

# Python API

— Tanya Sattaya-aphitan, Ph.D. —

Oct 6-7, 2020 @การยางแห่งประเทศไทย



# Agenda

- Day 1

- Overview & Installation
- Basic Python
- Call API :
  - BOT: Exchange Rate
  - AI : AI 4 Thai

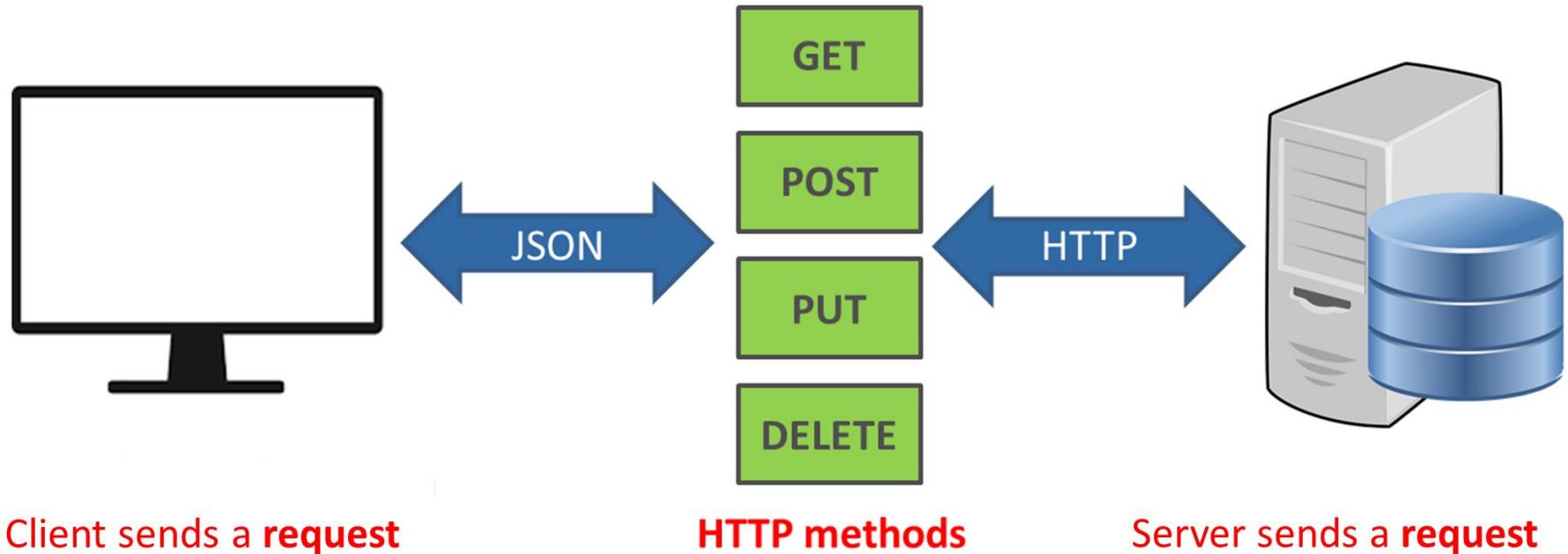
- Day 2

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



# API Overview

# API = Application Program Interface



# API vs Web Service

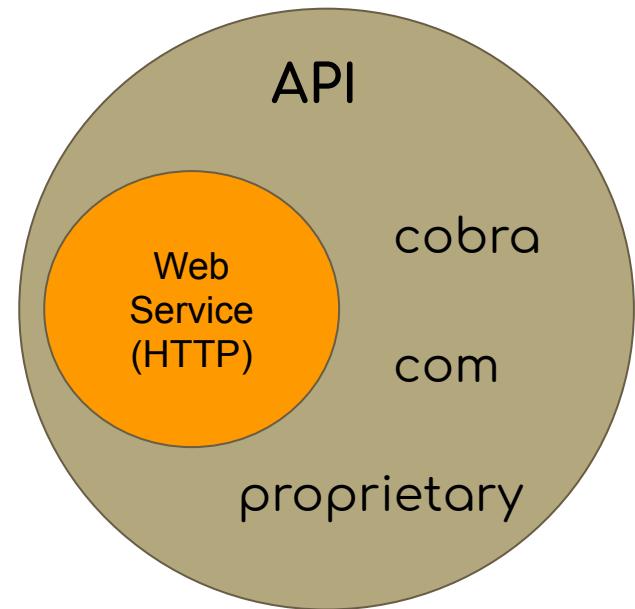
API = Application Program Interface

Web Service = API (operate over HTTP)

All Web Services = API

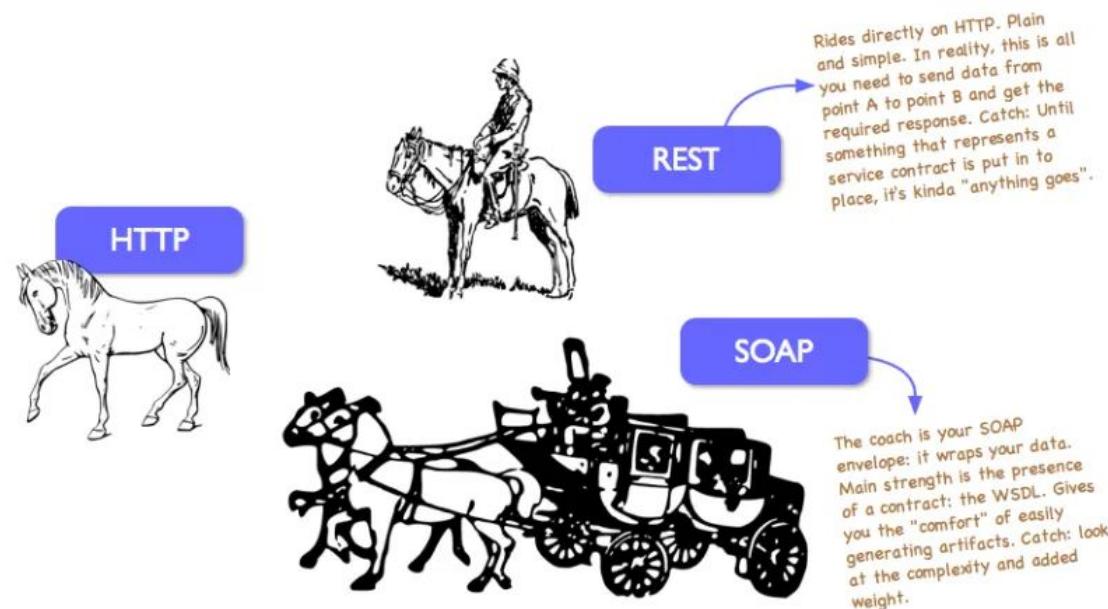
In this course !

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,
      }
    ]
  }
}
```



# Installation

# IDE: Visual Studio Code

https://code.visualstudio.com

Visual Studio Code

Docs

Updates

Blog

API

Extensions

FAQ

Search Docs

Download

Version 1.49 is now available! Read about the new features and fixes from August.

## Code editing. Redefined.

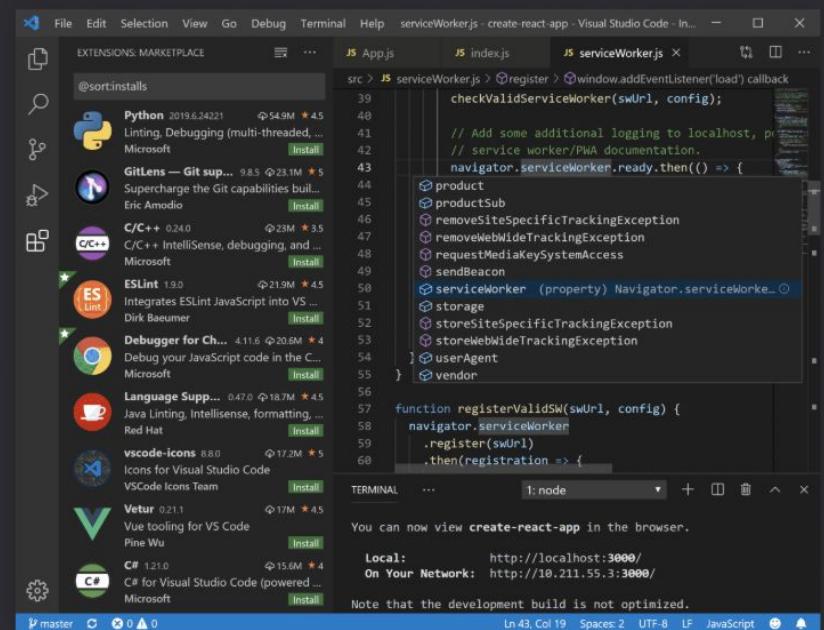
Free. Built on open source. Runs everywhere.

Download for Windows

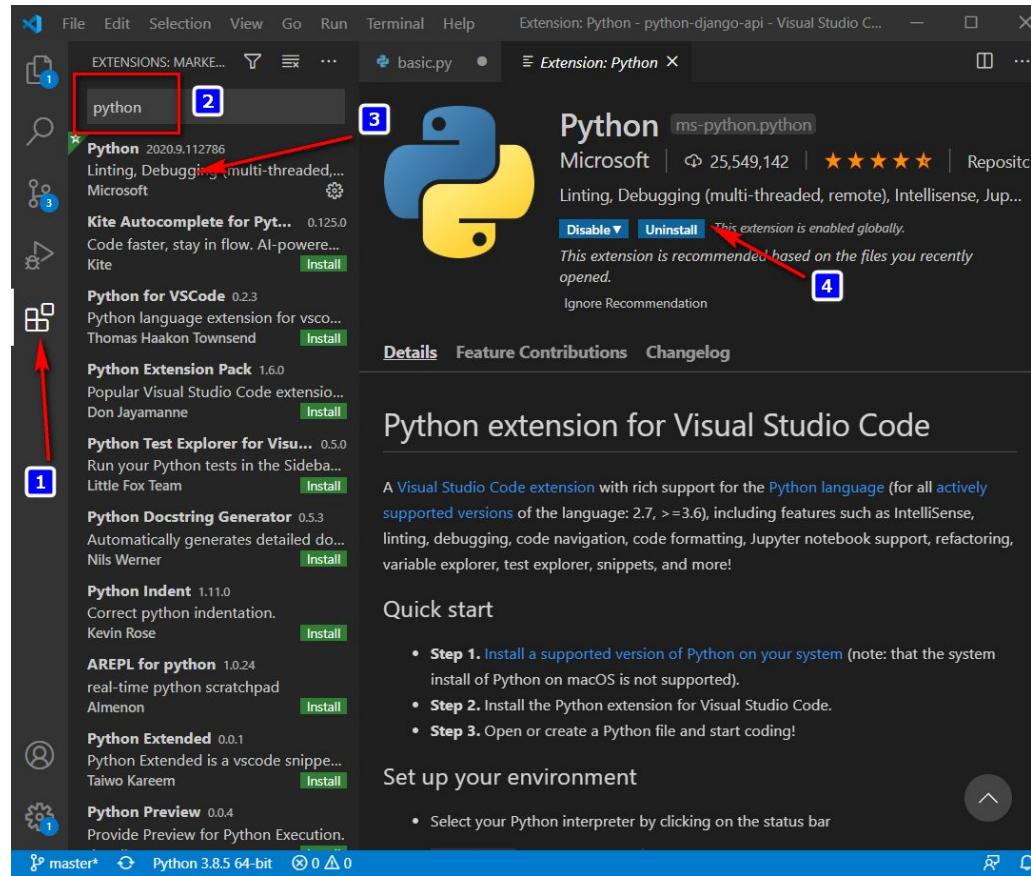
Stable Build

Other platforms and Insiders Edition

By using VS Code, you agree to its  
[license](#) and [privacy statement](#).



# 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 and a search bar with a 'GO' button. A prominent yellow button labeled 'Download Python 3.8.6' is highlighted with a red arrow. To the right of the button is a graphic of two parachutes carrying boxes. Below the button, text links to other OS versions and prereleases. A section titled 'Active Python Releases' lists the following table:

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
```



# Python Library

- flask
- django
- Djangorestframework
- django-cors-headers

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

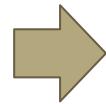
Check Version

```
python -m django --version  
3.1.2  
  
python -m flask --version  
Python 3.8.5  
Flask 1.1.2  
  
Werkzeug 1.0.1
```

# 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 over the "Download" button for the Windows MSI installer.



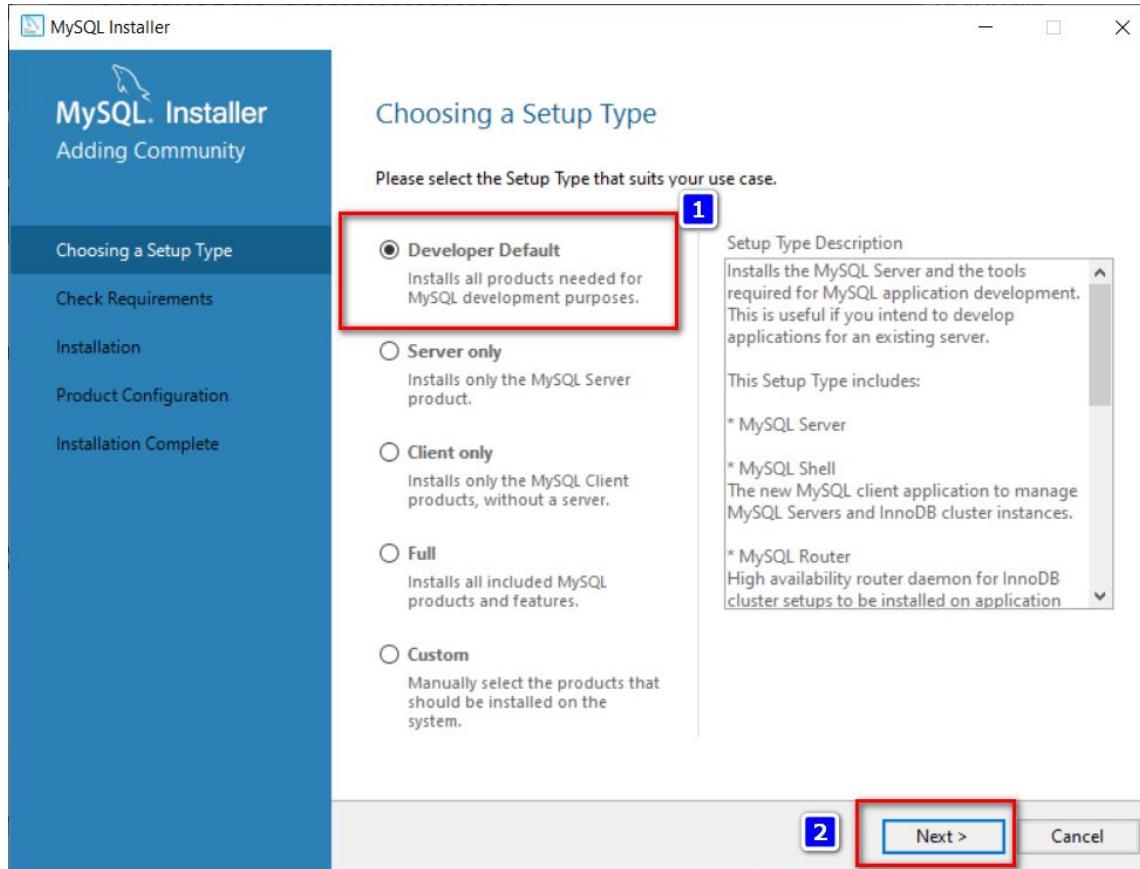
The screenshot shows the MySQL Community Downloads page for "MySQL Installer 8.0.21". The "General Availability (GA) Releases" tab is selected. The page lists two download options for "Windows (x86, 32-bit), MSI Installer":

File Type	Version	Size	Download Link	Signature
(mysql-installer-web-community-8.0.21.0.msi)	8.0.21	24.5M	<a href="#">Download</a>	<a href="#">Signature</a>
(mysql-installer-community-8.0.21.0.msi)	8.0.21	427.6M	<a href="#">Download</a>	<a href="#">Signature</a>

A blue box with the number "2" is placed over the "Download" button for the first file. A red box highlights the "Signature" links for both files. A callout 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

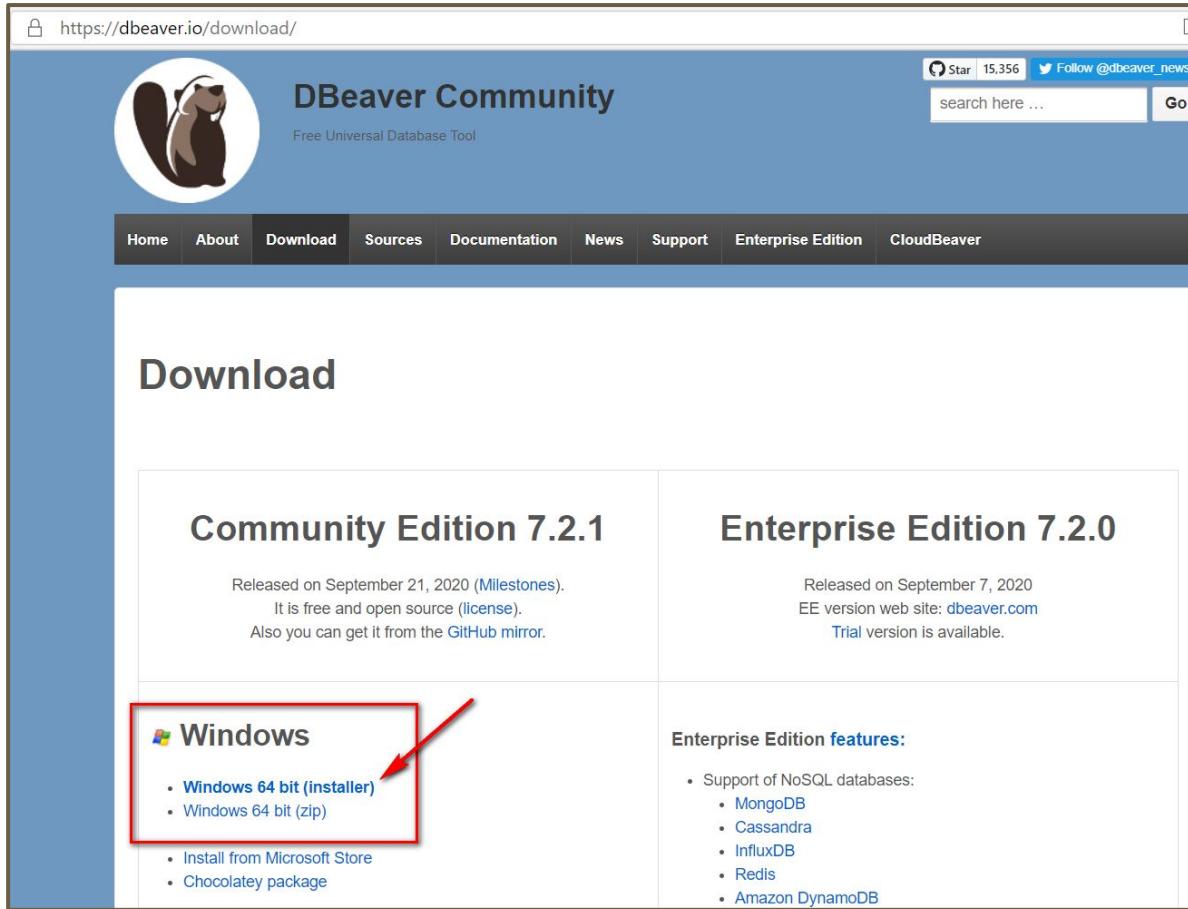


# DB GUI : DBeaver Community

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" underneath it. A navigation bar below the title includes links for Home, About, Download, Sources, Documentation, News, Support, Enterprise Edition, and CloudBeaver. On the right side, there's a "Star 15,356" button, a "Follow @dbeaver\_news" link, a search bar with placeholder text "search here ...", and a "Go" button. Below the navigation bar, a large section titled "Universal Database Tool" features a paragraph about the tool's capabilities and a list of supported databases. To the right of this text are two screenshots of the DBeaver interface: one showing the Database Manager with a list of tables and columns, and another showing a detailed Entity-Relationship (ER) diagram with various tables like Customer, Order, and OrderDetail, and their relationships.



# 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 and a navigation bar with 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 includes release details, a "Windows" subsection with download links for 64-bit installer and zip files, and a "Chocolatey package" link. A red box highlights the Windows section, and a red arrow points to the "Windows 64 bit (installer)" link. The "Enterprise Edition" section includes release details, a "Trial version is available" link, and a list of supported NoSQL databases.

## 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:**

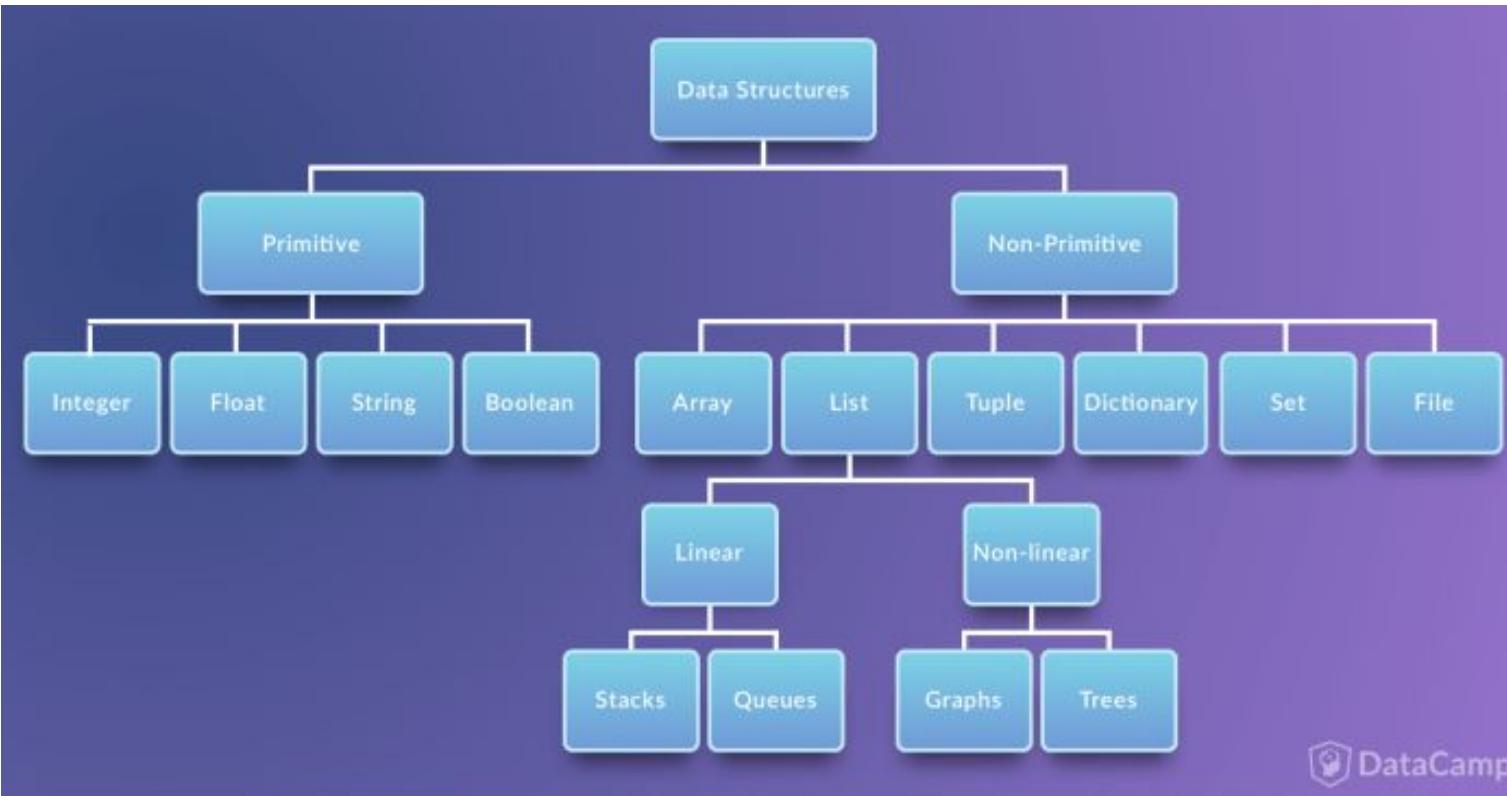
- Support of NoSQL databases:
  - MongoDB
  - Cassandra
  - InfluxDB
  - Redis
  - Amazon DynamoDB



# Basic Python

# Data Types

# Python Variables

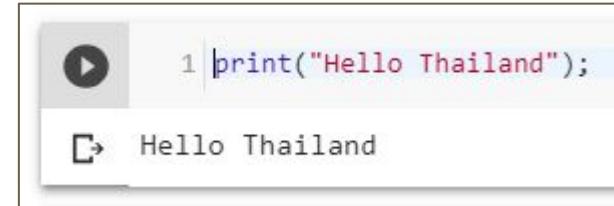


# Print

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

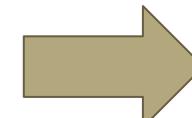
```
data = {  
    'day':['1/1/2018','2/1/2018','3/1/2018'],  
    'temp':[20,10,15],  
    'event':['cold','cold','cold']  
} #สร้างข้อมูล  
  
print(data['day'])  
  
#เพิ่มข้อมูล  
data['x']=[1,2,3]  
  
print(data)
```



# 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))
```

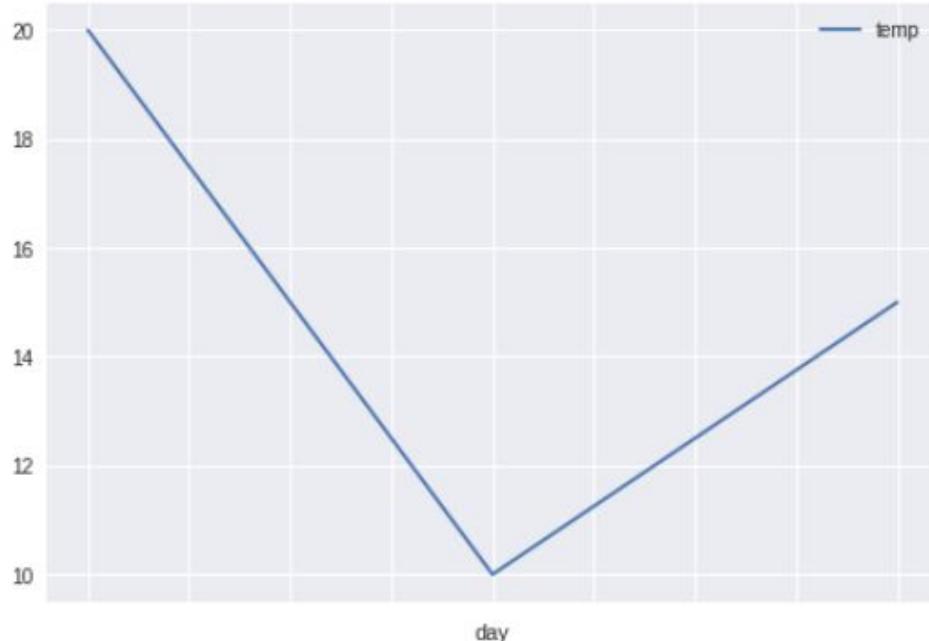
```
<class 'dict'>
<class 'list'>
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.series.Series'>
```



# 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

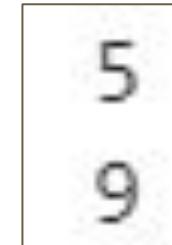
```
# ก้าเซ็นเดิม แต่เก็บค่าในรูปแบบ Int  
age={'Dad':42,'Mom':40}  
print(age['Dad'])  
print(age['Dad']+ age['Mom']) # คำนวณค่าที่ได้จากการเรียกข้อมูลโดยใช้ Key  
print(age)
```

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 <sup>2</sup>	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](http://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

Machine learning

NumPy

Scientific computing

matplotlib

2D plotting

#### Install Python



**ANACONDA**  
Leading open data science platform  
powered by Python



**spyder**  
Free IDE that is included  
with Anaconda



**jupyter**  
Create and share  
documents with live code,  
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
```

DataCamp

Learn Python for Data Science interactively



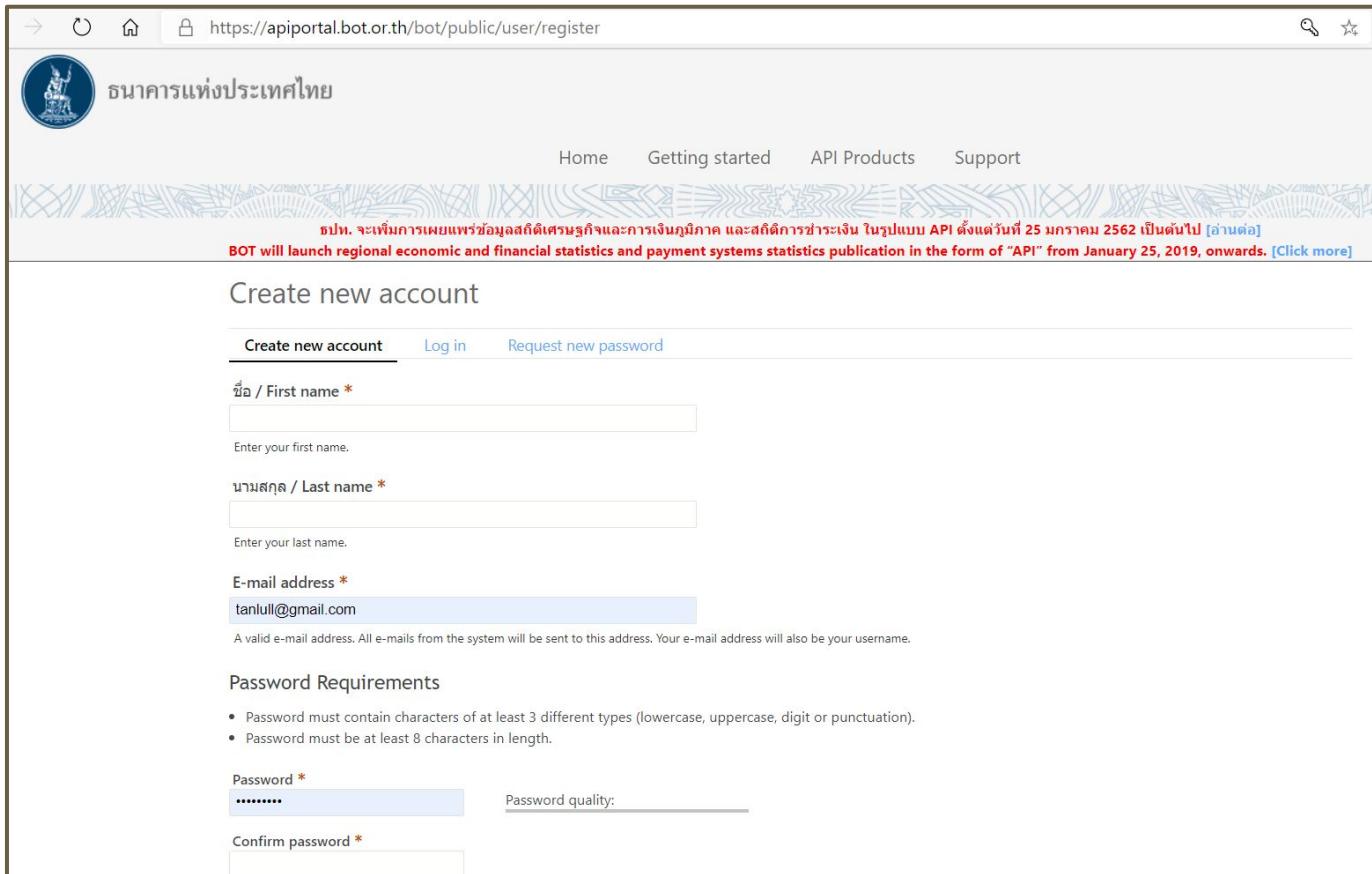
---

---

# 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 และผลติดการชำระเงิน 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:

- Create new account** (selected tab)
- Log in**
- Request new password**
- ชื่อ / First name \***: An input field with placeholder "Enter your first name."
- นามสกุล / Last name \***: An input field with placeholder "Enter your last name."
- E-mail address \***: An input field containing "tanlull@gmail.com". Below it, a note says: "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 \***: An input field showing masked text "\*\*\*\*\*".
- Password quality:** A progress bar indicating the strength of the password.
- Confirm password \***: An input field for password confirmation.



# 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 features a banner message in both Thai and English about the launch of regional economic and financial statistics and payment systems statistics publication via API. Below the banner, there is a card for an application named "Python Programming" with a status of "testing". At the bottom right of the card 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 Copyright information, along with a QR code.

ธนาคารแห่งประเทศไทย

tanlull@gmail.com Tanya Sattaya-aphitan

Home Getting started API Products Apps Support

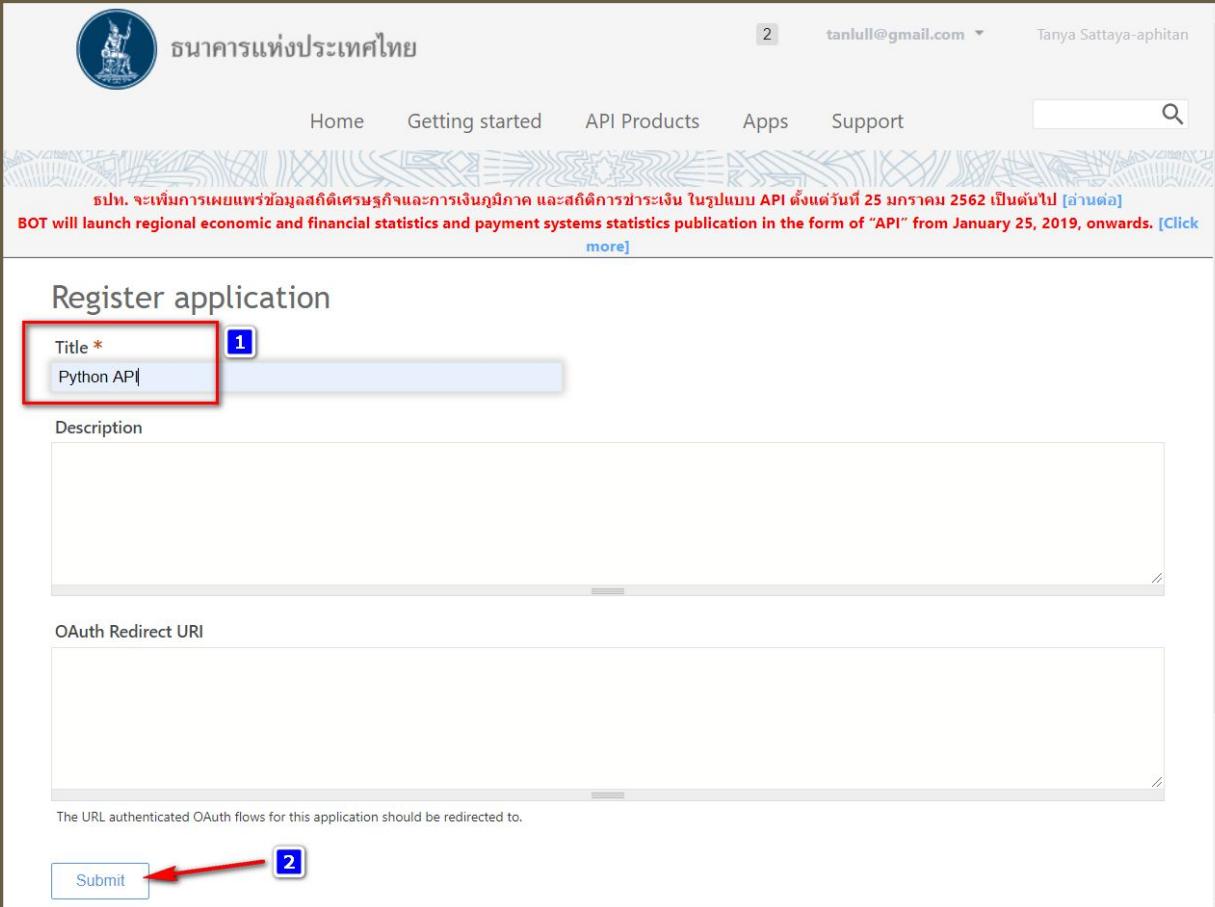
ธนาคารแห่งประเทศไทย จะเพิ่มการเผยแพร่ข้อมูลสถิติเศรษฐกิจและการเงินภูมิภาค และสถิติการชำระเงิน ในรูปแบบ API ตั้งแต่วันที่ 25 มกราคม 2562 เป็นต้นไป [อ่านต่อ]  
BOT will launch regional economic and financial statistics and payment systems statistics publication in the form of "API" from January 25, 2019, 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" and "Python Programming". 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 app details for "Python Programming" are shown, including its icon, name, update link, description ("testing"), credentials section (with a "Default" entry and a redacted "Client ID" field), and a "Reset" button.

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 items:

Plans	Default Plan
Weighted-average Interbank Exchange Rate - THB / USD	200 per hour
Average Exchange Rate - THB / Foreign Currency	200 per hour

Below the plans, there is a 'Free' row and a 'Subscribe' button. A note at the bottom states: '\* = Mouseover for more information'.

On the left sidebar, there is a menu with 'Exchange Rates 2.0.1' selected, and a list of other API products: APIs, Weighted-average Interbank Exchange Rate - THB / USD, and 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, and a placeholder for the client ID.
- Headers:** X-IBM-Client-Id: "REPLACE\_THIS\_KEY" (4)
- Example Response:** JSON response structure for the API call.

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"  
        }  
      ]  
    }  
  }  
}
```

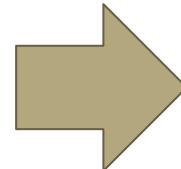
```
{  
  "period":"2018-10-29",  
  "rate":"33.1360000"  
},  
{  
  "period":"2018-10-26",  
  "rate":"33.0290000"  
},  
{  
  "period":"2018-10-25",  
  "rate":"32.9230000"  
},  
{  
  "period":"2018-10-24",  
  "rate":"32.8110000"  
},  
{  
  "period":"2018-10-03",  
  "rate":"32.3380000"  
},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



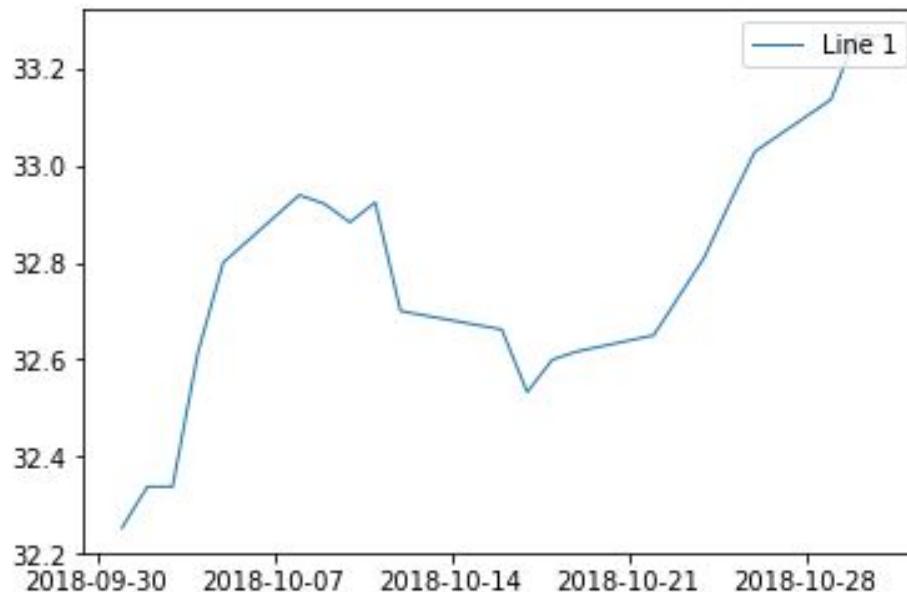
	period	rate
0	2018-10-31	33.266
1	2018-10-30	33.270
2	2018-10-29	33.136
3	2018-10-26	33.029
4	2018-10-25	32.923
5	2018-10-24	32.811
6	2018-10-22	32.650
7	2018-10-19	32.617
8	2018-10-18	32.600
9	2018-10-17	32.533
10	2018-10-16	32.662
11	2018-10-12	32.700
12	2018-10-11	32.924
13	2018-10-10	32.883
14	2018-10-09	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

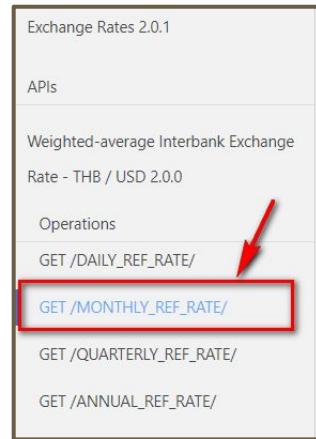
```
import matplotlib.pyplot as plt #plot graph เส้น และกำหนดขนาดของเส้น  
plt.plot(df.period,df.rate, linewidth=1) #กำหนด Description แต่ละเส้นพร้อมบอกตากำหนดที่แสดง  
plt.legend(["Line 1"], loc="upper right")  
plt.show()
```



# 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. A red arrow points from this box towards the right side of the screen. On the right, the Postman desktop application is shown running. The app window has several orange arrows pointing to its interface: one to the "Import" button in the top bar, one to the "Runner" tab, one to the "My Workspace" dropdown, one to the "History" tab, and one to the "Authorization" tab in the request configuration pane. The request configuration pane shows a GET request to "postman-echo.com/get". The bottom right corner of the screenshot features a QR code.

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

New Import Runner My Workspace ...

GET postman-echo.com/get

Params Authorization Headers (8) Body Pre-request

Type Inherit auth from parent

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 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/](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 includes information about frequency (Daily), lag time (---), and release schedule (Every business day at 6.00 p.m. (BKK)).
- 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 response details.

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](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. The main area is titled "My Workspace" with an "Invite" button. 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. Another red arrow points from the "Collection" section in the dialog to a blue box labeled "2". The dialog box contains several sections: "BUILDING BLOCKS" with Request (GET), Collection, Environment, API Documentation, Mock Server, Monitor, and API; "ADVANCED" with API Documentation, Mock Server, Monitor, and API; and a note at the bottom: "Not sure where to start? Use a [template](#) to see how Postman can help you in your work." A "Learn more on Postman Docs" link is at the bottom right. The left sidebar lists various collections: Express M, Google Sh, NodeJS wi, AI for Thai, Elastics, Google, LoRa, LPR, and NB-IoT, each with a count of requests.

Postman

File Edit View Help

New Import Runner

My Workspace Invite

Upgrade

Filter History New Collection

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

Create New Templates API Network

**BUILDING BLOCKS**

**Request** GET Create a basic request

**Collection** Save your requests in a collection for reuse and sharing

**Environment** Save values you frequently use in an environment

**ADVANCED**

**API Documentation** Create and publish beautiful documentation for your APIs

**Mock Server** Create a mock server for your in-development APIs

**Monitor** Schedule automated tests and check performance of your APIs

**API** Manage all aspects of API design, development, and testing

Not sure where to start? Use a [template](#) to see how Postman can help you in your work.

Learn more on Postman Docs

Find and Replace Console

Bootcamp Build Browse

QR code

# 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" and "Save" buttons, along with "Cookies" and "Code" tabs.



# Add Request

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

- 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 (blue box 1), Create a fork, Add Request (blue box 2, highlighted with a red arrow), Add Folder, Duplicate (Ctrl+D), Export, Monitor Collection, Mock Collection, Publish Docs, Remove from workspace, Delete (Del).
- Bottom Bar:** Find and Replace, Console, Bootcamp, Build, Browse, Help.

A red box labeled "Right Click" highlights the context menu. A red arrow points from the "Edit" option in the menu to the "Add Request" option, which is also highlighted with a blue box labeled "2".



# 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. The top navigation bar includes links for Home, About, Services, Demo, Use cases, Corpus, Developer, News & Events, Contact us, FAQs, and a user account icon labeled 'tanlull'. A red box highlights the user account icon. On the left, a sidebar lists various AI services: Dashboard, How to get started, Basic NLP, Tag suggestion, Machine translation, Sentiment Analysis, SSense (with a red arrow pointing to it), ThaiMoji, Character recognition, Object recognition, Person & activity, Face recognition, Speech to text, Text to speech, and ABDUL chatbot. The main content area displays the user's profile information: Username (tanlull), First name (Tanya), Last name (Sattaya-aphitana), Organization (TOT), Password (Password), Confirm password (Confirm Password), Email (tanlull@gmail.com), and API Key (bu---tUW' VGNK 5jL Frwv Z3). A blue box highlights the 'Save' button at the bottom. Three numbered callouts point to the 'SSense' service in the sidebar (3), the 'API Key' field (2), and the 'Save' button (1).

User Profile

Username : tanlull

First name : Tanya

Last name : Sattaya-aphitana

Organization : TOT

Password : \*รหัสผ่านต้องมีอย่างน้อย 6 ตัวอักษร

Confirm password : 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 ໂດຍວິເຄຣາະໜົກໃນເສັ່ນຈຸດປະສົງ (Intention) ວ່າເປັນປະເກດໃດ ໄດ້ແກ່ ສອບຄາມ ຮ້ອງຂອງປະກາດ ໄນບານ ແລະ/ເຊື້ອແສດງຄວາມຄິດເຫັນ ວິເຄຣາະໜົກວາມຄິດເຫັນ (Sentiment) ວ່າ ເປັນເສັ່ນບຸກທີ່ຮ້ອລບ ຮັບກັ້ງວິເຄຣາະໜົກນຳບໍ່ບອກຄຸນລັກບໍ່ນະຂອງສັນຄ້າເຊື້ອ ບໍລິການ (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

ພລັກພົກທີ່ໄດ້ຈາກ ຮະບບ SSense ຈະອຍໃນຮູບແບບຂອງ JSON Arrays ທີ່ສິ່ງແຕ່ລະອາເຮຍຈະນຶ່ງກົດປະກອບດັ່ງນີ້

- sentiment : ພລັກວິເຄຣາະໜົກວາມຄິດເຫັນວ່າເປັນເສັ່ນບຸກທີ່ຮ້ອລບ
  - polarity : ຫຼັງອານນີ້ນີ້ຄວາມຄິດເຫັນ (positive: ເສັ່ນບຸກ, negative: ເສັ່ນລົບ, ອ່າວ່າງ: ໃນເປັນກັ້ງບຸກແລະລົບ)
  - polarity-pos : ນີ້ຂ້ອງຄວາມເສັ່ນບຸກໃຫ້ເຊື້ອໄນ້ (true=ໃຫ້, false =ໄປໃຫ້)
  - polarity-neg : ນີ້ຂ້ອງຄວາມເສັ່ນບຸກໃຫ້ເຊື້ອໄນ້ (true=ໃຫ້, false =ໄປໃຫ້)



# 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

# Database (MySQL)

# Install Mysql Python library

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



# Prepare Database (Postgres)

```
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,'សមសុន',24,'female');  
commit;  
select * from source
```



# Result

The screenshot shows the DBeaver 7.2.0 interface with the following components and actions:

- 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.
- Script Editor:** A code editor window titled "\*<localhost> Script" (marked with a blue box 1) 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;
```
- Table View:** A grid view titled "source" (marked with a blue box 2) showing the data inserted into the "source" table:

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



# 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()
```



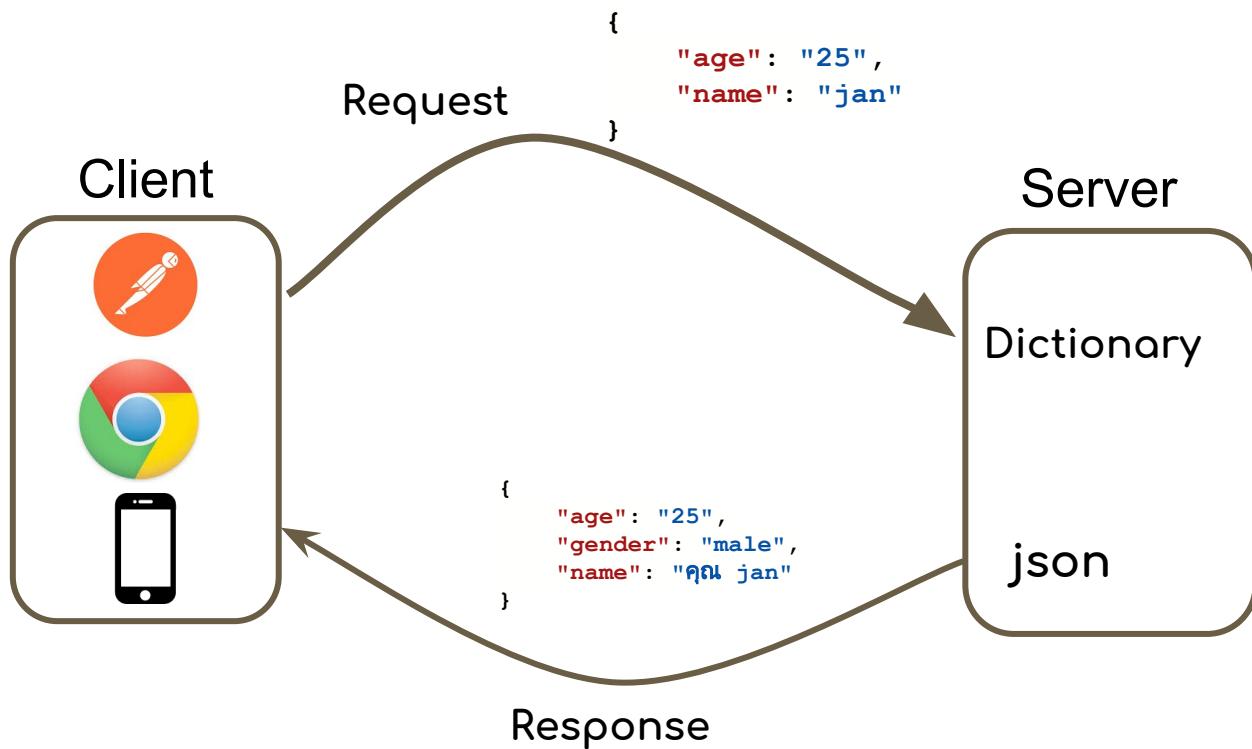
# Flask

# Prepare Environment

```
pip install flask
```



# 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

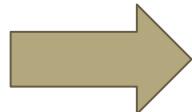
```
# dictionary
data = {'age':'25', 'name':'ເຈນ'}
print(type(data))

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

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

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

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



```
<class 'dict'>
ເຈນ
ນຸ້ມ
{'age': '25', 'name': 'ນຸ້ມ', 'friend': 'ໄລ'}
{'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 contains the value 'คุณสมปอง'.



# 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 areas:

- 1**: The method dropdown set to **GET**.
- 2**: The URL input field containing **http://localhost/test?name=jan&age=25**, which is also highlighted by a red box.
- 3**: The **Send** button, which is also highlighted by a red box.
- 4**: The JSON response body, which is also highlighted by a red box. The response content is:

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

The interface includes tabs for **Params**, **Auth**, **Headers (6)**, **Body**, **Pre-req.**, **Tests**, and **Settings**. Below the URL input, there is a **Query Params** table:

KEY	VALUE	DESCRIPTION	...	Bulk
<input checked="" type="checkbox"/> name	jan			
<input checked="" type="checkbox"/> age	25			
Key	Value	Description		

At the bottom, the response details are shown: **200 OK**, **552 ms**, **181 B**, and a **Save Response** button. The response body is displayed in **Pretty** format.



# 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 structure:

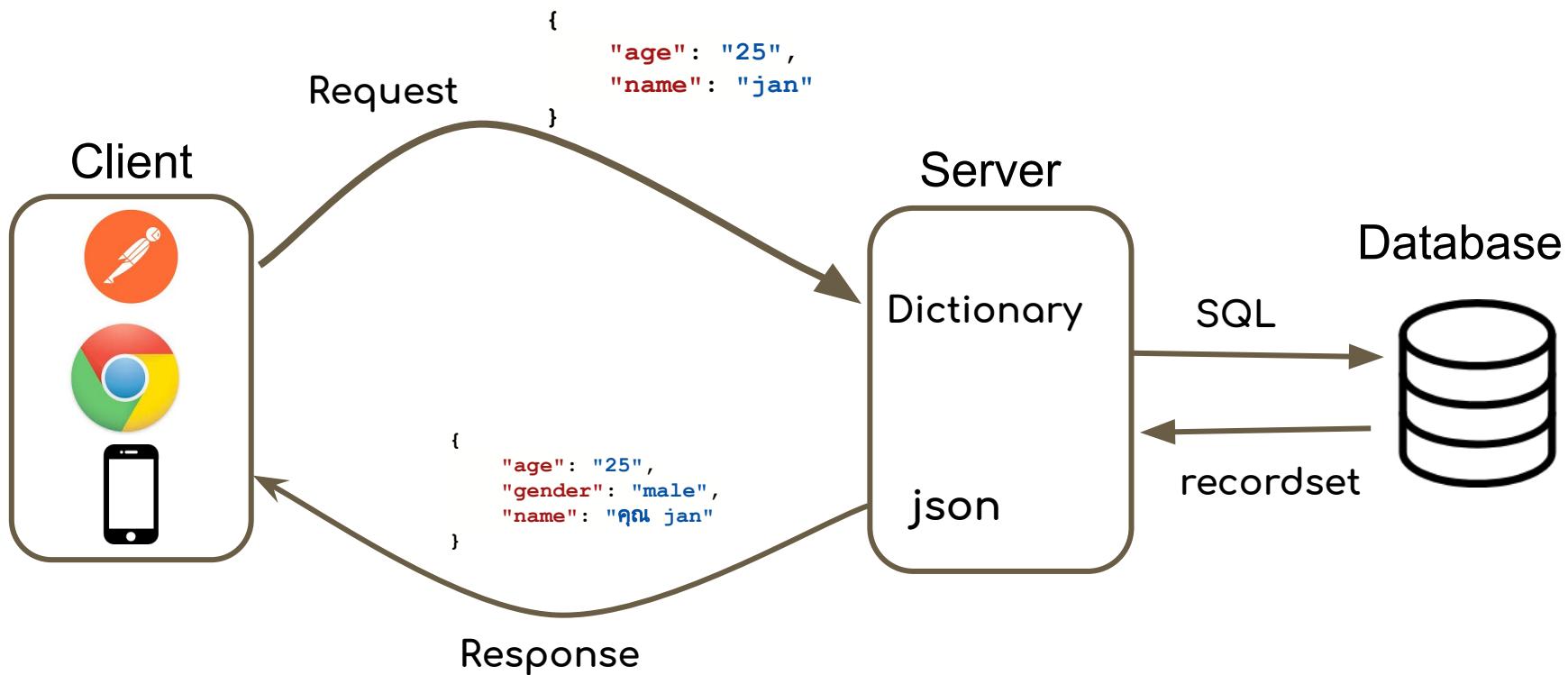
```
1 {  
2   "age": "25",  
3   "gender": "male",  
4   "name": "jan"  
5 }
```

The interface includes tabs for Params, Auth, Headers, Body, Pre-req., Tests, and Settings. The Body tab is currently 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



# 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 : flask\_api\_mysql.py

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

@app.route('/all', methods=['POST', '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 : flask\_api\_mysql.py

```
@app.route('/by_id', methods=['POST', '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 : Append to

## flask\_api\_mysql.py

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

def insert():
    data = request.args
    result =
    db_insert(data["id"], data["name"], data["age"], da-
ta["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

Success

Flask Insert Examples 0 BUILD

GET http://localhost/insert?id=10&name=ໂນຣ&age=22& Send

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

Query Params

KEY	VALUE	DESCRIPTION	...
<input checked="" type="checkbox"/> id	10		
<input checked="" type="checkbox"/> name	ໂນຣ		
<input checked="" type="checkbox"/> age	22		
<input checked="" type="checkbox"/> gender	female		

Body ▾ 200 OK 1549 ms 186 B Save

Pretty Raw Preview Visualize JSON

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

Fail

GET http://localhost/insert?id=10&name=ໂນຣ&age=22& Send

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

Query Params

KEY	VALUE	DESCRIPTION	...
<input checked="" type="checkbox"/> id	10		
<input checked="" type="checkbox"/> name	ໂນຣ		
<input checked="" type="checkbox"/> age	22		
<input checked="" type="checkbox"/> gender	female		
Key	Value	Description	

Body ▾ 200 OK 573 ms 234 B Save

Pretty Raw Preview Visualize JSON

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



# Update : Append to flask\_api\_mysql.py

```
# Update
@app.route('/update', methods=['POST', 'GET'])
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
```



The screenshot shows the Postman interface with the following details:

- Request Method:** GET
- URL:** http://localhost/update?id=10&name=Bow
- Query Params:**

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> id	10	
<input checked="" type="checkbox"/> name	Bow	
- Response Body (Pretty JSON):**

```
{  
  "count": 7,  
  "status": "Success"  
}
```

# Delete : Append to flask\_api\_mysql.py

```
# Delete
@app.route('/delete', methods=['POST', 'GET'])
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 Update Copy Examples 0 ▾ BUILD

GET http://localhost/delete?id=10 1 Send ▾ 3

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

Query Params

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> id	10	Key Value Description

Body

Pretty Raw Preview Visualize JSON ▾

```
1 [ { "count": 7, "status": "Success" } ]
```

200 OK 556 ms 186 B Save

A screenshot of the Postman API testing tool. The URL in the header is highlighted with a red box and labeled '1'. The 'id' query parameter in the 'Query Params' section is highlighted with a red box and labeled '2'. The 'Send' button is highlighted with a red box and labeled '3'. The response body, which is a JSON object with 'count': 7 and 'status': 'Success', is highlighted with a red box at the bottom.



---

---

# 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": "\u0915\u093f\u0922"}, {"age": 22, "gender": "male", "id": 2, "name": "\u0915\u093f\u0924"}, {"age": 23, "gender": "female", "id": 3, "name": "\u0915\u093f\u0921"}, {"age": 24, "gender": "female", "id": 4, "name": "\u0915\u093f\u0925"}, {"age": 26, "gender": "male", "id": 6, "name": "\u0915\u093f\u092e"}]
----- 1 -----
{"age": 22, "gender": "male", "id": 2, "name": "\u0915\u093f\u0924"}
----- 2 -----
{"age": 23, "gender": "female", "id": 3, "name": "\u0915\u093f\u0921"}
23
\u0915\u093f\u0921
----- Loop -----
{"age": 20, "gender": "male", "id": 1, "name": "\u0915\u093f\u0922"}, {"age": 22, "gender": "male", "id": 2, "name": "\u0915\u093f\u0924"}, {"age": 23, "gender": "female", "id": 3, "name": "\u0915\u093f\u0921"}, {"age": 24, "gender": "female", "id": 4, "name": "\u0915\u093f\u0925"}, {"age": 26, "gender": "male", "id": 6, "name": "\u0915\u093f\u092e"}]
-----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 taht API



# API ORM with Django

