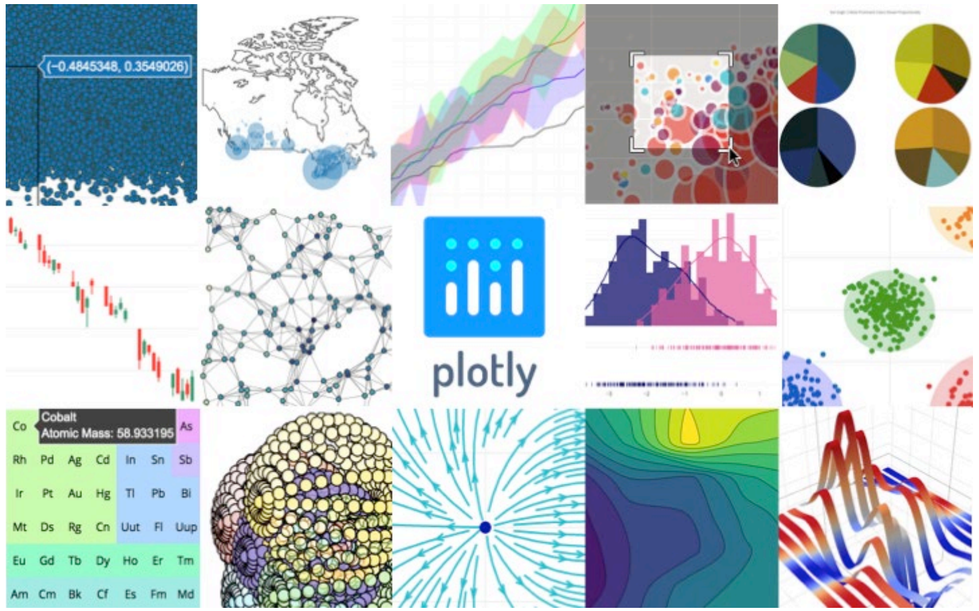
What have you learned?



Data Visualization
From Non-Coder to Coder

Plotly

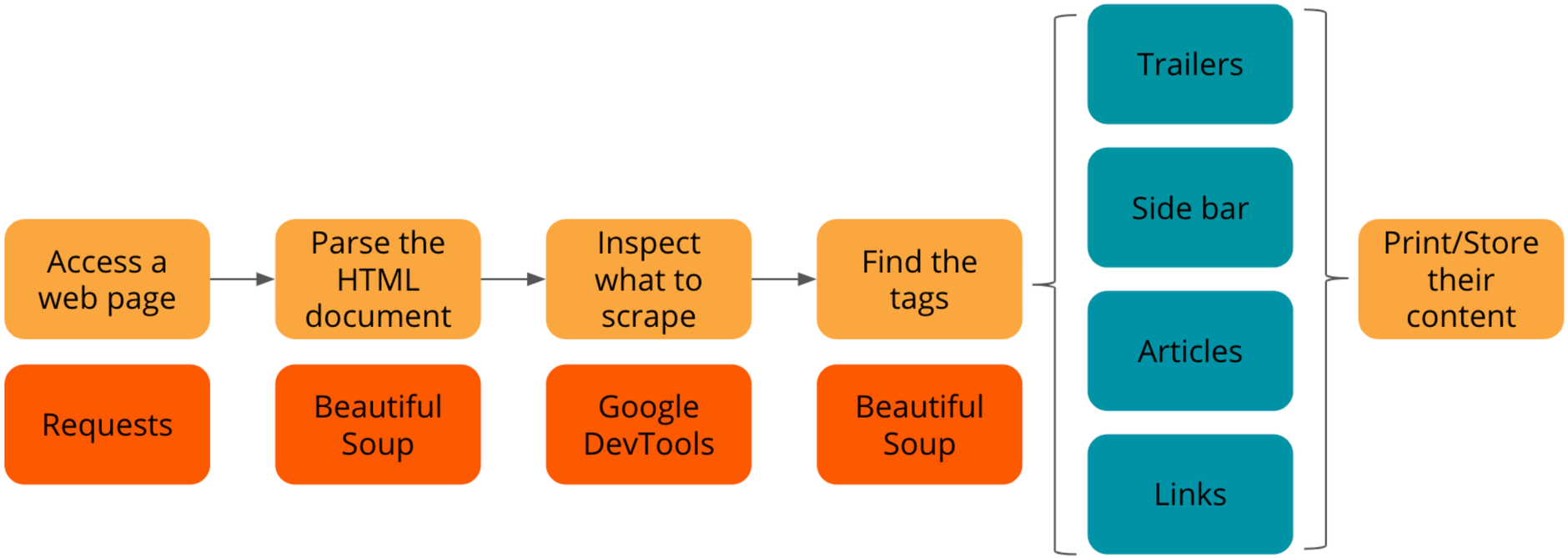




https://plotly.com/

9. Web Scraping with BeautifulSoup

Web Scraping with BeautifulSoup



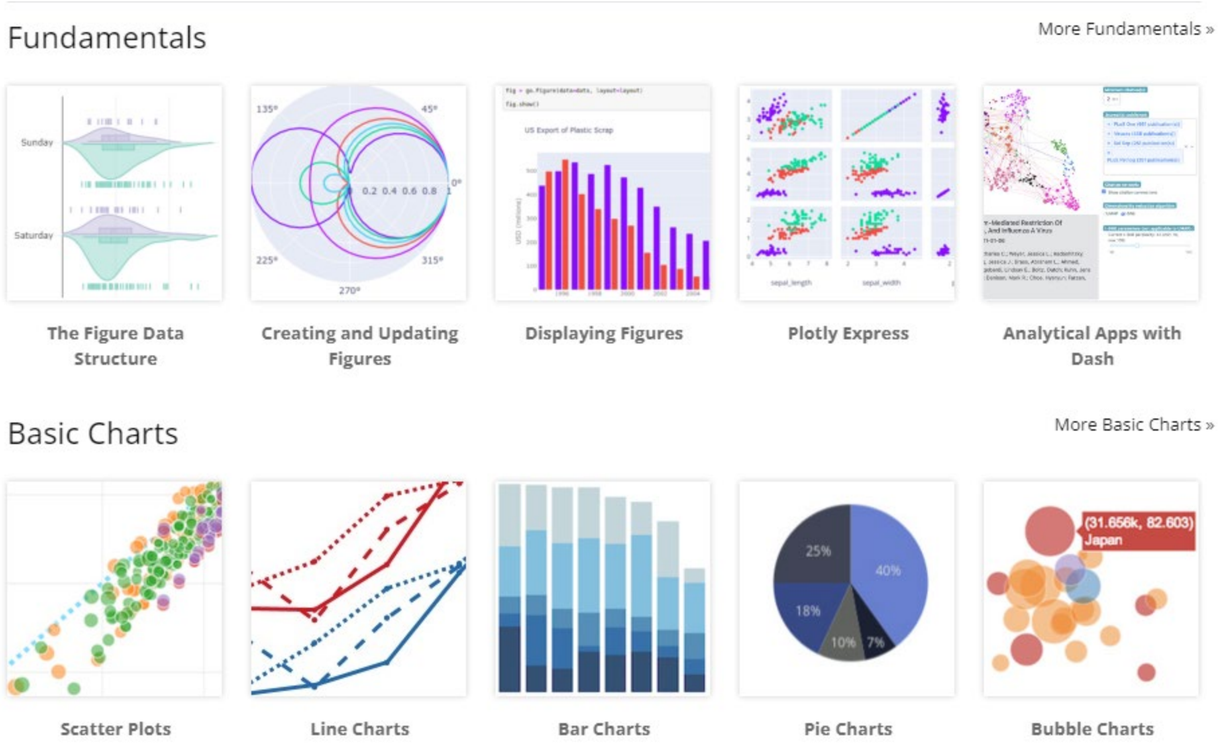
Additional

You can get free certificates

https://www.kaggle.com/learn

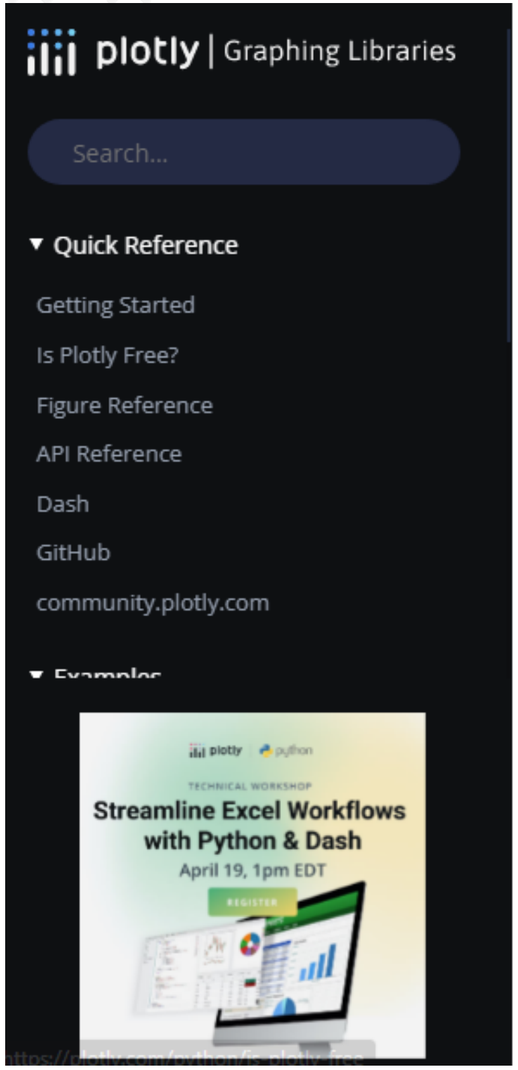
Visualization with Plotly

Visualization with Plotly



https://plotly.com/python/

Visualization with Plotly



Python (v5.14.1) Forum Pricing Dash Star 13,260

Plotly Open Source Graphing Library for Python

Plotly's Python graphing library makes interactive, publication-quality graphs. Examples of how to make line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axes, polar charts, and bubble charts.

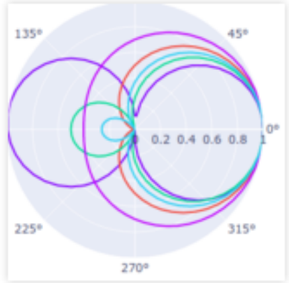
Plotly.py is free and open source and you can view the source, report issues or contribute on GitHub.

Deploy Python AI Dash apps on private Kubernetes clusters: Pricing | Demo | Overview | AI App Services

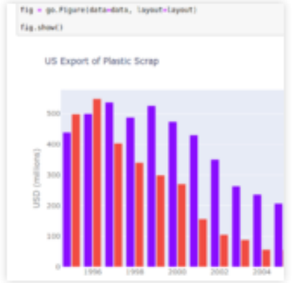
Fundamentals

More Fundamentals »

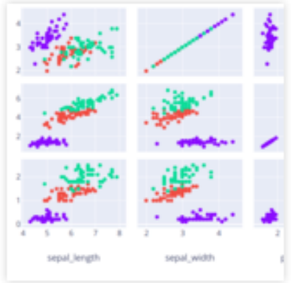
The Figure Data



Creating and Updating



Displaying Figures



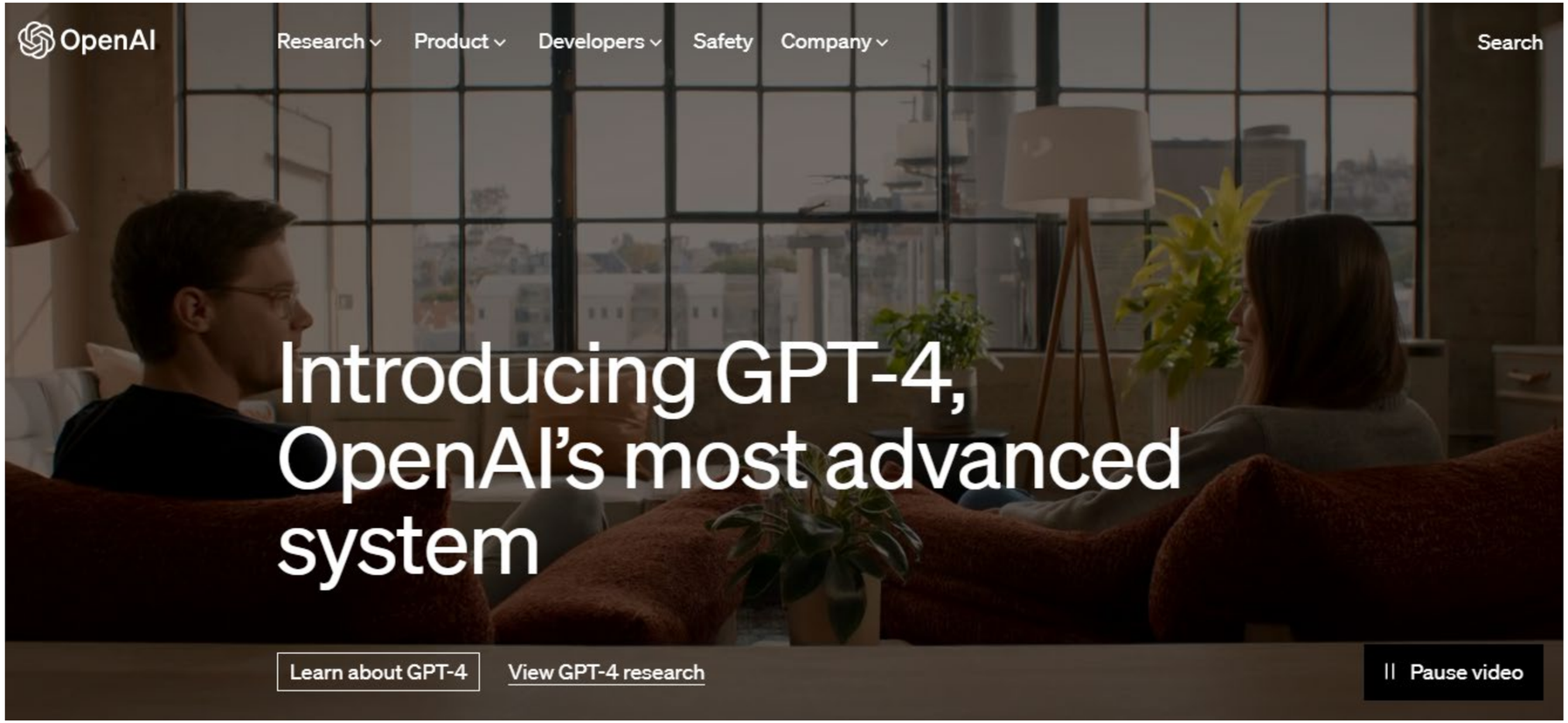
Plotly Express

Analytical Apps with

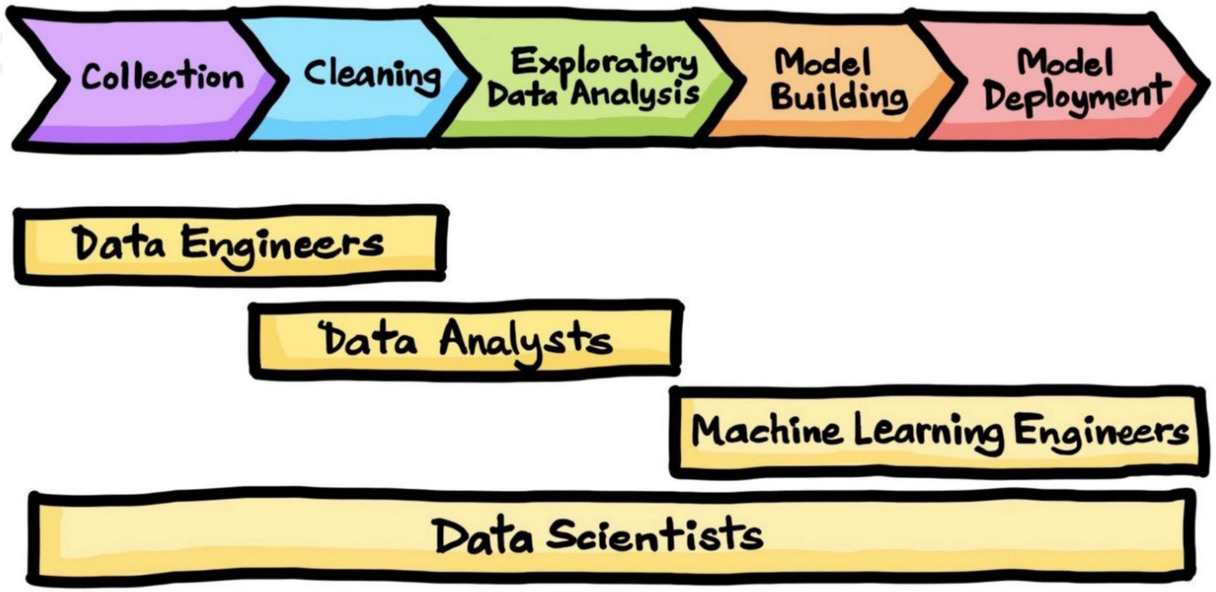
https://plotly.com/python/

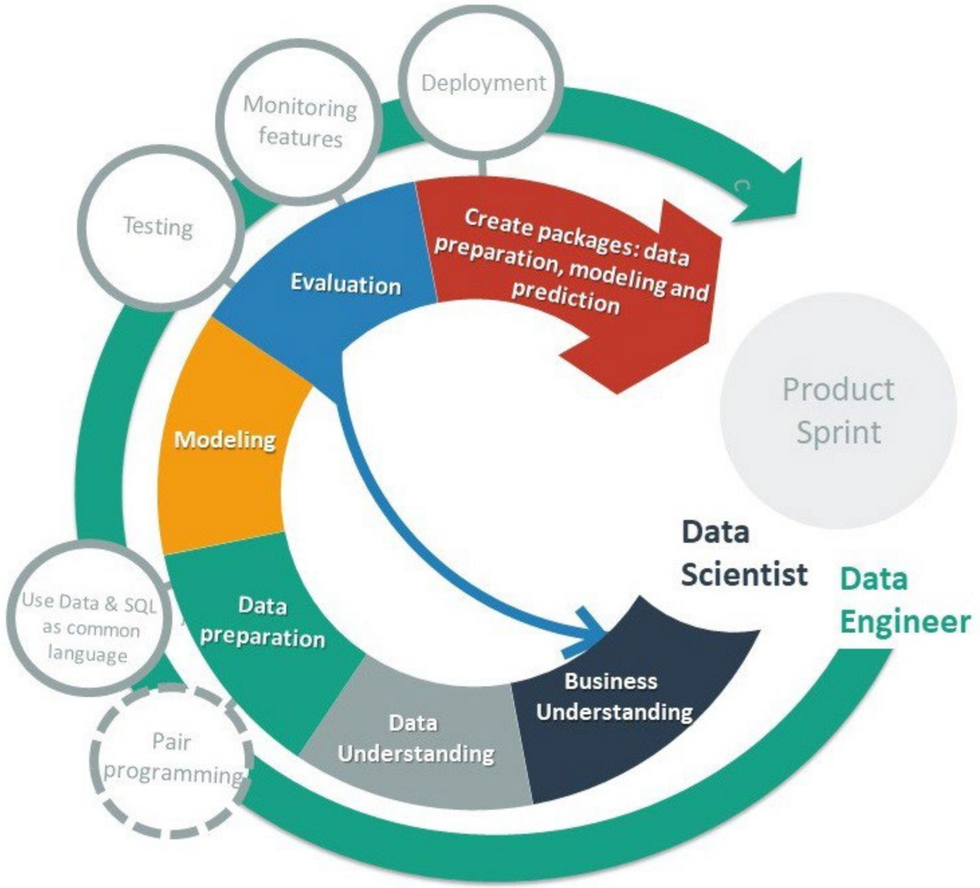
ChatGPT

- https://openai.com/research/overview
- https://openai.com/blog/chatgpt
- https://chat.openai.com/



Data Science Life Cycle





Credit: [The Data Science Process. A Visual Guide to Standard Procedures... | by Chanin Nantasenamat | Towards Data Science]

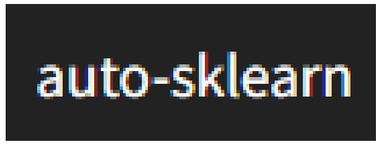
Credit: https://miro.medium.com/max/2400/1*-8sSf4po2pm-fzhBU7CUmQ.jpeg

Auto Model

https://pycaret.org/tutorial/

https://autokeras.com/

https://docs.fast.ai/tutorial.html



https://automl.github.io/auto-sklearn/master/

https://auto.gluon.ai/stable/index.html

https://optuna.readthedocs.io/en/stable/tutorial/index.html

Reference

- https://github.com/Mckiler
- https://medium.com/@mcmckiler

- https://www.projectpro.io/article/python-data-visualization-libraries/543

- https://www.simplilearn.com/top-python-libraries-for-data-science-article

- Learning How to Learn

Solution

- https://colab.research.google.com/drive/1L58uEGnZhSd2WU6jtK9_S3Hgaf3GkNG4?usp=sharing


Data Mommie