

Python API

— Tanya Sattaya-aphiton, Ph.D. —

Oct 6-7, 2020 @การยางแห่งประเทศไทย



Slide URL : <https://tinyurl.com/RAOT-API>



A little bit about me !

Education

- Doctor of Engineering (Software System Engineering, KMUTNB)
- Research works at RWTH University Aachen, Germany

Professional Certificates

- Data Engineering Professional Certification (RapidMiner)
- Machine Learning Professional Certification (RapidMiner)
- Applications & Use Cases Professional Certification (RapidMiner)
- Sun Certified Java Programmer (SCJP)

Work Experiences

- Database : Oracle, MySQL, Postgres, Elasticsearch
- Backend Developer: Java, Python, PHP, NodeJS
- Frontend Dev. : Vue.js
- Data Visualization : Tableau
- Smart Solutions : Video Analytics, IoTs, Connectivity (LoRaWan/NB-IoT) , AI/ML



Prof. Dr. Horst Licher, RWTH Aachen, Germany



Babysitter



Marathoner



Switzerland



Agenda

- Day 1

- Overview & Installation
- Basic Python
- Call API :
 - BOT: Exchange Rate
 - AI : AI for Thai

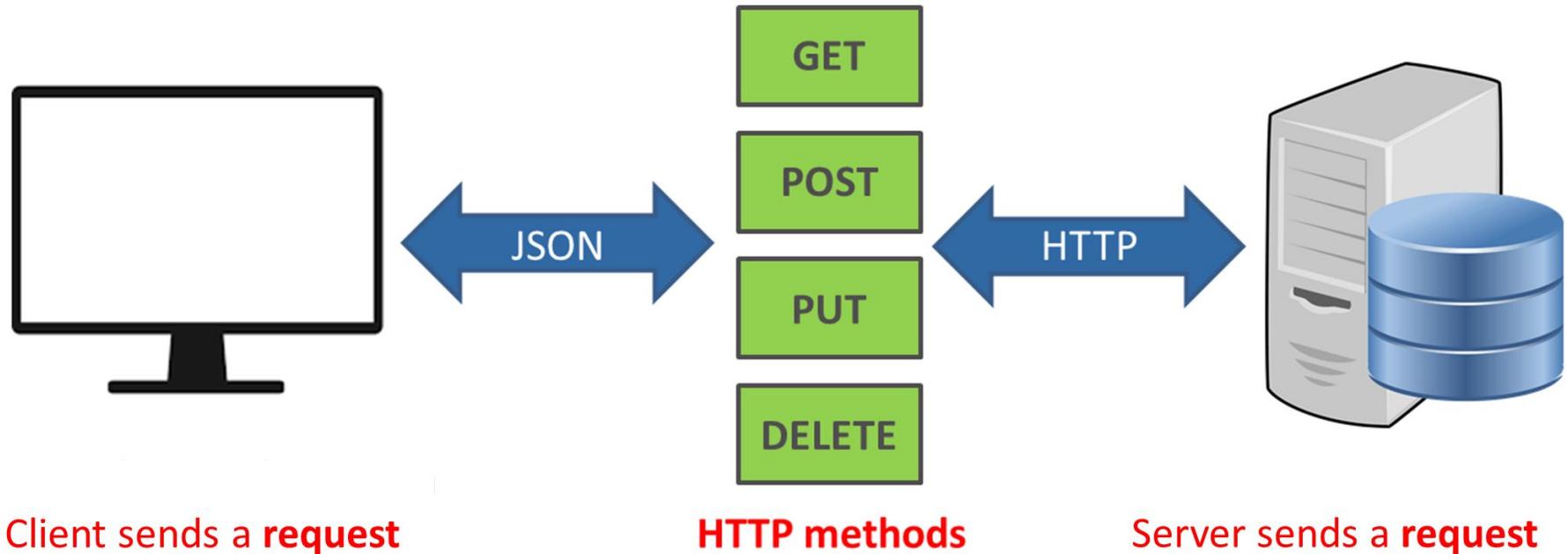
- Day 2

- Working with Database : MySQL
- Create Your First API
- Create API with Django framework



API Overview

API = Application Program Interface



API vs Web Service

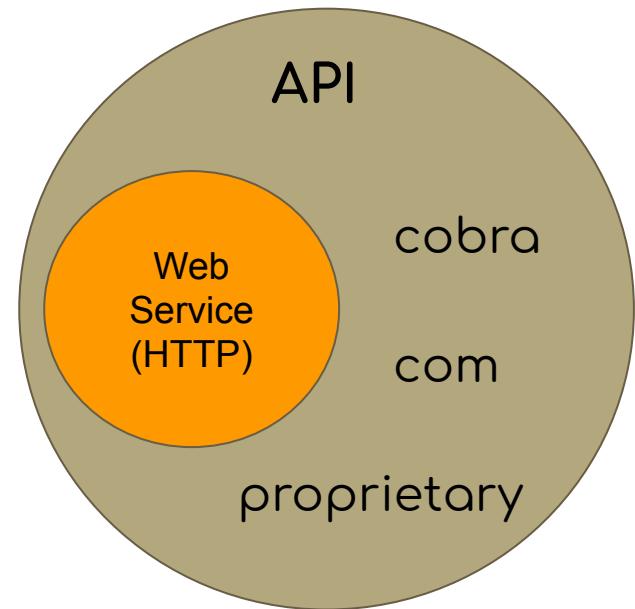
API = Application Program Interface

Web Service = API (operate over HTTP)

All Web Services = API

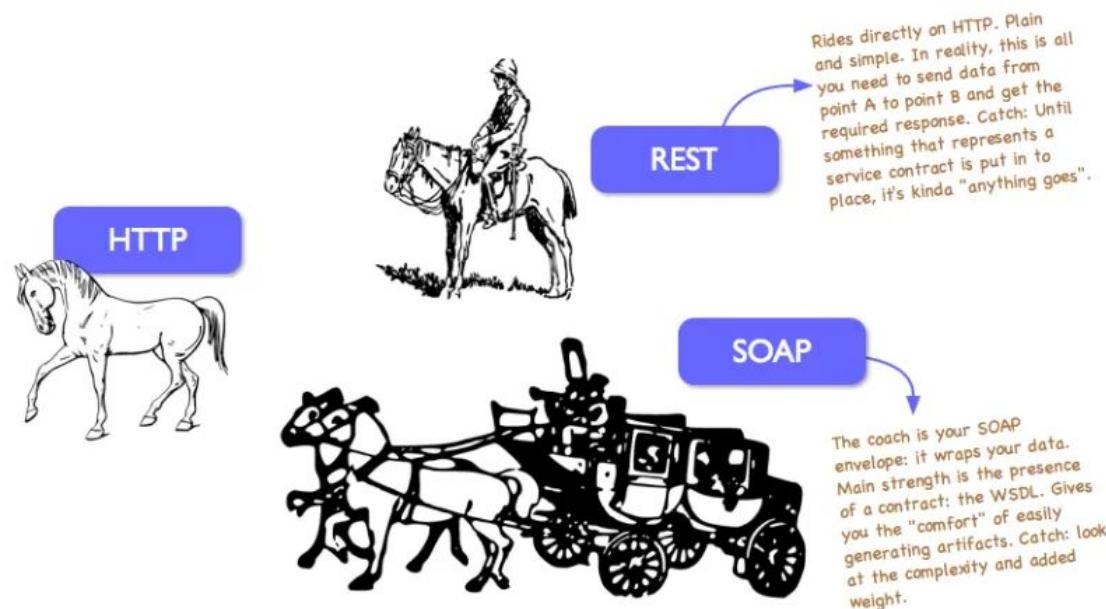
In this course !

API = Web Service



SOAP & REST

- SOAP (Simple Object Access P- REST (REpresentational State Transfer) - Data



More Information <https://rapidapi.com/blog/types-of-apis/>



SOAP

XML

```
<empinfo>
  <employees>
    <employee>
      <name>James Kirk</name>
      <age>40</age>
    </employee>
    <employee>
      <name>Jean-Luc Picard</name>
      <age>45</age>
    </employee>
    <employee>
      <name>Wesley Crusher</name>
      <age>27</age>
    </employee>
  </employees>
</empinfo>
```

REST

JSON

```
{ "empinfo" :
  {
    "employees": [
      {
        "name": "James Kirk",
        "age": 40,
      },
      {
        "name": "Jean-Luc Picard",
        "age": 45,
      },
      {
        "name": "Wesley Crusher",
        "age": 27,
      }
    ]
  }
}
```



Programming Language

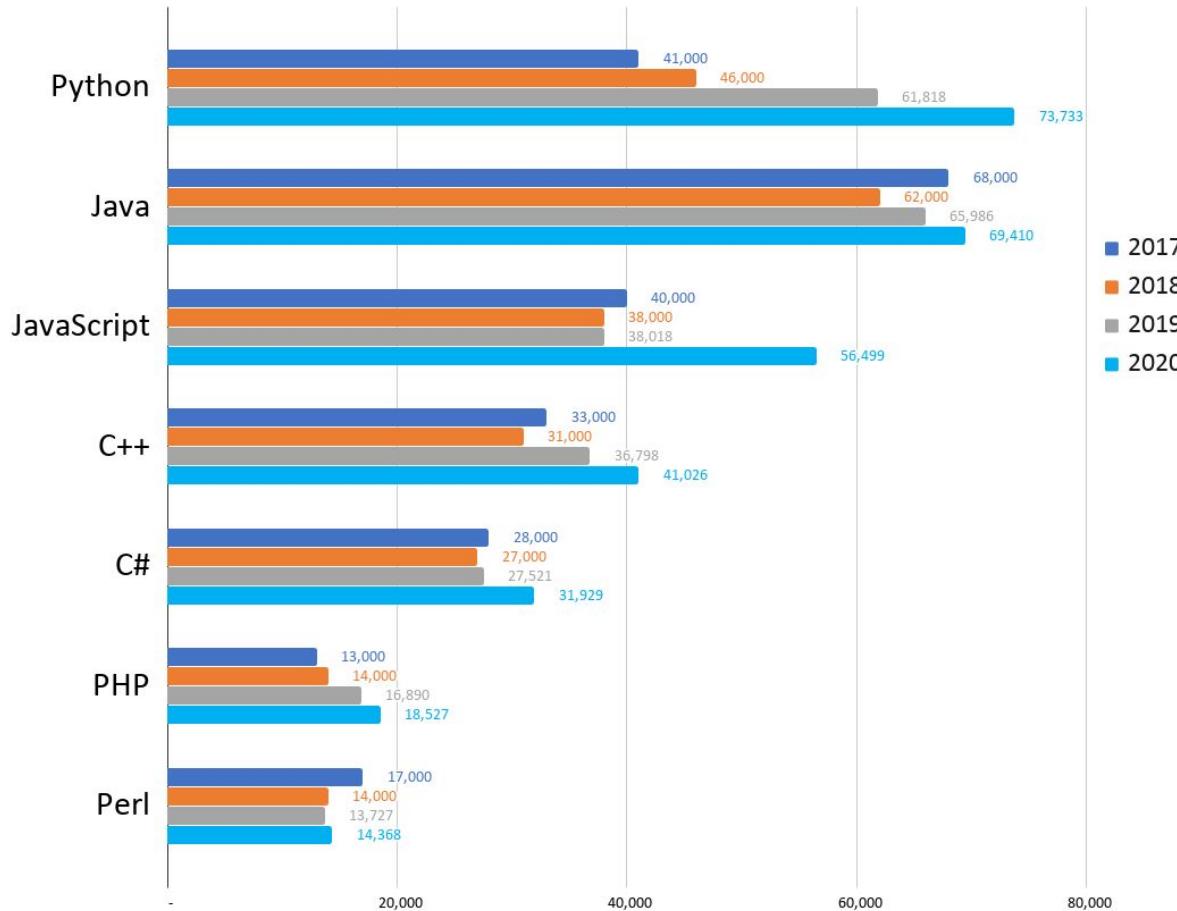
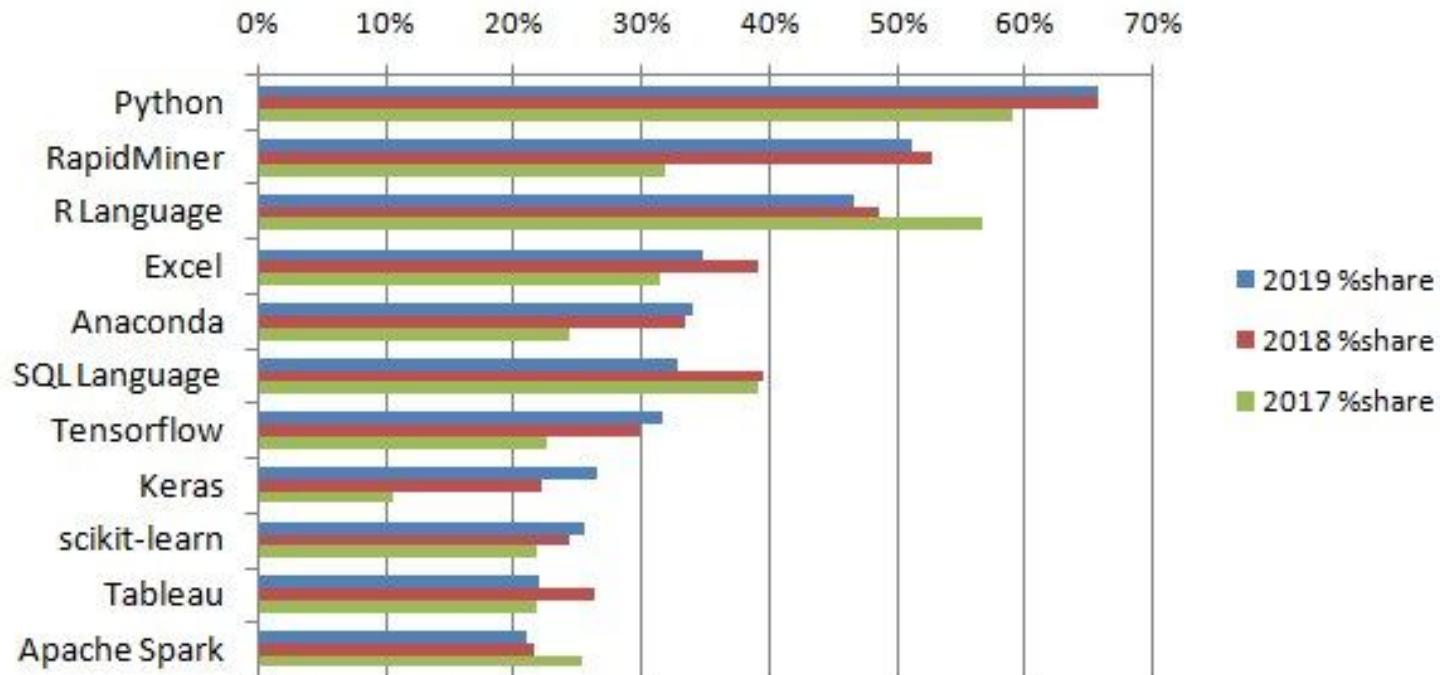


Image Source <https://www.codingdojo.com/blog/top-7-programming-languages-of-2020>



Data Science Language (Machine Learning / AI)

Top Analytics, Data Science, Machine Learning Software 2017-2019, KDnuggets Poll



Features of



Python

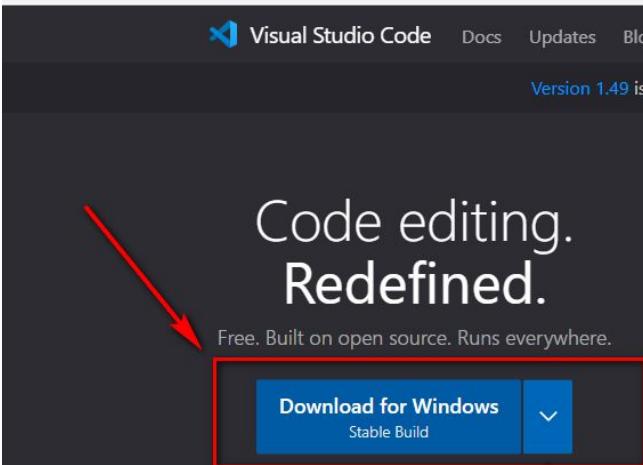
- | | | | |
|----------------------|----|------------------------|----|
| Easy | 01 | Extensible | 07 |
| Expressive | 02 | Embeddable | 08 |
| Free and Open Source | 03 | Interpreted | 09 |
| High-Level | 04 | Large Standard Library | 10 |
| Portable | 05 | GUI Programming | 11 |
| Object Oriented | 06 | Dynamically Typed | 12 |



Installation

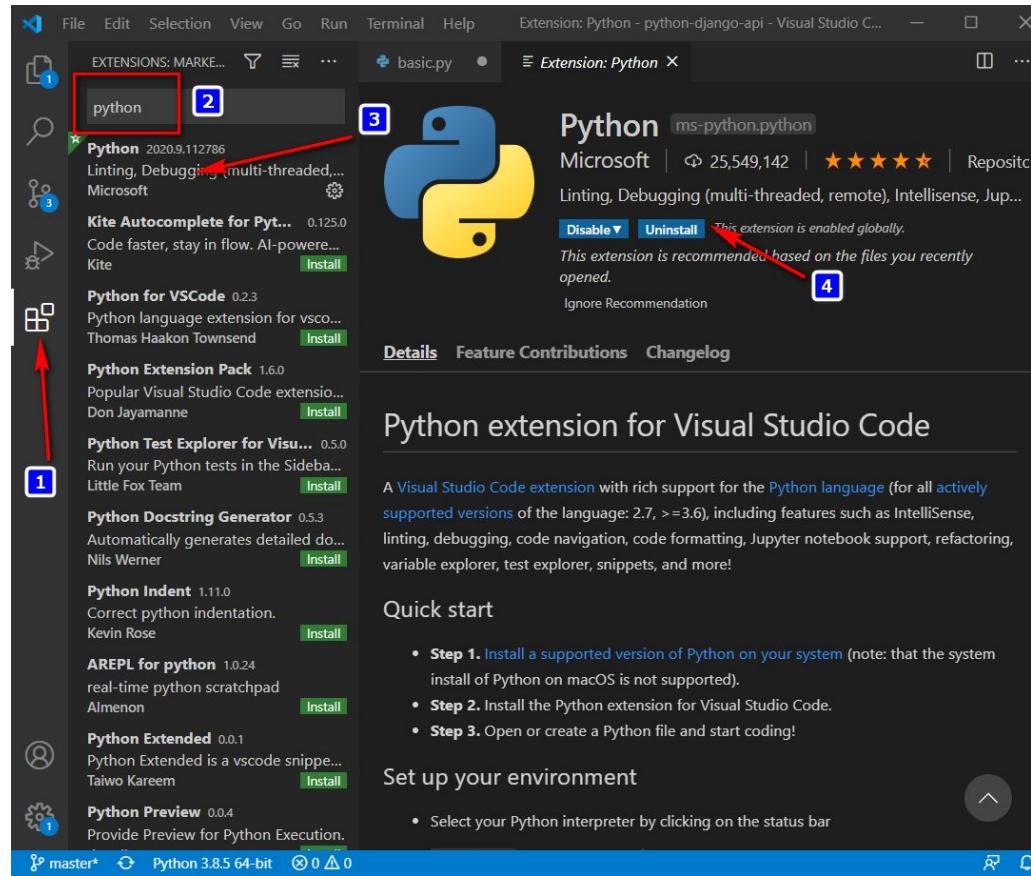
IDE: Visual Studio Code

https://code.visualstudio.com



The screenshot shows the official Visual Studio Code website. A large red arrow points to the "Download for Windows" button, which is highlighted with a red border. Below the button, there's a "Stable Build" link. To the right of the button, there's a "Download" button with a blue gradient background. The page features a dark header with links for Visual Studio Code, Docs, Updates, Blog, API, Extensions, and FAQ. A search bar and a "Download" button are also in the header. The main content area has a dark background with white text. It features the slogan "Code editing. Redefined." and a subtext "Free. Built on open source. Runs everywhere." Below this, there are two buttons: "Download for Windows" and "Other platforms and Insiders Edition". A note below the buttons states: "By using VS Code, you agree to its license and privacy statement." On the right side of the screenshot, a separate window of Visual Studio Code is shown. This window displays the code editor with several files open: "serviceWorker.js", "create-react-app", "Visual Studio Code - In...", and "index.js". The code editor shows a snippet of JavaScript related to service workers. The status bar at the bottom of the VS Code window indicates "Ln 43, Col 19, Spaces: 2, UTF-8, LF, JavaScript". A QR code is located in the bottom right corner of the screenshot.

IDE : Plugin python



Python

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

The screenshot shows the Python Downloads page. At the top, there's a navigation bar with links for Python, PSF, Docs, PyPI, Jobs, and Community. Below the navigation is the Python logo. A red arrow points to the "Download Python 3.8.6" button, which is highlighted with a red border. To the right of the button is a graphic of two parachutes descending from the sky, each carrying a large wooden crate. The page also features links for other OS versions and Docker images, and a table of active Python releases.

python™

Donate Search GO Socialize

About Downloads Documentation Community Success Stories News Events

Download the latest version for Windows

Download Python 3.8.6

Looking for Python with a different OS? Python for [Windows](#), [Linux/UNIX](#), [Mac OS X](#), [Other](#)

Want to help test development versions of Python? [Prereleases](#), [Docker images](#)

Looking for Python 2.7? See below for specific releases

Active Python Releases

For more information visit the Python Developer's Guide.

Python version	Maintenance status	First released	End of support	Release schedule
3.8	bugfix	2019-10-14	2024-10	PEP 569
3.7	security	2018-06-27	2023-06-27	PEP 537
3.6	security	2016-12-23	2021-12-23	PEP 494
3.5	security	2015-09-13	2020-09-13	PEP 478
2.7	end-of-life	2010-07-03	2020-01-01	PEP 373

```
python --version
```

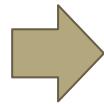
```
Python 3.8.6
```



Database : MySQL Community Server

<https://dev.mysql.com/downloads/mysql/>

The screenshot shows the MySQL Community Downloads page. At the top, there are tabs for "General Availability (GA) Releases" (which is selected), "Archives", and a help icon. Below the tabs, the title "MySQL Community Server 8.0.21" is displayed. A dropdown menu labeled "Select Operating System:" has "Microsoft Windows" selected and is highlighted with a red box. To the right of the dropdown is a link "Looking for previous GA versions?". Below the dropdown, under "Recommended Download:", there is a section for "MySQL Installer for Windows" featuring a thumbnail image of a Windows logo and a link "Go to Download Page >". A blue box with the number "1" is placed near the "Download" button in the second screenshot.



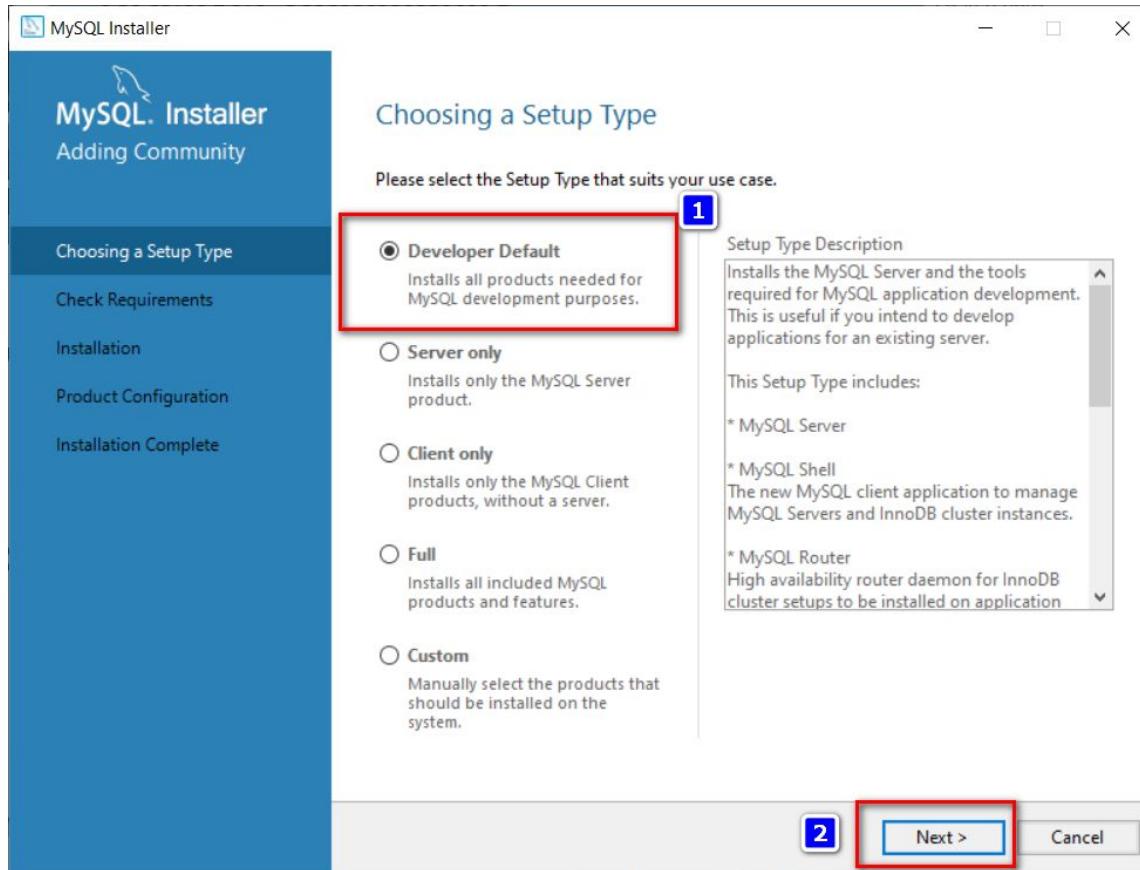
The screenshot shows the MySQL Community Downloads page for MySQL Installer 8.0.21. At the top, there are tabs for "General Availability (GA) Releases" (selected), "Archives", and a help icon. Below the tabs, the title "MySQL Installer 8.0.21" is displayed. A dropdown menu labeled "Select Operating System:" has "Microsoft Windows" selected. To the right is a link "Looking for previous GA versions?". The main area lists two download options:

Version	Type	Size	Download	Signature
8.0.21	Windows (x86, 32-bit), MSI Installer	24.5M	Download	Signature
8.0.21	Windows (x86, 32-bit), MSI Installer	427.6M	Download	Signature

A blue box with the number "2" is placed near the "Download" button for the 24.5M file. At the bottom, a warning message says: "We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download." A red arrow points to the "Signature" link next to the 24.5M download button.



Install : developer -> next -> next



Use Legacy

 MySQL Installer

MySQL Server 8.0.21

High Availability

Type and Networking

Authentication Method

Accounts and Roles

Windows Service

Apply Configuration

Authentication Method

Use Strong Password Encryption for Authentication (RECOMMENDED)
MySQL 8 supports a new authentication based on improved stronger SHA256-based password methods. It is recommended that all new MySQL Server installations use this method going forward.

 Attention: This new authentication plugin on the server side requires new versions of connectors and clients which add support for this new 8.0 default authentication (caching_sha2_password authentication).

Currently MySQL 8.0 Connectors and community drivers which use libmysqlclient 8.0 support this new method. If clients and applications cannot be updated to support this new authentication method, the MySQL 8.0 Server can be configured to use the legacy MySQL Authentication Method below.

Use Legacy Authentication Method (Retain MySQL 5.x Compatibility) 1
Using the old MySQL 5.x legacy authentication method should only be considered in the following cases:

- If applications cannot be updated to use MySQL 8 enabled Connectors and drivers.
- For cases where re-compilation of an existing application is not feasible.
- An updated, language specific connector or driver is not yet available.

Security Guidance: When possible, we highly recommend taking needed steps towards upgrading your applications, libraries, and database servers to the new stronger authentication. This new method will significantly improve your security. 2

< Back Next > Cancel

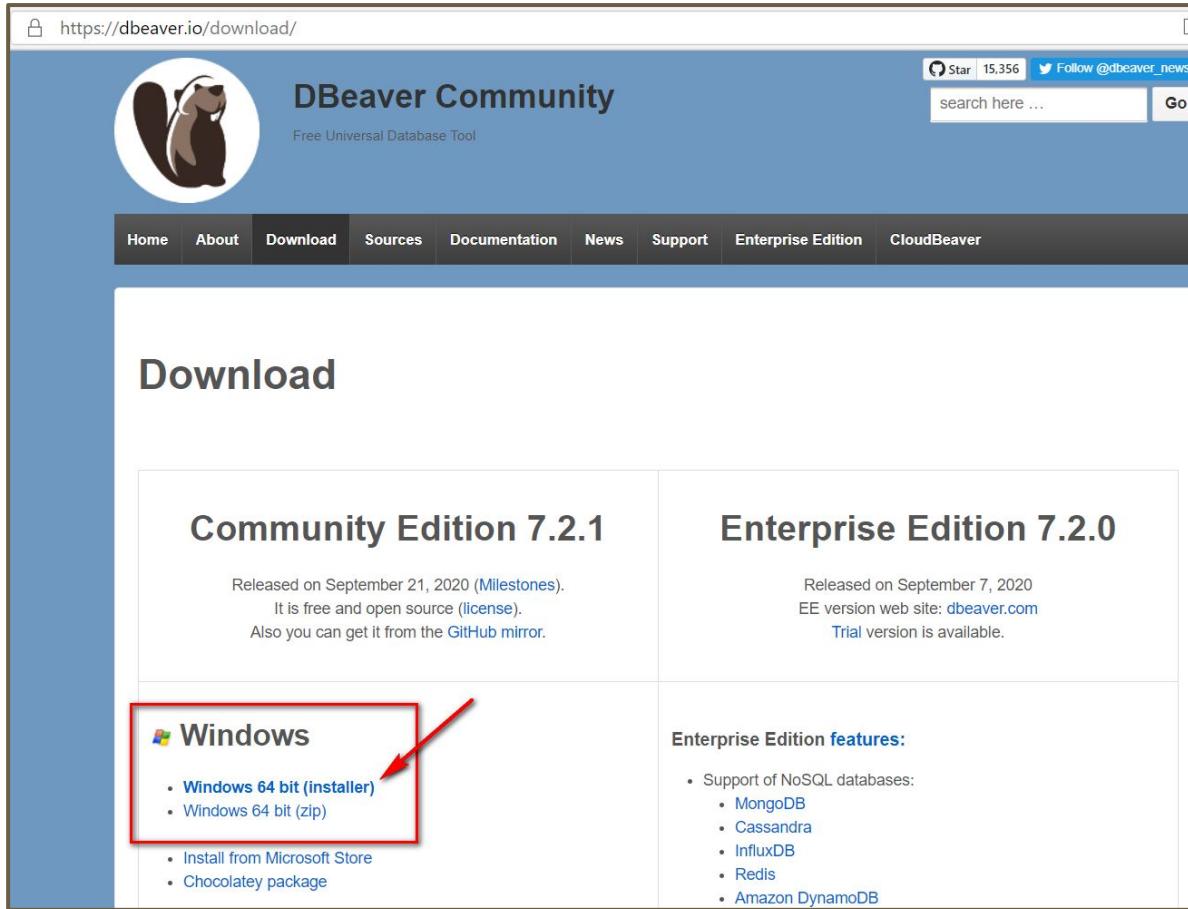


DB GUI : DBeaver Community

The screenshot shows the DBeaver Community website. At the top left is a logo of a squirrel. The title "DBeaver Community" is prominently displayed, with "Free Universal Database Tool" below it. A navigation bar includes links for Home, About, Download, Sources, Documentation, News, Support, Enterprise Edition, and CloudBeaver. On the right, there's a "Star 15,356" button, a "Follow @dbeaver_news" link, a search bar with placeholder "search here ...", and a "Go" button. Below the navigation is a large section titled "Universal Database Tool" with a descriptive paragraph about its capabilities. To the right of this text is a complex UML-style database diagram showing various tables like Address, City, Customer, and Order, with relationships between them. A red arrow points from a red-bordered "Download" button at the bottom left towards the database diagram.



Download & Install



The screenshot shows the DBeaver Community download page at <https://dbeaver.io/download/>. The page features a large logo of a beaver on the left, followed by the text "DBeaver Community" and "Free Universal Database Tool". A navigation bar below includes links for Home, About, Download, Sources, Documentation, News, Support, Enterprise Edition, and CloudBeaver. The main content area is titled "Download" and contains two sections: "Community Edition 7.2.1" and "Enterprise Edition 7.2.0". The "Community Edition" section notes its release on September 21, 2020, and mentions it's free and open source. It also links to GitHub. The "Enterprise Edition" section notes its release on September 7, 2020, and mentions a trial version is available. Below these sections, there's a "Windows" section with a red box around the "Windows 64 bit (installer)" link, which is highlighted with a red arrow. Other options in this section include "Windows 64 bit (zip)", "Install from Microsoft Store", and "Chocolatey package". To the right, under "Enterprise Edition features:", a list includes support for MongoDB, Cassandra, InfluxDB, Redis, and Amazon DynamoDB.

DBeaver Community
Free Universal Database Tool

Home | About | **Download** | Sources | Documentation | News | Support | Enterprise Edition | CloudBeaver

Download

Community Edition 7.2.1

Released on September 21, 2020 ([Milestones](#)).
It is free and open source ([license](#)).
Also you can get it from the [GitHub mirror](#).

Windows

- Windows 64 bit (installer)**
- Windows 64 bit (zip)

• Install from Microsoft Store
• Chocolatey package

Enterprise Edition 7.2.0

Released on September 7, 2020
EE version web site: [dbeaver.com](#)
Trial version is available.

Enterprise Edition features:

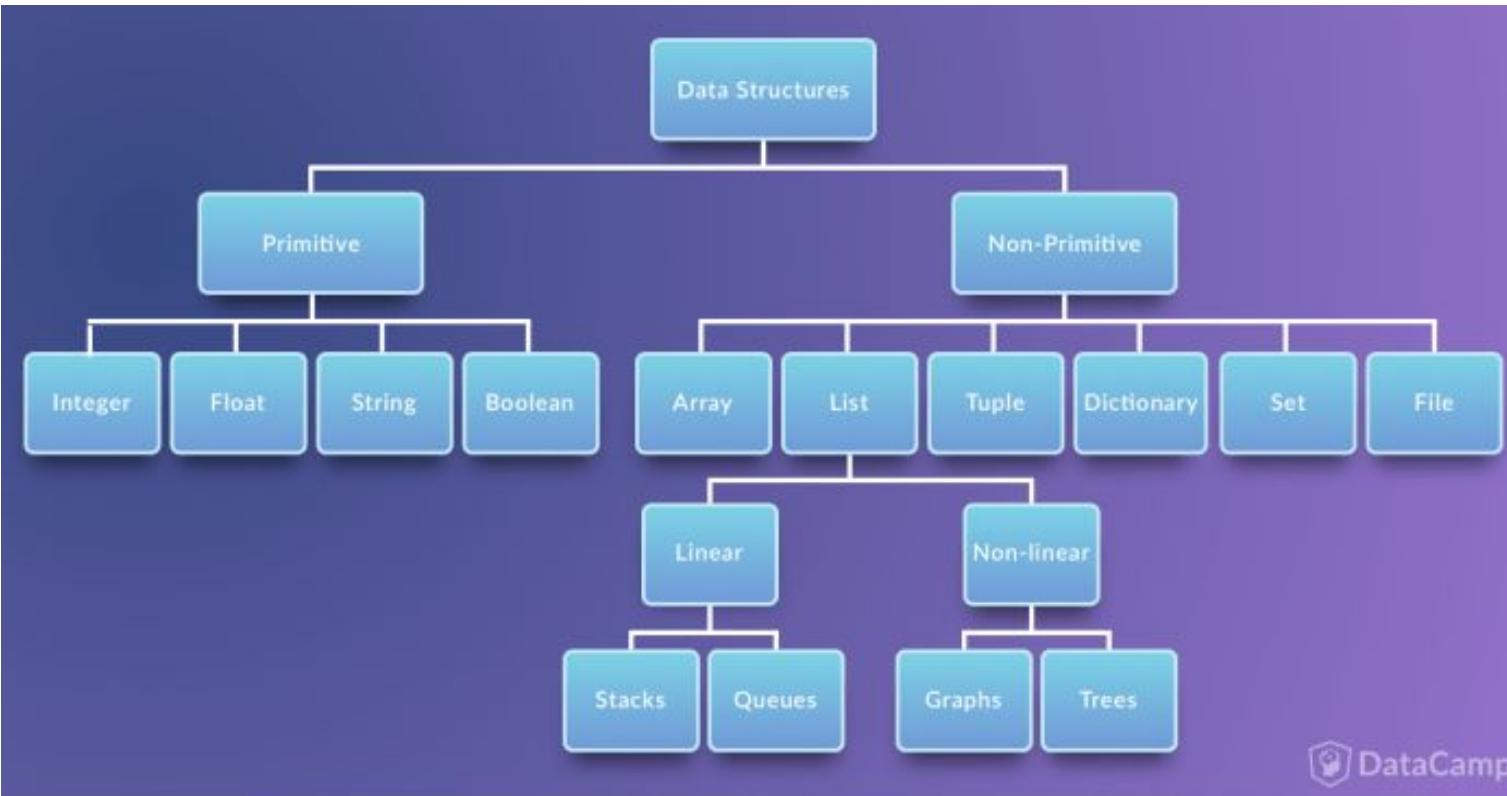
- Support of NoSQL databases:
 - MongoDB
 - Cassandra
 - InfluxDB
 - Redis
 - Amazon DynamoDB



Basic Python

Data Types

Python Variables

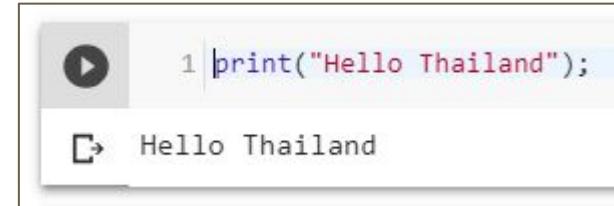


Print

Hello.py

```
print("Hello Thailand");
```

python Hello.py



A screenshot of a terminal window. On the left is a play button icon. To its right is a line number '1' followed by the Python code 'print("Hello Thailand");'. Below the code is the output 'Hello Thailand' preceded by a right-pointing arrow.

```
1 print("Hello Thailand");
   ▶ Hello Thailand
```



Primitive type

```
a = 150
```

```
a= a+20
```

```
print(type(a))
```

```
print (a)
```

```
b = 1.43
```

```
a = a + b
```

```
print(type(a))
```

```
print (a)
```

```
print("a="+str(a)+",b="+str(b))
```

```
[2] 1 a = 150  
2 a= a+20
```

```
[4] 1 print(type(a))  
2 print (a)
```

```
↳ <class 'int'>  
170
```

```
[5] 1 b = 1.43  
2 a = a + b
```

```
[6] 1 print(type(a)).  
2 print (a)
```

```
↳ <class 'float'>  
171.43
```

```
[7] 1 print("a="+str(a)+",b="+str(b)).
```

```
↳ a=171.43,b=1.43
```



Complex Type

```
print(type(5+4j))  
j=4  
print(type(j))
```

```
[ ]    1 print(type(5+4j))  
      □ <class 'complex'>  
  
[ ]    1 j=4  
      2 print(type(j))  
      □ <class 'int'>
```



Complex : Binary Octal Hexa

```
d = 0b1011 #ประกาศตัวเลขฐาน 2  
print(d) ##แสดงค่า d ที่เป็นเลขฐาน 2  
print(bin(11))
```

```
e = 0o13 #ประกาศตัวเลขฐาน 8  
print(e) ##แสดงค่า e ที่เป็นเลขฐาน 8  
print(oct(11))
```

```
f = 0xb #ประกาศตัวเลขฐาน 16  
print(f) ##แสดงค่า f ที่เป็นเลขฐาน 16  
print(hex(11))
```

```
[14] 1 d = 0b1011 #ประกาศตัวเลขฐาน 2  
2 print(d) ##แสดงค่า d ที่เป็นเลขฐาน 2  
3 print(bin(11))
```

```
↳ 11  
0b1011
```

```
[15] 1 e = 0o13 #ประกาศตัวเลขฐาน 8  
2 print(e) ##แสดงค่า e ที่เป็นเลขฐาน 8  
3 print(oct(11))
```

```
↳ 11  
0o13
```

```
▶ 1 f = 0xb #ประกาศตัวเลขฐาน 16  
2 print(f) ##แสดงค่า f ที่เป็นเลขฐาน 16  
3 print(hex(11))
```

```
↳ 11  
0xb
```



Concat String with Int

```
a = 1;  
b = 2;  
c = a+b;  
#Normal string concatenation  
print("sum of", a , "and" , b , "is" , c)
```

```
[18] 1 a = 1;  
2 b = 2;  
3 c = a+b;  
4 #Normal string concatenation  
5 print("sum of", a , "and" , b , "is" , c)
```

```
⇨ sum of 1 and 2 is 3
```



Concat String with Int (cont)

```
#convert variable into str  
print("sum of "+ str(a) + " and " + str(b) + " is " + str(c))  
# if you want to print in tuple way  
print("Sum of %s and %s is %s "%(a,b,c))  
  
print("Sum of {0} and {1} is {2} ".format(a,b,c))
```

```
[19] 1 #convert variable into str  
2 print("sum of "+ str(a) + " and " + str(b) + " is " + str(c))  
3 # if you want to print in tuple way  
4 print("Sum of %s and %s is %s "%(a,b,c))
```

```
↪ sum of 1 and 2 is 3  
Sum of 1 and 2 is 3
```



Concat String with Int (cont)

```
#New style string formatting  
print("sum of {0} and {1} is {2}".format(a,b,c))  
#in case you want to use repr()  
print("sum of "+ repr(a) + " and "+ repr(b) + " is "+ repr(c))
```

```
[20] 1 #New style string formatting  
2 print("sum of {0} and {1} is {2}".format(a,b,c))  
3 #in case you want to use repr()  
4 print("sum of "+ repr(a) + " and "+ repr(b) + " is "+ repr(c))  
  
⇒ sum of 1 and 2 is 3  
sum of 1 and 2 is 3
```



String

```
str1 = 'This is a literal string'#การประกาศตัวแปรแบบ String โดยใช้ Single Quote  
str2 = "This is another string"#การประกาศตัวแปรแบบ String โดยใช้ Double Quote  
#การประกาศตัวแปรแบบ String โดยใช้ Double Quote แบบมี Single Quote ในข้อความ  
str3 = "I'm a teacher"  
#การประกาศตัวแปรแบบ String โดยใช้ Single Quote แบบมี Single Quote ในข้อความ  
str4 = 'I don\'t like VB'  
#การประกาศตัวแปรแบบ String โดยใช้ Double Quote โดยมีการขึ้นบรรทัดใหม่ ในข้อความตอนประกาศตัวแปร  
str5 = "This is an example of \  
two lines string"
```

```
[21] 1 str1 = 'This is a literal string'#การประกาศตัวแปรแบบ String โดยใช้ Single Quote  
2 str2 = "This is another string"#การประกาศตัวแปรแบบ String โดยใช้ Double Quote  
3 #การประกาศตัวแปรแบบ String โดยใช้ Double Quote แบบมี Single Quote ในข้อความ  
4 str3 = "I'm a teacher"  
5 #การประกาศตัวแปรแบบ String โดยใช้ Single Quote แบบมี Single Quote ในข้อความ  
6 str4 = 'I don\'t like VB'  
7 #การประกาศตัวแปรแบบ String โดยใช้ Double Quote โดยมีการขึ้นบรรทัดใหม่ ในข้อความตอนประกาศตัวแปร  
8 str5 = "This is an example of \  
9 two lines string"
```

```
[22] 1 print(str1)  
2 print(str2)  
3 print(str3)  
4 print(str4)  
5 print(str5).
```

```
⇨ This is a literal string  
This is another string  
I'm a teacher  
I don't like VB  
This is an example of two lines string
```



List (Array)

Index	0	1	2	3	4
Value	banana	papaya	orange	apple	mango

```
fruits = ["banana", "papaya",
          "orange", "apple", "mango"]
```

```
print(fruits)
```

```
fruits[0] = "test"
```

```
print(fruits[0])
```

```
print(fruits[1])
```

```
print(fruits[-1])
```

```
print(fruits[-2])
```

```
print(fruits)
```

```
#add data to List
```

```
fruits.append('pen')
```

```
print(fruits)
```

```
['banana', 'papaya', 'orange', 'apple', 'mango']
```

```
test
```

```
papaya
```

```
mango
```

```
apple
```

```
['test', 'papaya', 'orange', 'apple', 'mango']
```



Functions

```
a = " Hello Thailand "
b = a.strip() #ตัด White Space
c = a.count('a') #นับว่ามี a กี่ตัวใน 1 คำ
#slicing index เริ่มที่ 0
e = a[0:10] #เป็นการ slice ตั้งแต่ 0-9 แต่ไม่รวม 10 (exclusion)
```

```
[41] 1 print(a)
      2 print(b)
      3 print(c)
      4 print(d)
```

```
>Hello Thailand
Hello Thailand
2
Hello Tha
```



Statistic Functions

```
!pip install statistics  
import statistics  
l = [15, 18, 2, 36, 12, 78, 5, 6, 9]  
print("Sum of l is : "+ str(sum(l)))  
print("Length of l is : "+ str(len(l)))  
print("Max of l is : "+ str(max(l)))  
print("Min of l is : "+ str(min(l)))  
print("AVG of l is : "+ str(sum(l) / float(len(l))))
```



```
Sum of l is : 181  
Lengthof l is : 9  
Max of l is : 78  
Min of l is : 2  
AVG of l is : 20.11111111111111
```



Statistic Functions

```
print("ค่าเฉลี่ย (Mean) = "+ str(statistics.mean(l)))  
print("ค่าเบี่ยงเบนมาตรฐาน (SD) = "+ str(statistics.stdev(l)))  
print("ค่าแปรปรวน (Variance) = "+ str(statistics.variance(l)))  
print("ค่าเฉลี่ยของประชากร (Population mean) = "+ str(statistics.pstdev(l)))  
print("ส่วนเบี่ยงเบนมาตรฐานของประชากร (Zigma)= "+ str(statistics.pvariance(l)))
```



```
ค่าเฉลี่ย (Mean) = 20.11111111111111  
ค่าเบี่ยงเบนมาตรฐาน (SD) = 23.924069702103594  
ค่าแปรปรวน (Variance) = 572.361111111111  
ค่าเฉลี่ยของประชากร (Population mean) = 22.555829226582766  
ส่วนเบี่ยงเบนมาตรฐานของประชากร (Zigma)= 508.76543209876536
```



Tuple -- data cannot be changed

```
animals = ("monkey", "rabbit", "cat",
           "kangaroo", "chicken")
print(animals[0])
print(animals[1])
print(animals[-1])
print(animals[-3])
animals[0] = "test" #cannot change
```



```
monkey
rabbit
chicken
cat
-----
TypeError                                     Traceback (most recent call last)
<ipython-input-54-bcf381ab8cec> in <module>()
      5 print(animals[-1])
      6 print(animals[-3])
----> 7 animals[0] = "test"

TypeError: 'tuple' object does not support item assignment
```

[SEARCH STACK OVERFLOW](#)



Dictionary

```
data = {  
    'day':['1/1/2018','2/1/2018','3/1/2018'],  
    'temp':[20,10,15],  
    'event':['cold','cold','cold']  
} #สร้างข้อมูล  
  
print(data['day'])  
  
#เพิ่มข้อมูล  
data['x']=[1,2,3]  
  
print(data)
```



Data Frame

```
import pandas as pd #Import pandas
data = {
'day':['1/1/2018','2/1/2018','3/1/2018'],
'temp':[20,10,15],
'event':['cold','cold','cold']
} #สร้างข้อมูล
df= pd.DataFrame(data)
print(df.shape) #ดูขนาดข้อมูล
print(df) #แสดงข้อมูลที่อยู่ใน DataFrame
print(df['temp'].max())
```

df.day
df['day']



	day	event	temp
0	1/1/2018	cold	20
1	2/1/2018	cold	10
2	3/1/2018	cold	15

(3, 3)

	day	event	temp
0	1/1/2018	cold	20
1	2/1/2018	cold	10
2	3/1/2018	cold	15
20			

0	1/1/2018
1	2/1/2018
2	3/1/2018
Name: day, dtype: object	



Data Frame

```
print(type(data))
print(type(data['temp']))
print(type(df))
print(type(df.temp))
```

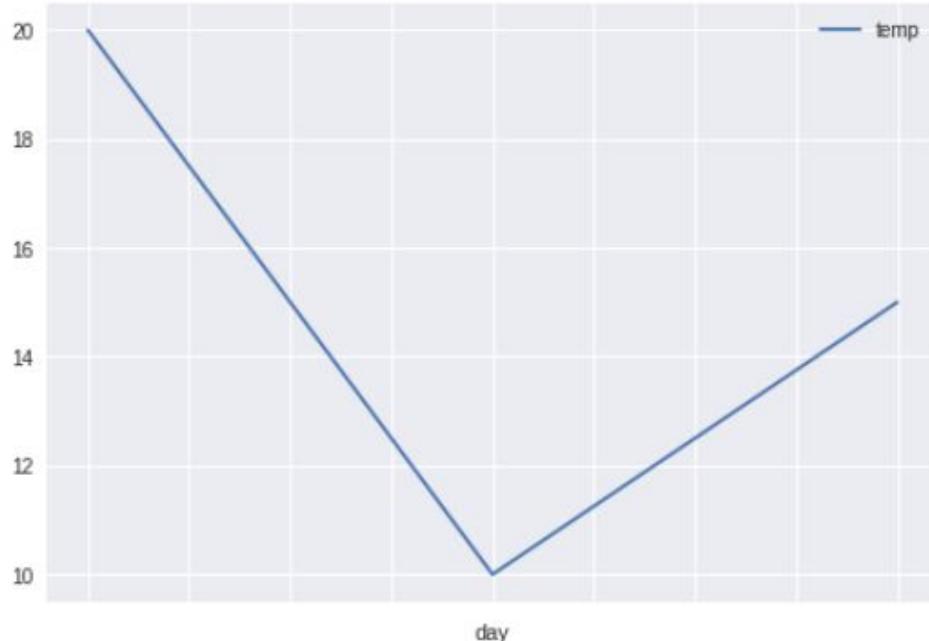
```
<class 'dict'>
<class 'list'>
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.series.Series'>
```



Plot graph (change index)

```
df.set_index('day',inplace=True)  
df.sort_index()  
df.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f07b80c1048>
```



Dictionary

```
name={'Dad':'Somchai','Mom':'Somsri','Bro':'John'} #สร้างข้อมูลเก็บไว้ใน Dictionary  
print(name) #แสดงข้อมูลเก็บไว้ใน Dictionary ที่มีชื่อว่า name  
print(name['Dad']) #แสดงข้อมูลเก็บไว้ใน Dictionary ที่มีชื่อว่า name โดยมี Key คือ Dad  
print(name['Mom']) #แสดงข้อมูลเก็บไว้ใน Dictionary ที่มีชื่อว่า name โดยมี Key คือ Mom
```

```
{'Dad': 'Somchai', 'Mom': 'Somsri', 'Bro': 'John'}  
Somchai  
Somsri
```



Dictionary

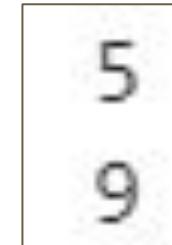
```
# ก้าเซ็นเดิม แต่เก็บค่าในรูปแบบ Int  
age={'Dad':42,'Mom':40}  
print(age['Dad'])  
print(age['Dad']+ age['Mom']) # คำนวณค่าที่ได้จากการเรียกข้อมูลโดยใช้ Key  
print(age)
```

42
82
{'Dad': 42, 'Mom': 40}



NumPy

```
import numpy as np #import numpy
a = np.array([1,2,3,4,5,6]) #สร้างข้อมูล
print(a[4]) #แสดงผลค่าที่อยู่ในตัวแปร a โดยมี index อยู่ที่ 5
print(a[3]+a[4]) #แสดงผลค่าผลบวกที่ index 3 และ 4 ในตัวแปร a
```



Control Flow

Control Flow

- Comparison Operators
 - == มีค่าเท่ากับ
 - != มีค่าไม่เท่ากับ
 - > มีค่ามากกว่า
 - >= มีค่ามากกว่าหรือเท่ากับ
 - < มีค่าน้อยกว่า
 - <= มีค่าน้อยกว่าหรือเท่ากับ
- Indent สำคัญใน Python



If-Else

```
a = 10  
b = 30  
c = 90  
#indent  
if(a == b):  
    print(a)  
elif(a > b):  
    print(a)  
elif(a < b):  
    print(c)
```

90



Loop

```
for x in range(3):
    print(x)

for x in range(5, 7):
    print(x)

for x in range(10, 15):
    if(x > 12):
        print(x)
```

```
0
1
2
5
6
13
14
```



Loop over List

```
fruits = ["banana","papaya","orange","apple","mango"]
for fruit in fruits:
    print(fruit)
```

banana
papaya
orange
apple
mango



Loop over String

```
name = "Tanya Sattayaaphitan"  
for char in name:  
    print(char)
```

T
a
n
y
a

S
a
t
t
a
y
a
a
p
h
i
t
a
n



Data Type Handling and Conversion

- `int(x [,base])` แปลง object x ให้เป็น integer
- `long (x [,base])` แปลง object x ให้เป็น long
- `float (X)` แปลง object x ให้เป็น float
- `complex (real [,imag])` ทำให้เป็นตัวเลข complex
- `str(x)` แปลง object x ให้เป็น string
- `repr(x)` แปลง object x ให้เป็น expression string
- `eval(str)` หากว่า string เป็น object
- `set (seq)` แปลง sequence ให้เป็น set
- `frozenset(seq)` แปลง sequence ให้เป็น frozenset
- `chr(x)` แปลง integer ให้เป็น character
- `unichr(x)` แปลง integer ให้เป็น Unicode character
- `ord(x)` แปลง character (ตัวเดียว) ให้เป็น integer
- `hex (x)` แปลง integer ให้เป็น hexadecimal string
- `oct(x)` แปลง integer ให้เป็น octal string



Function

```
def plus(a,b):  
    c = a + b  
    return c  
  
print(plus(1,2))
```

3



Local Variable

```
def plus(a,b):  
    c = a + b  
    return c  
  
print(plus(1,2))  
print(c)
```

```
3  
Traceback (most recent call last):  
  File "c:/Users/tanlu/git/python-django-api/basic.py", line 16, in <module>  
    print(c)  
NameError: name 'c' is not defined
```



Local Variable

```
age = 20
def plus10(a):
    age = a + 10
    return age

print(plus10(5))
print(age)
```

15
20



Global Variable

```
age = 20
def plus10(a):
    global age
    age = a + 10
    return age

print(plus10(5))
print(age)
```

15
15



Python Lab : Write a BMI program (bmi.py)

1. Input weight as “kg”
2. Input height as “m”
 - 177 cm = 1.77 m

$$BMI = \frac{Weight(kg)}{[Height(m)]^2}$$

Hint for Input

```
height = input("Enter your height: ")  
height = float(height)
```

```
Enter your weight: 77  
77.0  
Enter your height: 1.77  
1.77  
bmi = 24.58  
fat 1
```

BMI kg/m ²	อุปนิสัยที่	ภาวะเสี่ยงต่อโรค
น้อยกว่า 18.50	น้ำหนักน้อย / ผอม	มากกว่าคนปกติ
ระหว่าง 18.50 - 22.90	ปกติ (สุขภาพดี)	เท่าคนปกติ
ระหว่าง 23 - 24.90	ท้วม / โรคอ้วนระดับ 1	อันตรายระดับ 1
ระหว่าง 25 - 29.90	อ้วน / โรคอ้วนระดับ 2	อันตรายระดับ 2
มากกว่า 30	อ้วนมาก / โรคอ้วนระดับ 3	อันตรายระดับ 3



Python cheat sheet

Python For Data Science Cheat Sheet

Python Basics

Learn More Python for Data Science interactively at www.datacamp.com



Variables and Data Types

Variable Assignment

```
>>> x=5  
>>> x  
5
```

Calculations With Variables

>>> x+2 7	Sum of two variables
>>> x-2 3	Subtraction of two variables
>>> x*2 10	Multiplication of two variables
>>> x**2 25	Exponentiation of a variable
>>> x%2 1	Remainder of a variable
>>> x/float(2) 2.5	Division of a variable

Types and Type Conversion

str()	'5', '3.45', 'True'	Variables to strings
int()	5, 3, 1	Variables to integers
float()	5.0, 1.0	Variables to floats
bool()	True, True, True	Variables to booleans

Asking For Help

```
>>> help(str)
```

Strings

```
>>> my_string = 'thisStringIsAwesome'  
>>> my_string  
'thisStringIsAwesome'
```

String Operations

```
>>> my_string * 2  
'thisStringIsAwesome>thisStringIsAwesome'  
>>> my_string + 'Init'  
'thisStringIsAwesomeInit'  
>>> 'm' in my_string  
True
```

Lists

Also see NumPy Arrays

```
>>> a = 'is'  
>>> b = 'nice'  
>>> my_list = ['my', 'list', a, b]  
>>> my_list2 = [[4,5,6,7], [3,4,5,6]]
```

Selecting List Elements

Index starts at 0

Subset
>>> my_list[1]
>>> my_list[-3]
Slice
>>> my_list[1:3]
>>> my_list[1:]
>>> my_list[:3]
>>> my_list[]
Subset Lists of Lists
>>> my_list2[1][0]
>>> my_list2[1][:2]

Select item at index 1
Select 3rd last item
Select items at index 1 and 2
Select items after index 0
Select items before index 3
Copy my_list
my_list[list][itemOfList]

List Operations

```
>>> my_list + my_list  
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']  
>>> my_list * 2  
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']  
>>> my_list2 > 4  
True
```

List Methods

```
>>> my_list.index('a')  
>>> my_list.count('a')  
>>> my_list.append('!')  
>>> my_list.remove('!')  
>>> del(my_list[0])  
>>> my_list.reverse()  
>>> my_list.extend('!!')  
>>> my_list.pop(-1)  
>>> my_list.insert(0, '!!')  
>>> my_list.sort()  
Get the index of an item  
Count an item  
Append an item at a time  
Remove an item  
Remove an item  
Reverse the list  
Append an item  
Remove an item  
Insert an item  
Sort the list
```

String Operations

Index starts at 0

```
>>> my_string[3]  
>>> my_string[4:9]
```

String Methods

```
>>> my_string.upper()  
>>> my_string.lower()  
>>> my_string.count('w')  
>>> my_string.replace('e', 'i')  
>>> my_string.strip()  
String to uppercase  
String to lowercase  
Count String elements  
Replace String elements  
Strip whitespaces
```

Libraries

Import libraries

```
>>> import numpy  
>>> import numpy as np  
Selective import  
>>> from math import pi
```

pandas

Data analysis

NumPy

Scientific computing

matplotlib

2D plotting

Machine learning

jupyter

Create and share

documents with live

visualizations, text, ...

Install Python



Leading open data science platform
powered by Python



Free IDE that is included
with Anaconda



Create and share
documents with live
visualizations, text, ...

Numpy Arrays

Also see Lists

```
>>> my_list = [1, 2, 3, 4]  
>>> my_array = np.array(my_list)  
>>> my_2darray = np.array([[1,2,3],[4,5,6]])
```

Selecting Numpy Array Elements

Index starts at 0

Subset
>>> my_array[1]
2
Slice
>>> my_array[0:2]
array([1, 2])
Subset 2D Numpy arrays
>>> my_2darray[:,0]
array([1, 4])
Select Item at Index 1
Select items at index 0 and 1
my_2darray[rows, columns]

Numpy Array Operations

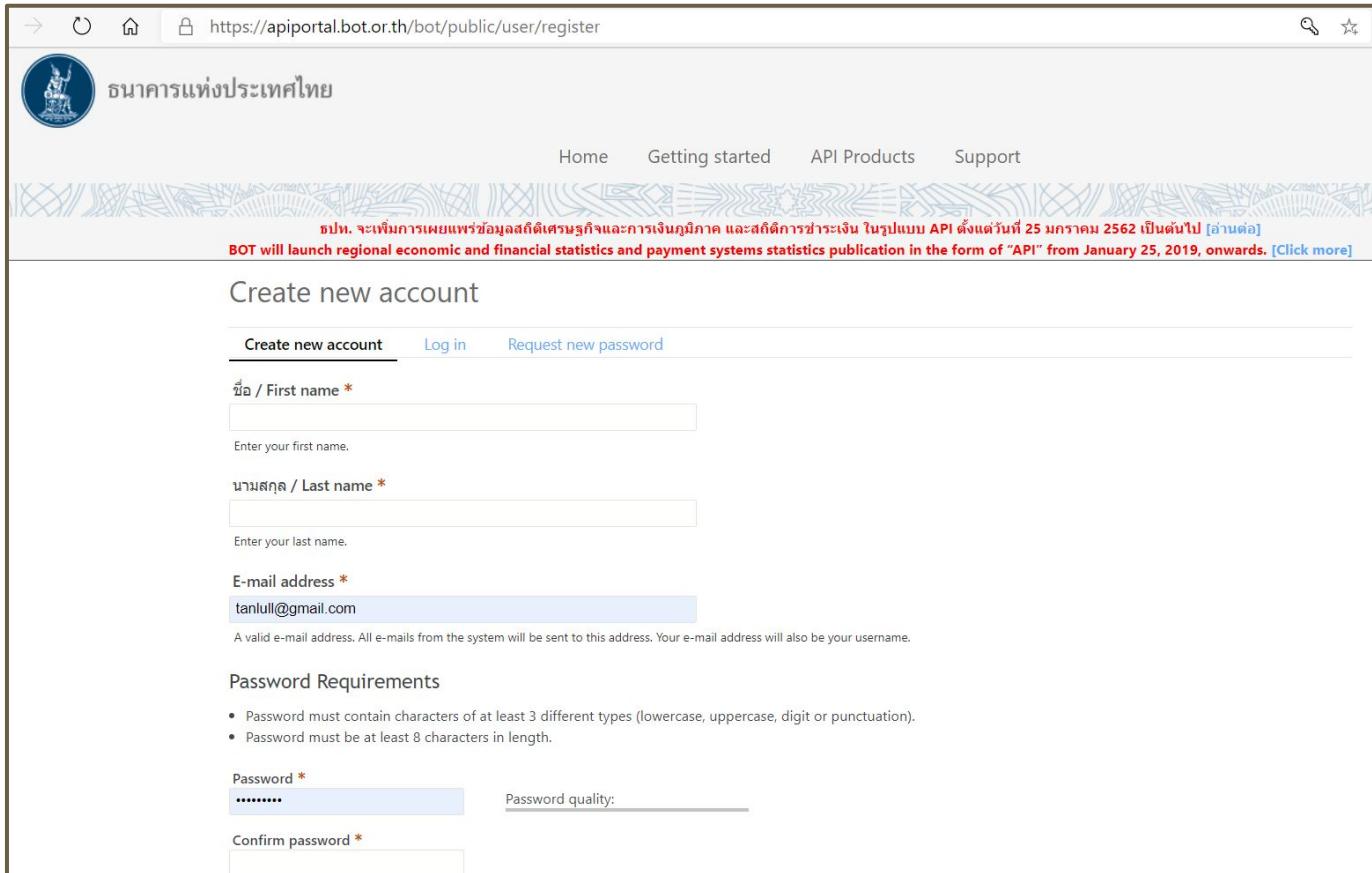
```
>>> my_array > 3  
array([False, False, False, True], dtype=bool)  
>>> my_array * 2  
array([2, 4, 6, 8])  
>>> my_array + np.array([5, 6, 7, 8])  
array([6, 8, 10, 12])
```

Numpy Array Functions

```
>>> my_array.shape  
>>> np.append(other_array)  
>>> np.insert(my_array, 1, 5)  
>>> np.delete(my_array, [1])  
>>> np.mean(my_array)  
>>> np.median(my_array)  
>>> my_array.correlate()  
>>> np.std(my_array)  
Get the dimensions of the array  
Append items to an array  
Insert items in an array  
Delete items in an array  
Mean of the array  
Median of the array  
Correlation coefficient  
Standard deviation
```



Call BOT API



The screenshot shows a web browser window with the URL <https://apiportal.bot.or.th/bot/public/user/register>. The page title is "Create new account". The header includes the logo of the Royal Thai Government and navigation links for Home, Getting started, API Products, and Support. A banner at the top states: "บก. จะเพิ่มการเผยแพร่องค์ประกอบเศรษฐกิจและการเงินภายนอก และผลติดารษาระเงิน ในรูปแบบ API ตั้งแต่วันที่ 25 มกราคม 2562 เป็นต้นไป [อ่านต่อ]" and "BOT will launch regional economic and financial statistics and payment systems statistics publication in the form of "API" from January 25, 2019, onwards. [Click more]". The main form fields include "First name *", "Last name *", "E-mail address *", and "Password *". The "E-mail address" field contains the value "tanlull@gmail.com". Below the form, there is a section titled "Password Requirements" with two bullet points: "Password must contain characters of at least 3 different types (lowercase, uppercase, digit or punctuation)." and "Password must be at least 8 characters in length." A QR code is located in the bottom right corner of the page.

ธนารถแห่งประเทศไทย

Home Getting started API Products Support

บก. จะเพิ่มการเผยแพร่องค์ประกอบเศรษฐกิจและการเงินภายนอก และผลติดารษาระเงิน ในรูปแบบ API ตั้งแต่วันที่ 25 มกราคม 2562 เป็นต้นไป [อ่านต่อ]
BOT will launch regional economic and financial statistics and payment systems statistics publication in the form of "API" from January 25, 2019, onwards. [Click more]

Create new account

Create new account Log in Request new password

ชื่อ / First name *

นามสกุล / Last name *

E-mail address *

tanlull@gmail.com

A valid e-mail address. All e-mails from the system will be sent to this address. Your e-mail address will also be your username.

Password Requirements

- Password must contain characters of at least 3 different types (lowercase, uppercase, digit or punctuation).
- Password must be at least 8 characters in length.

Password *

Confirm password *

QR code

Register API

The screenshot shows a web browser window for the Bank of Thailand API portal at <https://apiportal.bot.or.th/bot/public/application>. The page header includes the Bank of Thailand logo and the text "ธนาคารแห่งประเทศไทย". The top navigation bar has links for Home, Getting started, API Products, Apps (highlighted with a red arrow and labeled '1'), and Support. A search bar is also present. The main content area contains a message about the launch of regional economic and financial statistics and payment systems statistics publication via API from January 25, 2019. Below this message, there is a section titled "Python Programming" with a thumbnail icon of a smartphone and the text "testing". At the bottom right of this section is a blue button labeled "+ Create new App" with a red arrow pointing to it and the number '2'. The footer contains links for Terms of use and ©2018 Bank of Thailand. All rights reserved.

ธนาคารแห่งประเทศไทย

tanlull@gmail.com Tanya Sattaya-aphitan

Home Getting started API Products Apps Support

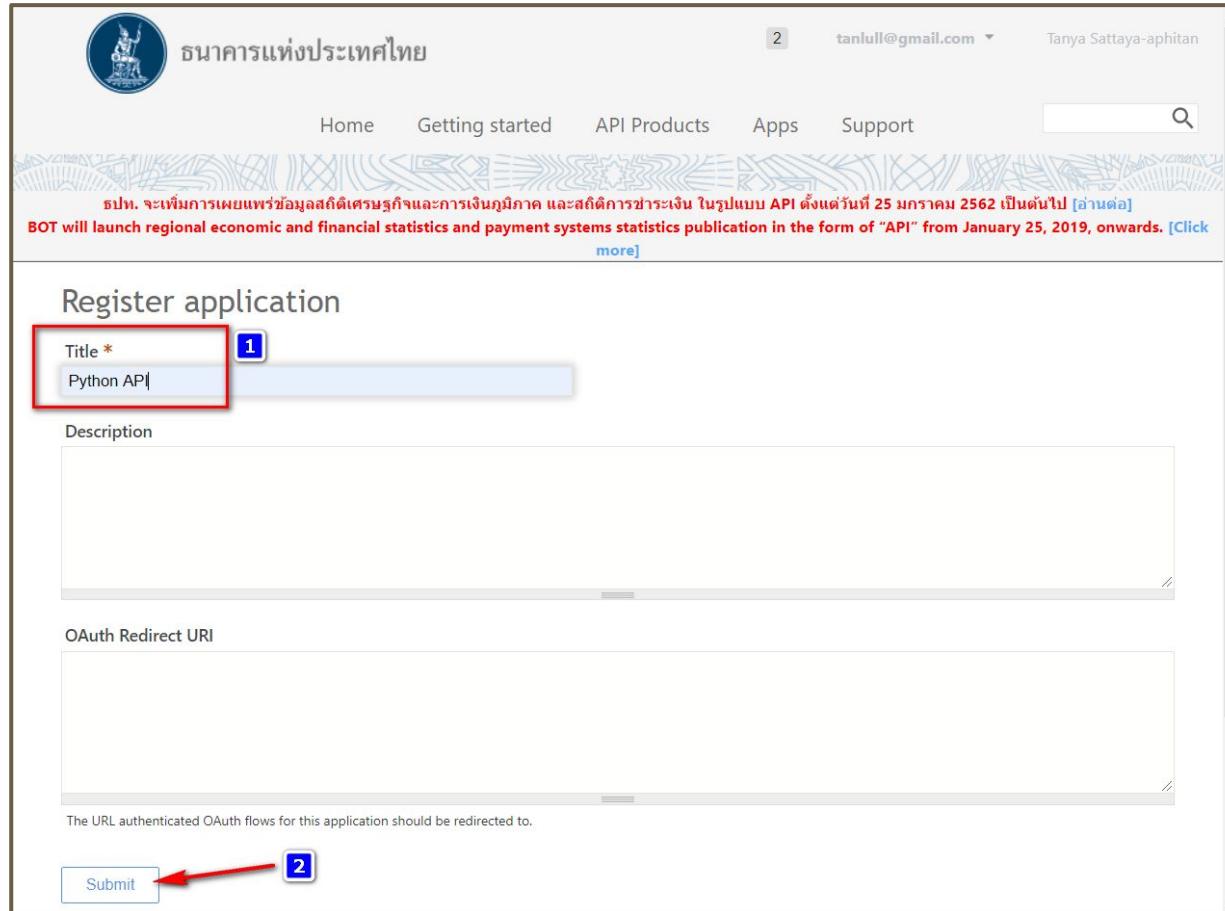
ธปท. จะเพิ่มการเผยแพร่องค์ความรู้และสารสนเทศทางเศรษฐกิจและการเงินภูมิภาค และสถิติการชำระเงิน ในรูปแบบ API ตั้งแต่วันที่ 25 มกราคม 2562 เป็นต้นไป [อ่านต่อ]
BOT will launch regional economic and financial statistics and payment systems statistics publication in the form of "API" from January 25, 2019, onwards. [Click more]

+ Create new App

Python Programming
testing

Terms of use
©2018 Bank of Thailand. All rights reserved.

Register Application : Python API



ธนาคารแห่งประเทศไทย

tanlull@gmail.com Tanya Sattaya-aphitana

Home Getting started API Products Apps Support

ปีน. จะเพิ่มการเผยแพร่องค์กรและเศรษฐกิจและการเงินภูมิภาค และอัตติการชำระเงิน ในรูปแบบ API ตั้งแต่วันที่ 25 มกราคม 2562 เป็นต้นไป [อ่านต่อ]
BOT will launch regional economic and financial statistics and payment systems statistics publication in the form of "API" from January 25, 2019, onwards. [Click more]

Register application

Title * 1
Python API

Description

OAuth Redirect URI

The URL authenticated OAuth flows for this application should be redirected to.

Submit 2



The screenshot shows the Thailand Government API registration interface. At the top, there is a logo and the text "ธนาคารแห่งประเทศไทย". Below the logo, there are navigation links: Home, Getting started, API Products, Apps (which has a red arrow pointing to it), and Support. On the left, there is a sidebar with "Apps" selected, and under "API Products", "Python Programming" is listed. The main content area displays a success message: "Application created successfully." followed by "Your client secret is:" and a redacted string. There is also a "Show Client Secret" link. Below this, instructions say: "Now that you've registered your app, you can browse the [available APIs](#) and subscribe." It advises users to make a note of their client ID and client secret for application access. A note states: "Your client secret will only be displayed once. If you forget or lose it, you can verify the secret to see if it's correct or reset it to get a new one." The page title is "Python Programming". Under the "Details" tab, there is a thumbnail icon, the app name "Python Programming", an "Update" button, a "Description" field containing "testing", and a "Credentials" section with a "Default" entry and a "Client ID" field which is also redacted.

Copy Client ID to field in the next step : 1edd90f8-f587-4e21-9433-81db0f322825

Activate API

ธนาคารแห่งประเทศไทย

Home Getting started API Products Apps Support

Statistics (1.0.2) (2 APIs included)

Exchange Rates (2.0.1) (2 APIs included)

Interest Rates (2.0.0) (9 APIs included)

Debt Securities Auction (2.0.0) (1 API included)

Financial Product Comparison (1.0.4) (3 APIs included)



Activate API

The screenshot shows a web application interface for activating an API. A modal window titled "Subscribe" is open, overlaid on a main page displaying API plans.

Main Page (Background):

- Header: ธนาคารแห่งประเทศไทย
- User: tanlull@gmail.com
- Navigation: Products, Apps, Support
- Plans:
 - Default Plan**: 200 per hour (locked)
 - Free**: 200 per hour

Modal Window ("Subscribe"):

- Step 1:** Shows the "Default Plan" with a lock icon and "200 per hour".
- Step 2:** Shows the "Free" plan with a lock icon and "200 per hour".
- Step 3:** The "Subscribe" button in the modal window is highlighted with a red arrow.

Text in Modal:

- "Select an application to sign up to this plan."
- "Python Programming" (radio button selected)
- "There are no available applications to subscribe to this plan. [Register a new application](#)"
- "Subscribe"



Browse API

The screenshot shows the Bank of Thailand's API products page. At the top right, there is a user icon with a red arrow pointing to it labeled '1'. Below the header, there are navigation links: Home, Getting started, API Products (which is highlighted with a blue box and a red arrow labeled '2'), Apps, and Support.

The main content area displays the details for the 'Exchange Rates 2.0.1' product. It features a logo, the product name, and a 'Plans' section. The 'Plans' section includes two plans:

Plan	Cost
Default Plan	200 per hour
Average Exchange Rate - THB / Foreign Currency	200 per hour

Below the plans, there is a 'Subscribe' button and a note: '* = Mouseover for more information'. A 'Bookmark this' link is at the bottom left.

On the left sidebar, there is a menu with the following items:

- Exchange Rates 2.0.1
- APIs
- Weighted-average Interbank Exchange Rate - THB / USD
- Average Exchange Rate - THB / Foreign Currency



Code Example

The screenshot shows the Bank of Thailand API documentation and a corresponding Python code example.

Left Panel (API Documentation):

- APIs:** Weighted-average Interbank Exchange (1)
- Operations:** GET /DAILY_REF_RATE/ (2)
- Description:** Daily Weighted-average Interbank Exchange Rate - THB / USD (3)
- Details:** More information about the rate calculation, frequency (Daily), lag time, release schedule, source of data, and security.
- Definitions:** X-IBM-Client-Id (apiKey located in header) (clientIdHeader)
- Parameters:**
 - start_period:** Start Period Date (YYYY-MM-DD) EX. 2017-06-30 (Required in query, string)
 - end_period:** End Period Date (YYYY-MM-DD) EX. 2017-06-30 (Required in query, string)
 - Accept:** application/json

Right Panel (Python Code Example):

- cURL:** curl command for the API endpoint.
- Python:** Sample Python code using the `http.client` module to make a request to the API. It includes headers for X-IBM-Client-ID and Accept.
- Headers:** The code snippet shows the `headers` section of the Python script, specifically the `X-IBM-Client-ID` header which is annotated with a red box and labeled (4).
- Example Response:** Definition of the response structure, including the URL and the JSON schema for the result.

Client ID from previous step : 1edd90f8-f587-4e21-9433-81db0f322825

Request API -- BOT_API_Daily.py

```
import http.client
conn = http.client.HTTPSConnection("apigw1.bot.or.th")
headers = {
    'x-ibm-client-id': "1edd90f8-f587-4e21-9433-81db0f322825",
    'accept': "application/json"
}
conn.request("GET",
"/bot/public/Stat-ReferenceRate/v2/DAILY_REF_RATE/?start_period=2018-10-01&
end_period=2018-10-31", headers=headers)
res = conn.getresponse()
data = res.read()
print(data.decode("utf-8"))
```



Response JSON

```
{  
  "result":{  
    "timestamp":"2018-11-04 09:35:56",  
    "api":"Daily Weighted-average Interbank Exchange Rate - THB / USD",  
    "data":{  
      "data_header":{  
        "report_name_eng":"Rates of Exchange of Commercial Banks in Bangkok  
Metropolis (2002-present)",  
        "report_name_th":"อัตราแลกเปลี่ยนและลีบของธนาคารพาณิชย์ในกรุงเทพมหานคร  
(2545-ปัจจุบัน)",  
        "report_uoq_name_eng":"(Unit : Baht / 1 Unit of Foreign Currency)",  
        "report_uoq_name_th":"(หน่วย : บาท ต่อ 1 หน่วยเงินตราต่างประเทศ)",  
        "report_source_of_data": [  
          {  
            "source_of_data_eng":"Bank of Thailand",  
            "source_of_data_th":"ธนาคารแห่งประเทศไทย"  
          }  
        ],  
        "report_remark": [  
        ],  
        "last_updated":"2018-11-02"  
      },  
      "data_detail": [  
        {  
          "period":"2018-10-31",  
          "rate":"33.2660000"  
        },  
        {  
          "period":"2018-10-30",  
          "rate":"33.2700000"  
        },  
        {  
          "period":"2018-10-29",  
          "rate":"33.1360000"  
        },  
        {  
          "period":"2018-10-26",  
          "rate":"33.0290000"  
        },  
        {  
          "period":"2018-10-25",  
          "rate":"32.9230000"  
        },  
        {  
          "period":"2018-10-24",  
          "rate":"32.8110000"  
        },  
        {  
          "period":"2018-10-03",  
          "rate":"32.3380000"  
        },  
        {  
          "period":"2018-10-02",  
          "rate":"32.3380000"  
        },  
        {  
          "period":"2018-10-01",  
          "rate":"32.2530000"  
        }  
      ]  
    }  
  }  
}
```

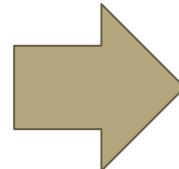
```
{  
  "period":"2018-10-29",  
  "rate":"33.1360000"  
},  
{  
  "period":"2018-10-26",  
  "rate":"33.0290000"  
},  
{  
  "period":"2018-10-25",  
  "rate":"32.9230000"  
},  
{  
  "period":"2018-10-24",  
  "rate":"32.8110000"  
},  
{  
  "period":"2018-10-03",  
  "rate":"32.3380000"  
},  
{  
  "period":"2018-10-02",  
  "rate":"32.3380000"  
},  
{  
  "period":"2018-10-01",  
  "rate":"32.2530000"  
}  
]
```



Format Response to DataFrame

```
pip install pandas
```

```
import pandas as pd
j=pd.io.json.loads(data)
df=pd.DataFrame(j['result']['data']['data_detail'])
#แปลงให้เป็น Type DateTime
df.period= pd.to_datetime(df.period)
#แปลงให้เป็น Type Numeric
df.rate= pd.to_numeric(df.rate)
print(df)
```



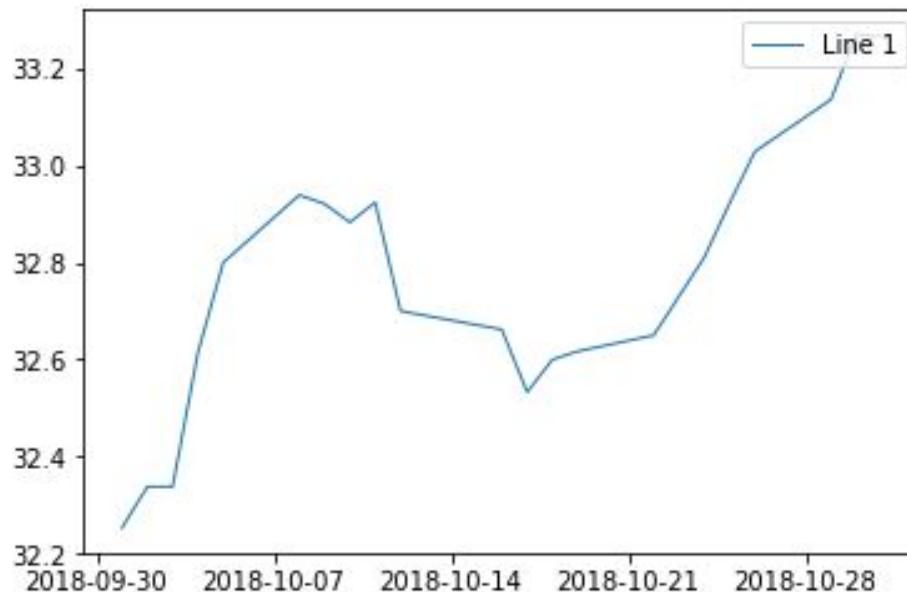
	period	rate
0	2018-10-31	33.266
1	2018-10-30	33.270
2	2018-10-29	33.136
3	2018-10-26	33.029
4	2018-10-25	32.923
5	2018-10-24	32.811
6	2018-10-22	32.650
7	2018-10-19	32.617
8	2018-10-18	32.600
9	2018-10-17	32.533
10	2018-10-16	32.662
11	2018-10-12	32.700
12	2018-10-11	32.924
13	2018-10-10	32.883
14	2018-10-09	32.921
15	2018-10-08	32.939
16	2018-10-05	32.800
17	2018-10-04	32.614
18	2018-10-03	32.338
19	2018-10-02	32.338
20	2018-10-01	32.253



Plot graph

pip install matplotlib

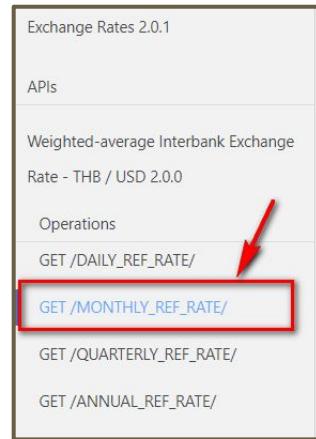
```
import matplotlib.pyplot as plt #plot graph เส้น และกำหนดขนาดของเส้น  
plt.plot(df.period,df.rate, linewidth=1) #กำหนด Description แต่ละเส้นพร้อมบอกตากำหนดที่แสดง  
plt.legend(["Line 1"], loc="upper right")  
plt.show()
```



Request API Lab 1

1. Create [BOT_API_Monthly.py](#)

2. Using



3. Retrieve data from January 2017 to September 2020

4. Print data

5. Plot Graph



Test with postman

Test with Postman

<https://www.postman.com/downloads/>

The screenshot shows the Postman download page. On the left, there's a large callout box with the heading "The Postman app". It contains text about the app's improvements and a prominent orange "Download the App" button. Below the button, there's a red box containing the text: "By downloading and using Postman, I agree to the [Privacy Policy](#) and [Terms](#)". To the right of this box, a red arrow points towards the Postman desktop application interface. The interface shows a window with a request URL "get postman-echo.com/get", a "Params" tab, and a "Headers" tab. The "Authorization" tab is selected, showing "Inherit auth from parent". A QR code is located in the bottom right corner of the page.

POSTMAN Product Use Cases Pricing Enterprise Explore Learning Center

Launch Postman

Download Postman

Download the app to quickly get started using the Postman API Platform. Or, if you prefer a browser experience, you can try the new web version of Postman.

The Postman app

The ever-improving Postman app (a new release every two weeks) gives you a full-featured Postman experience.

[Download the App](#)

By downloading and using Postman, I agree to the [Privacy Policy](#) and [Terms](#).

Version 7.33.1 | [Release Notes](#) | [Product Roadmap](#)

Not your OS? Download for Mac ([macOS](#)) or Linux (x64)

History Collections APIs

GET postman-echo.com/get

Params Authorization Headers (8) Body Pre-request

Type Inherit auth from parent

This request is not inh use the parent's auth

Body Cookies (1) Headers (7) Test Results

Pretty Raw Preview Visualize JSON

Get postman-echo.com/get

Inherit auth from parent

The authorization header will be automatically generated when you send the request. Learn more about authorization

Body Cookies (1) Headers (7) Test Results

Pretty Raw Preview Visualize JSON

QR code

Get Endpoint

The screenshot shows a web browser window with the URL <https://apiportal.bot.or.th/bot/public/node/470>. The page is titled "ธนานาชาติแห่งประเทศไทย" and features a navigation menu with links to Home, Getting started, API Products, Apps, and Support.

A banner at the top states: "เรบ. จะเพิ่มการเผยแพร่ข้อมูลเกี่ยวกับเศรษฐกิจและการเงินภูมิภาค และสกุลเงินชั้นนำ ในรูปแบบ API ตั้งแต่วันที่ 25 มกราคม 2562 เป็นต้นไป [อ่านต่อ]" and "BOT will launch regional economic and financial statistics and payment systems statistics publication in the form of "API" from January 25, 2019, onwards. [Click more]".

The main content area is divided into sections:

- Endpoints:** Shows the endpoint <https://apigw1.bot.or.th/bot/public> in PRODUCTION mode.
- Paths:** Shows the path https://apigw1.bot.or.th/bot/public/DAILY_REF_RATE/.
- Description:** Details the Daily Weighted-average Interbank Exchange Rate - THB / USD, stating it is calculated from daily interbank purchases and sales of US dollar (against THB) of transactions worth more than or equal to 1 million USD. It uses a weighted-average between trading volume and exchange rate specified.
- Example Request:** Displays a curl command:

```
curl --request GET \
--url 'https://apigw1.bot.or.th/bot/public/Stat-ReferenceRate/v2/DAILY_REF_RATE?start_period=REPLACE_THIS_VALUE&end_period=REPLACE_THIS_VALUE' \
--header 'accept: application/json' \
--header 'x-ibm-client-id: REPLACE_THIS_KEY'
```
- Example Response:** Placeholder for the response content.

A red arrow points to the "GET /DAILY_REF_RATE/" link in the Paths section, and another red arrow points to the curl command in the Example Request section.



Extract Information

```
curl --request GET \  
  --url 'https://apigw1.bot.or.th/bot/public/Stat-  
ReferenceRate/v2/DAILY_REF_RATE/?start_period=REPL  
ACE_THIS_VALUE&end_period=REPLACE_THIS_VALUE' \  
  --header 'accept: application/json' \  
  --header 'x-ibm-client-id: REPLACE_THIS_KEY'
```

Method: GET

Endpoint:

https://apigw1.bot.or.th/bot/public/Stat-ReferenceRate/v2/DAILY_REF_RATE/?start_period=REPLACE_THIS_VALUE&end_period=REPLACE_THIS_VALUE

Header: accept: application/json

x-ibm-client-id: **1edd90f8-f587-4e21-9433-81db0f322825**



Postman

The screenshot shows the Postman application window. At the top, there is a menu bar with File, Edit, View, Help, and a toolbar with New, Import, Runner, and other icons. A red arrow points from the 'New' button in the toolbar to a blue box labeled '1' on the 'Create New' tab of the dialog.

The main area displays a 'Create New' dialog with several options:

- BUILDING BLOCKS**
 - Request**: Create a basic request (GET icon)
 - Collection**: Save your requests in a collection for reuse and sharing (Collection icon). A red arrow points from this option to a blue box labeled '2'.
 - Environment**: Save values you frequently use in an environment (Environment icon)
- ADVANCED**
 - API Documentation**: Create and publish beautiful documentation for your APIs (Document icon)
 - Mock Server**: Create a mock server for your in-development APIs (Mock Server icon)
 - Monitor**: Schedule automated tests and check performance of your APIs (Monitor icon)
 - API**: Manage all aspects of API design, development, and testing (API icon)

At the bottom of the dialog, there is a note: "Not sure where to start? Use a [template](#) to see how Postman can help you in your work." and a link "Learn more on Postman Docs".

The background shows a list of collections in the workspace:

- Express M 6 requests
- Google Sh 5 requests
- NodeJS wi 5 requests
- AI for Thai 5 requests
- Elastics 16 requests
- Google 7 requests
- LoRa 16 requests
- LPR 4 requests
- NB-IoT 1 request

At the bottom of the window, there are tabs for Find and Replace, Console, Bootcamp, Build, Browse, and Help.



New Collection

The screenshot shows the Postman application interface with the title "New Collection" at the top. The main workspace on the left lists various collections: "AI for Thai" (5 requests), "Elastics" (16 requests), "Google" (7 requests), "LoRa" (16 requests), "LPR" (4 requests), "NB-IoT" (1 request), "OData" (1 request), "Postman Echo" (37 requests), and "PythonDjango" (5 requests). A "Find and Replace" and "Console" button are also visible.

A central modal window titled "CREATE A NEW COLLECTION" is open. It contains a "Name" input field with the value "Python API" highlighted by a red box and marked with a blue number "1". Below the name is a "Description" section with a text area containing placeholder text: "Make things easier for your teammates with a complete collection description." A note below says "Descriptions support Markdown". At the bottom right of the modal are "Cancel" and "Create" buttons, with the "Create" button highlighted by a red arrow and marked with a blue number "2".

The top navigation bar includes "New", "Import", "Runner", "My Workspace", "Invite", "Upgrade", and several status icons. On the right side, there are "BUILD", "Save", "Cookies", "Code", and "Bulk Edit" buttons.



Add Request

The screenshot shows the Postman application interface with the following elements:

- Header Bar:** Postman, File, Edit, View, Help, New, Import, Runner, My Workspace, Invite, Upgrade.
- Left Sidebar:** History, Collections (selected), APIs, Trash. Collections list includes: AI for Thai (5 requests), Elastics (16 requests), Google (7 requests), LoRa (16 requests), LPR (4 requests), NB-IoT (1 request), OData (1 request), Postman Echo (37 requests), Python API (0 requests).
- Central Area:** Untitled Request, Method: GET, URL: https://reqres.in/api/users, Send, Save buttons.
- Request Details:** Params, Auth, Headers (7), Body, Pre-req., Tests, Settings, Cookies, Code tabs.
- Query Params Table:** KEY, VALUE, DESCRIPTION columns.
- Context Menu (Open by Right Clicking 'Python API'):** Share Collection, Manage Roles, Rename (Ctrl+E), Edit (2), Create a fork, Add Request (highlighted with a red arrow and blue box), Add Folder, Duplicate (Ctrl+D), Export, Monitor Collection, Mock Collection, Publish Docs, Remove from workspace, Delete (Del).
- Bottom Navigation:** Find and Replace, Console, Bootcamp, Build, Browse, Help.

A red arrow points from the "Right Click" label to the context menu, and another red arrow points from the "1" label to the "Python API" collection in the sidebar.



GET / URI

Updated Request

1

2

3

4

5

Send

Save

Params ● Authorization Headers (8) Body Pre-request Script Tests Settings Cookies Code

Headers Hide auto-generated headers

KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets ▾
Cookie ⓘ	visid_incap_1778672=e5cAmrbuTXy65VvKq0Z17Np0eF8AAAAAQUI				
Postman-Token ⓘ	<calculated when request is sent>				
Host ⓘ	<calculated when request is sent>				
User-Agent ⓘ	PostmanRuntime/7.26.5				
Accept ⓘ	*/*				
Accept-Encoding ⓘ	gzip, deflate, br				
Connection ⓘ	keep-alive				
x-ibm-client-id	1edd90f8-f587-4e21-9433-81db0f322825		4		
Key	Value	Description			



Response

Body Cookies (2) Headers (21) Test Results

Pretty Raw Preview Visualize JSON 

```
1  {
2      "result": {
3          "timestamp": "2020-10-03 19:56:20",
4          "api": "Daily Weighted-average Interbank Exchange Rate - THB/USD"
5          "data": {
6              "data_header": {
7                  "report_name_eng": "Rates of Exchange of Commercial Banks in Thailand"
8                  "report_name_th": "อัตราแลกเปลี่ยนและลีบของธนาคารพาณิชย์ไทย"
9                  "report_uoq_name_eng": "(Unit: Baht / 1 Unit of Foreign Currency)"
10                 "report_uoq_name_th": "(หน่วย: บาท ต่อ 1 หน่วยเงินต่างประเทศ)"
11                 "report_source_of_data": [
12                     {
13                         "source_of_data_eng": "Bank of Thailand",
14                         "source_of_data_th": "ธนาคารแห่งประเทศไทย"
15                     }
16                 ],
17                 "report_remark": [],
18                 "last_updated": "2020-10-02"
19             },
20             "data_detail": [
```

```
18             {
19                 "last_updated": "2020-10-02"
20             },
21             "data_detail": [
22                 {
23                     "period": "2018-10-31",
24                     "rate": "33.266000"
25                 },
26                 {
27                     "period": "2018-10-30",
28                     "rate": "33.270000"
29                 },
30                 {
31                     "period": "2018-10-29",
32                     "rate": "33.136000"
33                 },
34                 {
35                     "period": "2018-10-26",
36                     "rate": "33.029000"
37                 },
--
```



AI 4 Thai API

Request API Lab 2

<https://aiforthai.in.th/>

The screenshot shows the 'User Profile' section of the AI FOR THAI website. At the top right, there's a user profile icon with the name 'tanlull'. A red box labeled '1' highlights this area. On the left sidebar, under the 'Services' section, there's a list of services: Dashboard, How to get started, Basic NLP, Tag suggestion, Machine translation, Sentiment Analysis, SSense (with a red arrow and box labeled '3' pointing to it), and ThaiMoji. Below the sidebar, the main content area has fields for Username (tanlull), First name (Tanya), Last name (Sattaya-aphitana), Organization (TOT), Password, Confirm password, Email (tanlull@gmail.com), and API Key (bu---tUW' VGNK 5jL Frwv Z3). A red box labeled '2' highlights the API Key field. At the bottom right of the content area is a 'Save' button.

AI FOR THAI

Home About Services Demo Use cases Corpus Developer News & Events Contact us FAQs tanlull

User Profile

Dashboard

How to get started

Basic NLP

Tag suggestion

Machine translation

Sentiment Analysis

SSense

ThaiMoji

Character recognition

Object recognition

Person & activity

Face recognition

Speech to text

Text to speech

ABDUL chatbot

Username : tanlull

First name : Tanya

Last name : Sattaya-aphitana

Organization : TOT

Password : *รหัสผ่านต้องมีอักษรตัวอักษร 6 อักขระ

Confirm password :

Email : tanlull@gmail.com

API Key : bu---tUW' VGNK 5jL Frwv Z3

Save



Sentiment Analysis



AI FOR THAI

Home About Services Demo Use cases Corpus Developer News & Events Contact us FAQs tanlull

Dashboard

How to get started

Basic NLP

Tag suggestion

Machine translation

Sentiment Analysis

SSense 1

ThaiMoji

Character recognition

Object recognition

Person & activity

Face recognition

Speech to text

Text to speech

ABDUL chatbot

SSense

ເອສເຊັນສ໌ ຮະບບວິເຄຣາະໜົກວາມຄິດເຫັນຈາກຂ້ອງຄວາມ (Social Sensing: SSENSE)

ຮະບບວິເຄຣາະໜົກວາມຄິດເຫັນຂ້ອງຄວາມພາຫາໄທ ໂດຍວິເຄຣາະໜົກໃນເສັງຈຸດປະສົງ ຂອງຂ້ອງຄວາມ (Intention) ວ່າເປັນປະເກດໃດ ໄດ້ແກ່ ສອບຄາມ ຮ້ອງຂອງປະກາດ ໄນບານ ແລະ/ຫຼືວິເຄຣາະໜົກວາມຄິດເຫັນ ວິເຄຣາະໜົກວາມຄິດເຫັນ (Sentiment) ວ່າ ເປັນເສັງບຸກຫຼືລູບ ຮົວກັ້ງວິເຄຣາະໜົກນຳບໍ່ບອກຄຸນລັກບໍ່ບໍ່ຂອງສັນຄ້າຫຼື ບັນດາ (Feature words) ດ້ວຍ

Host	https://api.aiforthai.in.th/ssense
Method	GET/POST
Header	Apikey : buZc JV JwGN' 35jLesFoF ik' 3CnE vbB Content-Type : application/x-www-form-urlencoded
Parameter	text : ຂ້ອງຄວາມທີ່ຕ້ອງການວິເຄຣາະໜົກ

2

OUTPUT

ພລັພຽດທີ່ໄດ້ຈາກ ຮະບບ SSense ຈະອຍໃນຮູບແບບຂອງ JSON Arrays ທີ່ສັງແຕ່ລະອາເຮຍຈະນັອງກຳປະກອບດັ່ງນີ້

- sentiment : ພລັພຽດໜົກວາມຄິດເຫັນວ່າເປັນເສັງບຸກຫຼືລູບ
 - polarity : ຫຼັງວ່າຮຽນນີ້ນົກວາມຄິດເຫັນ (positive: ເສັງບຸກ, negative: ເສັງລູບ, ອ່າວ່າງ: ໃນເປັນກັ້ງບຸກແລະລູບ)
 - polarity-pos : ນີ້ນົກວາມເສັງບຸກໃຫ້ຫຼືໃນ (true=ໃຫ້, false = ໃບ້ໃຫ້)
 - polarity-neg : ນີ້ນົກວາມເສັງລູບໃຫ້ຫຼືໃນ (true=ໃຫ້, false = ໃບ້ໃຫ້)



Python Example

A
I FOR THAI

Home About Services Demo Use cases Corpus Developer News & Events Contact us FAQs tanlull

```
17.     .addHeader('Cache-Control', 'no-cache')
18.     .build();
19.
20. Response response = client.newCall(request).execute();
```

Python

GET

```
1. import requests
2.
3. url = "https://api.aiforthai.in.th/ssense"
4.
5. text = 'สาขาที่พนักงานนำรักให้บริการดี'
6.
7. params = {'text':text}
8.
9. headers = {
10.     'Apikey': "bl_5tUW_GI_35jL_sFo_rkZ3_gvbB"
11. }
12.
13. response = requests.get(url, headers=headers, params=params)
14.
15. print(response.json())
```



Output

Terminal Help AI4Thai_Sentiment.py - python-django-api - Visual Studio Code

JS json.js AI4Thai_Sentiment.py

AI4Thai_Sentiment.py > ...

```
1 import requests
2
3 url = "https://api.aiforthai.in.th/ssense"
4
5 text = 'ສຸດໆນາຮັກຈັງເລຍ'
6
7 params = {'text':text}
8
9 headers = {
10     'Apikey': "buz5tUwbWGNK35jLesFoFwkZ3Cn5gvbB"
11 }
12
13 response = requests.get(url, headers=headers, params=params)
14
15 print(response.json())
```

PROBLEMS 30 OUTPUT DEBUG CONSOLE TERMINAL

Windows PowerShell Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/pscore6>

PS C:\Users\tanlu\git\python-django-api> & C:/Python38/python.exe c:/Users/tanlu/git/python_django/api/AI4Thai_Sentiment.py

{'sentiment': {'score': '80', 'polarity-neg': False, 'polarity-pos': True, 'polarity': 'positive'}, 'preprocess': {'input': 'ສຸດໆນາຮັກຈັງເລຍ', 'neg': [], 'pos': ['ນາຮັກ'], 'segmented': ['ສຸ', 'ດໆ', 'ນາ', 'ຮັກ', 'ຈັງ', 'ເລຍ'], 'keyword': ['ສຸ', 'ດໆ', 'ນາ', 'ຮັກ', 'ຈັງ', 'ເລຍ'], 'alert': []}, 'comparative': [], 'associative': [], 'intention': {'request': '0', 'sentiment': '80', 'question': '0', 'announcement': '0'}}

PS C:\Users\tanlu\git\python-django-api>

API Lab 2

1. Make your own sentiment analysis
 2. Test Sentiment with Postman
 3. Try another API.



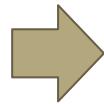
Day 2

Connect Database (MySQL)

Database : MySQL Community Server

<https://dev.mysql.com/downloads/mysql/>

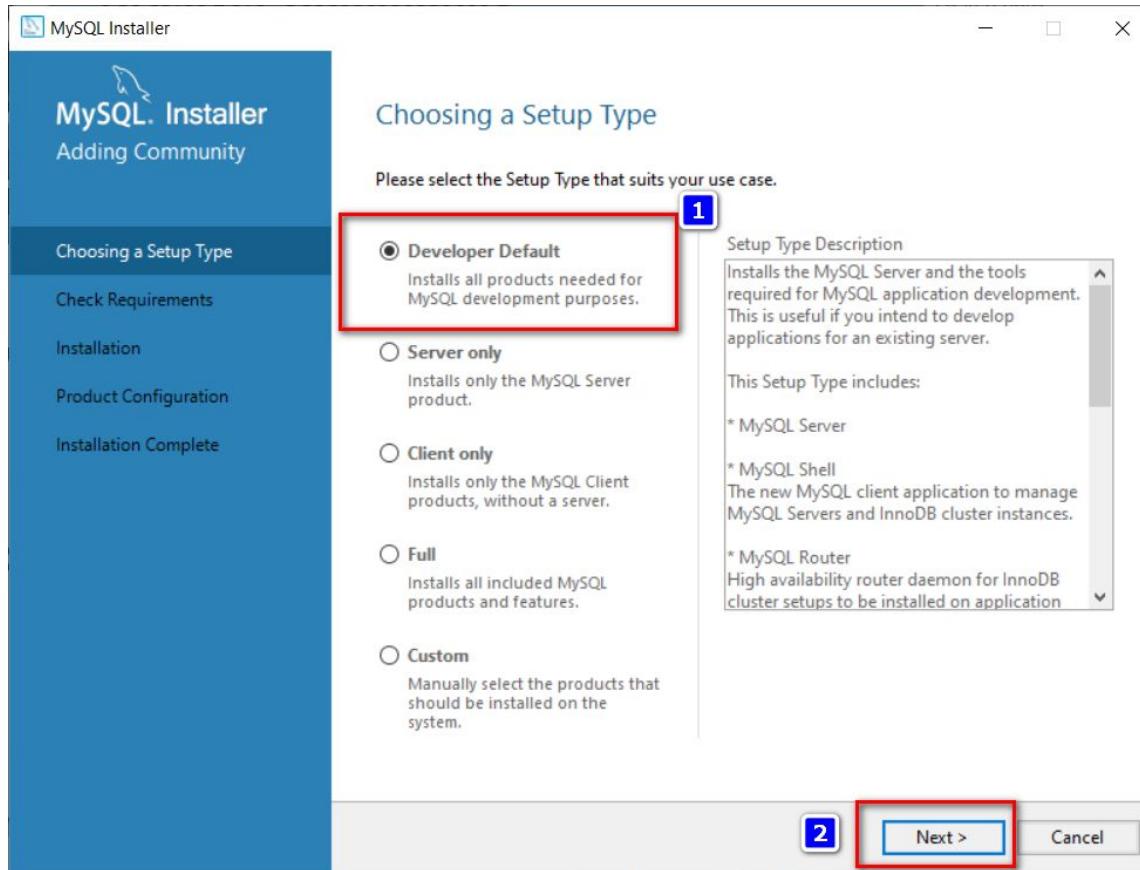
The screenshot shows the MySQL Community Downloads page. At the top, there are tabs for "General Availability (GA) Releases" (which is selected), "Archives", and a help icon. Below the tabs, the title "MySQL Community Server 8.0.21" is displayed. A dropdown menu labeled "Select Operating System:" has "Microsoft Windows" selected and is highlighted with a red box. To the right of the dropdown is a link "Looking for previous GA versions?". Below the dropdown, under "Recommended Download:", there is a section for "MySQL Installer for Windows" featuring a thumbnail image of a Windows logo and a link "Go to Download Page >". A blue box with the number "1" is placed near the "Download" button in the second screenshot.



The screenshot shows the MySQL Community Downloads page for the MySQL Installer 8.0.21. At the top, it says "MySQL Community Downloads" and "MySQL Installer". Below that, there are tabs for "General Availability (GA) Releases" (selected), "Archives", and a help icon. The main content area is titled "MySQL Installer 8.0.21". It asks "Select Operating System:" with "Microsoft Windows" selected. To the right is a link "Looking for previous GA versions?". Below this, there are two download options for "Windows (x86, 32-bit), MSI Installer": one for "mysql-installer-web-community-8.0.21.0.msi" (size 24.5M) and another for "mysql-installer-community-8.0.21.0.msi" (size 427.6M). Each download link is preceded by a "Download" button (highlighted with a red box and a blue box with "2") and a "Signature" link (also highlighted with a red box). A note at the bottom suggests using MD5 checksums and GnuPG signatures to verify package integrity.



Install : developer -> next -> next



Use Legacy

 MySQL Installer

MySQL Server 8.0.21

High Availability

Type and Networking

Authentication Method

Accounts and Roles

Windows Service

Apply Configuration

Authentication Method

Use Strong Password Encryption for Authentication (RECOMMENDED)
MySQL 8 supports a new authentication based on improved stronger SHA256-based password methods. It is recommended that all new MySQL Server installations use this method going forward.

 Attention: This new authentication plugin on the server side requires new versions of connectors and clients which add support for this new 8.0 default authentication (caching_sha2_password authentication).

Currently MySQL 8.0 Connectors and community drivers which use libmysqlclient 8.0 support this new method. If clients and applications cannot be updated to support this new authentication method, the MySQL 8.0 Server can be configured to use the legacy MySQL Authentication Method below.

Use Legacy Authentication Method (Retain MySQL 5.x Compatibility) 1
Using the old MySQL 5.x legacy authentication method should only be considered in the following cases:

- If applications cannot be updated to use MySQL 8 enabled Connectors and drivers.
- For cases where re-compilation of an existing application is not feasible.
- An updated, language specific connector or driver is not yet available.

Security Guidance: When possible, we highly recommend taking needed steps towards upgrading your applications, libraries, and database servers to the new stronger authentication. This new method will significantly improve your security. 2

< Back Next > Cancel



DB GUI : DBeaver Community

https://dbeaver.io

The screenshot shows the DBeaver Community website at https://dbeaver.io. The top navigation bar includes links for Home, About, Download, Sources, Documentation, News, Support, Enterprise Edition, and CloudBeaver. A search bar and social media links for GitHub and Twitter are also present. Below the navigation, a large section titled "Universal Database Tool" describes the software as a free multi-platform database tool for developers, database administrators, analysts, and others who need to work with databases. It supports MySQL, PostgreSQL, SQLite, Oracle, DB2, SQL Server, Sybase, MS Access, Teradata, Firebird, Apache Hive, Phoenix, Presto, etc. A prominent blue "Download" button is highlighted with a red box and arrow. To the right, a screenshot of the DBeaver interface displays a database navigation tree and a detailed entity-relationship (ER) diagram showing tables like Customer, Order, and Product with their relationships.

DBeaver Community

Free Universal Database Tool

Home | About | Download | Sources | Documentation | News | Support | Enterprise Edition | CloudBeaver

Universal Database Tool

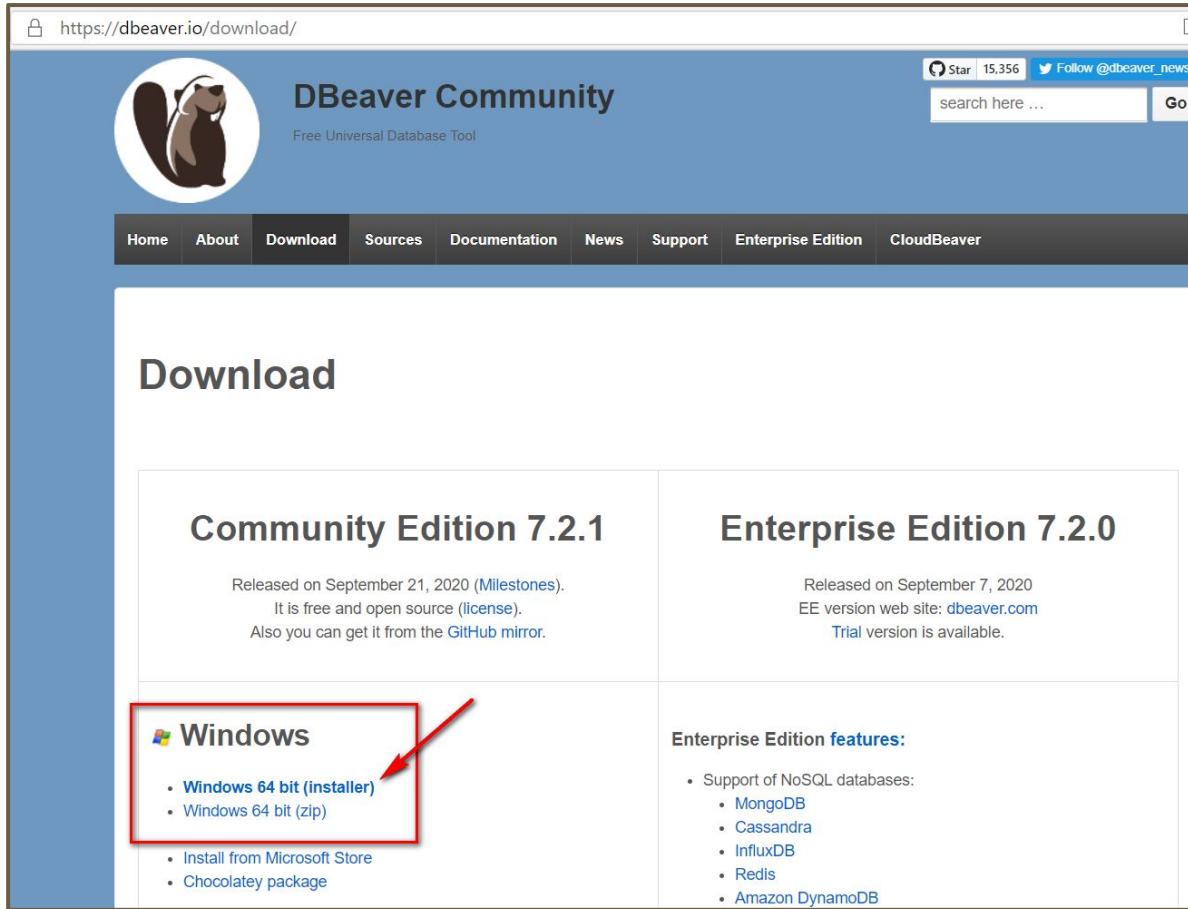
Free multi-platform database tool for developers, database administrators, analysts and all people who need to work with databases. Supports all popular databases: MySQL, PostgreSQL, SQLite, Oracle, DB2, SQL Server, Sybase, MS Access, Teradata, Firebird, Apache Hive, Phoenix, Presto, etc.

[Download](#)

The screenshot of the DBeaver interface shows the Database Navigator on the left, displaying a tree view of database objects for a "chinook" database, including tables like Artist, Album, and Track. On the right, an ER diagram visualizes the database schema with entities such as Customer, Order, and Product, connected by various relationships. A navigation bar at the bottom includes tabs for General, Structure, Diagrams, and Scripts.



Download & Install



The screenshot shows the DBeaver Community download page at <https://dbeaver.io/download/>. The page features a large logo of a beaver on the left, followed by the text "DBeaver Community" and "Free Universal Database Tool". A navigation bar below includes links for Home, About, Download, Sources, Documentation, News, Support, Enterprise Edition, and CloudBeaver. The main content area is titled "Download" and contains two sections: "Community Edition 7.2.1" and "Enterprise Edition 7.2.0". The "Community Edition" section notes its release on September 21, 2020, and mentions it's free and open source. It also links to GitHub. The "Enterprise Edition" section notes its release on September 7, 2020, and mentions a trial version is available. Below these sections, there's a "Windows" section with a red box around the "Windows 64 bit (installer)" link, which has a red arrow pointing to it. Other options in this section include "Windows 64 bit (zip)", "Install from Microsoft Store", and "Chocolatey package". To the right, under "Enterprise Edition features:", a list includes support for MongoDB, Cassandra, InfluxDB, Redis, and Amazon DynamoDB.

DBeaver Community
Free Universal Database Tool

Home | About | **Download** | Sources | Documentation | News | Support | Enterprise Edition | CloudBeaver

Download

Community Edition 7.2.1

Released on September 21, 2020 ([Milestones](#)).
It is free and open source ([license](#)).
Also you can get it from the [GitHub mirror](#).

Windows

- Windows 64 bit (installer)**
- Windows 64 bit (zip)

• Install from Microsoft Store
• Chocolatey package

Enterprise Edition 7.2.0

Released on September 7, 2020
EE version web site: [dbeaver.com](#)
Trial version is available.

Enterprise Edition features:

- Support of NoSQL databases:
 - MongoDB
 - Cassandra
 - InfluxDB
 - Redis
 - Amazon DynamoDB



Install Mysql Python library

```
pip install mysql-connector-python
```



Prepare Database (Mysql)

```
Create database api;  
Use api;  
create table source(  
    id int4,  
    name text,  
    age int4,  
    gender text  
);
```

```
insert into source values(1,'សមបែង',20,'male');  
insert into source values(2,'សមចាយ',22,'male');  
insert into source values(3,'សមគី',23,'female');  
insert into source values(4,'សមសុខ',24,'female');  
commit;  
select * from source
```



Result

The screenshot shows the DBeaver 7.2.0 interface with the following details:

- Top Bar:** File, Edit, Navigate, Search, SQL Editor, Database, Window, Help.
- Toolbar:** Includes icons for New, Open, Save, Commit, Rollback, Auto, and localhost/api.
- Database Navigator:** Shows the database structure. A red box highlights the "Tables" node under the "api" schema, which contains a single table named "source". A yellow button labeled "Refresh" is positioned next to the table node. A blue number "3" is placed above the "source" table entry.
- Script Editor:** A code editor window titled "*<localhost> Script" containing the following SQL script:

```
Create database api;
Use api;
create table source(
id int4,
name text,
age int4,
gender text
);
insert into source values(1,'អារុន',20,'male');
insert into source values(2,'អារុន',22,'male');
insert into source values(3,'អារុន',23,'female');
insert into source values(4,'អារុន',24,'female');
commit;
select * from source;
```

A red box surrounds this entire section.
- Result Grid:** A table grid titled "source" showing the data inserted into the "source" table. The data is:

	id	name	age	gender
1	1	អារុន	20	male
2	2	អារុន	22	male
3	3	អារុន	23	female
4	4	អារុន	24	female

A red box surrounds this grid.



Select -- select_mysql.py

```
import mysql.connector as mysql

try:
    #Connection
    conn = mysql.connect(user='root', password='toor',host='127.0.0.1',database='api')
    cursor = conn.cursor()

    #Select from database
    sql = "select * from source"
    cursor.execute(sql)
    records = cursor.fetchall()
    for row in records:
        print ("Id = {0} , Name = {1}, Age = {2}, Gender = {3}".format(row[0],row[1],row[2],row[3]))

except (Exception, mysql.Error) as error :
    print ("Error while fetching data from Mysql.", error)
finally:
    #closing database connection.
    if(conn):
        cursor.close()
        conn.close()
        print("MySQL connection is closed")
```

Insert -- insert_mysql.py

```
import psycopg2 as pg
try:
    conn =
pg.connect(user="postgres",password="postgres",host="127.0.0.1",port="5432",database="postgres")
    cursor = conn.cursor()
    insert_sql = "insert into source values(%s,%s,%s,%s)"
    data_to_insert = (5, 'สมเจตน์', 25,'male')
    cursor.execute(insert_sql, data_to_insert)
    conn.commit()
    count = cursor.rowcount
    print (count, "Record inserted successfully into source table")
except (Exception, pg.Error) as error :
    if(conn):
        print("Failed to insert record into source table", error)
finally:
    #closing database connection.
    if(conn):
        cursor.close()
        conn.close()
        print("PostgreSQL connection is closed")
```



Update -- update_mysql.py

```
import mysql.connector as mysql

def updateTable(id, newid):
    try:
        conn = mysql.connect(user='root', password='toor',
                             host='127.0.0.1', database='api')
        cursor = conn.cursor()
        sql_update = "Update source set id = %s where id = %s"""""
        cursor.execute(sql_update, (newid, id))
        conn.commit()
        count = cursor.rowcount
        print(count, "Record Updated successfully ")
    except (Exception, pg.Error) as error:
        print("Error in update operation", error)
    finally:
        # closing database conn.
        if (conn):
            cursor.close()
            conn.close()
            print("PostgreSQL conn is closed")

updateTable(5, 10)
```

Delete -- delete_mysql.py

```
import mysql.connector as mysql

def deleteData(id):
    try:
        conn = mysql.connect(user='root', password='toor',
                             host='127.0.0.1', database='api')
        cursor = conn.cursor()
        # Update single record now
        sql_delete_query = "Delete from source where id = %s"
        cursor.execute(sql_delete_query, (id,))
        conn.commit()
        count = cursor.rowcount
        print(count, "Record deleted successfully ")
    except (Exception, pg.Error) as error:
        print("Error in Delete operation", error)
    finally:
        # closing database conn.
        if (conn):
            cursor.close()
            conn.close()
            print("PostgreSQL conn is closed")

deleteData(10)
```

Put it all together

Put it on together (Part 1) -- crud_mysql.py

```
import mysql.connector as mysql
conn = {}
cursor = {}
def connect():
    global conn,cursor
    try:
        conn = mysql.connect( user='root', password='toor',
                            host='127.0.0.1',
                            database='api')
        cursor = conn.cursor()
        cursor.execute( 'SET NAMES utf8;' )
        cursor.execute( 'SET CHARACTER SET utf8;' )
        cursor.execute( 'SET character_set_connection=utf8;' )
        return True
    except (Exception, mysql.Error) as error:
        print("Error ", error)
        return False

def close():
    global conn,cursor
    if(conn):
        cursor.close()
        conn.close()
        print("---- closed connection --" )

def selectAll():
    global conn,cursor
    try:
        sql = "select * from source"
        cursor.execute(sql)
        records = cursor.fetchall()
        for row in records:
            print ("Id = {0} , Name = {1}, Age = {2}, Gender = {3}".format(row[0],row[1],row[2],row[3]))
    except (Exception, mysql.Error) as error :
        print ("Error while fetching data from Mysql." , error)
```



Put it on together (Part 2)-- crud_mysql.py

```
def insert(data_to_insert):
    global conn,cursor
    try:
        insert_sql = "insert into target values(%s,%s,%s,%s)"
        #data_to_insert = (5, 'ਸਮੇਤਾਨੀ', 25,'male')
        cursor.execute(insert_sql, data_to_insert)
        conn.commit()
        count = cursor.rowcount
    except (Exception, mysql.Error) as error :
        print ("Error while fetching data from Mysql.", error)

def updateTable(id, newid):
    global conn,cursor
    try:
        sql_update = "Update target set id = %s where id = %s"""
        cursor.execute(sql_update, (newid,id))
        conn.commit()
        count = cursor.rowcount
        print(count, "Record Updated successfully ")
    except (Exception, mysql.Error) as error:
        print("Error in update operation", error)

def deleteData(id):
    global conn,cursor
    try:
        cursor = conn.cursor()
        sql_delete_query = "Delete from target where id = %s"
        cursor.execute(sql_delete_query, (id,))
        conn.commit()
        count = cursor.rowcount
        print(count, "Record deleted successfully ")
    except (Exception, mysql.Error) as error:
        print("Error in Delete operation", error)
```



Call from other file

Put it on together (Part 3)-- test_mysql.py

```
import crud_mysql as my

my.connect()

data_to_insert = (6, 'สมเจตน์', 26,'male')

my.insert(data_to_insert)

#my.updateTable(6,10)

#my.deleteData(10)

my.selectAll()

my.close()
```





Flask

Prepare Environment

```
pip install flask
```

```
python -m flask --version
```

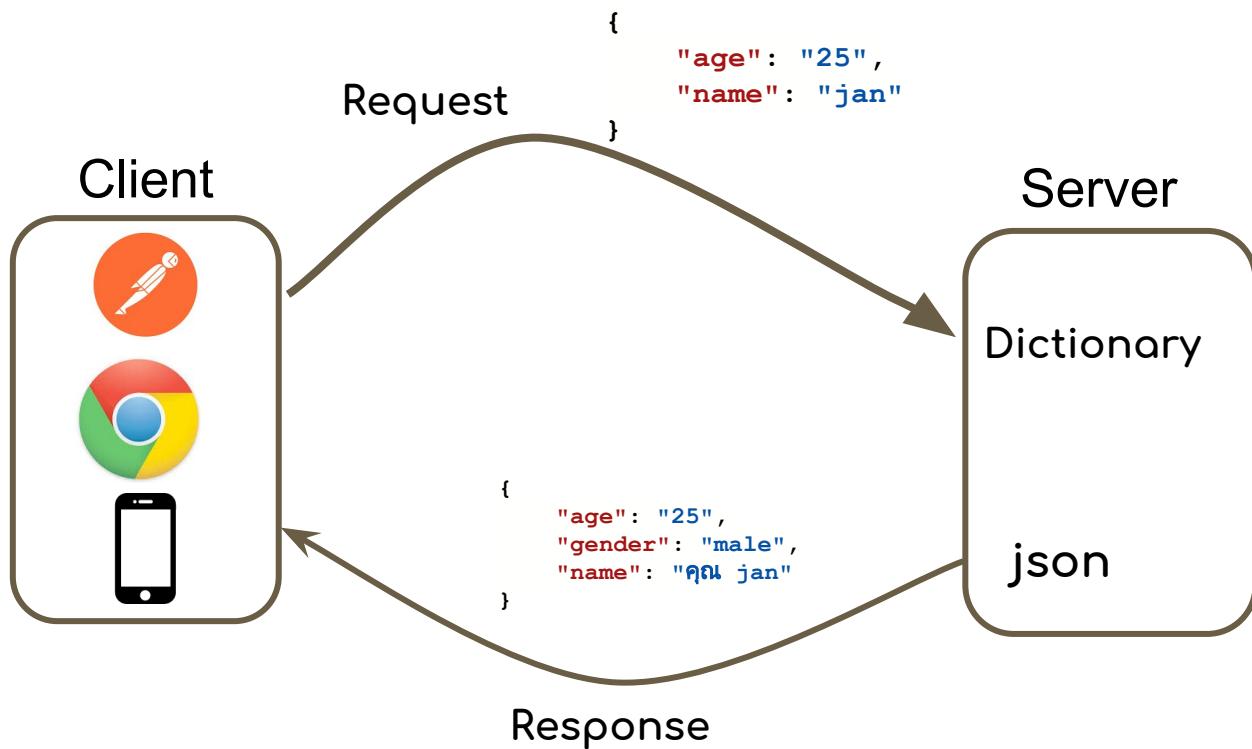
```
Python 3.8.5
```

```
Flask 1.1.2
```

```
Werkzeug 1.0.1
```



API summary



Request with json

Flask API : flask_echo.py

```
from flask import Flask, request, jsonify
import json
app = Flask(__name__)
#test
@app.route('/echo', methods=['POST','GET'])
def foo():
    data = request.json
    print(type(data))
    print()
    print("----- ECHO -----")
    print(data)
    print("-----")
    return jsonify(data)

if __name__ == '__main__':
    app.run(debug=True, port=80, host='0.0.0.0')
```

```
PS C:\Users\tanlu\git\python-django-api> & C:/Python38/python.exe c:/Users/tanlu/git/python-django-api/flask_echo.py
* Serving Flask app "flask_echo" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 319-922-802
* Running on http://0.0.0.0:80/ (Press CTRL+C to quit)
```



Run & Allow Access

```
from flask import Flask, request, jsonify
import json
app = Flask(__name__)

@app.route('/ea/members', methods=['POST','GET'])
def forest_fire():
    data = request.args['account_no']
    print()
    print("-----")
    print(data)
    print("-----")
    return jsonify(data)

@test
@app.route('/echo', method="POST")
def foo():
    data = request.json
    print()
    print("-----")
    print(data)
    print("-----")
    return jsonify(data)

if __name__ == '__main__':
    app.run(debug=True, po
```

Windows Security Alert

Windows Defender Firewall has blocked some features of this app

Windows Defender Firewall has blocked some features of Python on all public and private networks.

Name: Python
Publisher: Python Software Foundation
Path: C:\python38\python.exe

Allow Python to communicate on these networks:

Private networks, such as my home or work network

Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security)

[What are the risks of allowing an app through a firewall?](#)

Allow access **Cancel**

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Serving Flask app "flask_app" (lazy loading)
Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
Debug mode: on
Restarting with stat
Debugger is active!
Debugger PIN: 319-922-802
Running on http://0.0.0.0:80/ (Press CTRL+C to quit)
```



Test with Postman

The screenshot shows the Postman application interface with various numbered steps indicating the process:

- 1: Method dropdown set to GET.
- 2: URL input field containing http://localhost/echo.
- 3: Body tab selected.
- 4: JSON content type selected in the body dropdown.
- 5: JSON payload: { "name": "สมปอง", "age": "25" }.
- 6: Send button.
- 7: Response body: { "age": "25", "name": "สมปอง" }.
- 8: Response status: 200 OK, 600 ms, 208 B.

Below the main interface, there is a preview section with tabs: Pretty, Raw, Preview, Visualize, and JSON. The JSON tab is selected, showing the same response structure.



Recall dictionary data type :

dictionary_test.py

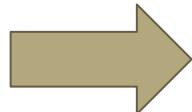
```
# dictionary
data = {'age':'25', 'name':'ເຈນ'}
print(type(data))

#Access Dictionary
print(data["name"])

#Modify Dict
data["name"] = "ນຸ້ມ"
print(data["name"])

# add dict
data["friend"] = "ໄມ"
print(data)

#delete dict
del data["friend"]
print(data)
```



```
<class 'dict'>
ເຈນ
ນຸ້ມ
{'age': '25', 'name': 'ນຸ້ມ', 'friend': 'ໄມ'}
{'age': '25', 'name': 'ນຸ້ມ'}
```



To Stop Running Server -> Ctrl+C

```
127.0.0.1 - - [04/Oct/2020 15:26:21] "GET /echo HTTP/1.1" 200 -
PS C:\Users\tanlu\git\python-django-api> & C:/Python38/python.exe c:/Users/tanlu/git/python-django-api/flask_add_data.py
 * Serving Flask app "flask_add_data" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 319-922-802
 * Running on http://0.0.0.0:80/ (Press CTRL+C to quit)
PS C:\Users\tanlu\git\python-django-api>
```

CTRL + C



Add data: flask_add_data.py

```
from flask import Flask, request, jsonify
import json
app = Flask(__name__)

#test
@app.route('/echo', methods=['POST', 'GET'])
def foo():
    data = request.json
    data["gender"] = "male"
    print()
    print("----- ECHO -----")
    print(data)
    print("-----")
    return jsonify(data)

if __name__ == '__main__':
    app.run(debug=True, port=80, host='0.0.0.0')
```



Test with Postman

The screenshot shows the Postman application interface. At the top, it displays a 'GET' request to 'Flask Echo'. The 'Body' tab is selected, showing a JSON payload:

```
1 {  
2   "name": "สมปอง",  
3   "age": "25"  
4 }  
5
```

The response section shows a 200 OK status with a response time of 711 ms and a response size of 229 B. The response body is displayed in JSON format:

```
1 [  
2   "age": "25",  
3   "gender": "male",  
4   "name": "สมปอง"  
5 ]
```

A red arrow points to the 'gender' field in the response body.



Modify data : flask_add_data.py

```
from flask import Flask, request, jsonify
import json
app = Flask(__name__)

@test
@app.route('/echo', methods=['POST', 'GET'])
def foo():
    data = request.json
    data["gender"] = "male"
    data["name"] = "ମୁଖ" + data["name"]
    print()
    print("----- ECHO -----")
    print(data)
    print("-----")
    return jsonify(data)

if __name__ == '__main__':
    app.run(debug=True, port=80, host='0.0.0.0')
```



Test with Postman

The screenshot shows the Postman application interface. At the top, there's a header with 'GET Flask Echo' and a status indicator (red dot). To the right are buttons for '+', 'No Environment', 'Examples 0', 'BUILD', and a settings icon.

The main workspace has a 'Send' button and a 'Save' button. Below them, tabs include 'Params', 'Auth', 'Headers (8)', 'Body' (which is selected and highlighted in green), 'Pre-req.', 'Tests', and 'Settings'. The 'Cookies' tab is also visible.

The 'Body' section has dropdowns for 'raw' and 'JSON'. The JSON content is:

```
1 {  
2   "name": "สมปอง",  
3   "age": "25"  
4 }  
5
```

Below this, the 'Body' tab is expanded, showing the response details. It includes a globe icon, '200 OK', '590 ms', '248 B', and a 'Save Response' button. The response body is displayed in 'Pretty' format:

```
1 {  
2   "age": "25",  
3   "gender": "male",  
4   "name": "คุณสมปอง"  
5 }
```

A red arrow points to the 'name' field in the response body, which is highlighted in a red box.



Request with parameter

flask_parameter.py

```
from flask import Flask, request, jsonify
import json
app = Flask(__name__)

@app.route('/test', methods=['POST', 'GET'])
def bar():
    data = request.args
    print()
    print(type(data))
    print("----- Get Parameters-----")
    print(data)
    print("-----")
    return jsonify(data)

if __name__ == '__main__':
    app.run(debug=True, port=80, host='0.0.0.0')
```

```
<class 'werkzeug.datastructures.ImmutableMultiDict'>
----- Get Parameters-----
ImmutableMultiDict([('name', 'jan'), ('age', '25')])
-----
```



Test with Postman

The screenshot shows the Postman application interface with the following highlighted elements:

- Method:** GET (highlighted by a red box)
- URL:** http://localhost/test?name=jan&age=25 (highlighted by a red box)
- Send button:** A large blue "Send" button with a dropdown arrow (highlighted by a red box).
- Response Body:** A JSON object with keys "age" and "name". The value for "age" is "25" and for "name" is "jan". (highlighted by a red box)

Below the URL input, there are tabs for Params (selected), Auth, Headers (6), Body, Pre-req., Tests, and Settings. The Params section shows two checked Query Params: "name" with value "jan" and "age" with value "25".

The bottom section displays the response details: 200 OK, 552 ms, 181 B, and a "Save Response" button. The response body is shown in Pretty, Raw, Preview, and Visualize formats, with JSON selected.



ImmutableMultiDict -> Dict

```
<class 'werkzeug.datastructures.ImmutableMultiDict'>
```

```
----- Get Parameters -----
```

```
ImmutableMultiDict([('name', 'jan'), ('age', '25')])
```

```
-----
```

```
from flask import Flask, request, jsonify
app = Flask(__name__)

@app.route('/test', methods=['POST', 'GET'])
def bar():
    data = request.args.to_dict()
    print()
    print(type(data))
    data["gender"] = "male"
    data["name"] = "Qasim " + data["name"]
    print("----- Get Parameters -----")
    print(data)
    print("-----")
    return jsonify(data)

if __name__ == '__main__':
    app.run(debug=True, port=80, host='0.0.0.0')
```



Test with Postman

The screenshot shows the Postman application interface with the following highlights:

- Step 1:** The URL input field contains "http://localhost/test?name=jan&age=25".
- Step 2:** The "Send" button is highlighted.
- Step 3:** The JSON response body is highlighted, showing the following data:

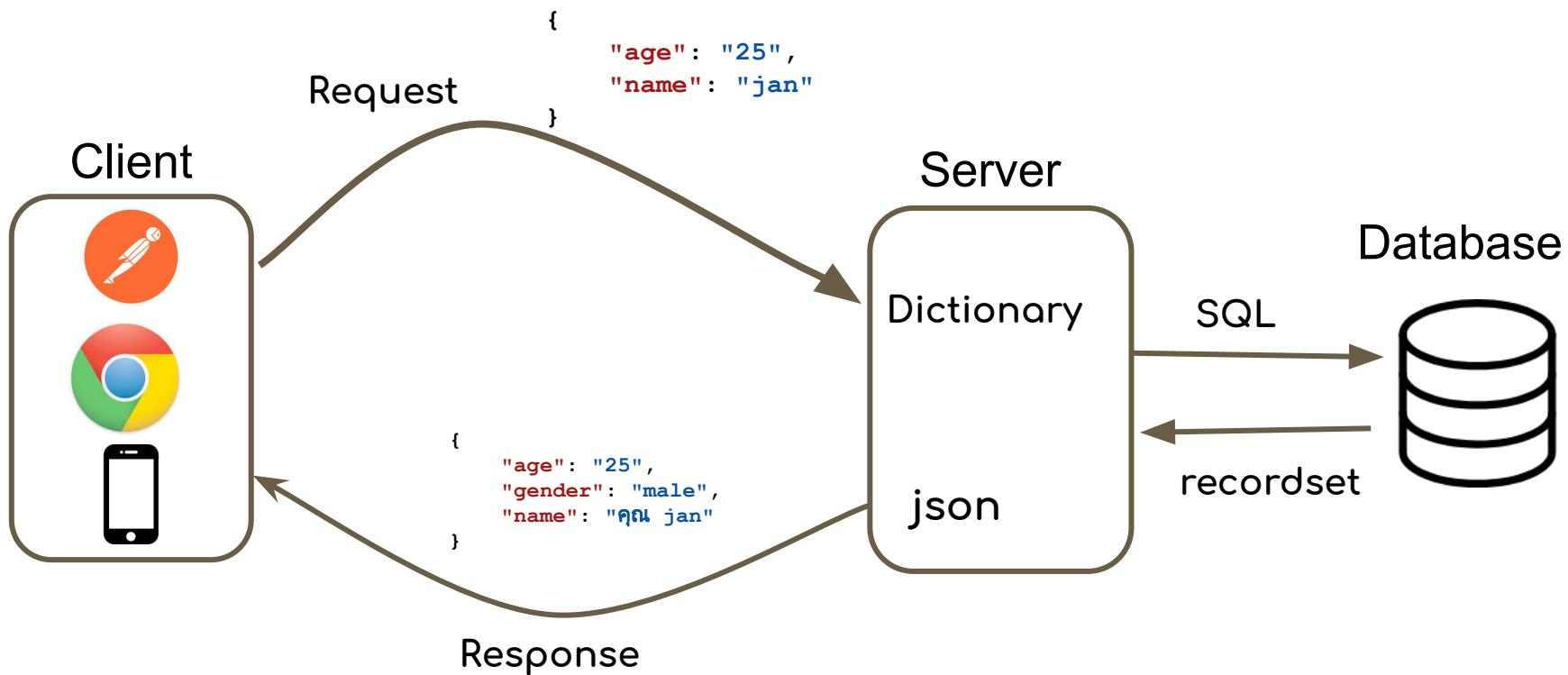
```
1 {  
2   "age": "25",  
3   "gender": "male",  
4   "name": "jan"  
5 }
```

The interface includes tabs for Params, Auth, Headers, Body, Pre-req., Tests, and Settings. The Body tab is selected. The response section shows a status of 200 OK with a response time of 735 ms and a size of 221 B. A "Save Response" button is also visible.

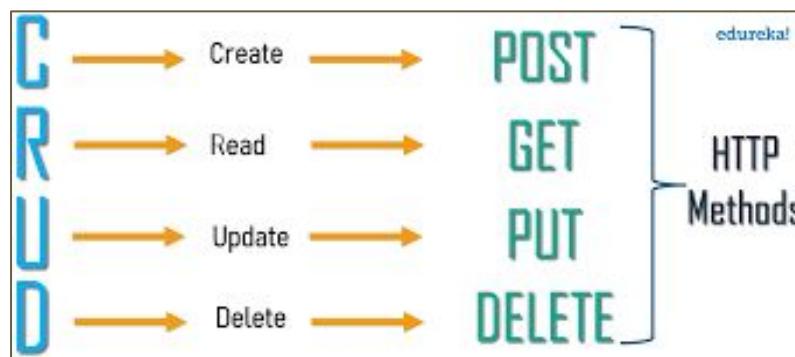
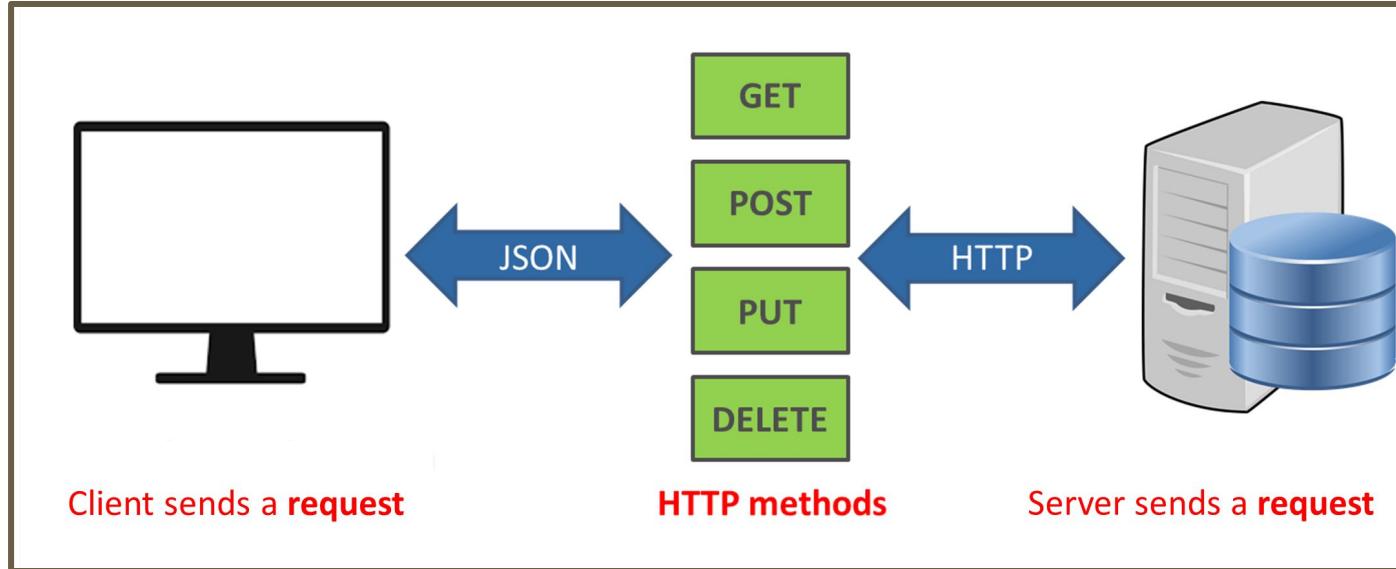


Your First API (MySQL)

API summary



HTTP Request Methods



Create MySQL Library : mysql_lib.py

```
import mysql.connector as mysql
conn = {}
cursor = {}

def connect():
    global conn, cursor
    try:
        conn = mysql.connect(user='root',
password='toor',host='127.0.0.1', database='api')
        cursor = conn.cursor()
        return True
    except (Exception, mysql.Error) as error:
        print("Error ", error)
        return False

def close():
    global conn, cursor
    if(conn):
        cursor.close()
        conn.close()
        print("---- closed connection --")

def select(sql):
    global conn, cursor
    try:
        cursor.execute(sql)
        data = list2dict(cursor)
        return data
    except (Exception, mysql.Error) as error:
        print("Error while fetching data from Mysql.", error)
```

```
def list2dict(cursor):
    desc = cursor.description
    column_names = [col[0] for col in desc]
    data_dict = [dict(zip(column_names, row)) for
row in cursor.fetchall()]
    return data_dict

def executeSQL(sql):
    global conn, cursor
    data = {}
    try:
        cursor.execute(sql)
        conn.commit()
        count = cursor.rowcount
        data["status"] = "Success"
        data["count"] = count
        return data
    except (Exception, mysql.Error) as error:
        print("Error while fetching data from
Mysql.", error)
        data["status"] = "Error"
        data["message"] = str(error)
    finally:
        return data
```

Select All [GET]: flask_api_mysql.py

```
from flask import Flask, request, jsonify
import mysql_lib
app = Flask(__name__)

@app.route('/all', methods=['GET'])
def all():
    result = db_get_all()
    return jsonify(result)

def db_get_all():
    mysql_lib.connect()
    sql = "select * from source"
    data = mysql_lib.select(sql)
    mysql_lib.close()
    return data

if __name__ == '__main__':
    app.run(debug=True, port=80, host='0.0.0.0')
```



GET http://localhost/all

Send

Params Auth Headers (6) Body Pre-req. Tests Settings

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body ▾ 200 OK 554 ms 704 B Save

Pretty Raw Preview Visualize JSON ▾

```
1 {  
2   "age": 20,  
3   "gender": "male",  
4   "id": 1,  
5   "name": "สมปอง"  
6 },  
7 {  
8   "age": 22,  
9   "gender": "male",  
10  "id": 2,  
11  "name": "สมชาย"  
12 },  
13 {  
14   "age": 23,  
15   "gender": "female",  
16 }
```

Select by ID [GET]: flask_api_mysql.py

```
@app.route('/by_id', methods=['GET'])

def by_id():
    data = request.args
    id = data["id"]
    result = db_get_by_id(id)
    return jsonify(result)

def db_get_by_id(id):
    mysql_lib.connect()
    sql = "select * from source where
id={ }".format(id)
    print(sql)
    data = mysql_lib.select(sql)
    mysql_lib.close()
    return data
```



GET http://localhost/by_id?id=2 Send

Params ● Auth Headers (6) Body Pre-req. Tests Settings

Query Params

KEY	VALUE	DESCRIPTION

Body ▾ 200 OK 638 ms 256 B

Pretty Raw Preview Visualize JSON ☰

```
1 {  
2   "age": 22,  
3   "gender": "male",  
4   "id": 2,  
5   "name": "สมชาย"  
6 }  
7 ]  
8 }
```



Insert [POST] : Append to

flask_api_mysql.py

```
@app.route('/insert', methods=['POST'])

def insert():
    data = request.args
    result =
    db_insert(data["id"],data["name"],data["age"],data["gender"])
    return jsonify(result)

def db_insert(id,name,age,gender):
    mysql_lib.connect()
    insert_sql = "insert into source
values ('{}','{}','{}','{}')".format(id,name,age,
gender)
    print(insert_sql)
    data = mysql_lib.executeSQL(insert_sql)
    mysql_lib.close()
    return data
```

mysql_lib.py

```
def executeSQL(sql):
    global conn, cursor
    data = {}
    try:
        cursor.execute(sql)
        conn.commit()
        count = cursor.rowcount
        data["status"] = "Success"
        data["count"] = count
        return data
    except (Exception, mysql.Error) as error:
        print("Error while fetching data from
Mysql.", error)
        data["status"] = "Error"
        data["message"] = str(error)
    finally:
        return data
```



Insert [POST]

Success

Flask Insert 1

POST http://localhost/insert?id=10&name=ໂນງ&age=22&gender=female Send 3

Params ● Auth Headers (7) Body Pre-req. Tests Settings

<input checked="" type="checkbox"/> id	10
<input checked="" type="checkbox"/> name	ໂນງ
<input checked="" type="checkbox"/> age	22
<input checked="" type="checkbox"/> gender	female

Key Value Description

Body ▾ 2

Pretty Raw Preview Visualize JSON 4

```
1 {  
2   "count": 1,  
3   "status": "Success"  
4 }
```

Fail

POST http://localhost/insert?id=10&name=ໂນງ&age=22&gender=female Send 1

Params ● Auth Headers (7) Body Pre-req. Tests Settings

<input checked="" type="checkbox"/> id	10
<input checked="" type="checkbox"/> name	ໂນງ
<input checked="" type="checkbox"/> age	22
<input checked="" type="checkbox"/> gender	female

Key Value Description

Body ▾ 2

Pretty Raw Preview Visualize JSON 3

```
1 {  
2   "message": "1146 (42S02): Table 'api.source1' doesn't exist",  
3   "status": "Error"  
4 }
```

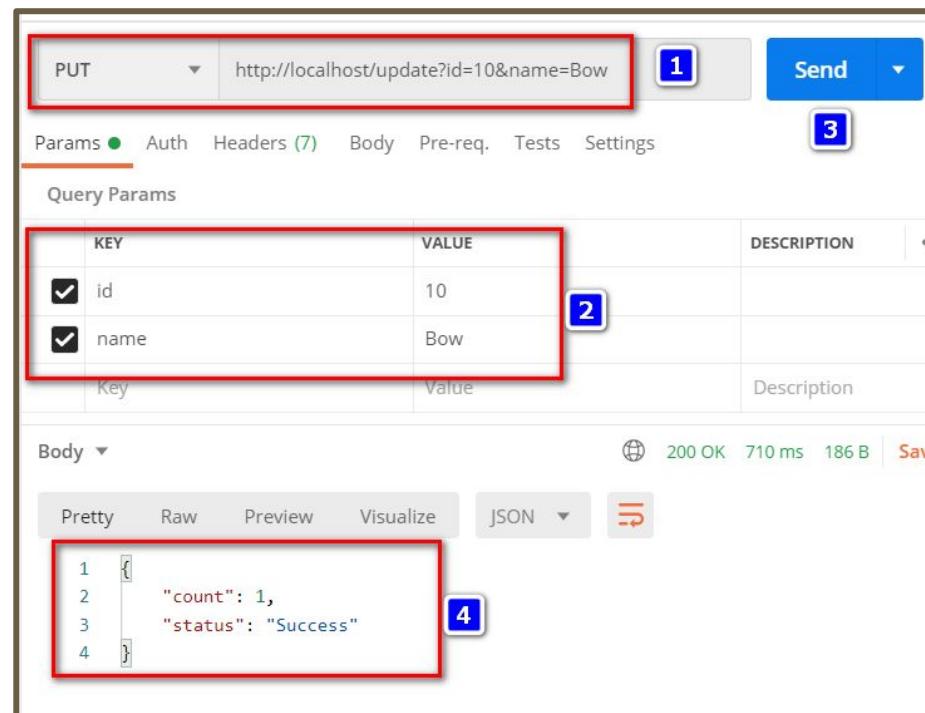


Update [PUT]: Append to

flask_api_mysql.py

```
# Update
@app.route('/update', methods=['PUT'])
def update():
    data = request.args
    result = db_update(data["id"], data["name"])
    return jsonify(result)

def db_update(id, name):
    mysql_lib.connect()
    sql = "update source set name='{}' where id = {}".format(name, id)
    print(sql)
    data = mysql_lib.executeSQL(sql)
    mysql_lib.close()
    return data
```



PUT http://localhost/update?id=10&name=Bow 1 Send 2

Params ● Auth Headers (7) Body Pre-req. Tests Settings 3

Query Params

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> id	10	2
<input checked="" type="checkbox"/> name	Bow	

Body ▼

Pretty Raw Preview Visualize 4 JSON

```
1 {
2   "count": 1,
3   "status": "Success"
4 }
```



Delete [DELETE]: Append to flask_api_mysql.py

```
# Delete
@app.route('/delete', methods=['DELETE'])
def delete():
    data = request.args
    result = db_delete(data["id"])
    return jsonify(result)

def db_delete(id):
    mysql_lib.connect()
    sql = "delete from source where id
    ={ }".format(id)
    print(sql)
    data = mysql_lib.executeSQL(sql)
    mysql_lib.close()
    return data
```



Flask DELETE

DELETE http://localhost/delete?id=10

Params Auth Headers (6) Body Pre-req. Tests Settings

Query Params

KEY	VALUE
<input checked="" type="checkbox"/> id	10

Body

Pretty Raw Preview Visualize JSON

```
1 {  
2   "count": 1,  
3   "status": "Success"  
4 }
```

200 OK 659 ms 186 B Save

1 2 3 4



Access our own API via Python

flask_api_call.py

Run with new
Terminal!

```
import requests
URL = "http://localhost/all"

r = requests.get(url = URL)
data = r.json()

print(data)
print("----- 1 -----")
print(data[1])
print("----- 2 -----")
print(data[2])
print(data[2] ["age"])
print(data[2] ["name"])
print("----- Loop -----")
for d in data:
    print(d)
print("-----END LOOOP-----")
```

```
ps C:\Users\tanlu\git\python-django-api> python flask_api_call.py
[{"age": 20, "gender": "male", "id": 1, "name": "\udcda"}, {"age": 22, "gender": "male", "id": 2, "name": "\udcda"}, {"age": 23, "gender": "female", "id": 3, "name": "\udcda"}, {"age": 24, "gender": "female", "id": 4, "name": "\udcda"}, {"age": 26, "gender": "male", "id": 6, "name": "\udcda"}]
----- 1 -----
{"age": 22, "gender": "male", "id": 2, "name": "\udcda"}
----- 2 -----
{"age": 23, "gender": "female", "id": 3, "name": "\udcda"}
23
\udcda
----- Loop -----
{"age": 20, "gender": "male", "id": 1, "name": "\udcda"}, {"age": 22, "gender": "male", "id": 2, "name": "\udcda"}, {"age": 23, "gender": "female", "id": 3, "name": "\udcda"}, {"age": 24, "gender": "female", "id": 4, "name": "\udcda"}, {"age": 26, "gender": "male", "id": 6, "name": "\udcda"}
-----END LOOOP-----
```



API Lab

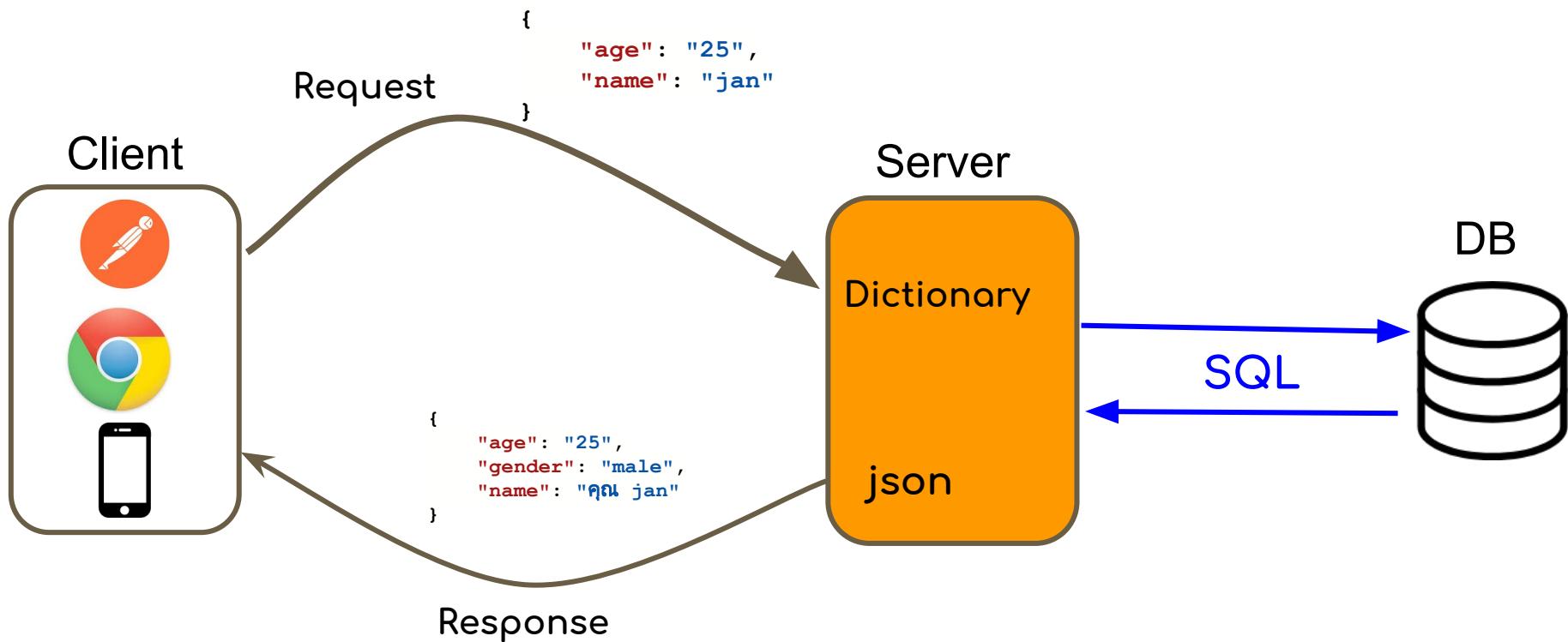
1. Create employee table with
 - a. emp_id, name, department, salary, emp_start
2. Create API : CRUD
3. Create Python that can call that API



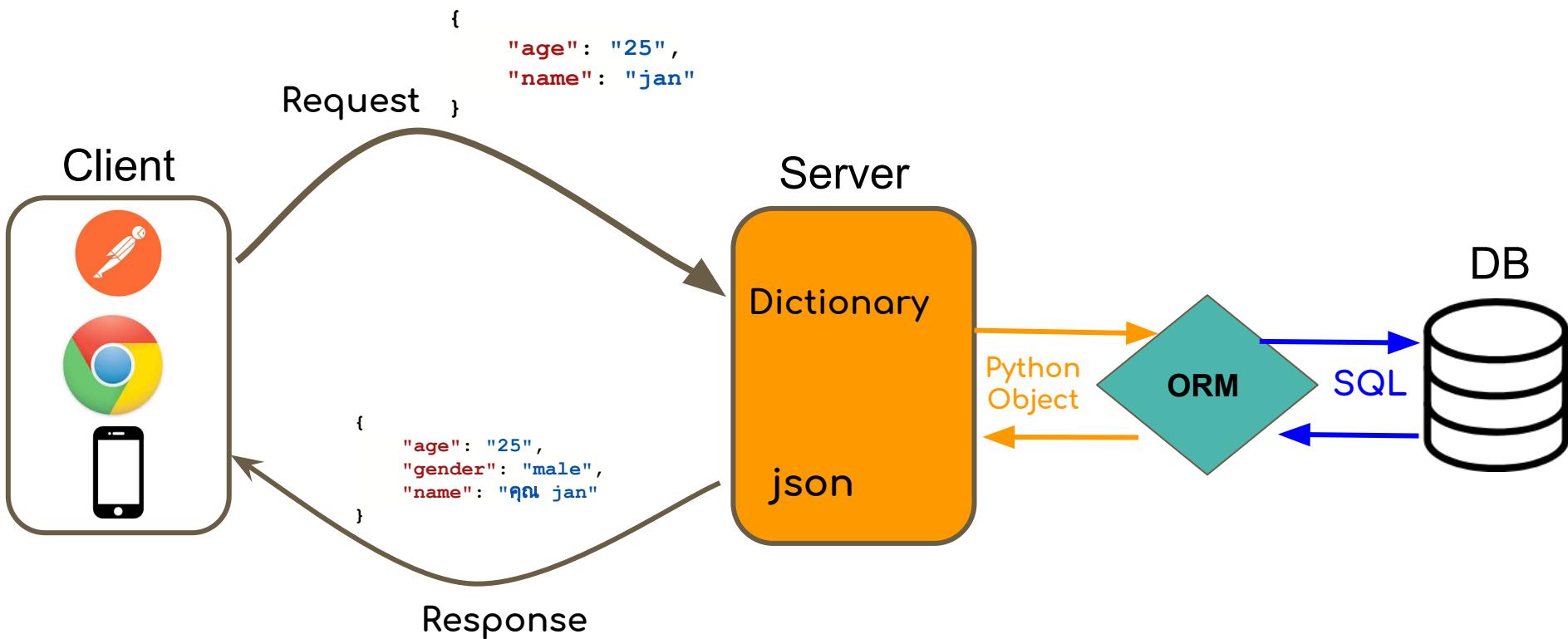


API : ORM with Django

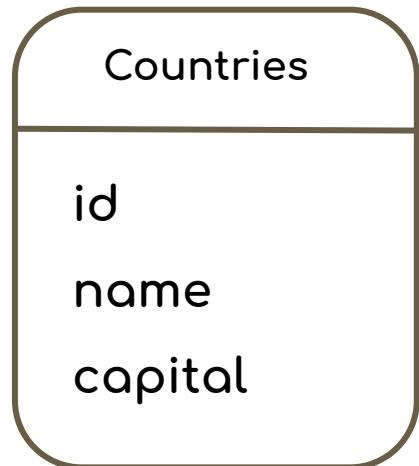
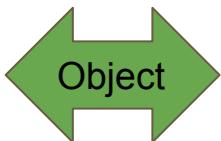
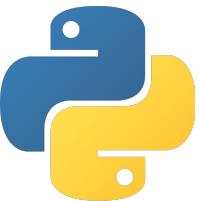
Traditional Architecture



ORM = Object Relational Mapping



ORM Concept



Object



Countries		
id	name	capital
1	Thailand	Bangkok
2	Austria	Vienna
3	Loas	Vienchan

Table



Create Virtual Environment

```
python -m venv venv  
.\venv\Scripts\activate
```

```
PS C:\Users\tanlu\git\python-django-api\venv\Scripts> .\activate  
(venv) PS C:\Users\tanlu\git\python-django-api\venv\Scripts> pip install django  
Collecting django
```



Django Library

- django
- Djangorestframework
- django-cors-headers

```
pip install django  
pip install djangorestframework  
pip install django-cors-headers
```

Check Version

```
python -m django --version  
3.1.2
```



Create Django Project

```
django-admin startproject PythonDjango  
cd PythonDjango  
dir
```

```
(venv) PS C:\Users\tanlu\git\python-django-api> django-admin startproject PythonDjango  
(venv) PS C:\Users\tanlu\git\python-django-api> cd PythonDjango  
(venv) PS C:\Users\tanlu\git\python-django-api\PythonDjango> dir  
  
Directory: C:\Users\tanlu\git\python-django-api\PythonDjango  
  
Mode LastWriteTime      Length Name  
----   
d----  10/4/2020 10:56 PM          PythonDjango  
-a---  10/4/2020 10:56 PM    690 manage.py
```



Create App

```
python manage.py startapp countries  
dir
```

```
(venv) PS C:\Users\tanlu\git\python-django-api\PythonDjango> python manage.py startapp countries  
(venv) PS C:\Users\tanlu\git\python-django-api\PythonDjango> dir
```

```
Directory: C:\Users\tanlu\git\python-django-api\PythonDjango
```

Mode	LastWriteTime	Length	Name
----	-----	-----	
d----	10/4/2020 10:59 PM		countries
d----	10/4/2020 10:59 PM		PythonDjango
-a---	10/4/2020 10:56 PM	690	manage.py



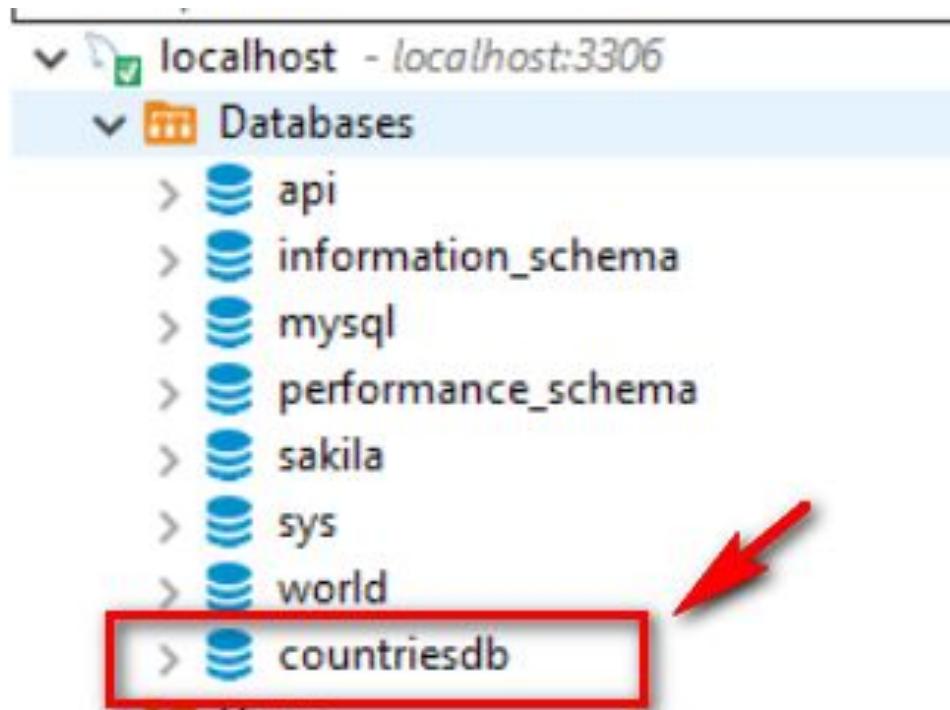
Register app with Django

PythonDjango/PythonDjango/settings.py

```
INSTALLED_APPS = [
    'django.contrib.admin',
    ...
    'django.contrib.staticfiles',
    'rest_framework',
    'countries.apps.CountriesConfig',
    'corsheaders'
]
MIDDLEWARE = [
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.security.SecurityMiddleware',
    ...
]
```



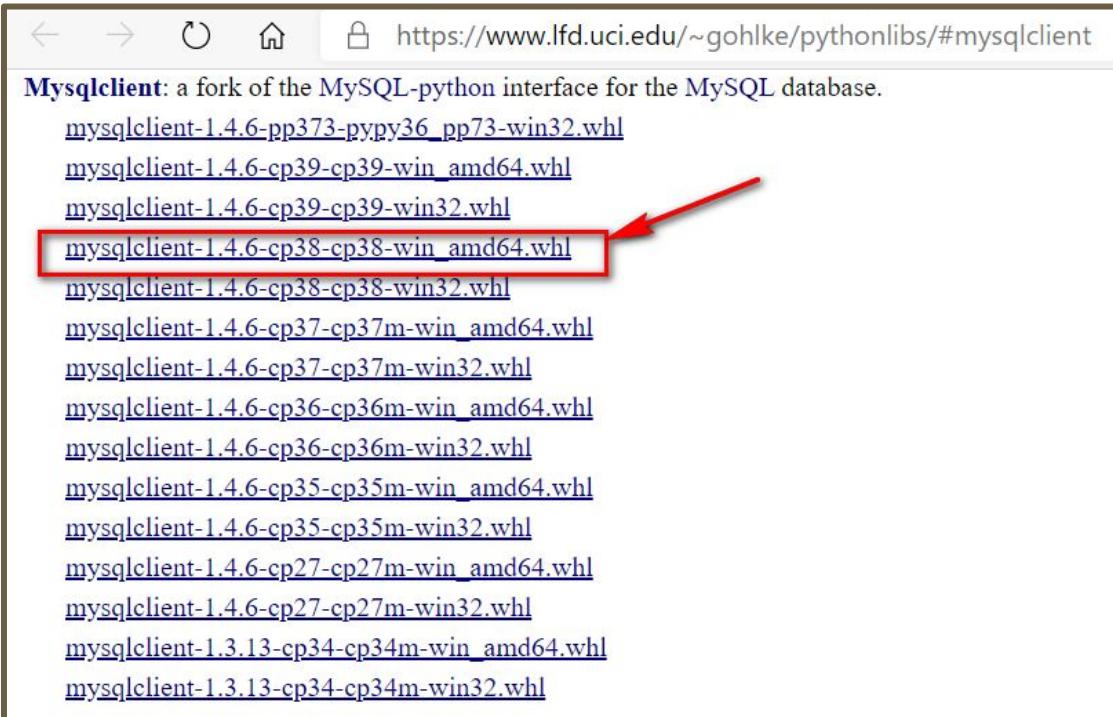
On MySQL Create “countriesdb” Database



Install MySQL Client for Django

<https://www.lfd.uci.edu/~gohlke/pythonlibs/#mysqlclient>

mysqlclient-1.4.6-cp38-cp38-win_amd64.whl



https://www.lfd.uci.edu/~gohlke/pythonlibs/#mysqlclient

Mysqlclient: a fork of the MySQL-python interface for the MySQL database.

- [mysqlclient-1.4.6-pp373-pypy36_pp73-win32.whl](#)
- [mysqlclient-1.4.6-cp39-cp39-win_amd64.whl](#)
- [mysqlclient-1.4.6-cp39-cp39-win32.whl](#)
- [mysqlclient-1.4.6-cp38-cp38-win_amd64.whl](#)
- [mysqlclient-1.4.6-cp38-cp38-win32.whl](#)
- [mysqlclient-1.4.6-cp37-cp37m-win_amd64.whl](#)
- [mysqlclient-1.4.6-cp37-cp37m-win32.whl](#)
- [mysqlclient-1.4.6-cp36-cp36m-win_amd64.whl](#)
- [mysqlclient-1.4.6-cp36-cp36m-win32.whl](#)
- [mysqlclient-1.4.6-cp35-cp35m-win_amd64.whl](#)
- [mysqlclient-1.4.6-cp35-cp35m-win32.whl](#)
- [mysqlclient-1.4.6-cp27-cp27m-win_amd64.whl](#)
- [mysqlclient-1.4.6-cp27-cp27m-win32.whl](#)
- [mysqlclient-1.3.13-cp34-cp34m-win_amd64.whl](#)
- [mysqlclient-1.3.13-cp34-cp34m-win32.whl](#)



Install Mysql Client

```
pip install mysqlclient-1.4.6-cp38-cp38-win32.whl
```

```
(venv) PS C:\Users\tanlu\git\python-django-api\PythonDjango> pip install .\mysqlclient-1.4.6-cp38-cp38-win_amd64.whl
Processing c:\users\tanlu\git\python-django-api\pythondjango\mysqlclient-1.4.6-cp38-cp38-win_amd64.whl
Installing collected packages: mysqlclient
Successfully installed mysqlclient-1.4.6
WARNING: You are using pip version 20.1.1; however, version 21.0.1 is available.
```



Config Database Connection

PythonDjango/PythonDjango/settings.py

```
DATA BASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.mysql',  
        'NAME': 'countriesdb',  
        'USER': 'root',  
        'PASSWORD': 'toor',  
        'HOST': 'localhost',  
        'PORT': '3306'  
    }  
}
```



Create Model file:

PythonDjango/countries/models.py

```
from django.db import models

# Create your models here.

class Countries(models.Model):
    name=models.CharField(max_length=50,blank=False,default='')
    capital = models.CharField(max_length=50,blank=False,default='')

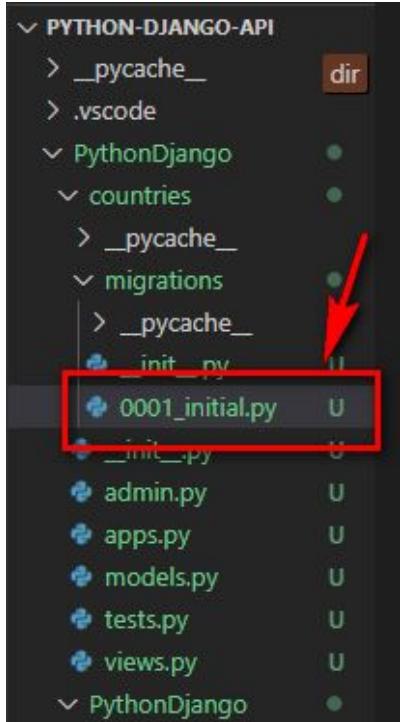
    class Meta:
        ordering = ('id',)
```



Migration model DB => Create Model File

```
python manage.py makemigrations countries
```

It created model file: PythonDjango/countries/migration/**0001_initial.py**



```
# Generated by Django 3.1.2 on 2020-10-04 16:33

from django.db import migrations, models


class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='Countries',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False,
verbose_name='ID')),
                ('name', models.CharField(default='', max_length=50)),
                ('capital', models.CharField(default='', max_length=50)),
            ],
            options={
                'ordering': ('id',),
            },
        ),
    ]
```

Apply Migration Model (Create table in DB)

```
python manage.py migrate countries
```

Operations to perform:

 Apply all migrations: countries

Running migrations:

 Applying countries.0001_initial... OK

The screenshot shows the MySQL Workbench interface with the following details:

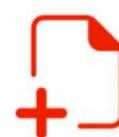
- Connection: localhost - localhost:3306
- Databases:
 - api
 - information_schema
 - mysql
 - performance_schema
 - sakila
 - sys
 - world
 - countriesdb** (selected, highlighted with a red border)
 - Tables
 - countries_countries (16K)
 - django_migrations (16K)
 - Views
 - Indexes
 - Procedures
 - Triggers
 - Events



Working with Object

ORM shell

```
python manage.py shell
```



CREATE



READ



UPDATE



DELETE

C

R

U

D

```
(venv) PS C:\users\tanlu\git\python-django-api\PythonDjango> python manage.py shell
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win3
2
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> 
```



Create : Insert

```
from countries.models import Countries

r1 = Countries(name='Afghanistan', capital='Kabul')
r1.save()
r2 = Countries(name='Austria', capital='Vienna')
r2.save()
r3 = Countries(name='Thailand', capital='Bangkok')
r3.save()
r4 = Countries(name='Laos', capital='Vientiane')
r4.save()
```

```
>>> from countries.models import Countries
>>> r1 = Countries(name='Afghanistan', capital='Kabul')
>>> r1.save()
>>> r2 = Countries(name='Austria', capital='Vienna')
>>> r2.save()
>>> r3 = Countries(name='Thailand', capital='Bangkok')
>>> r3.save()
>>> r4 = Countries(name='Laos', capital='Vientiane')
>>> r4.save()
```



Select ALL

```
call = Countries.objects.all()  
call
```

```
>>> call = Countries.objects.all()  
>>> call  
<QuerySet [<Countries: Thailand>, <Countries: Lao>, <Countries: Vietnam>, <Countries: AFGHANI  
STAN>, <Countries: AFGHANISTAN>, <Countries: AUSTRIA>]>  
>>> []
```



Select By Name : where

```
c1 =  
Countries.objects.get(name="Thailand")  
c1.capital
```

```
>>> c1 = Countries.objects.get(name="Thailand")  
>>> c1.capital  
'Bangkok'  
>>>
```



Select Filter (contains) : Like

```
c1 = Countries.objects.filter(name__contains="an")  
c1
```

```
>>> c1 = Countries.objects.filter(name__contains="an")  
>>> c1  
<QuerySet [<Countries: Thailand>, <Countries: Afghanistan>, <Countries: Afghanistan>]>  
>>> █
```



Select Filter Multiple Conditions : where and

```
c1 = Countries.objects.filter(id=1, capital="Bangkok")  
c1
```

```
>>> c1 = Countries.objects.filter(id=1, capital="Bangkok")  
>>> c1  
<QuerySet [<Countries: Thailand>]>  
>>> |
```



Update One

```
c1 = Countries.objects.get(name="Thailand")
c1.capital = "กรุงเทพฯ"
c1.capital
c1.save()
```

```
>>> c1 = Countries.objects.get(name="Thailand")
>>> c1.capital = "กรุงเทพฯ"
>>> c1.capital
'กรุงเทพฯ'
>>> c1.save()
```



Delete Multiple

```
c6 = Countries.objects.filter(name='Austria')
c6.delete()
call
```

```
>>> c6 = Countries.objects.filter(name='Austria')
>>> c6
<QuerySet [<Countries: AUSTRIA>, <Countries: Austria>, <Countries: Austria>, <Countries: Austria>]>
>>> c6.delete()
(4, {'countries.Countries': 4})
>>> call
<QuerySet [<Countries: Thailand>, <Countries: Lao>, <Countries: Vietnam>, <Countries: AFGHANISTAN>, <Countries: AFGHANISTAN>, <Countries: Afghanistan>, <Countries: Afghanistan>]>
>>> |
```



Delete One

```
c5 = Countries.objects.get(name='lao')
c5.delete()
call
```

```
>>> c5 = Countries.objects.get(name='lao')
>>> c5.delete()
(1, {'countries.Countries': 1})
>>> call
<QuerySet [<Countries: Thailand>, <Countries: Vietnam>, <Countries: AFGHANISTAN>, <Countries: AFGHANISTAN>, <Countries: Afghanistan>, <Countries: Afghanistan>]>
```



Exit shell

```
exit()
```

```
>>> exit()  
(venv) PS C:\Users\tanlu\git\python-django-api\PythonDjango> █
```



API Django

Serializer (Model <-> JSON)

PythonDjango/countries/serializers.py

```
from rest_framework import serializers
from countries.models import Countries

class CountriesSerializer(serializers.ModelSerializer):
    class Meta:
        model = Countries
        fields = ('id', 'name', 'capital')
```

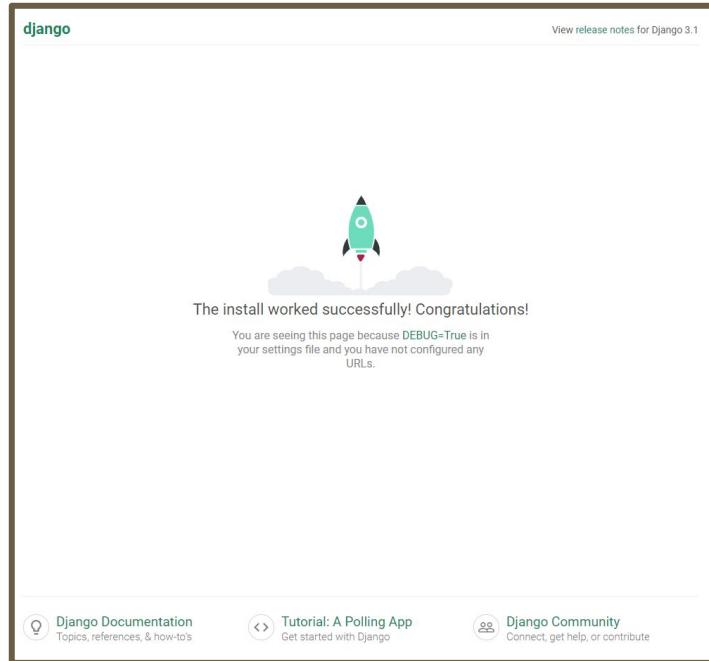
```
# Run Migration
python manage.py migrate
```



Run Server

```
python manage.py runserver
```

<http://127.0.0.1:8000/>



Django Admin

```
python manage.py createsuperuser
```

<http://127.0.0.1:8000/admin>

The screenshot shows the Django administration login interface. It features a teal header bar with the text "Django administration". Below it is a light gray form area. The first field is labeled "Username:" with the value "admin" entered. The second field is labeled "Password:" with the value "....." (represented by four dots). At the bottom center is a blue "Log in" button.

Username : admin
Password : admin



Modify View : /countries/view.py

```
from django.shortcuts import render
from django.http.response import JsonResponse
from rest_framework.parsers import JSONParser
from rest_framework import status

from countries.models import Countries
from countries.serializers import CountriesSerializer
from rest_framework.decorators import api_view

@api view(['GET','POST'])
def countries_list(request):
    if request.method == 'GET':
        countries = Countries.objects.all()
        name = request.GET.get('name',None)
        if name is not None:
            countries = countries.filter(name__icontains=name)
        countries_serializer = CountriesSerializer(countries,many=True)
        return JsonResponse(countries_serializer.data,safe=False)
    elif request.method == 'POST':
        countries_data = JSONParser().parse(request)
        countries_serializer = CountriesSerializer(data=countries_data)
        if countries_serializer.is_valid():
            countries_serializer.save()
            return JsonResponse(countries_serializer.data,status=status.HTTP_201_CREATED)
        return JsonResponse(countries_serializer.errors, status=status.HTTP_400_BAD_REQUEST)

@api view(['GET','PUT','DELETE'])
def countries_detail(request,pk):
    try:
        countries = Countries.objects.get(pk=pk)
    except Countries.DoesNotExist:
        return JsonResponse({'message':'Countries does not exist'},status=status.HTTP_404_NOT_FOUND)
    if request.method == 'GET':
        countries_serializer = CountriesSerializer(countries)
        return JsonResponse(countries_serializer.data)
    elif request.method == 'PUT':
        countries_data = JSONParser().parse(request)
        countries_serializer = CountriesSerializer(countries,data=countries_data)
        if countries_serializer.is_valid():
            countries_serializer.save()
            return JsonResponse(countries_serializer.data)
        return JsonResponse(countries_serializer.errors, status=status.HTTP_400_BAD_REQUEST)
    elif request.method == 'DELETE':
        countries.delete()
        return JsonResponse({'message':'Countries was deleted successfully'},status=status.HTTP_204_NO_CONTENT)
```



Create : /countries/urls.py

```
from django.conf.urls import url
from countries import views
urlpatterns = [
    url(r'^api/countries$',views.countries_list),
    url(r'^api/countries/(?P<pk>[0-9]+)$',views.countries_detail)
]
```



Modify admin: /countries/admin.py

```
from django.contrib import admin
from .models import Countries
# Register your models here.
admin.site.register(Countries)
```



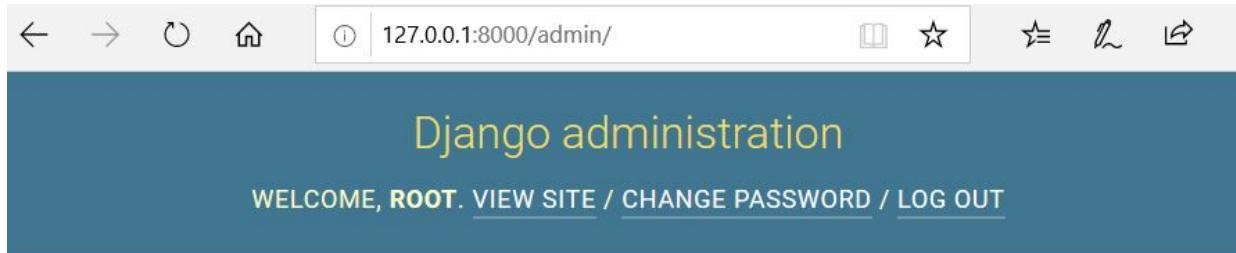
Create : /PythonDjango/urls.py

```
from django.contrib import admin
from django.urls import path
from django.conf.urls import url,include

urlpatterns = [
    path('admin/', admin.site.urls),
    url(r'^',include('countries.urls')) ,
]
```



Check : 127.0.0.1:8000/admin



A screenshot of a web browser showing the Django administration login page. The address bar displays "127.0.0.1:8000/admin/". The main title is "Django administration" with a subtitle "WELCOME, ROOT. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)".

Site administration

AUTHENTICATION AND AUTHORIZATION

[Groups](#)

[+ Add](#) [Change](#)

[Users](#)

[+ Add](#) [Change](#)

COUNTRIES

[Countries](#)

[+ Add](#) [Change](#)



Add Countries ie., Thailand, Bangkok

Django administration

WELCOME, ADI

Home > Countries > Countries > Add countries

AUTHENTICATION AND AUTHORIZATION

- Groups [+ Add](#)
- Users [+ Add](#)

COUNTRIES

- Countries [+ Add](#)

Add countries 2

Name: 3

Capital:

[Save and add another](#)

1

2

3



View inserted data

Django administration

Home > Countries > Countries

AUTHENTICATION AND AUTHORIZATION

Groups	+ Add
Users	+ Add

COUNTRIES

Countries	+ Add
Countress	+ Add

Select countries to change

Action: ----- Go 0 of 3 selected

COUNTRIES

Countries object (1)

Countries object (2)

Countries object (3)

3 countries

1

2



Modify Model file: /countries/models.py

```
from django.db import models

# Create your models here.

class Countries(models.Model):
    name=models.CharField(max_length=50,blank=False,default='')
    capital = models.CharField(max_length=50,blank=False,default='')

    def __str__(self):
        return self.name

    class Meta:
        ordering = ('id',)
```

C:\Users\tanlu\git\python-django-api\PythonDjango\countries\models.py changed, reloading.
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
October 05, 2020 - 00:26:09
Django version 3.1.2, using settings 'PythonDjango.settings'



Refresh Admin Page

Django administration

Home > Countries > Countries

AUTHENTICATION AND AUTHORIZATION

Groups	+ Add
Users	+ Add

COUNTRIES

Countries	+ Add
-----------	-------

Select countries to change

Action: ----- Go 0 of 3 selected

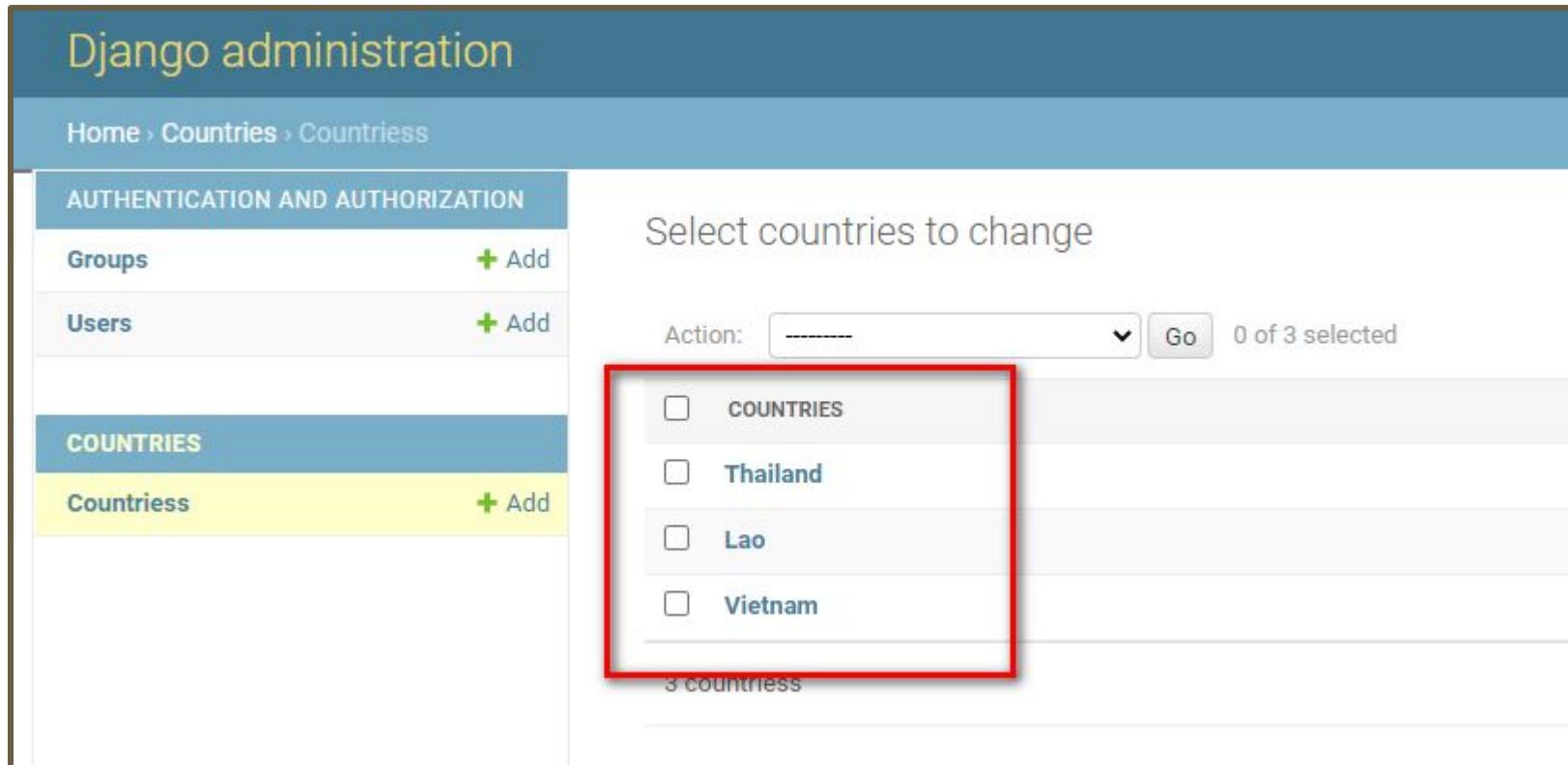
COUNTRIES

Thailand

Lao

Vietnam

3 countries



Test with Postman : Select All

GET <http://127.0.0.1:8000/api/countries>

The screenshot shows the Postman application interface. At the top, there is a header with 'GET' and the URL 'http://127.0.0.1:8000/api/countries'. Below the header, there are tabs for 'Params', 'Auth', 'Headers (6)', 'Body', 'Pre-req.', 'Tests', and 'Settings'. The 'Params' tab is currently selected. Under 'Query Params', there is a table with columns 'KEY', 'VALUE', and 'DESCRIPTION'. The table has one row with 'Key' and 'Value' columns. Below the table, there is a 'Body' section with tabs for 'Pretty', 'Raw', 'Preview', 'Visualize', and 'JSON'. The 'JSON' tab is selected, showing a JSON response. The response is a list of three countries: Thailand, Lao, and Vietnam, each with an id, name, and capital. The JSON output is as follows:

```
1 [  
2 {  
3   "id": 1,  
4   "name": "Thailand",  
5   "capital": "Bangkok"  
6 },  
7 {  
8   "id": 2,  
9   "name": "Lao",  
10  "capital": "Viencang"  
11 },  
12 {  
13   "id": 3,  
14   "name": "Vietnam",  
15   "capital": "Hanoi"  
16 }  
17 ]
```



Test with Postman : Select

GET <http://127.0.0.1:8000/api/countries/2>

The screenshot shows the Postman application interface. At the top, there is a header with 'GET' and the URL 'http://127.0.0.1:8000/api/countries'. Below the header, there is a navigation bar with tabs: Params (which is selected), Auth, Headers (6), Body, Pre-req., Tests, and Settings. Under the 'Params' tab, there is a section for 'Query Params' with a table:

KEY	VALUE	DESCRIPTION
Key	Value	Description

Below the table, there is a 'Body' section with a dropdown menu set to 'Pretty'. The response body is displayed in a JSONpretty-printed format:

```
1 [  
2 {  
3   "id": 1,  
4   "name": "Thailand",  
5   "capital": "Bangkok"  
6 },  
7 {  
8   "id": 2,  
9   "name": "Lao",  
10  "capital": "Vienschang"  
11 },  
12 {  
13   "id": 3,  
14   "name": "Vietnam",  
15   "capital": "Hanoi"  
16 }  
17 ]
```

At the bottom right of the response area, there are status indicators: 200 OK, 33 ms, 443 B, and a 'Save' button.



Test with Postman : Insert

POST http://127.0.0.1:8000/api/countries

Django POST Examples 0 ▾ | BUILD

POST http://127.0.0.1:8000/api/countries ▾

Send ▾

Params Authorization Headers (8) Body (●)

none form-data x-www-form-urlencoded raw binary GraphQL JSON

1 {"name": "USA",
2 "capital": "Washington"
3 }

Body Cookies Headers (9) Test Results

201 Created 28 ms 342 B Sav

Pretty Raw Preview Visualize JSON

1 {
2 "id": 4,
3 "name": "USA",
4 "capital": "Washington"
5 }

1

2

3

4

```
{  
  "name": "USA",  
  "capital": "Washington"  
}
```



Test with Postman : Update

PUT http://127.0.0.1:8000/api/countries/4

PUT http://127.0.0.1:8000/api/countries/4 Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {"name": "USA",
2 "capital": "Washington DC"
3 }
```

Body Cookies Headers (9) Test Results 200 OK 27 ms 347 B Sav

Pretty Raw Preview Visualize JSON

```
1 {
2     "id": 4,
3     "name": "USA",
4     "capital": "Washington DC"
5 }
```

```
{
  "name": "USA",
  "capital": "Washington DC"
}
```



Test with Postman : Delete

DELETE http://127.0.0.1:8000/api/countries/4

The screenshot shows the Postman application interface. At the top, there is a toolbar with a 'DELETE' button, a URL input field containing 'http://127.0.0.1:8000/api/countries/4', and a 'Send' button. Below the toolbar, there are tabs for 'Params', 'Authorization', 'Headers (6)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Params' tab is currently selected. Under 'Query Params', there is a table with columns 'KEY', 'VALUE', and 'DESCRIPTION'. One row in the table has 'Key' in the KEY column and 'Value' in the VALUE column. Below this table, there are tabs for 'Body', 'Cookies', 'Headers (9)', and 'Test Results'. The 'Body' tab is selected. In the 'Body' section, there are buttons for 'Pretty', 'Raw', 'Preview', 'Visualize', and 'JSON'. The 'JSON' button is selected, and the response body is displayed in JSON format:

```
1 {  
2     "message": "Countries was deleted successfully"  
3 }
```

At the bottom right of the main area, there are status indicators: a globe icon, '204 No Content', '34 ms', '353 B', and a 'Save' button.



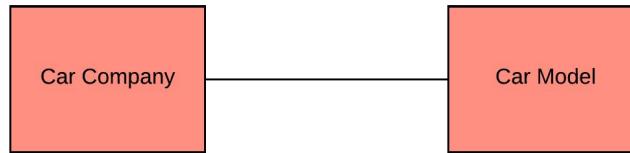
ORM Summary

- Create Project (working under /PythonDjango)
- Create App : `/countries`
- Config Library + Database Connection : `setting.py`
- Create Database (`countriesdb`)
- Create Model file (table metadata) : `countries/models.py`
- Migration model : create table : `python manage.py migrate`
- Serializer: map model <-> json : `countries/serializers.py`
- Create Django admin user : `admin/admin`
- Modify View : handle method `GET, POST, PUT, DELETE`
- Create country router : `countries/urls.py`
- Modify main router : `urls.py`
- Register app : `countries/admin.py`
- Run server : `python manage.py runserver`

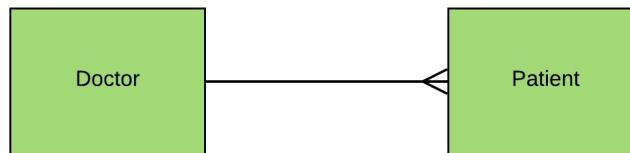


Relationship

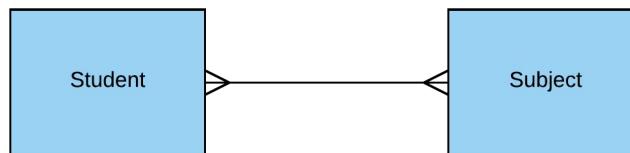
One to One



One to Many



Many to Many



https://docs.djangoproject.com/en/3.1/topics/db/examples/many_to_one/

https://docs.djangoproject.com/en/3.1/topics/db/examples/many_to_many/



Exit virtual environment

```
.. \venv\Scripts\deactivate
```



Resources

Resources

Git : <https://github.com/tanlull/python-django-api>

```
python -m venv venv  
.\venv\Scripts\activate
```

```
pip install flask  
pip install mysql-connector-python  
pip install django  
pip install djangorestframework  
pip install django-cors-headers  
  
# https://www.lfd.uci.edu/~gohlke/pythonlibs/#mysqlclient  
pip install mysqlclient-1.4.6-cp38-cp38-win32.whl
```

```
django-admin startproject PythonDjango  
cd PythonDjango  
python manage.py startapp countries  
python manage.py migrate  
python manage.py runserver
```

