Python API

Tanya Sattaya-aphitan, Ph.D.

Oct 6-7, 2020 @การยางแห่งประเทศไทย

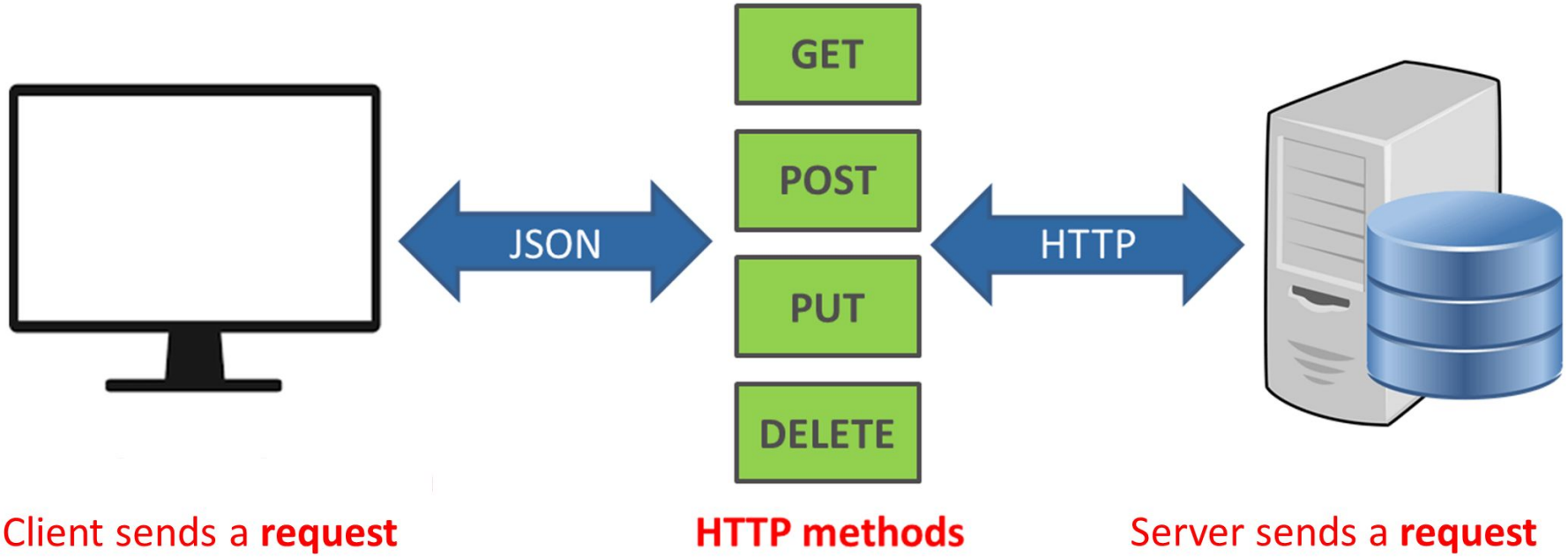
My blog: **http://drtan.cc** e-mail: **Tanyas@tot.co.th** Line id : **tanlull**

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



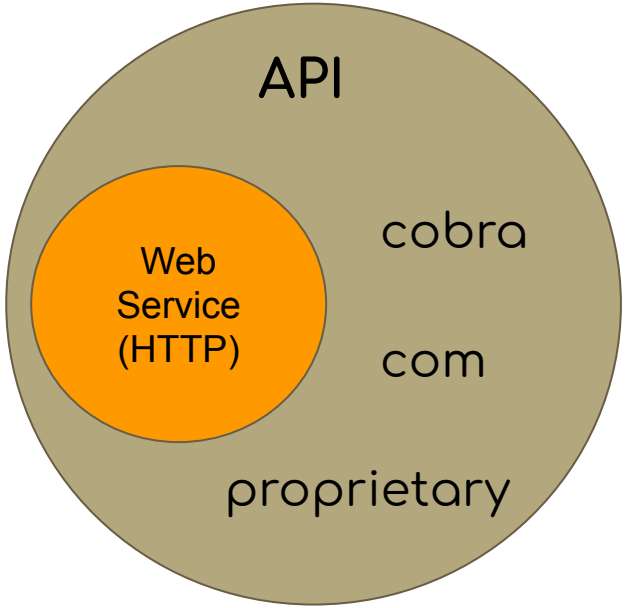
Img src: https://www.dailytech.in.th/api-%E0%B8%84%E0%B8%B7%E0%B8%AD%E0%B8%AD%E0%B8%B0%E0%B9%84%E0%B8%A3

API vs Web Service

API = Application Program Interface

Web Service = API (operate over HTTP)

All Web Services = API

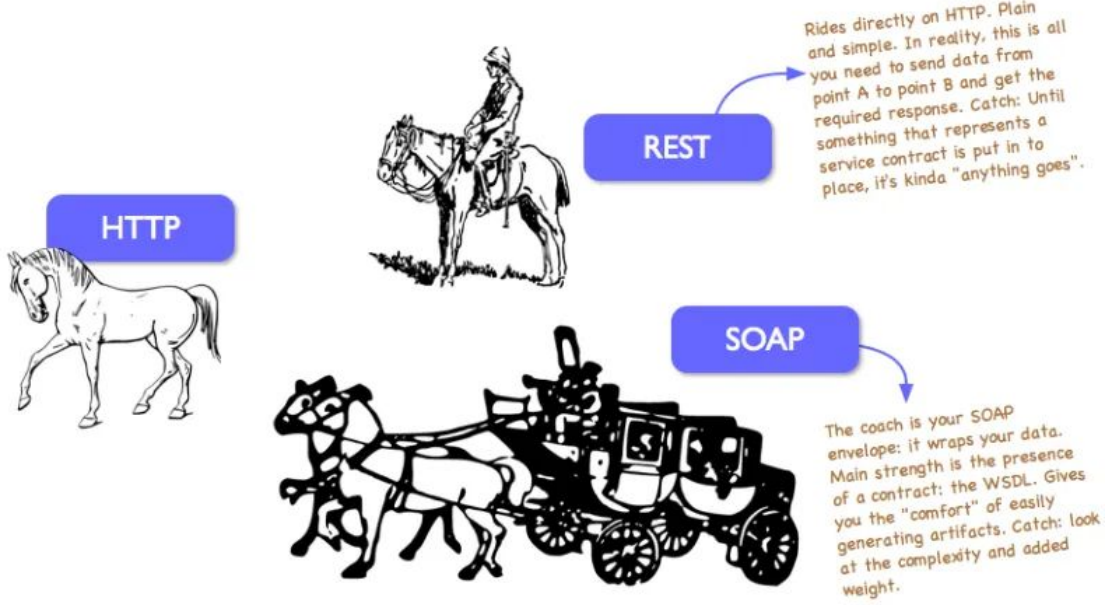


In this course !

API = Web Service

SOAP & REST

- SOAP (Simple Object Access Protocol) - Standard
- 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>
```

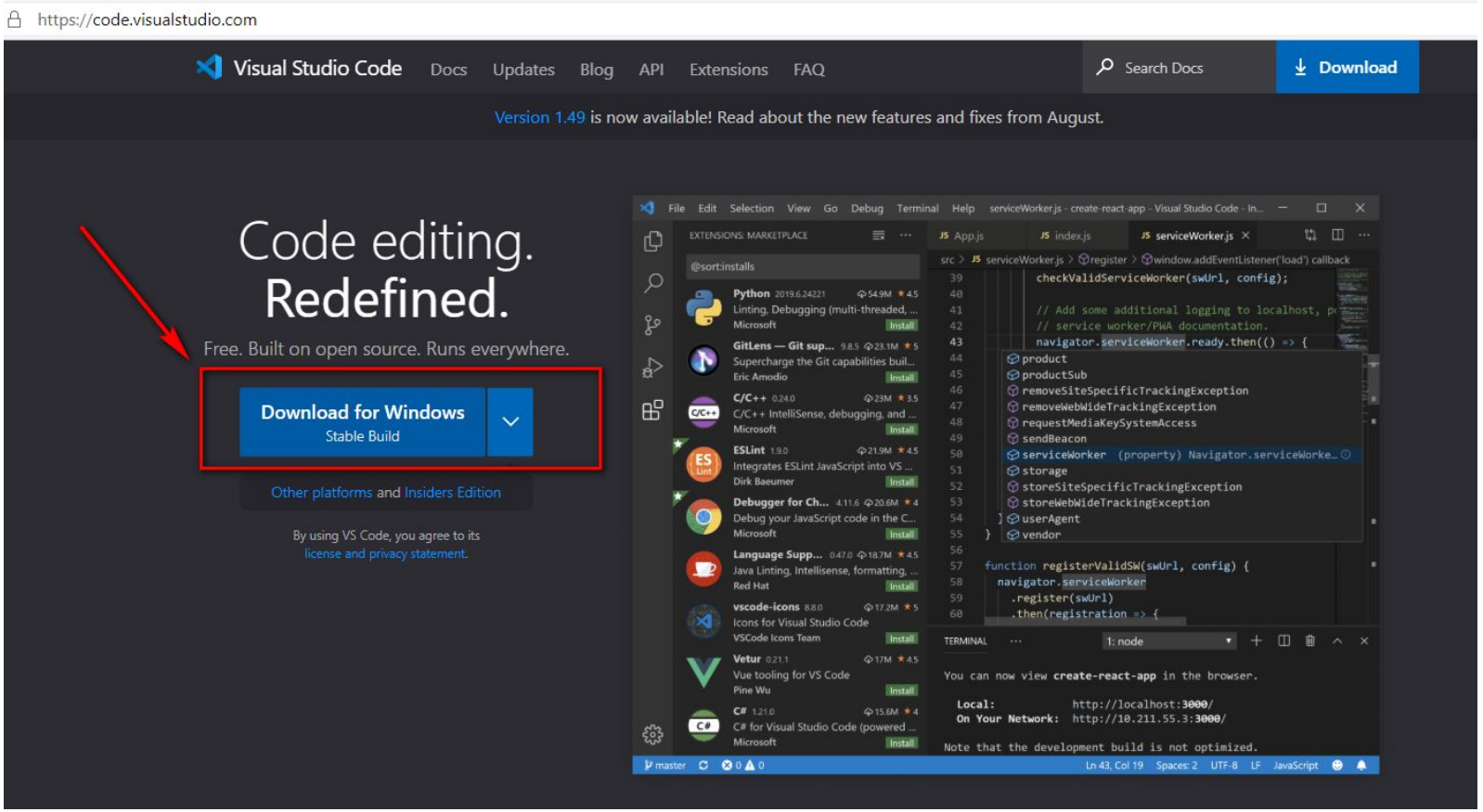
REST

JSON

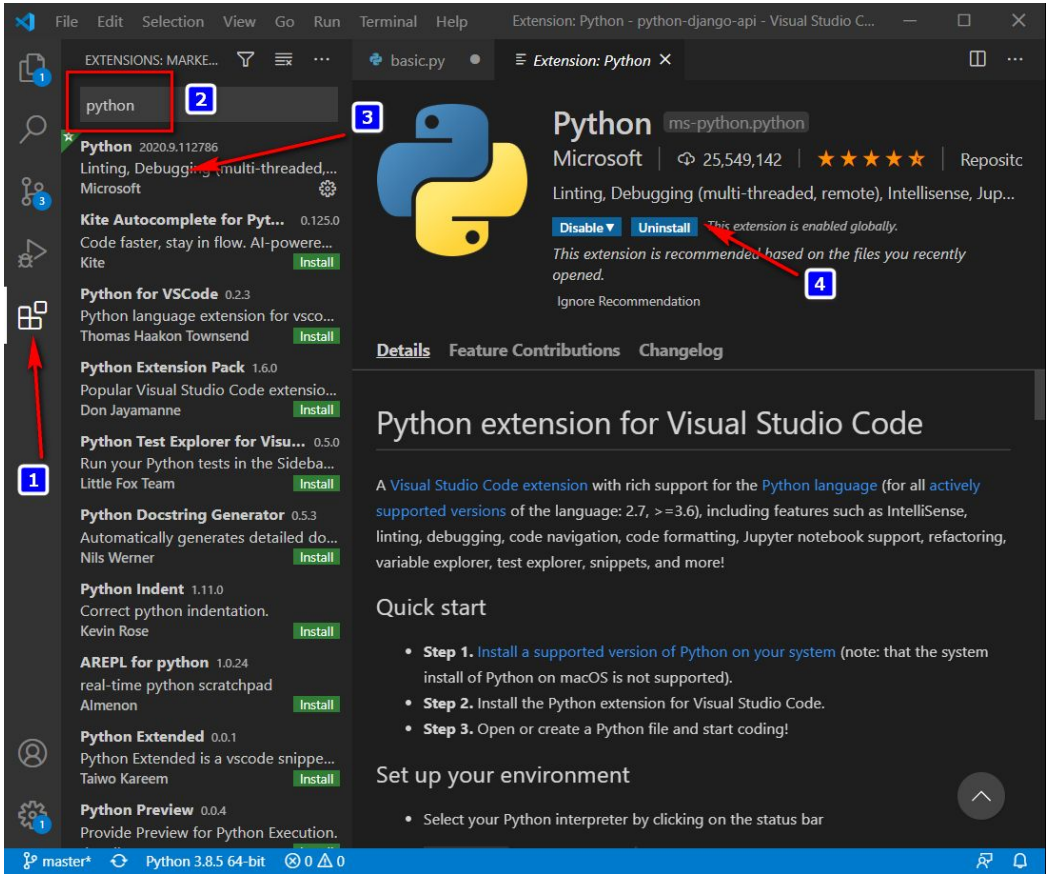
```
{  "empinfo" :
      {
            "employees" :  [
            {
                  "name" : "James Kirk",
                  "age" : 40,
            },
            {
                  "name" : "Jean-Luc Picard",
                  "age" : 45,
            },
            {
                  "name" :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

Installation

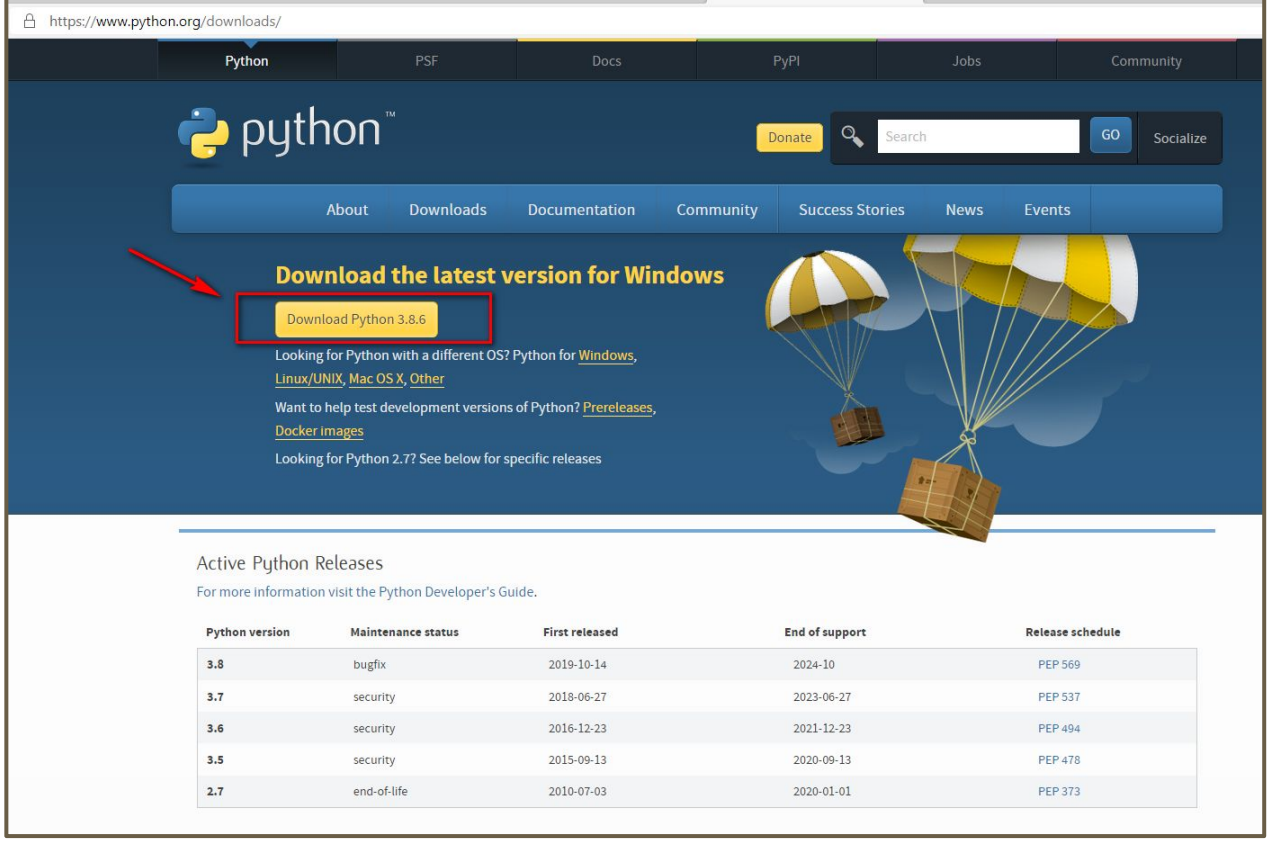
IDE: Visual Studio Code



IDE : Plugin python



Python




```
python --version
  Python 3.8.6
```

Python Library

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

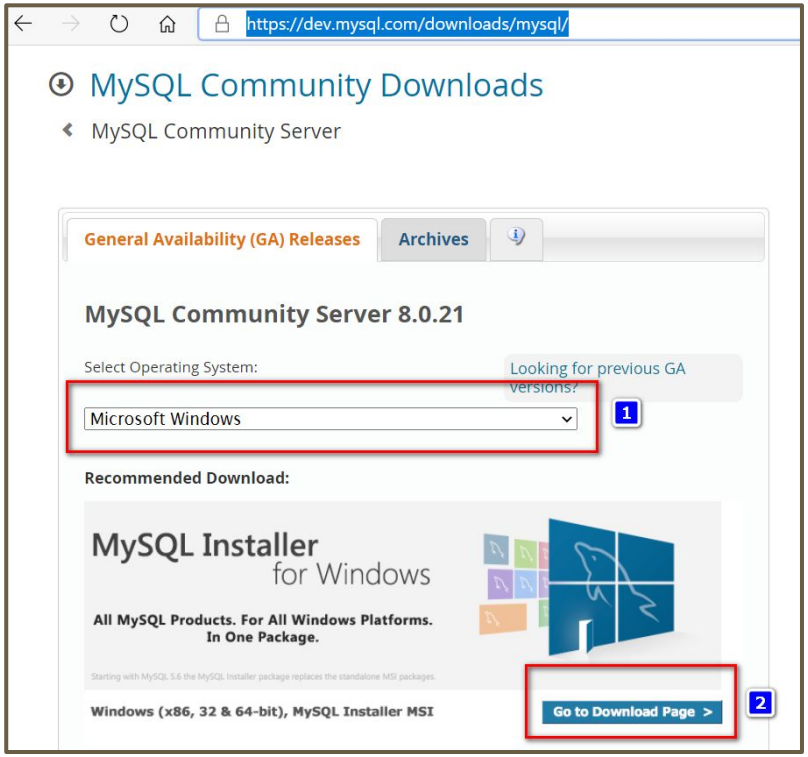
```
pip install flask
pip install django
pip install djangorestframework
pip install django-cors-headers
```

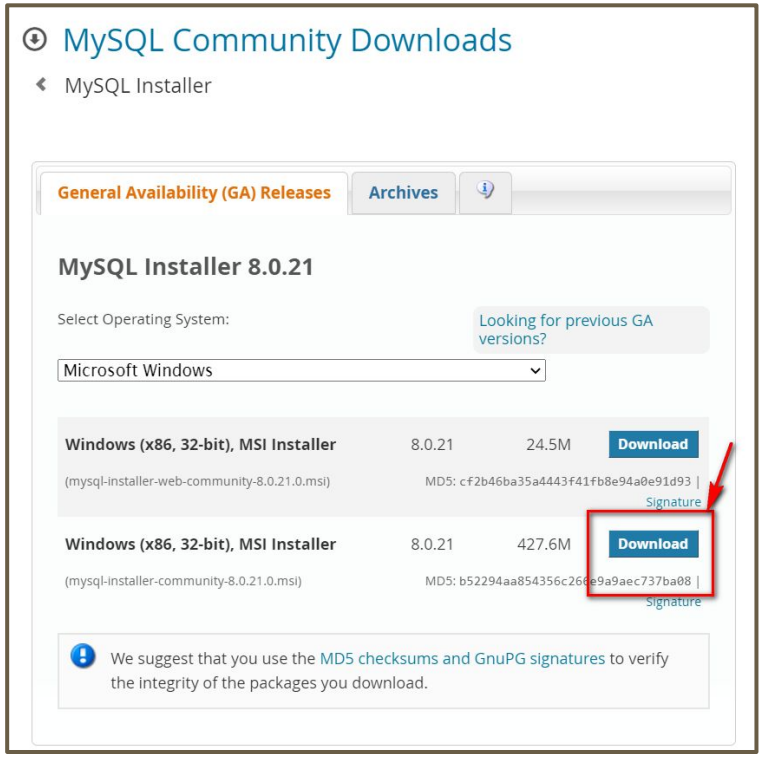
Check Version

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

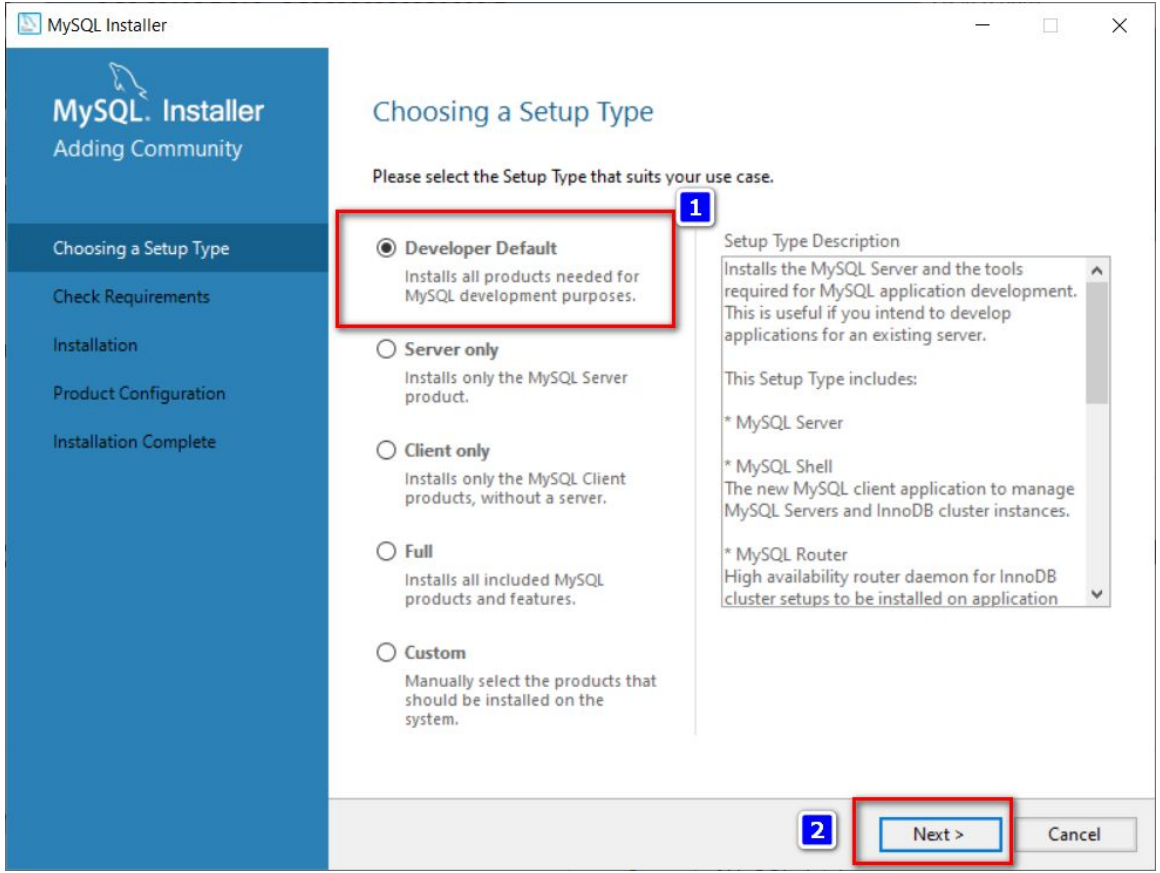
Database : MySQL Community Server

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

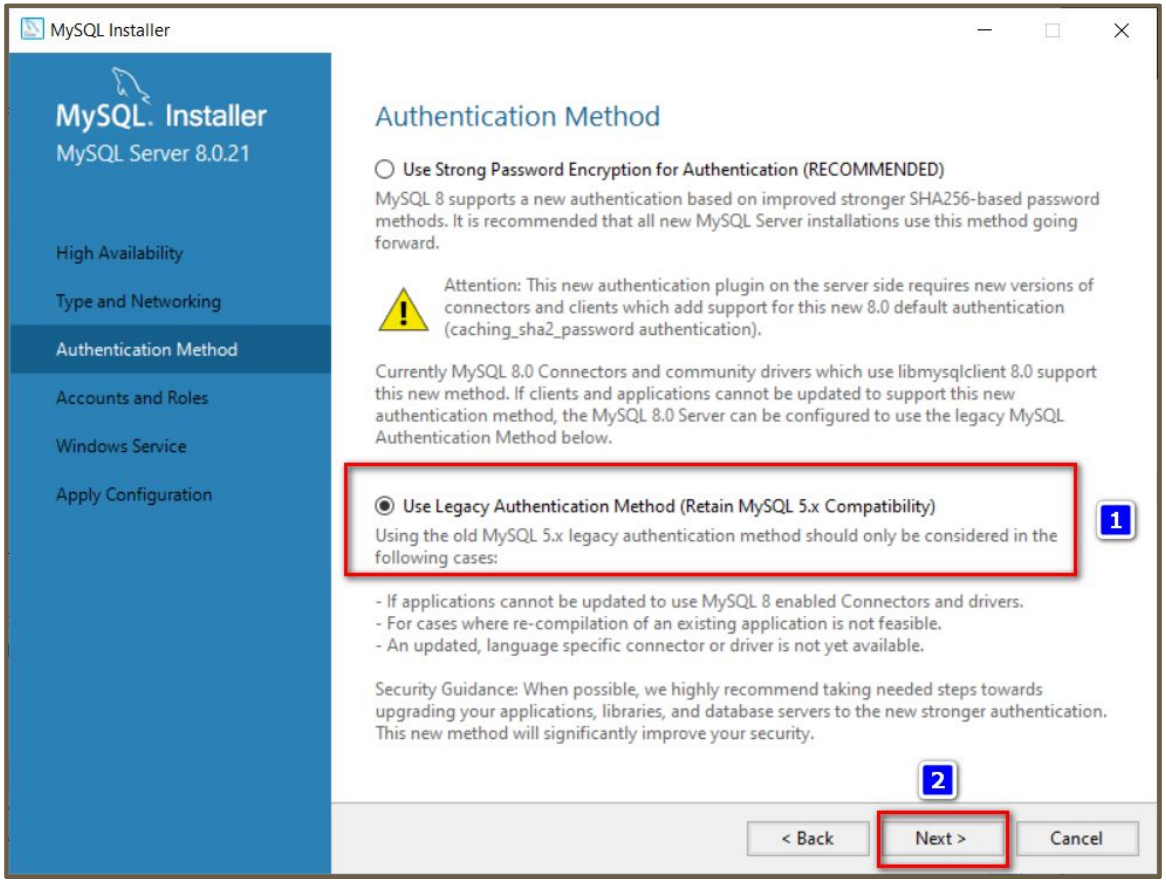




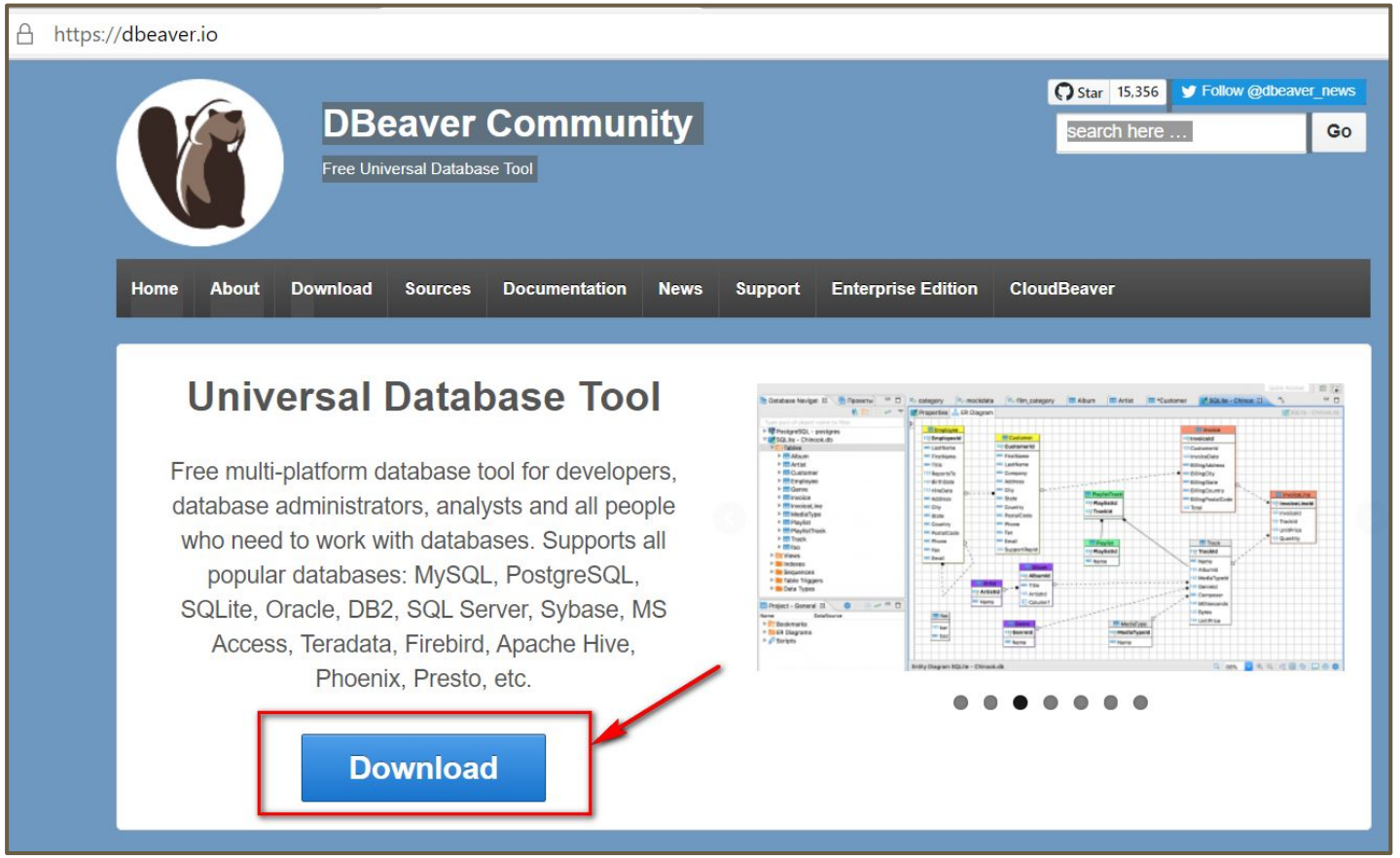
Install : developer -> next -> next



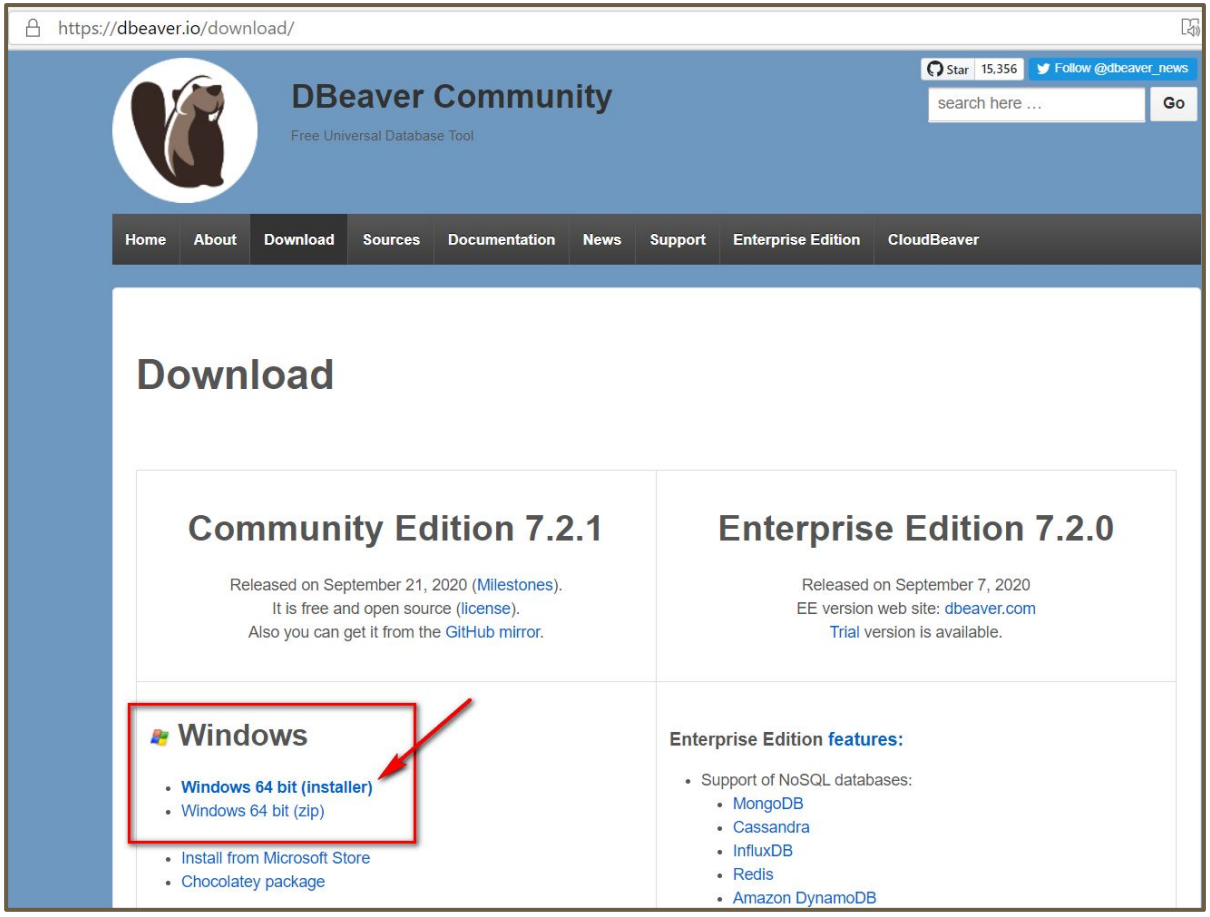
Use Legacy


MySQL Installer
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)
Using the old MySQL 5.x legacy authentication method should only be considered in the following cases:
1
- 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



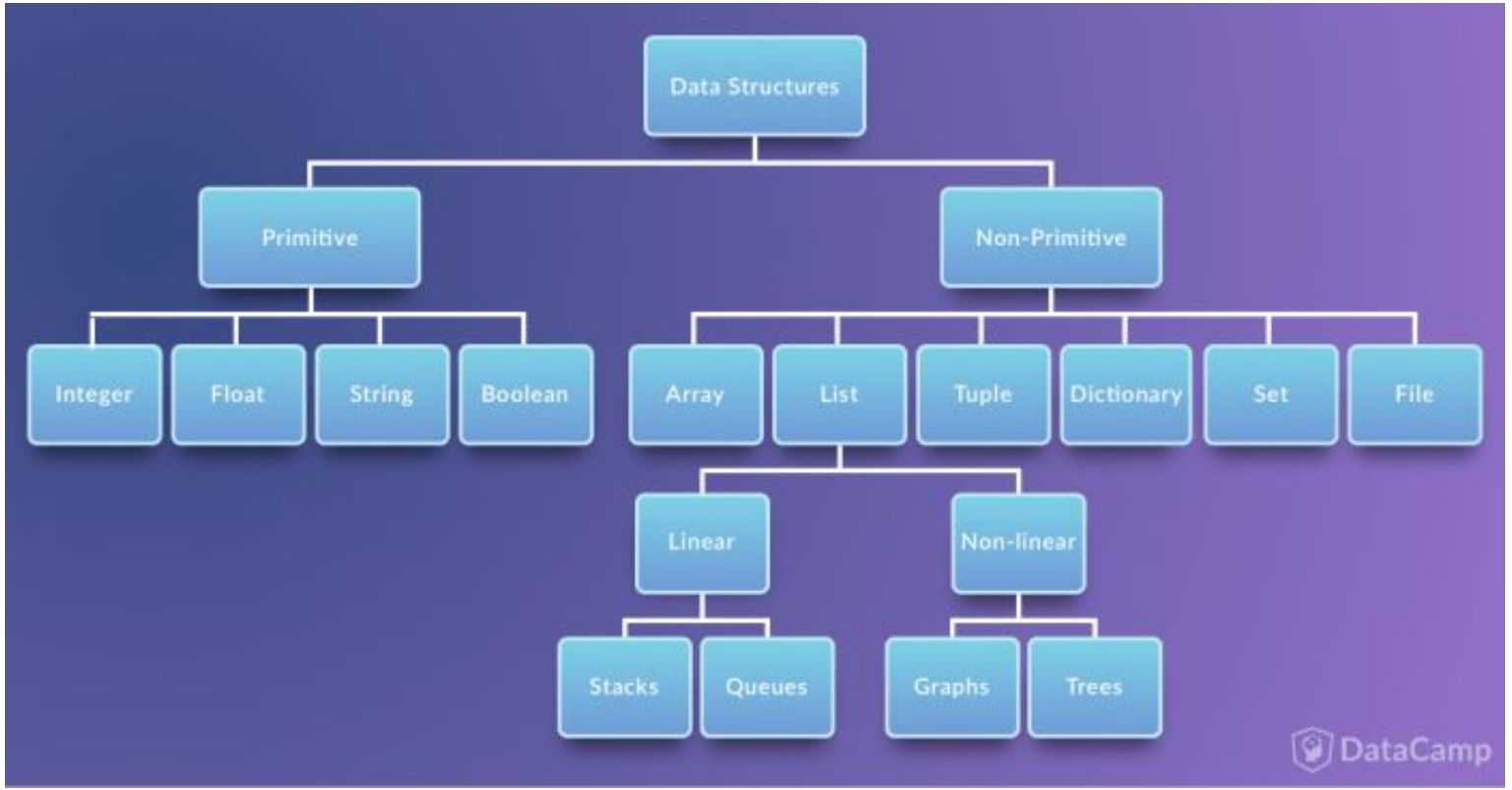
Download & Install



Basic Python

Data Types

Python Variables



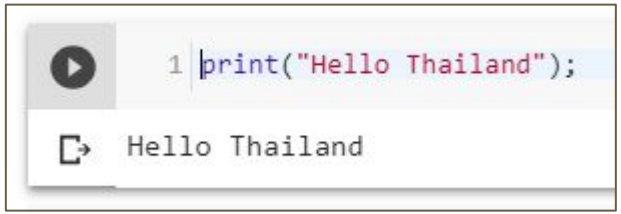
Source: datacamp.com

Print

Hello.py

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

```
python Hello.py
```



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))
```

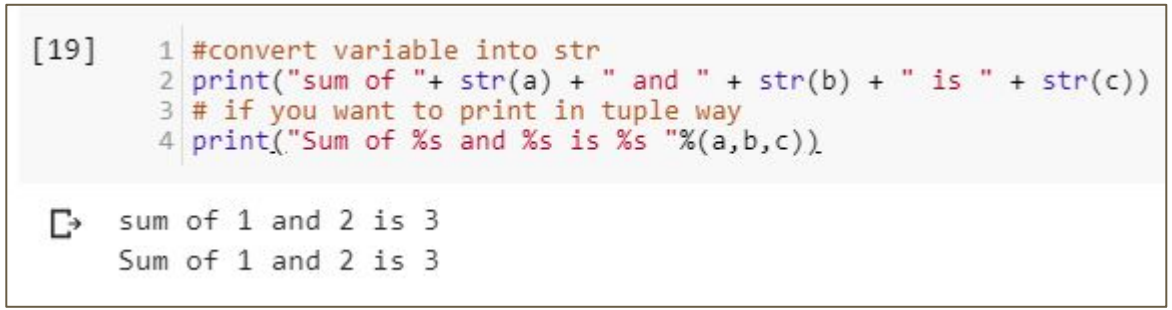
Concat String with Int

```
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
```

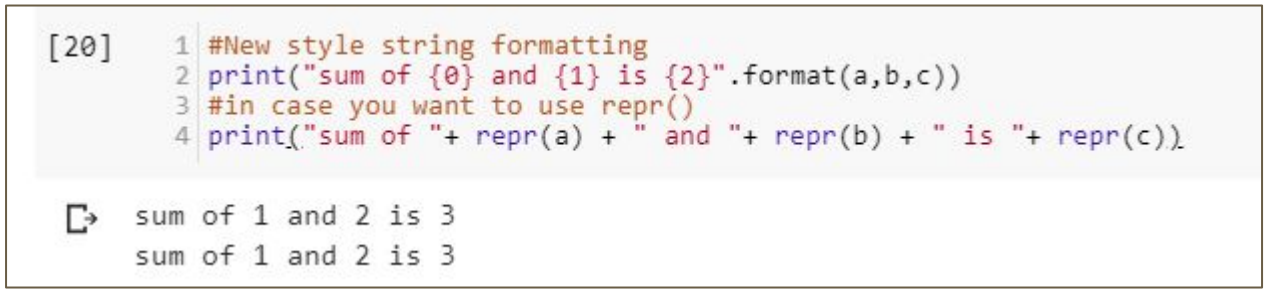
Concat String with Int (cont)

```
#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))
```



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))
```



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]
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"

[22]
print(str1)
print(str2)
print(str3)
print(str4)
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)
```

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
Lenghtof 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) = 572.3611111111111
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())
```

	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
```

```
df.day
df['day']
```

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

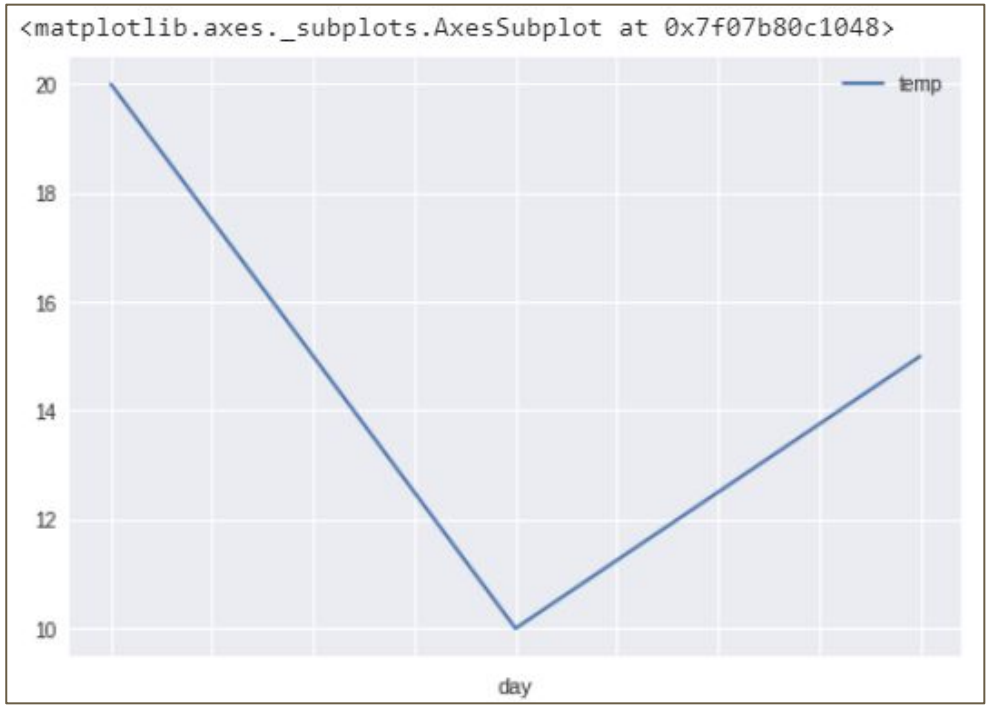
Data Frame

```
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()
```



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

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

```
5
9
```

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]) ทาให้เป็นตัวเลข complex
- str(x) แปลง object x ให้เป็น string
- repr(x) แปลง object x ให้เป็น expression string
- eval(str) หาค่า 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

```python
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

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

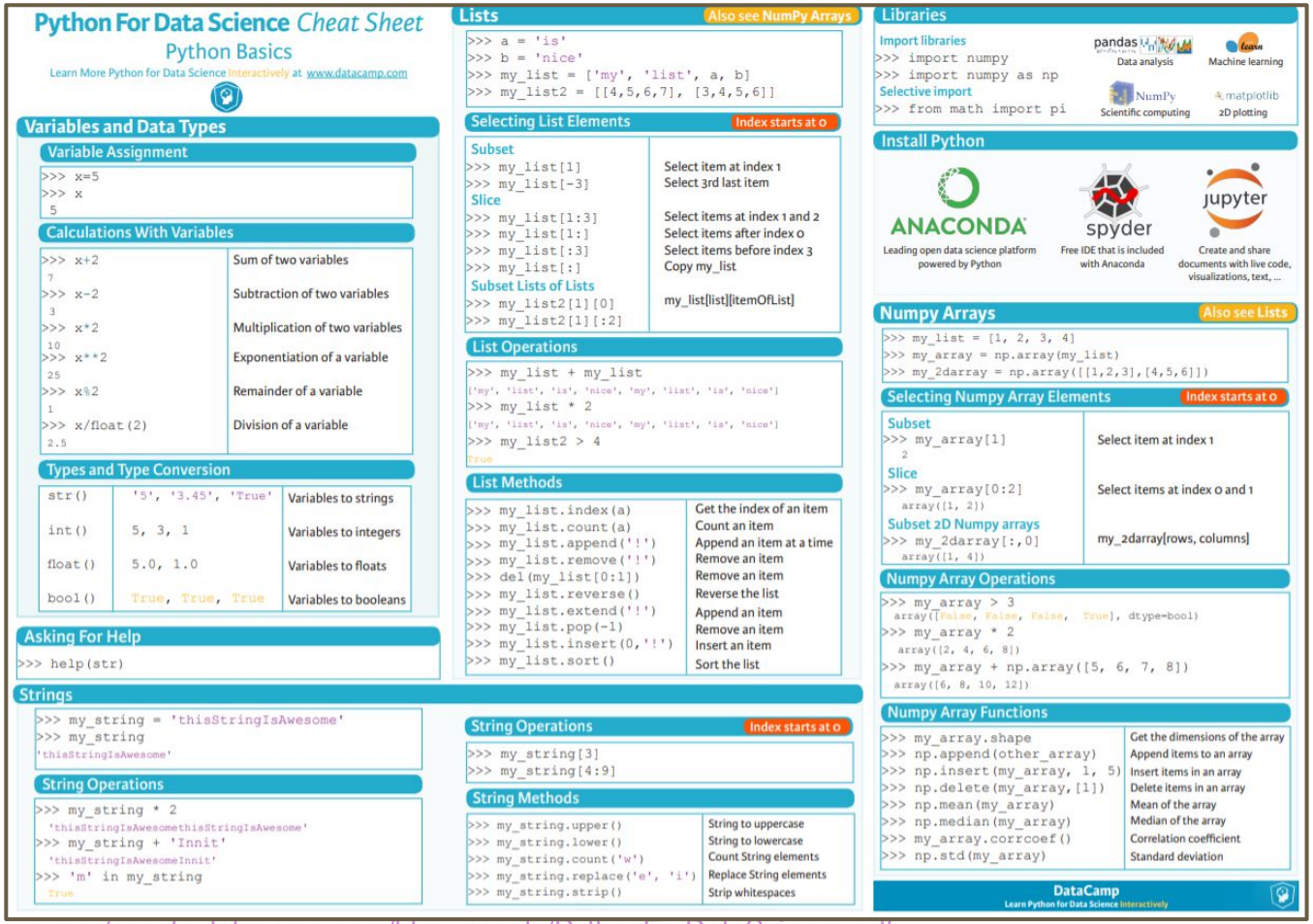
BMI kg/m²	อยู่ในเกณฑ์	ภาวะเสี่ยงต่อโรค
น้อยกว่า 18.50	น้ำหนักน้อย / ผอม	มากกว่าคนปกติ
ระหว่าง 18.50 - 22.90	ปกติ (สุขภาพดี)	เท่าคนปกติ
ระหว่าง 23 - 24.90	ท้วม / โรคอ้วนระดับ 1	อันตรายระดับ 1
ระหว่าง 25 - 29.90	อ้วน / โรคอ้วนระดับ 2	อันตรายระดับ 2
มากกว่า 30	อ้วนมาก / โรคอ้วนระดับ 3	อันตรายระดับ 3

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
```

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
>>> x-2
3
>>> x*2
10
>>> x**2
25
>>> x%2
1
>>> x/float(2)
2.5
```

Sum of two variables
Subtraction of two variables
Multiplication of two variables
Exponentiation of a variable
Remainder of a variable
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
'thisStringIsAwesomethisStringIsAwesome'
>>> my_string + 'Innit'
'thisStringIsAwesomeInnit'
>>> '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]
```

Select item at index 1
Select 3rd last item

Slice

```
>>> my_list[1:3]
>>> my_list[1:]
>>> my_list[:3]
>>> my_list[:]
```

Select items at index 1 and 2
Select items after index 0
Select items before index 3
Copy my_list

Subset Lists of Lists

```
>>> my_list2[1][0]
>>> my_list2[1][:2]
```

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

String Operations

Index starts at 0

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

String Methods

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

Libraries

Import libraries

```
>>> import numpy
>>> import numpy as np
```

Selective import

```
>>> from math import pi
```

pandas Data analysis
learn 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
```

Select item at index 1

Slice

```
>>> my_array[0:2]
array([1, 2])
```

Select items at index 0 and 1

Subset 2D Numpy arrays

```
>>> my_2darray[:,0]
array([1, 4])
```

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

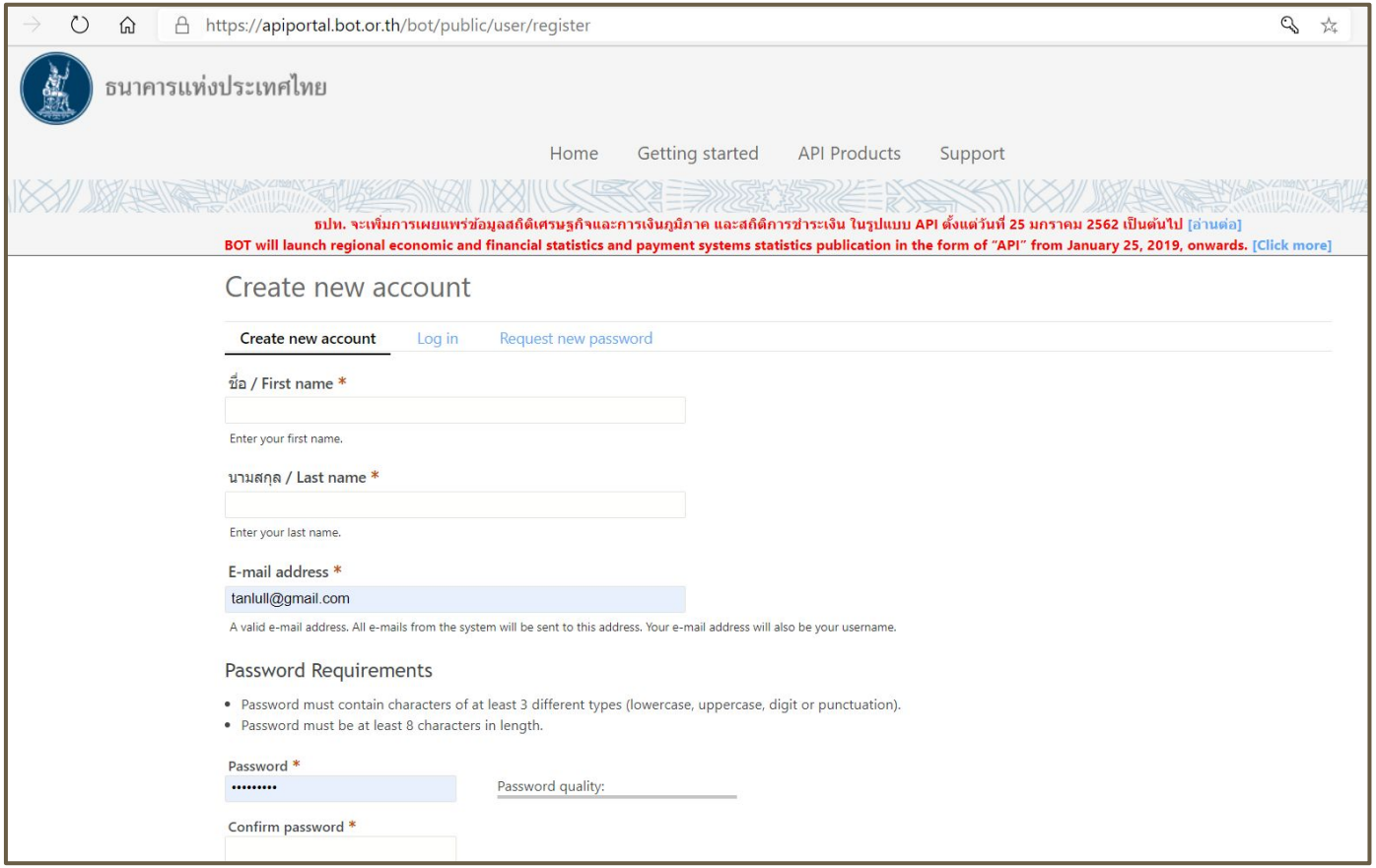
DataCamp
Learn Python for Data Science Interactively

https://s3.amazonaws.com/assets.datacamp.com/blog_assets/PythonForDataScience.pdf

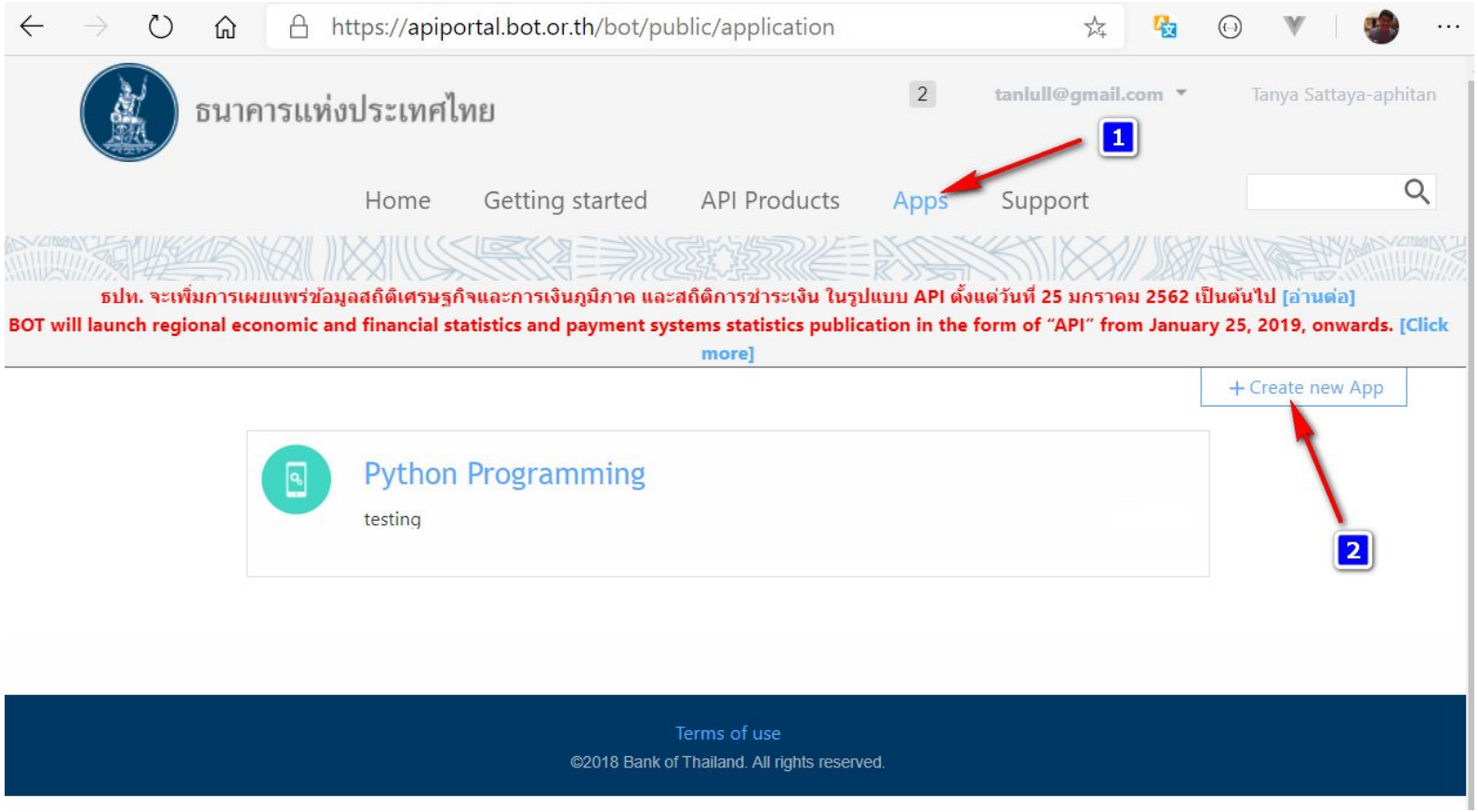
Call BOT API

BOT API

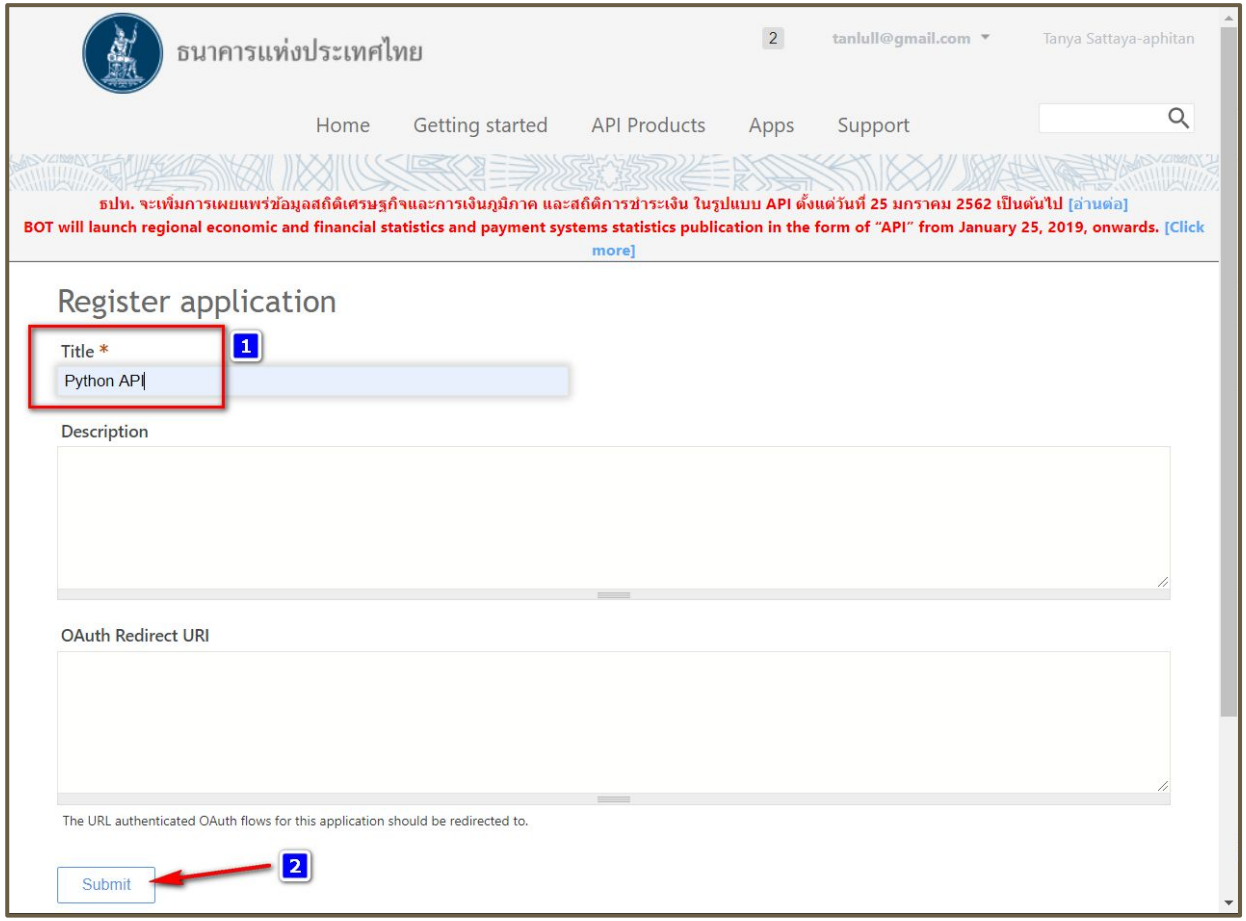
https://apiportal.bot.or.th/bot/public/user/register

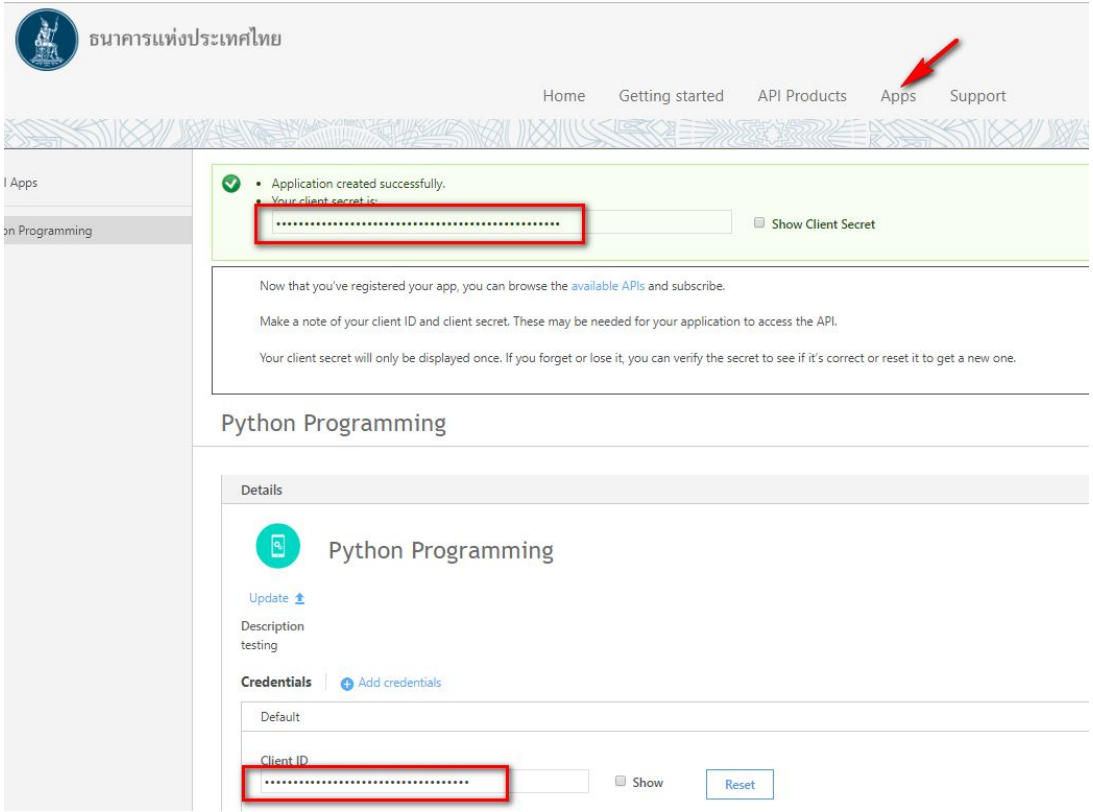


Register API



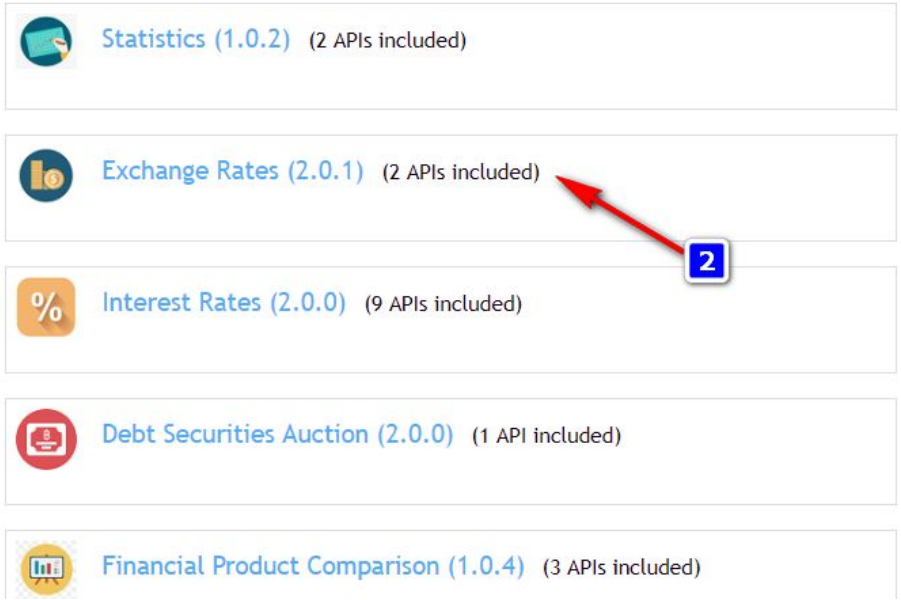
Register Application : Python API



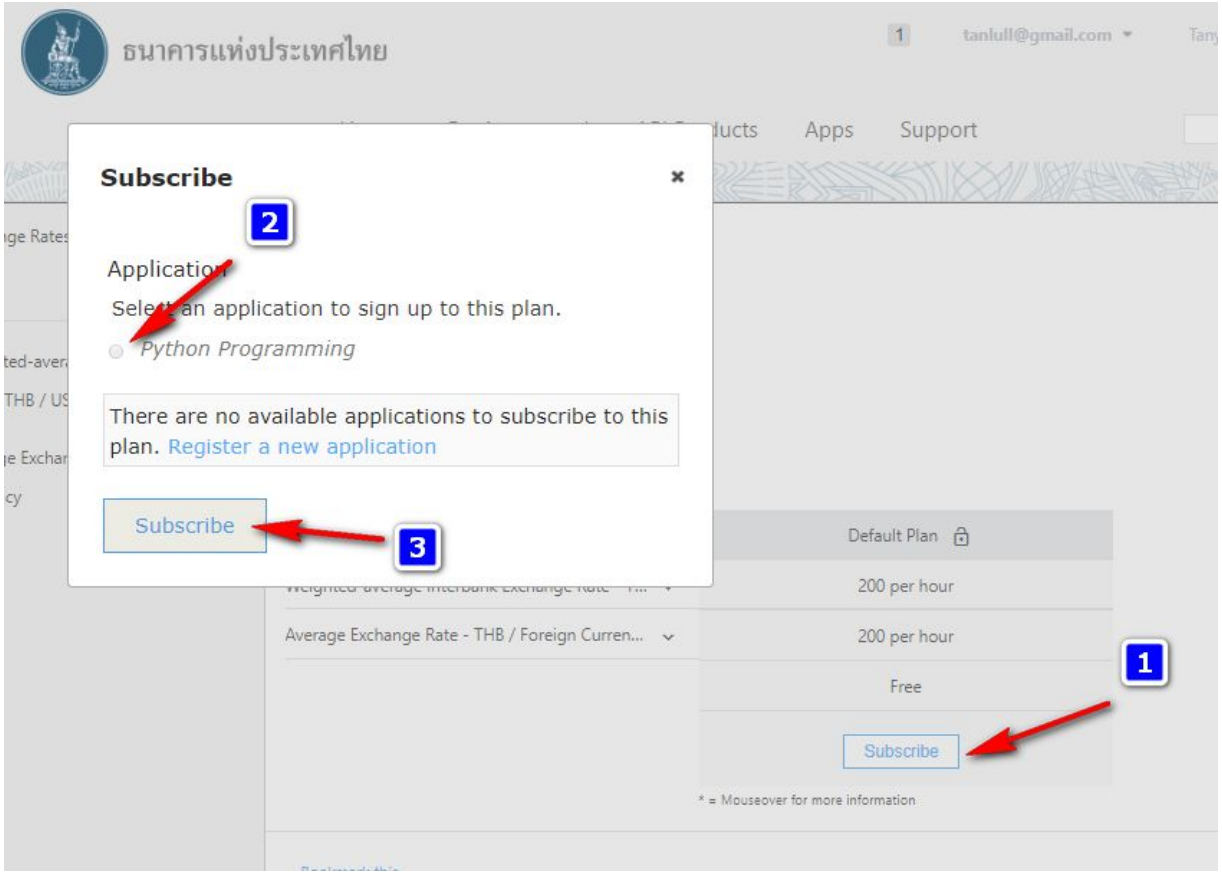


Copy Client ID to field in the next step : 1edd90f8-f587-4e21-9433-81db0f322825

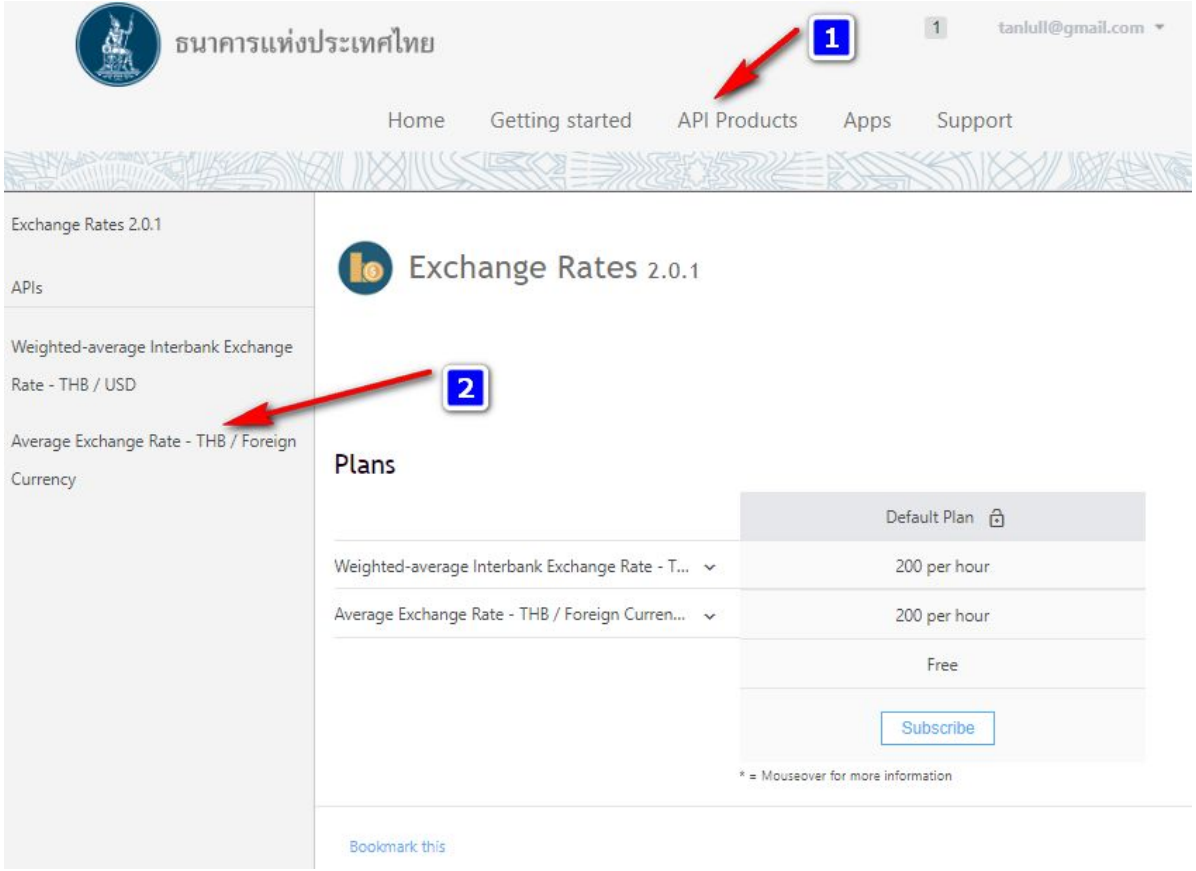
Activate API



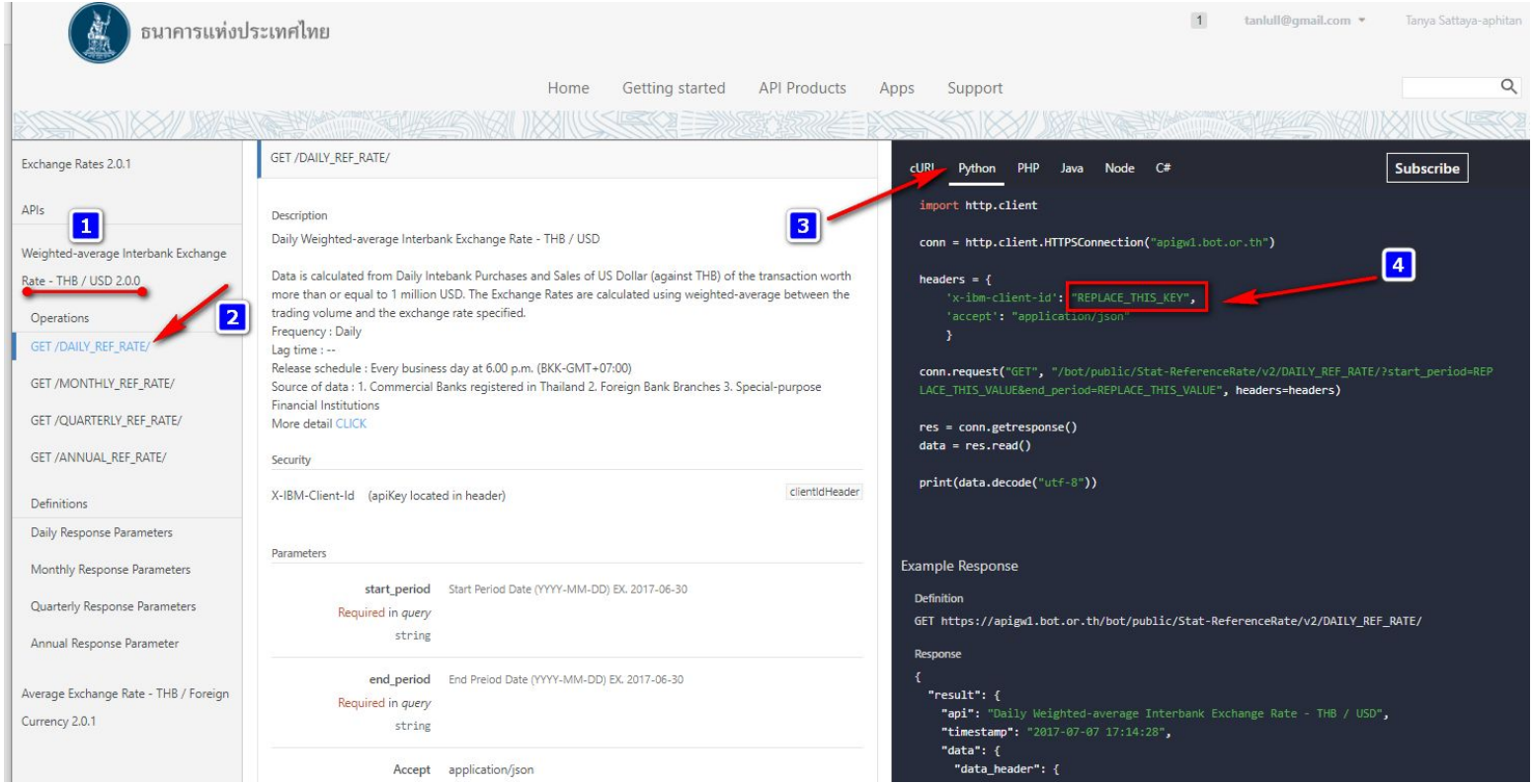
Activate API



Browse API


ธนาคารแห่งประเทศไทย
1
tanlull@gmail.com
Home
Getting started
API Products
Apps
Support
Exchange Rates 2.0.1
APIs
Weighted-average Interbank Exchange Rate - THB / USD
Average Exchange Rate - THB / Foreign Currency
2
Exchange Rates 2.0.1
Plans
Default Plan
Weighted-average Interbank Exchange Rate - T...
200 per hour
Average Exchange Rate - THB / Foreign Curren...
200 per hour
Free
Subscribe
* = Mouseover for more information
Bookmark this

Code Example



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"))
```

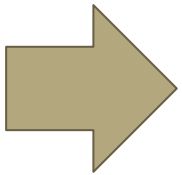
Response JSON


```
{
  "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":"อัตราแลกเปลี่ยนเฉลี่ยของธนาคารพาณิชย์ในกรุงเทพมหานคร
(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"
        }
      ]
    }
  }
}
```

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)
```

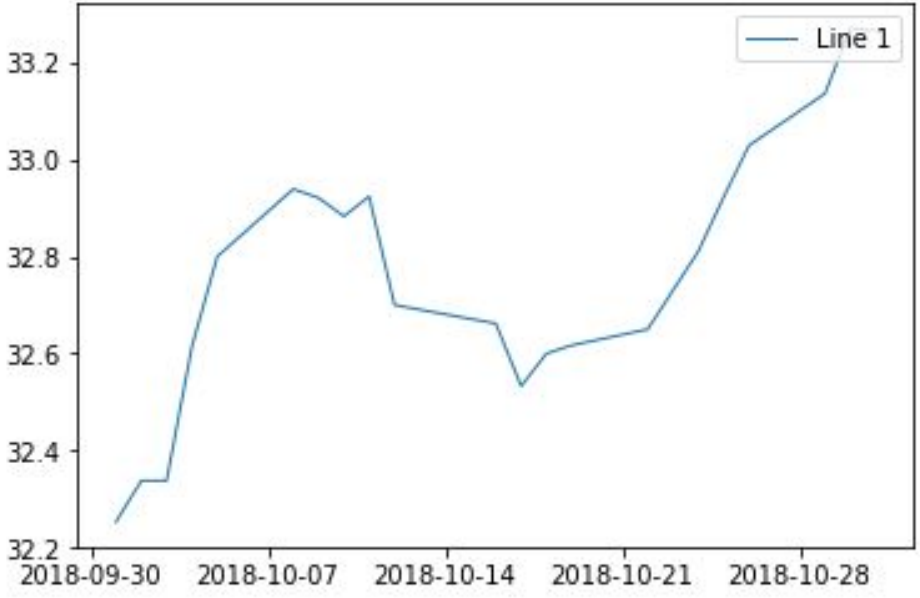


	period	rate
0	2018-10-31	33.266
1	2018-10-30	33.270
2	2018-10-29	33.136
3	2018-10-26	33.029
4	2018-10-25	32.923
5	2018-10-24	32.811
6	2018-10-22	32.650
7	2018-10-19	32.617
8	2018-10-18	32.600
9	2018-10-17	32.533
10	2018-10-16	32.662
11	2018-10-12	32.700
12	2018-10-11	32.924
13	2018-10-10	32.883
14	2018-10-09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

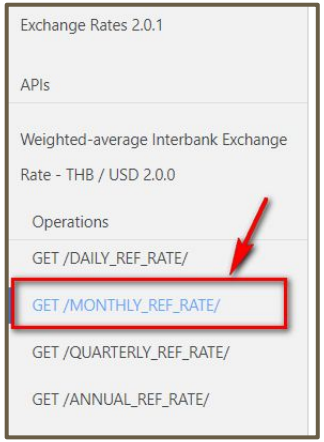
```
pip install matplotlib
```

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



Request API Lab 1

1. Create BOT_API_Monthly.py
2. Using

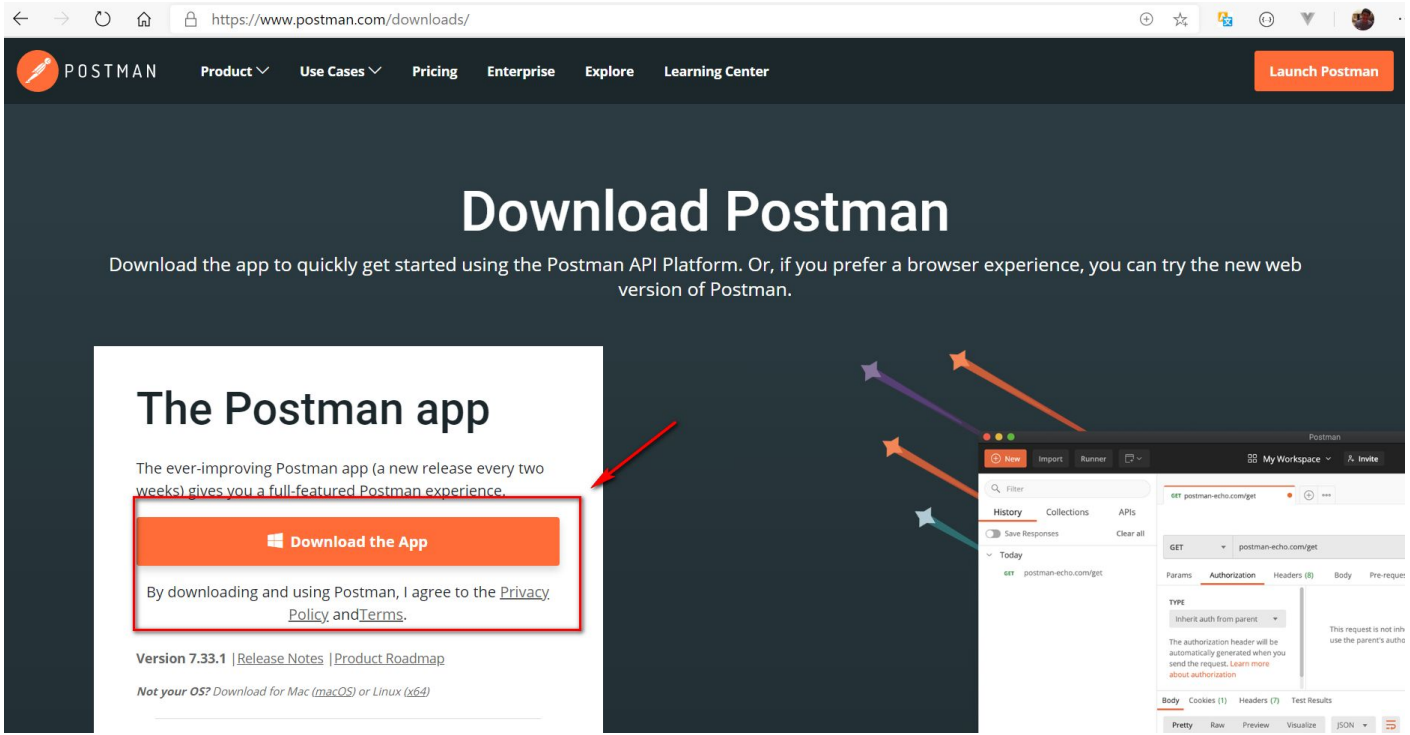


3. Retrieve data from January 2017 to September 2020
4. Print data
5. Plot Graph

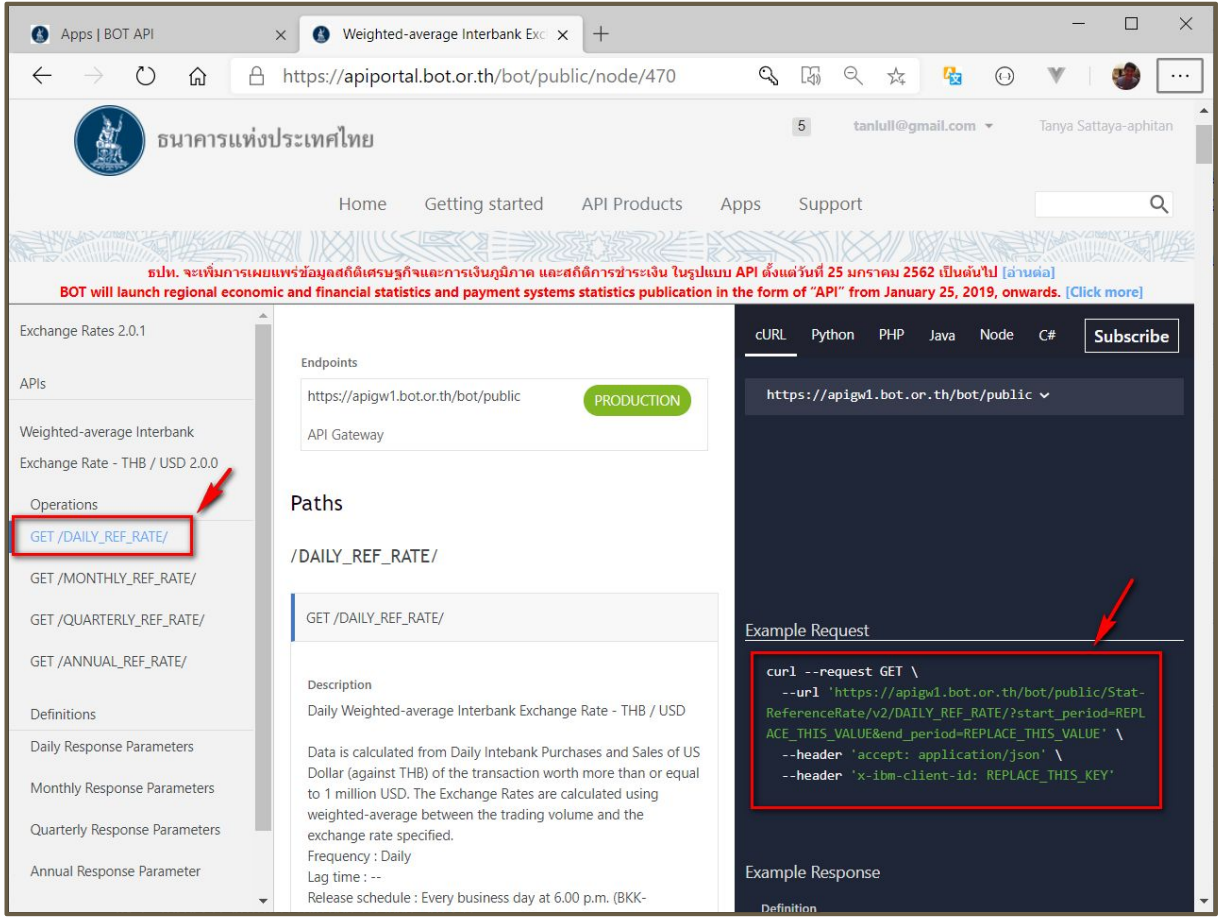
Test with postman

Test with Postman

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



Get Endpoint



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'
```

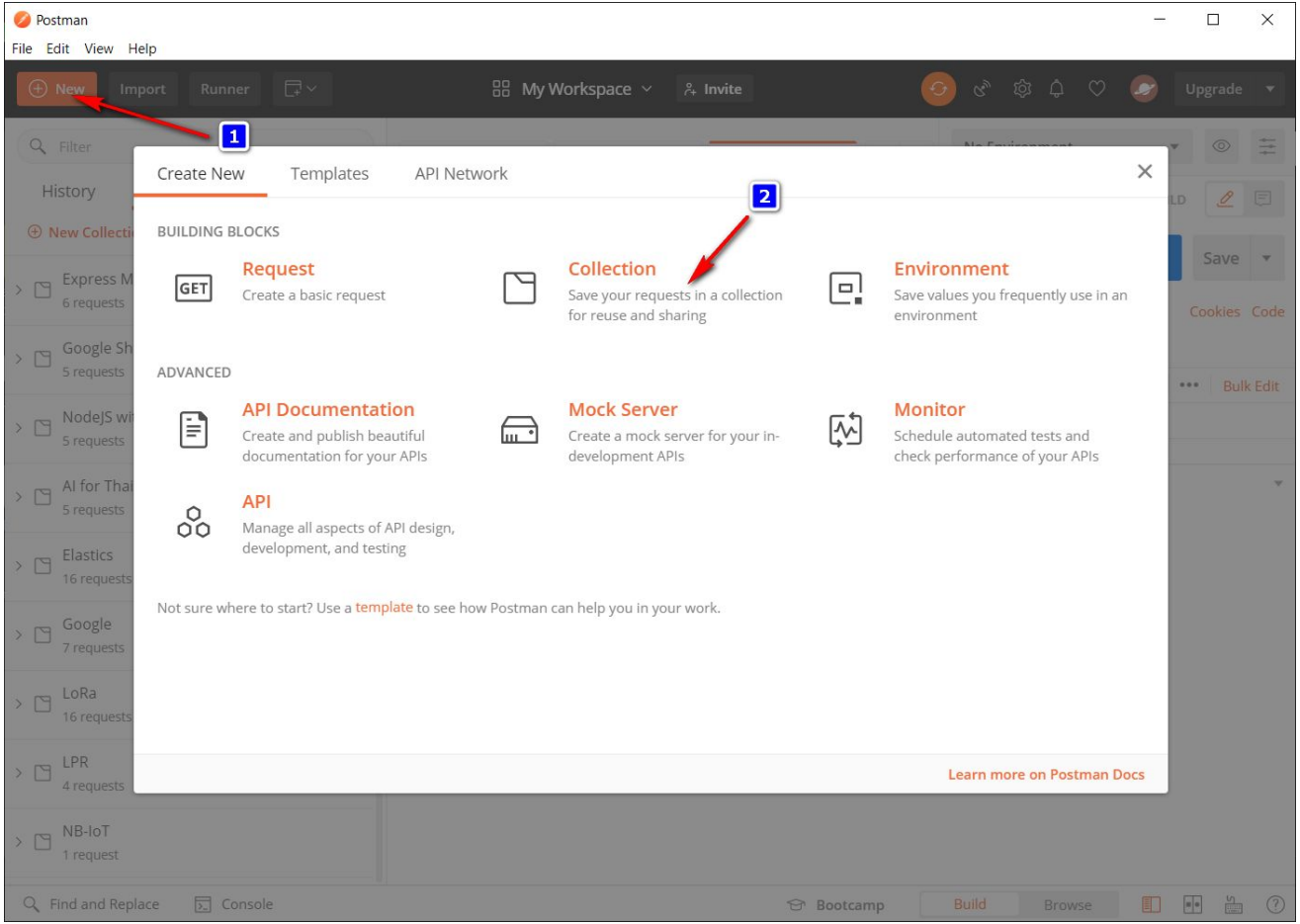
Method: GET

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

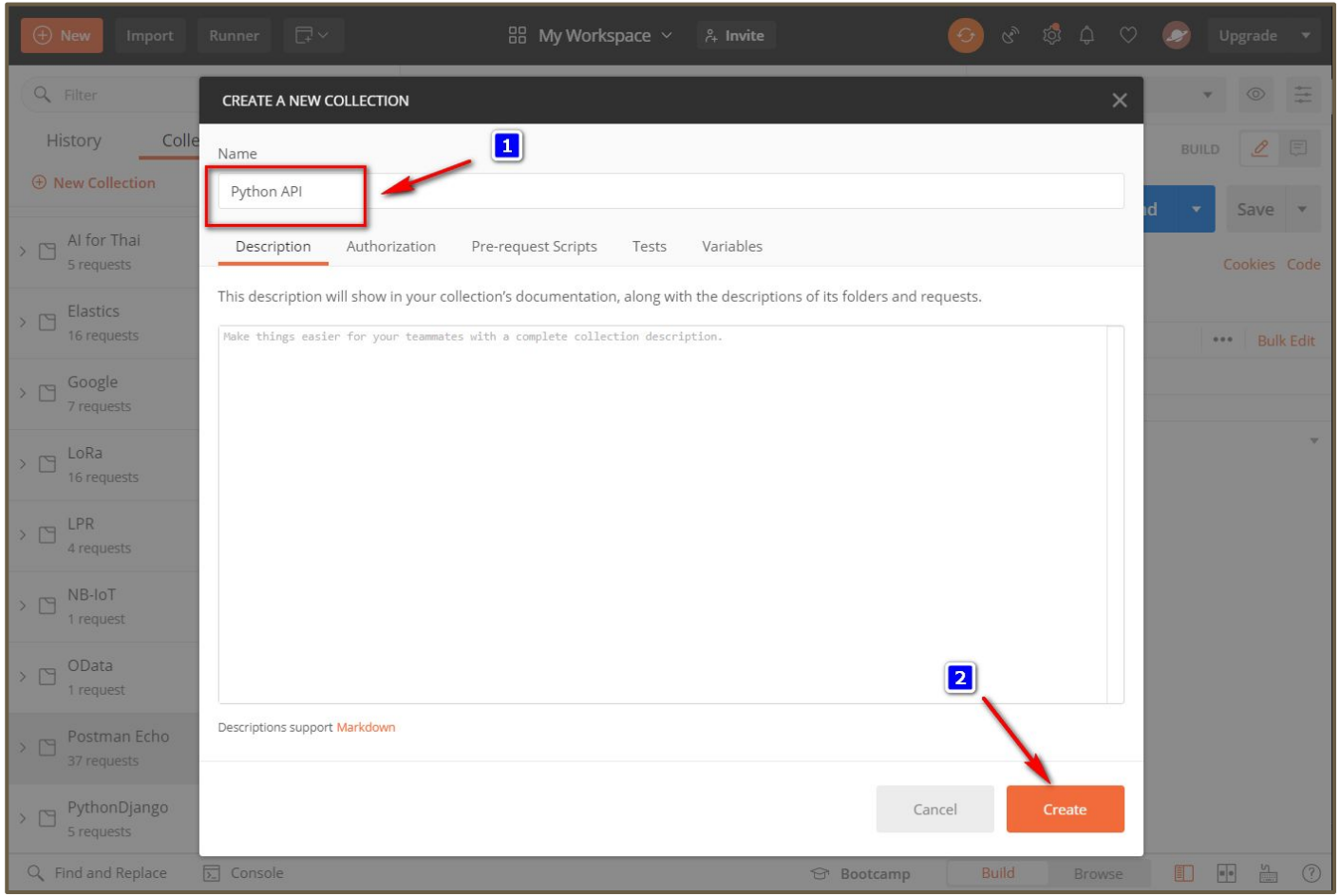
Header: accept: application/json

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

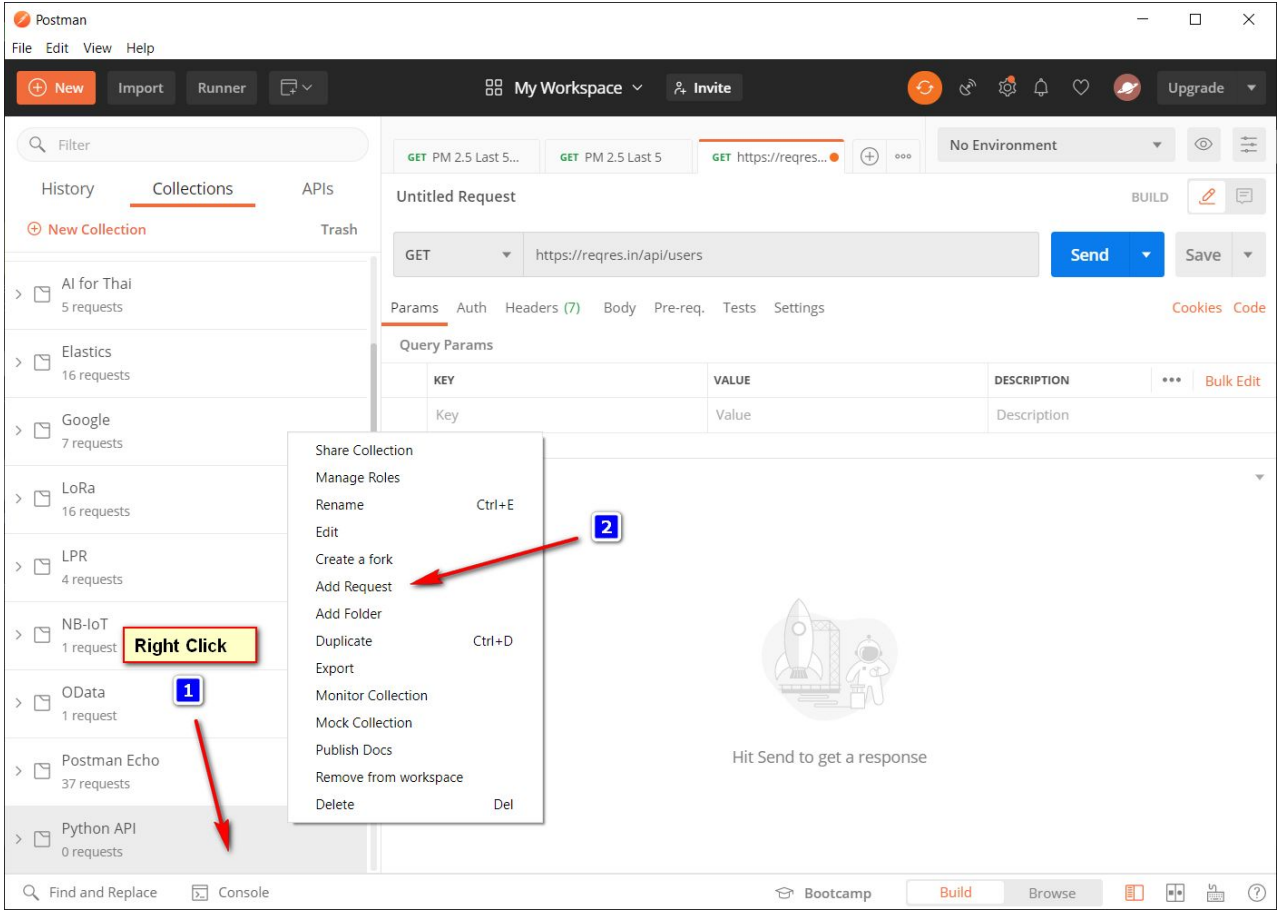
Postman


Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
1
2
Create New
Templates
API Network
BUILDING BLOCKS
Request
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
History
Save
Cookies
Code
Bulk Edit
Find and Replace
Console
Bootcamp
Build
Browse

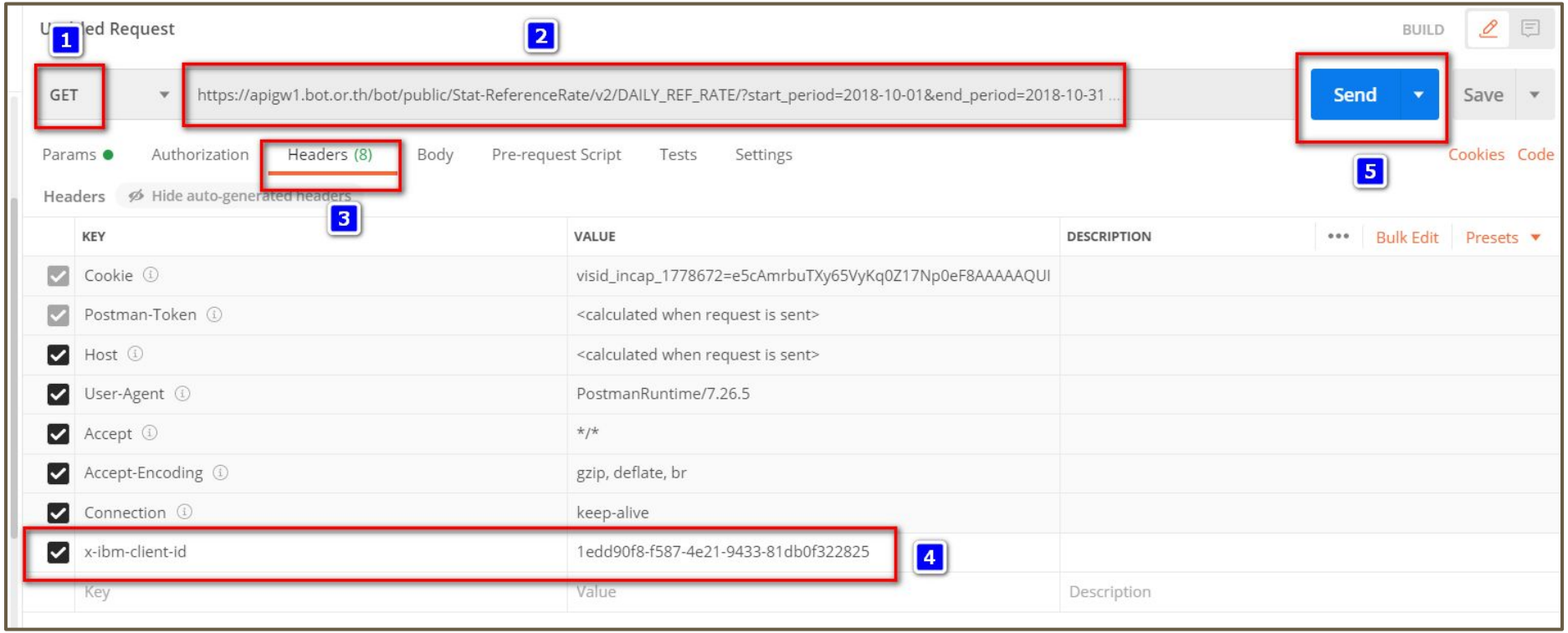
New Collection



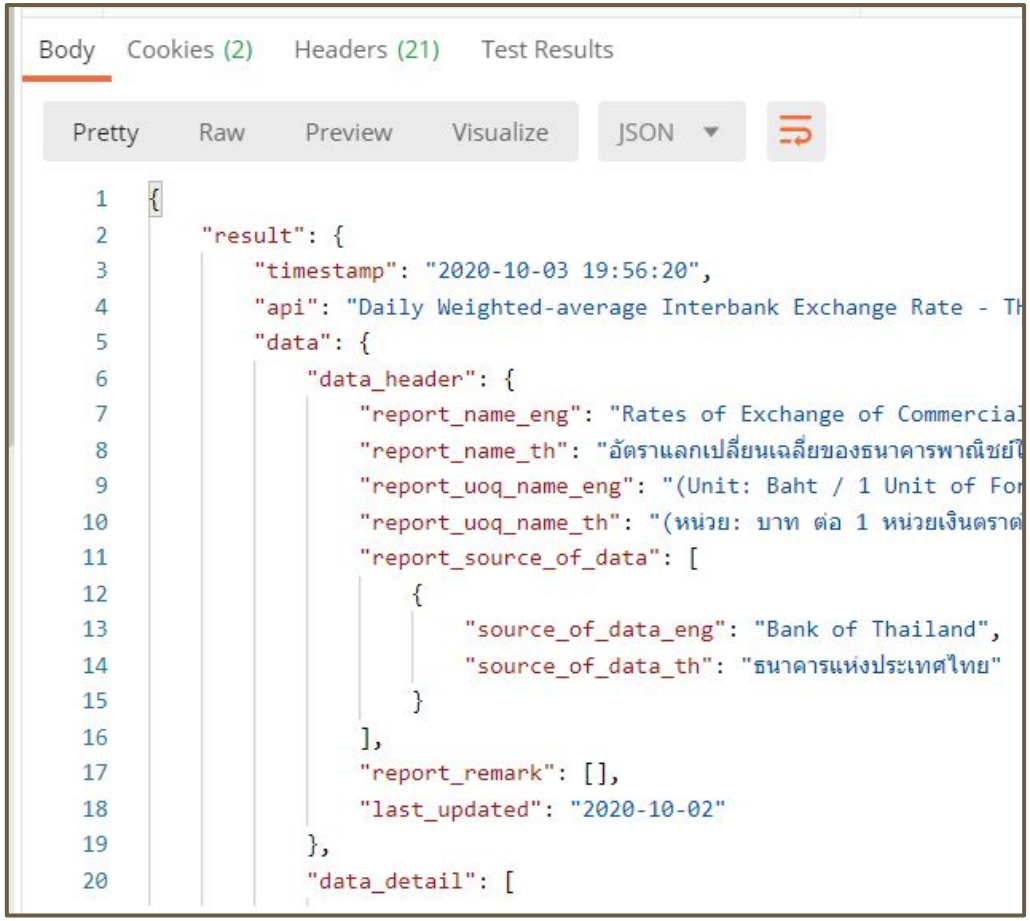
Add Request


Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
Filter
History
Collections
APIs
New Collection
Trash
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
Right Click
1
2
Share Collection
Manage Roles
Rename Ctrl+E
Edit
Create a fork
Add Request
Add Folder
Duplicate Ctrl+D
Export
Monitor Collection
Mock Collection
Publish Docs
Remove from workspace
Delete Del
GET PM 2.5 Last 5...
GET PM 2.5 Last 5
GET https://reqres...
No Environment
Untitled Request
BUILD
GET
https://reqres.in/api/users
Send
Save
Params Auth Headers (7) Body Pre-req. Tests Settings
Cookies Code
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Hit Send to get a response
Find and Replace
Console
Bootcamp
Build
Browse

GET / URI


1
2
BUILD
GET
https://apigw1.bot.or.th/bot/public/Stat-ReferenceRate/v2/DAILY_REF_RATE/?start_period=2018-10-01&end_period=2018-10-31 ...
Send
Save
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
Code
5
Headers
Hide auto-generated headers
3
KEY
VALUE
DESCRIPTION
Bulk Edit
Presets
Cookie
visid_incap_1778672=e5cAmrbuTXy65VyKq0Z17Np0eF8AAAAAQUI
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



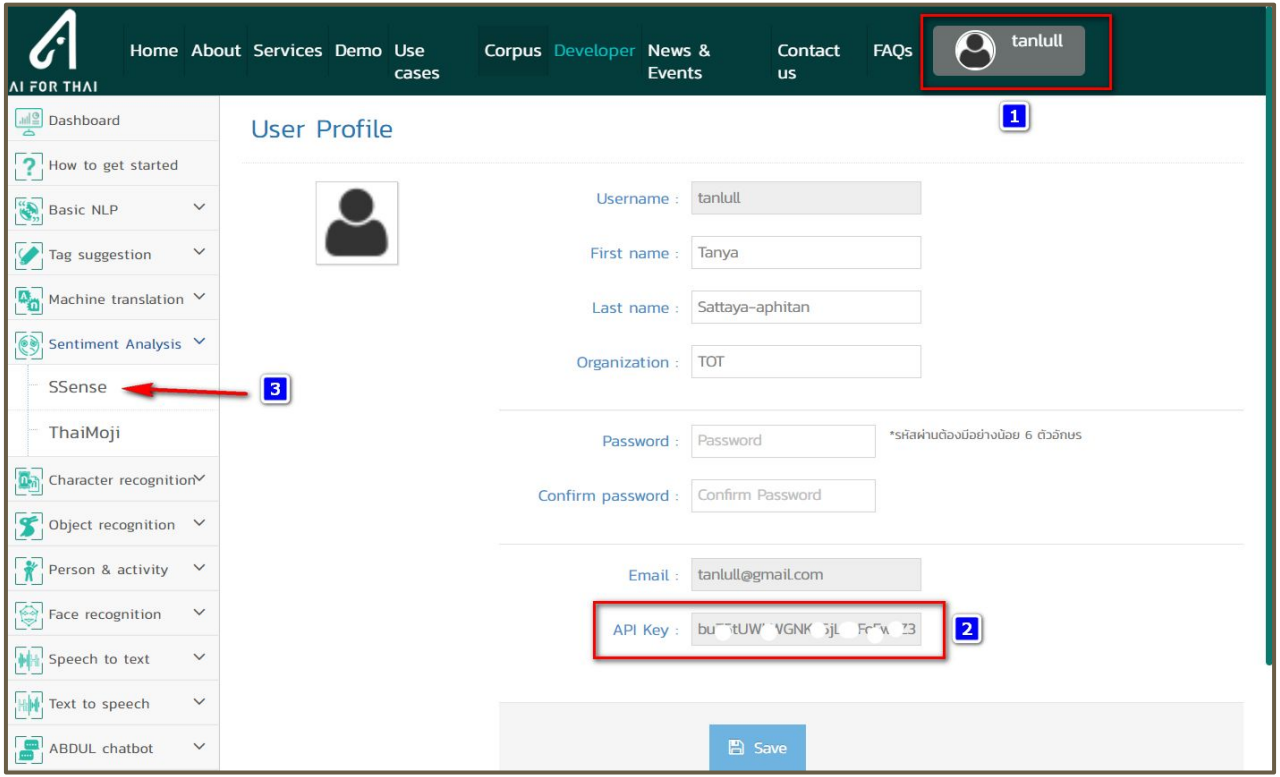

```
{
    "result": {
        "timestamp": "2020-10-03 19:56:20",
        "api": "Daily Weighted-average Interbank Exchange Rate - Th
        "data": {
            "data_header": {
                "report_name_eng": "Rates of Exchange of Commercial
                "report_name_th": "อัตราแลกเปลี่ยนเฉลี่ยของธนาคารพาณิชย์ใ
                "report_uoq_name_eng": "(Unit: Baht / 1 Unit of For
                "report_uoq_name_th": "(หน่วย: บาท ต่อ 1 หน่วยเงินตราต่
                "report_source_of_data": [
                    {
                        "source_of_data_eng": "Bank of Thailand",
                        "source_of_data_th": "ธนาคารแห่งประเทศไทย"
                    }
                ],
                "report_remark": [],
                "last_updated": "2020-10-02"
            },
            "data_detail": [
```

```
                "last_updated": "2020-10-02"
            },
            "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"
                },
                {
```

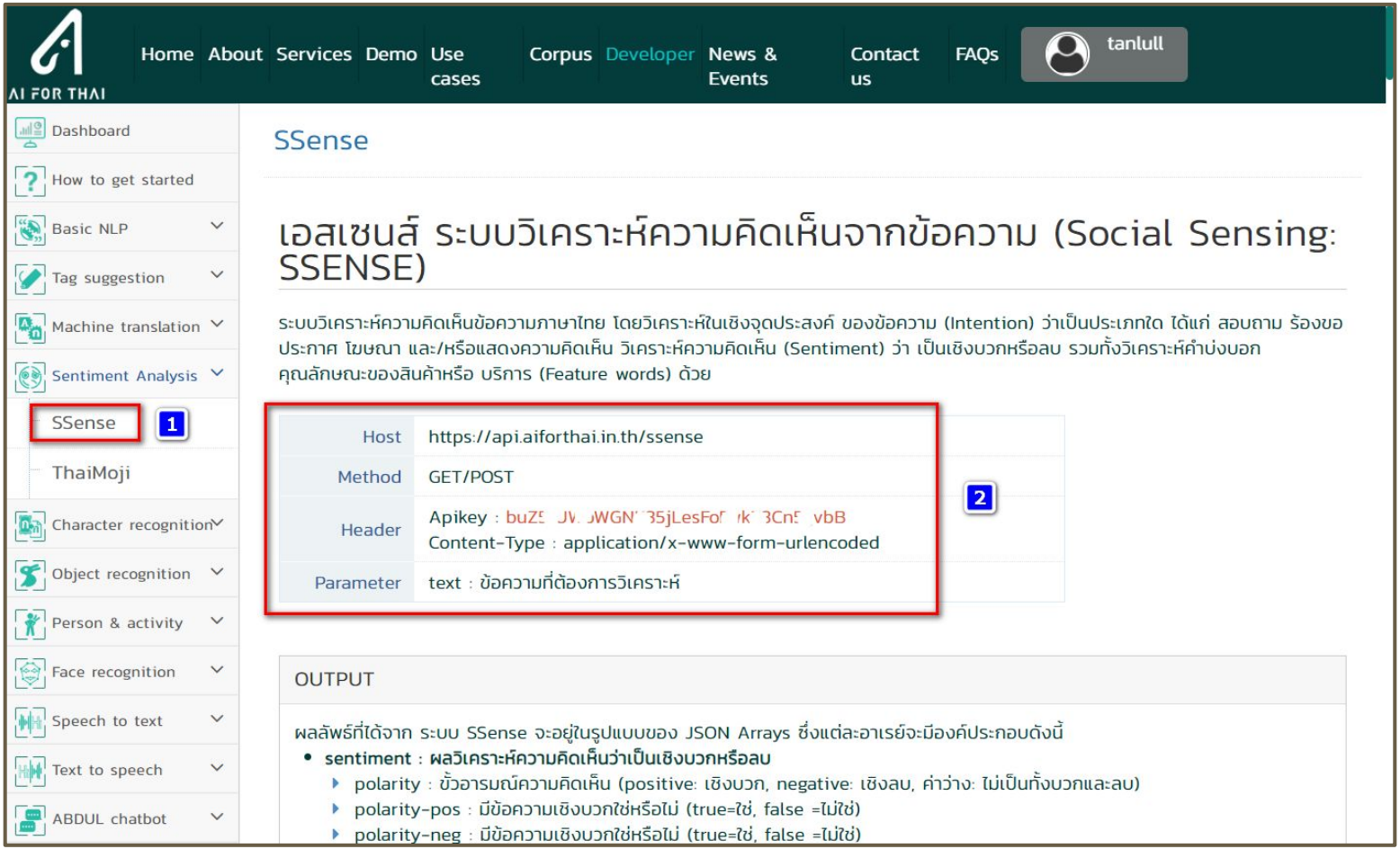
AI 4 Thai API

Request API Lab 2

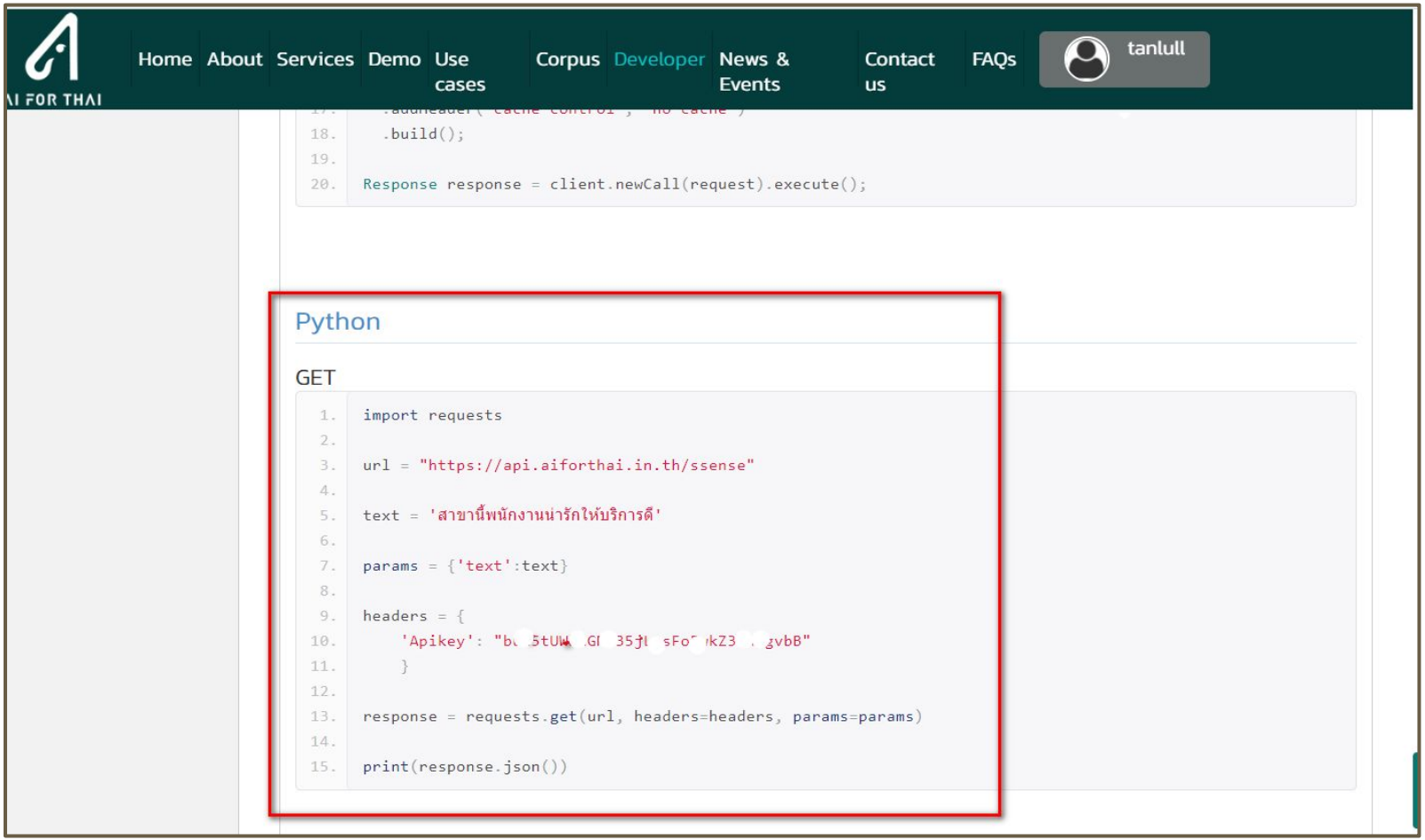
https://aiforthai.in.th/



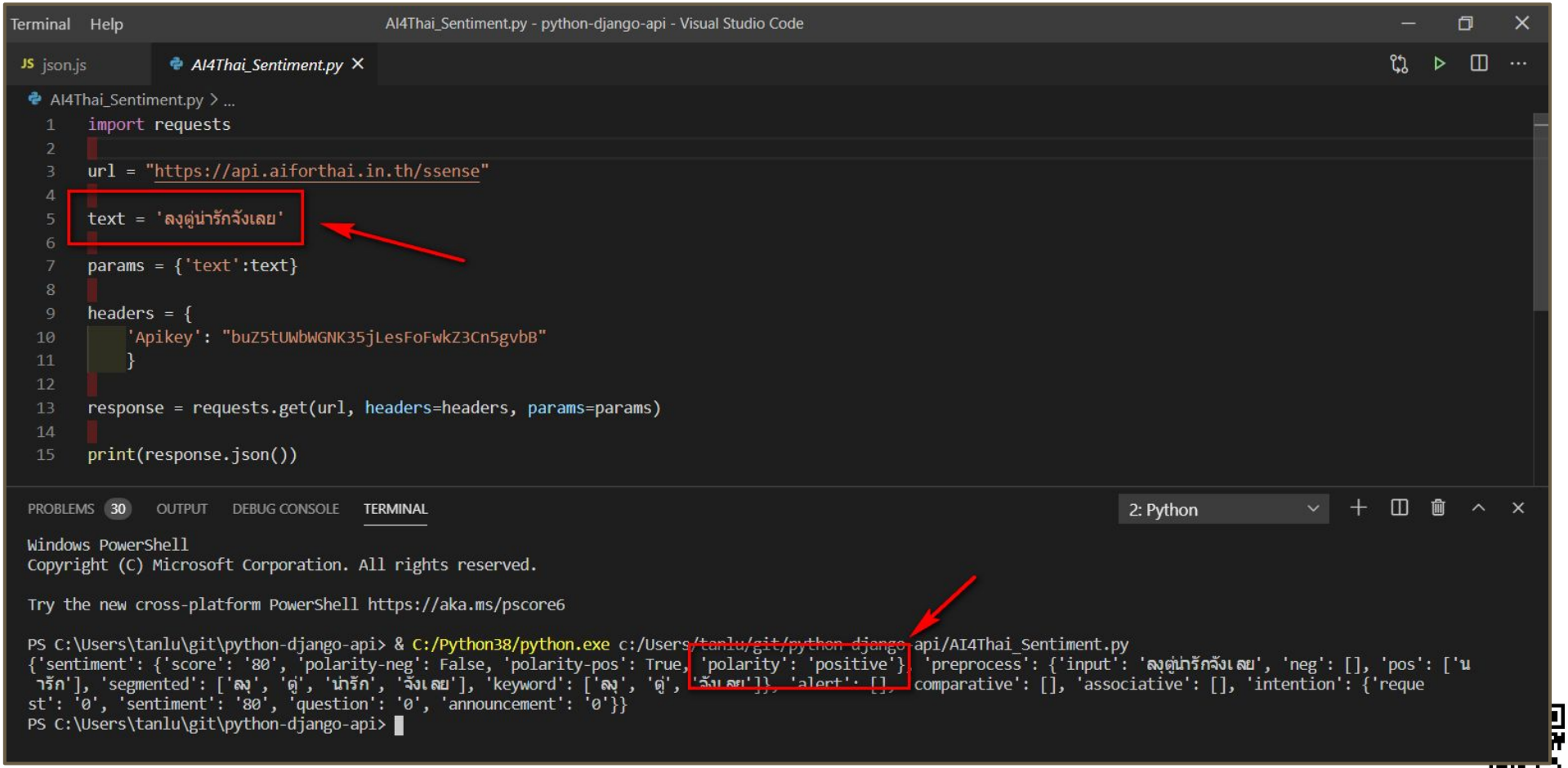
Sentiment Analysis



Python Example

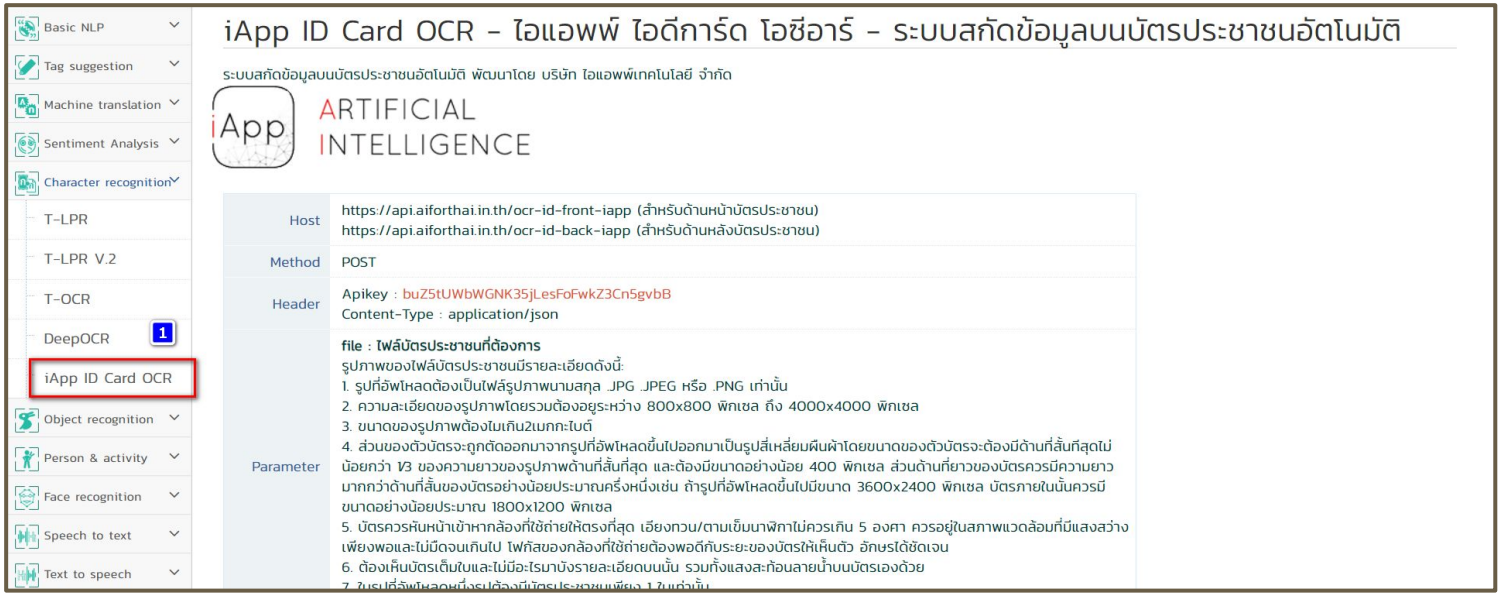


Output


Terminal Help
AI4Thai_Sentiment.py - python-django-api - Visual Studio Code
json.js
AI4Thai_Sentiment.py
AI4Thai_Sentiment.py > ...
1 import requests
2
3 url = "https://api.aiforthai.in.th/ssense"
4
5 text = 'ลุงตู่น่ารักจังเลย'
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
2: Python
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': 'ลุงตู่น่ารักจังเลย', 'neg': [], 'pos': ['น
ารัก'], 'segmented': ['ลุง', 'ตู่', 'น่ารัก', 'จังเลย'], 'keyword': ['ลุง', 'ตู่', 'จังเลย']}, 'alert': [], 'comparative': [], 'associative': [], 'intention': {'reque
st': '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

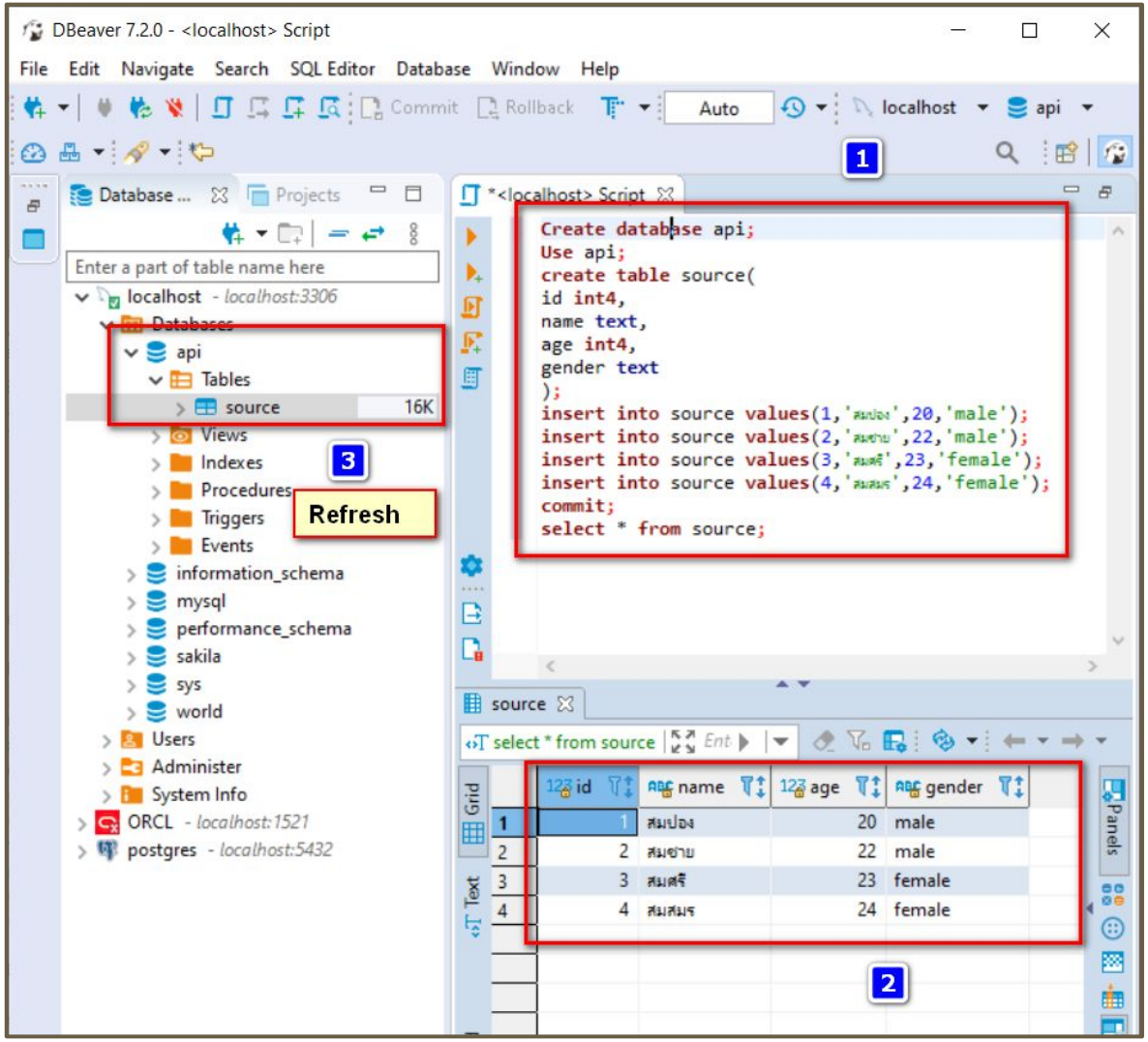
```
pip install mysql-connector-python
```

Prepare Database (Postgres)

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

```
insert into source values(1,'สมปอง',20,'male');
insert into source values(2,'สมชาย',22,'male');
insert into source values(3,'สมศรี',23,'female');
insert into source values(4,'สมสมร',24,'female');
commit;
select * from source
```

Result


DBeaver 7.2.0 - <localhost> Script
File Edit Navigate Search SQL Editor Database Window Help
Commit Rollback Auto localhost api
1
Database ... Projects
Enter a part of table name here
localhost - localhost:3306
Databases
api
Tables
source 16K
Views
Indexes
Procedures
Triggers
Events
information_schema
mysql
performance_schema
sakila
sys
world
Users
Administer
System Info
ORCL - localhost:1521
postgres - localhost:5432
3
Refresh
*<localhost> Script
Create database api;
Use api;
create table source(
id int4,
name text,
age int4,
gender text
);
insert into source values(1,'สมปอง',20,'male');
insert into source values(2,'สมชาย',22,'male');
insert into source values(3,'สมศรี',23,'female');
insert into source values(4,'สมสมร',24,'female');
commit;
select * from source;
source
select * from source
id name age gender
1 1 สมปอง 20 male
2 2 สมชาย 22 male
3 3 สมศรี 23 female
4 4 สมสมร 24 female
Grid
Text
Panels
2

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")
```

https://raw.githubusercontent.com/tanlull/Python-Postgres-Mysql/master/insert_postgres.py

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, 'สมเจตน์', 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

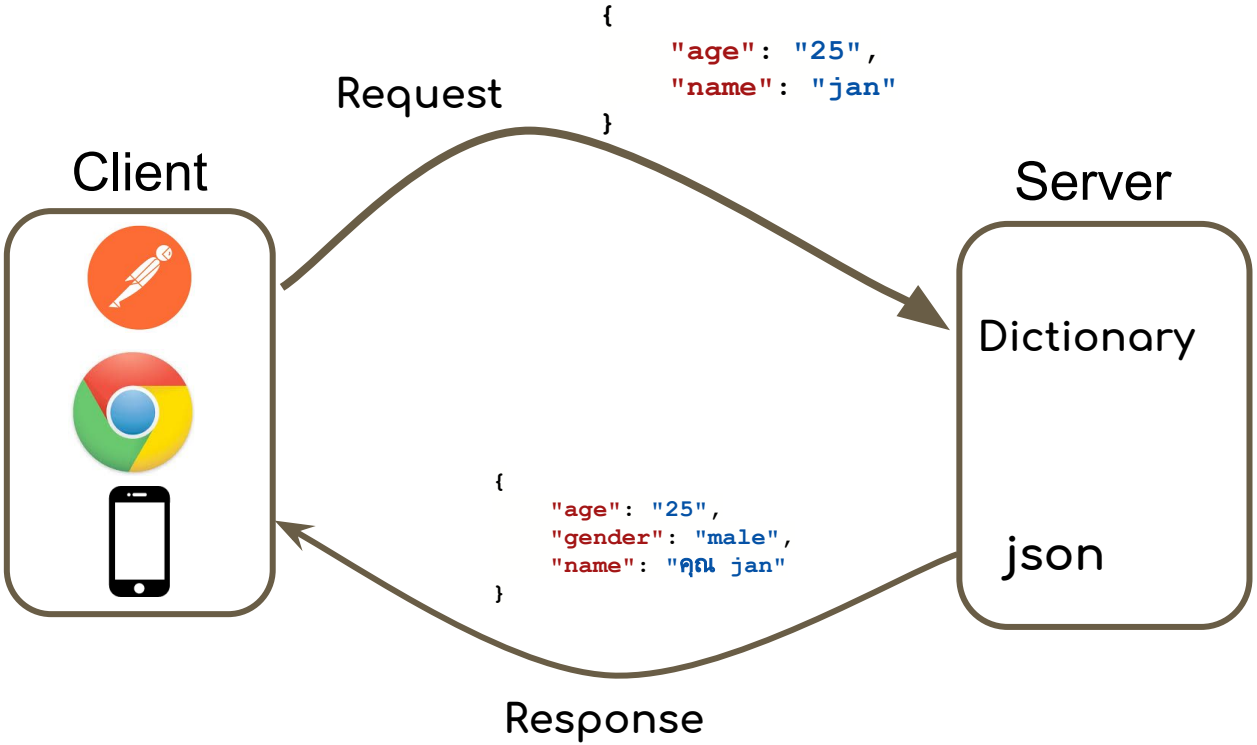
```
import crud_mysql as my


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


{
"age": "25",
"name": "jan"
}
Request
Client
Server
Dictionary
{
"age": "25",
"gender": "male",
"name": "คุณ jan"
}
json
Response

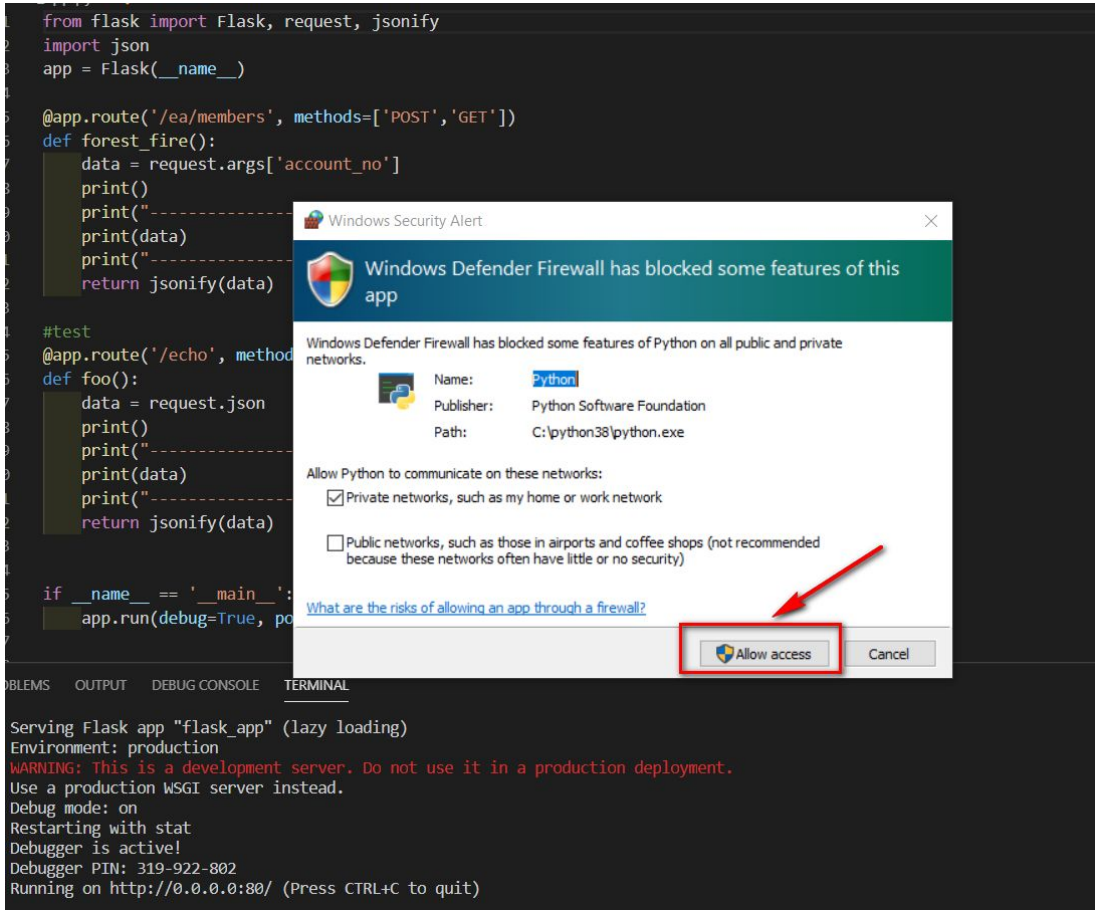
Request with json

Flask API : flask_echo.py

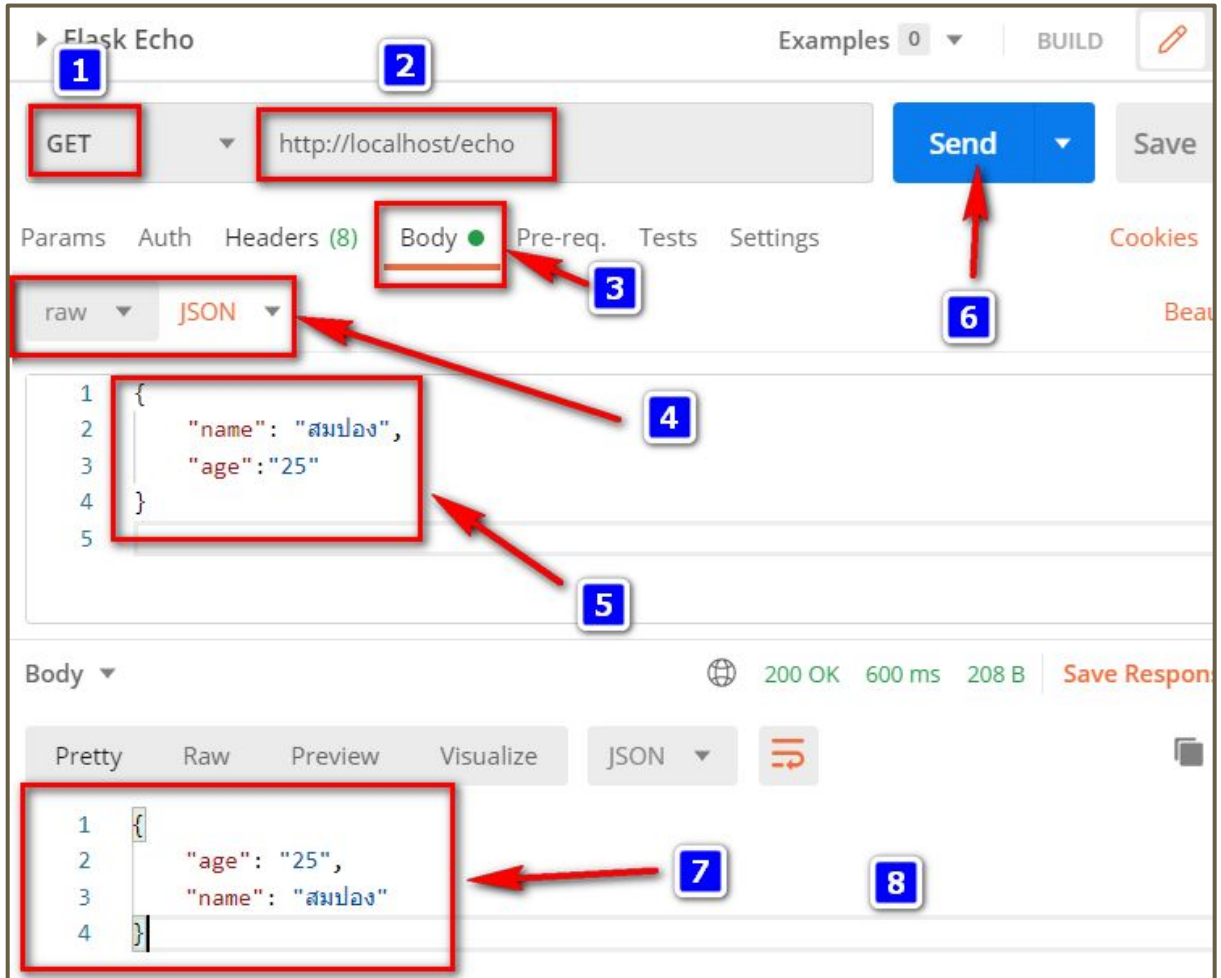
```python
from flask import Flask, request, jsonify
import json
app = Flask(__name__)
#test
@app.route('/echo', methods=['POST','GET'])
def foo():
    data = request.json
    print(type(data))
    print()
    print("------------------------- ECHO -------------------------")
    print(data)
    print("--------------------------------------------------------")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



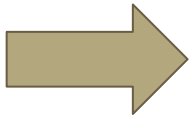
Test with Postman



Recall dictionary data type :

dictionary_test.py

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

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

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

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

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



```
<class 'dict'>
เจน
นุ่น
{'age': '25', 'name': 'นุ่น', 'friend': 'โบ'}
{'age': '25', 'name': 'นุ่น'}
```

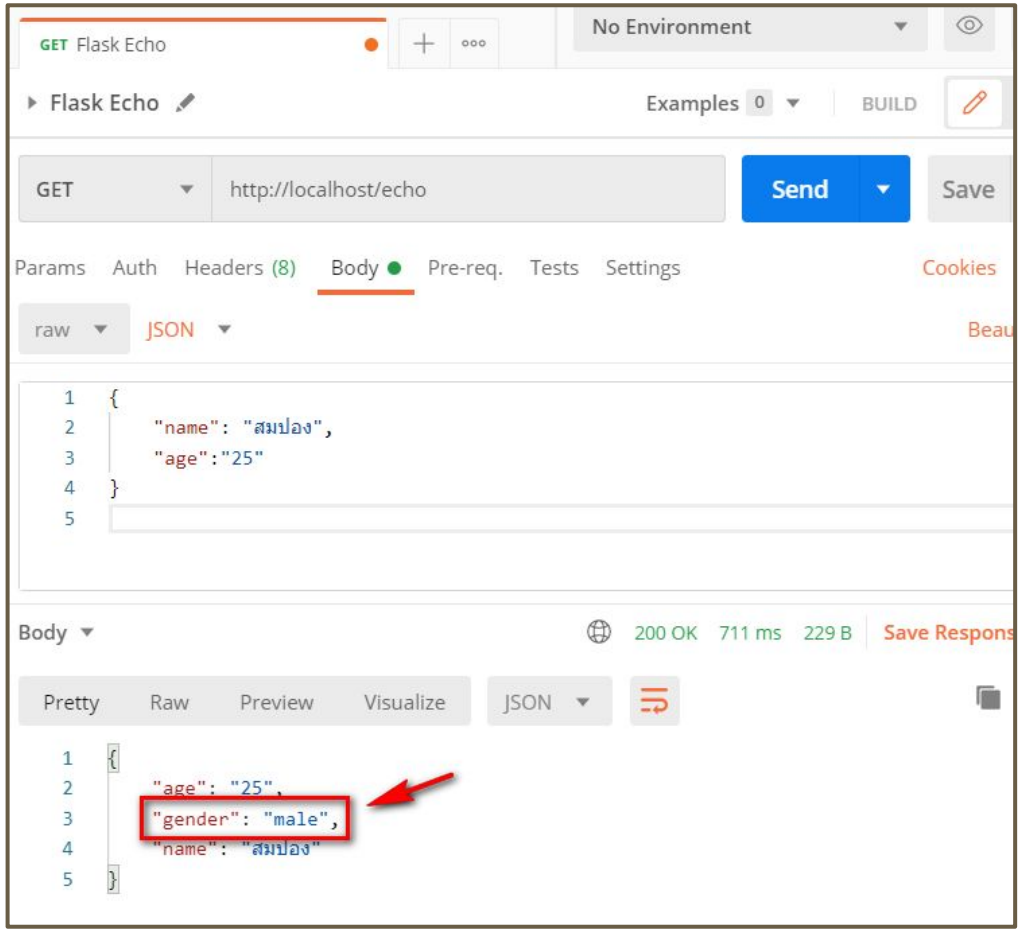
To Stop Running Server -> Ctrl+C

```
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>
```

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')
```

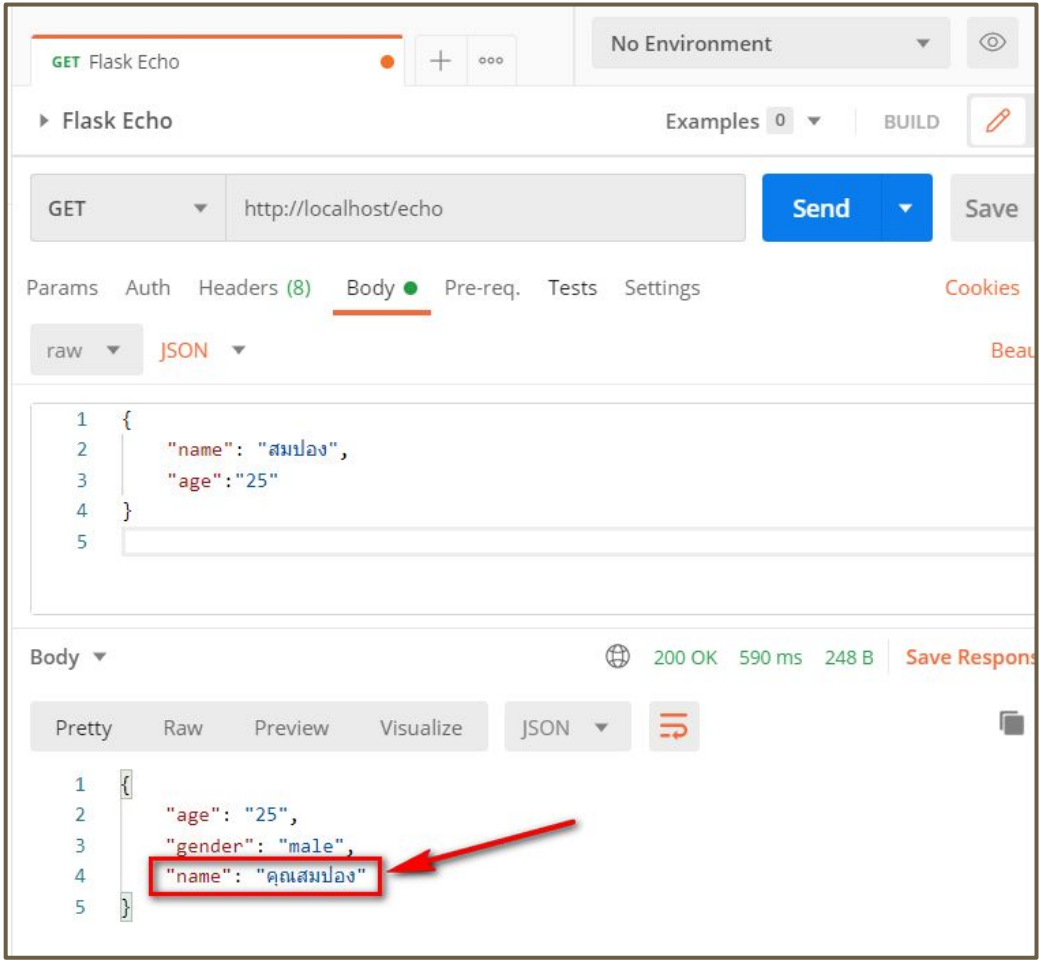
Test with Postman



Modify data : flask_add_data.py

```python
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"] = "คุณ" + 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



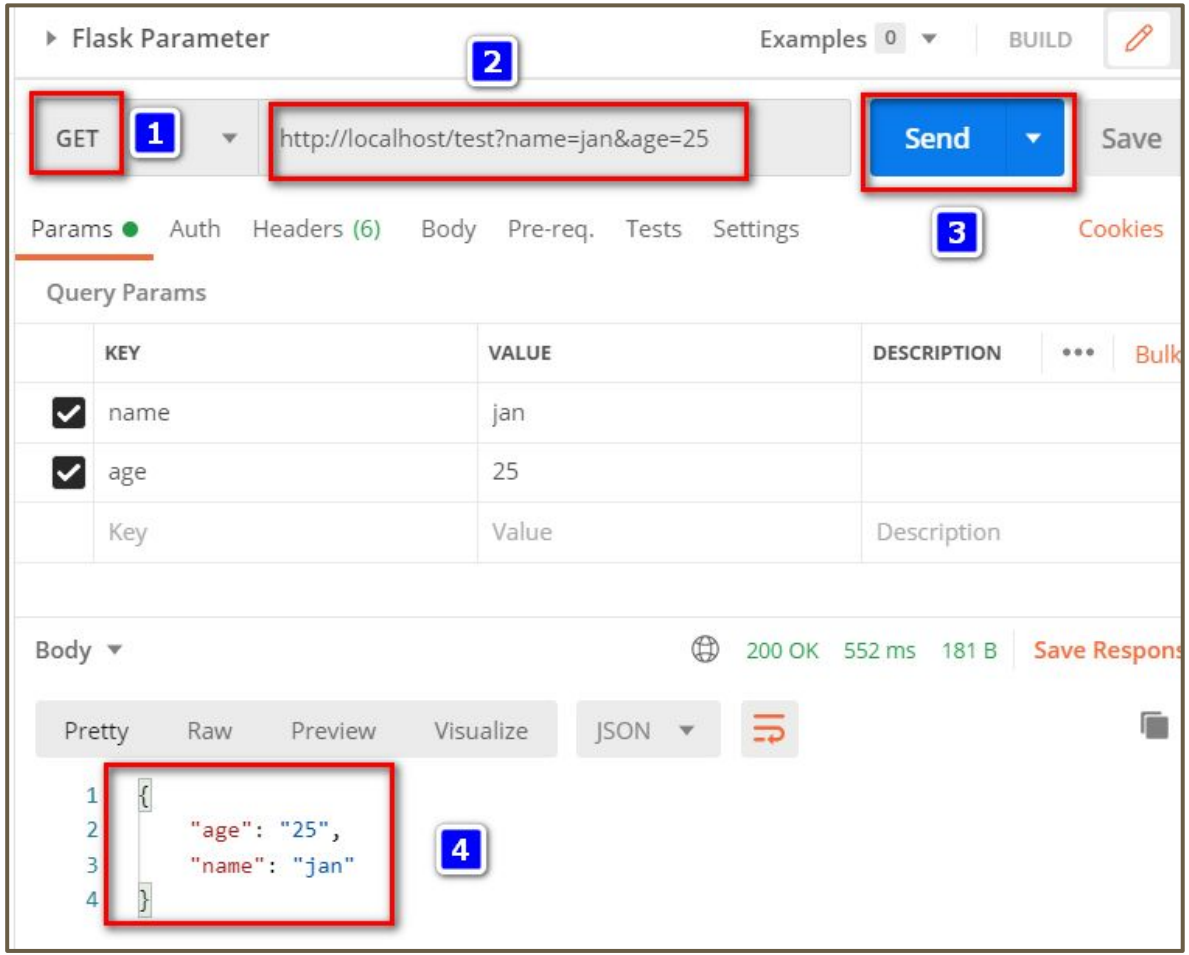
Request with parameter

flask_parameter.py

```python
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

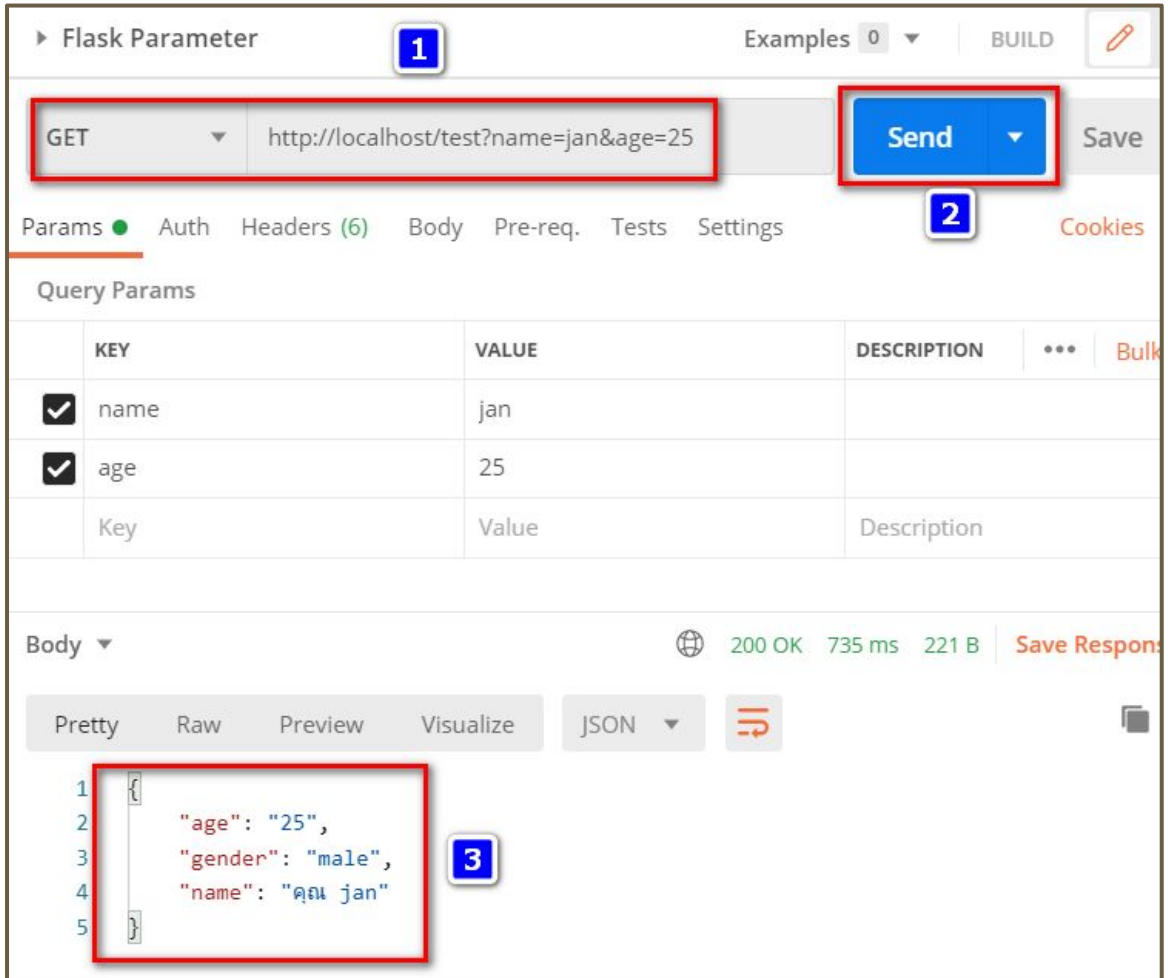


ImmutableMultiDict -> Dict

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

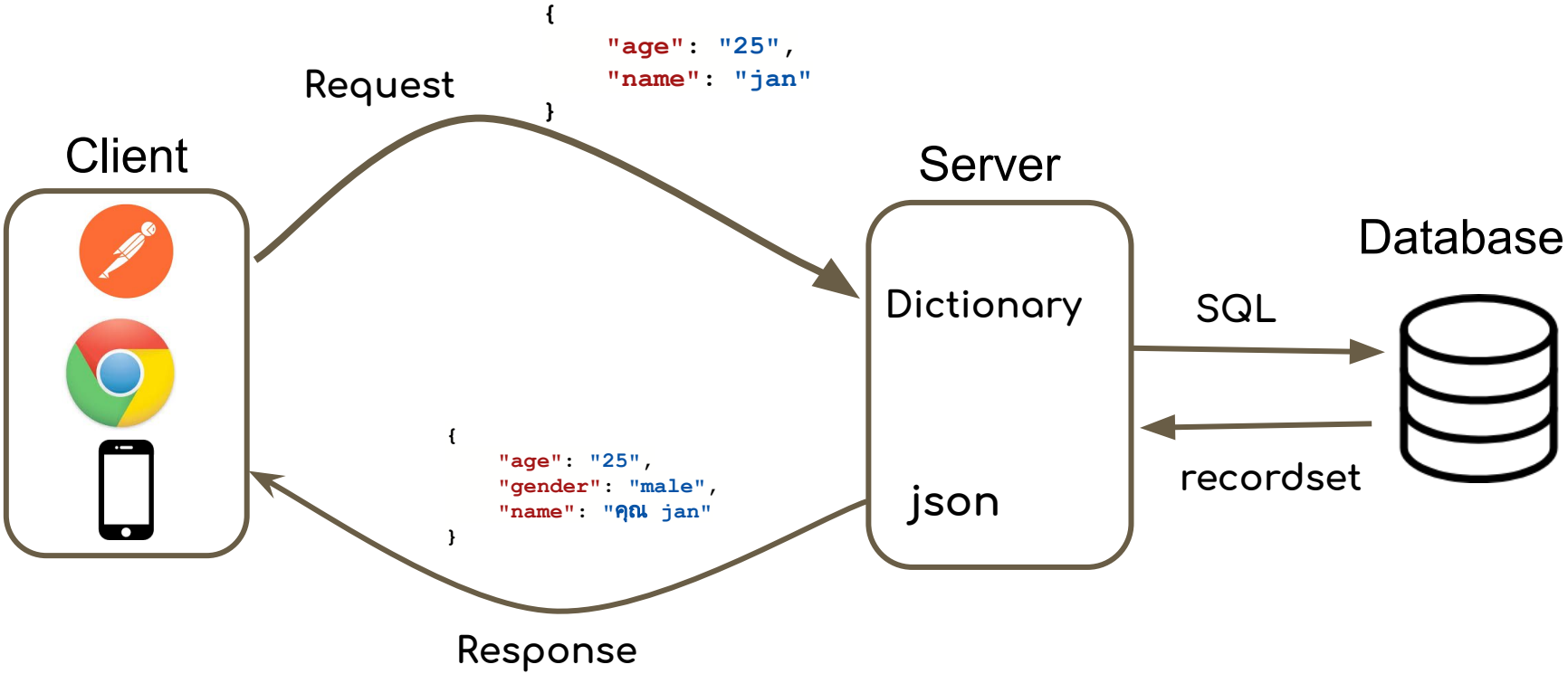
```python
from flask import Flask, request, jsonify
app = Flask(__name__)

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



Your First API (MySQL)

API summary


{
"age": "25",
"name": "jan"
}
Request
Client
Server
Database
Dictionary
SQL
{
"age": "25",
"gender": "male",
"name": "คุณ jan"
}
recordset
json
Response

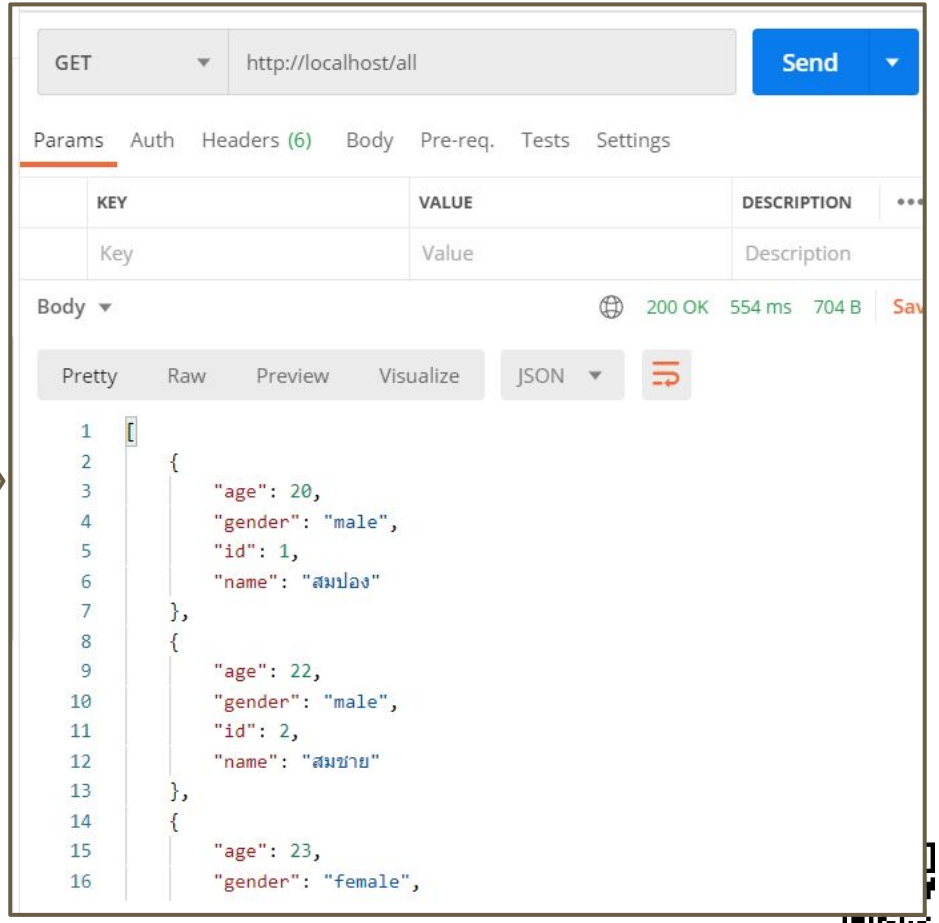
Create MySQL Library : mysql_lib.py

```
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
```

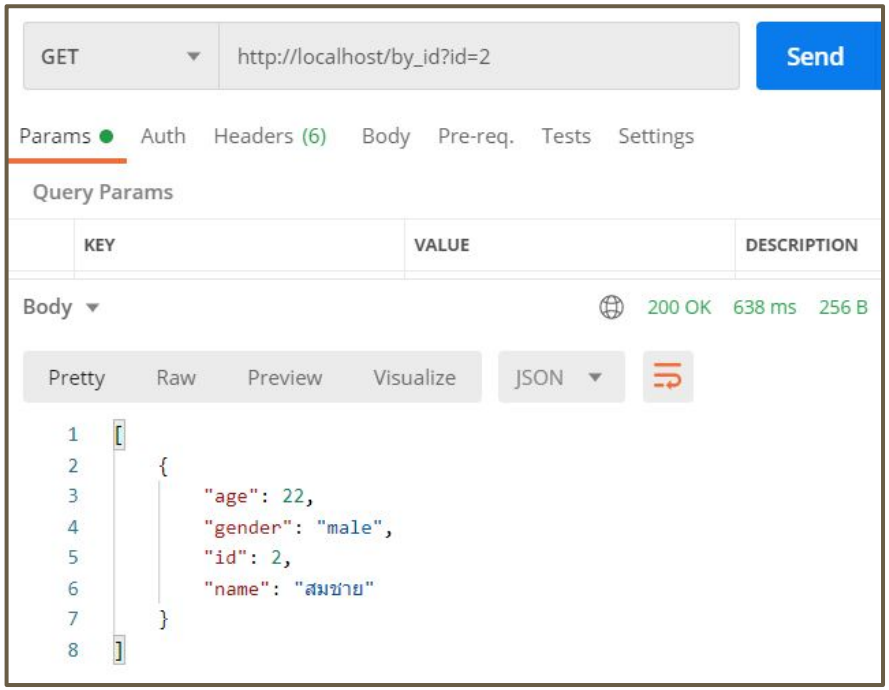
Select All : flask_api_mysql.py

```python
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')
```



Select by ID : flask_api_mysql.py

```python
@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
```



Insert : Append to

flask_api_mysql.py

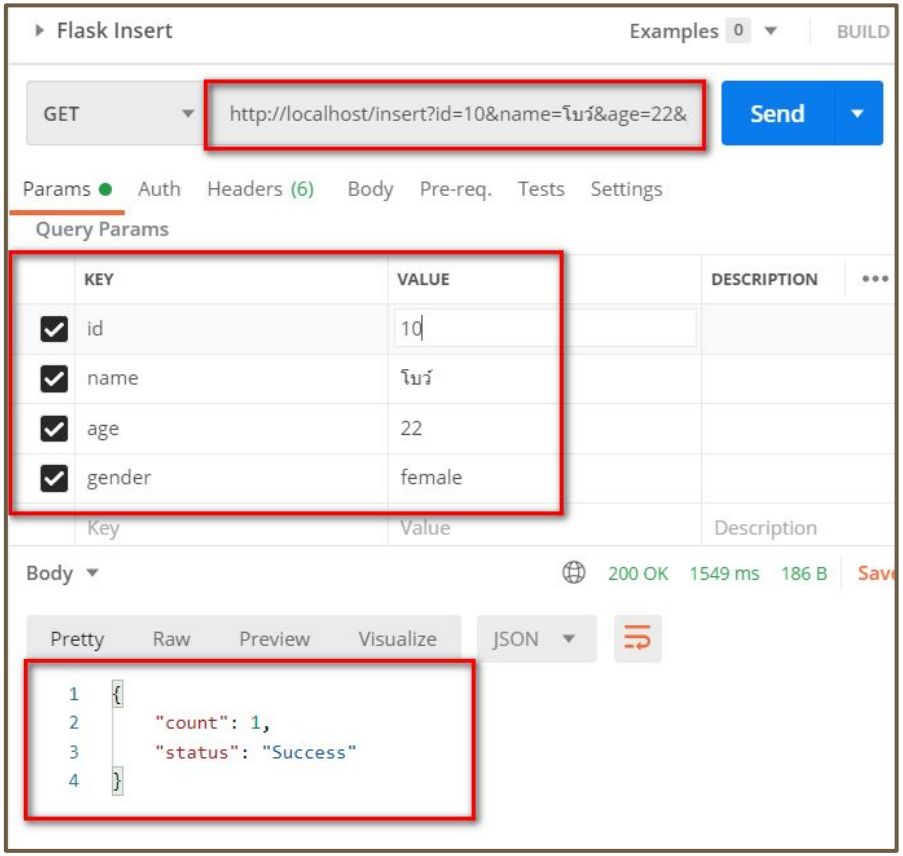
```
@app.route('/insert', methods=['POST','GET'])
def insert():
    data = request.args
    result =
db_insert(data["id"],data["name"],data["age"],da
ta["gender"])
    return jsonify(result)

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

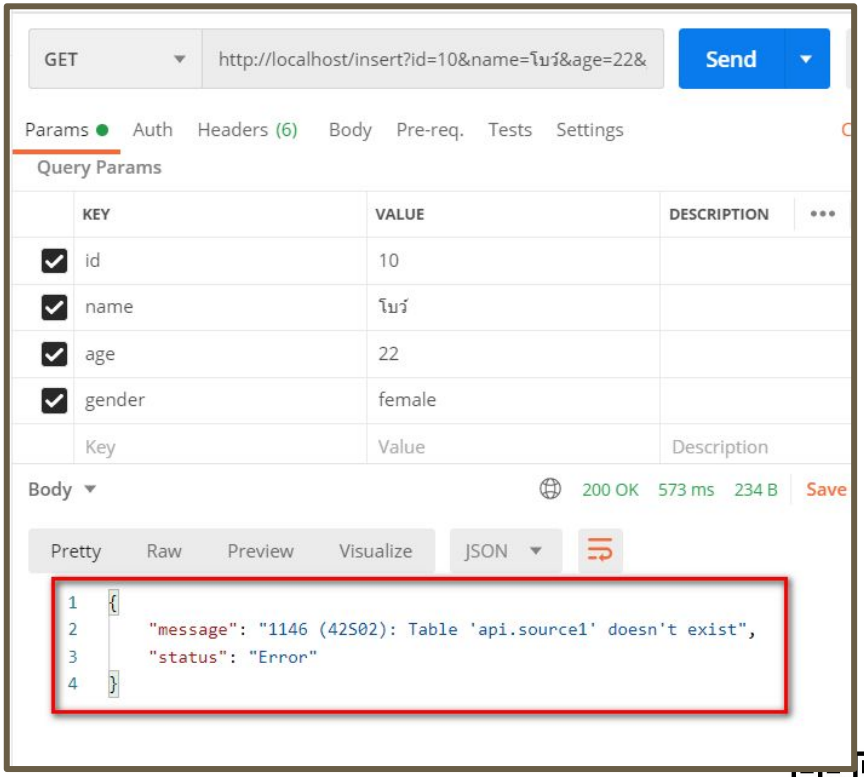
```
def executeSQL(sql):
    global conn, cursor
    data = {}
    try:
        cursor.execute(sql)
        conn.commit()
        count = cursor.rowcount
        data["status"] = "Success"
        data["count"] = count
        return data
    except (Exception, mysql.Error) as error:
        print("Error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

Insert

Success

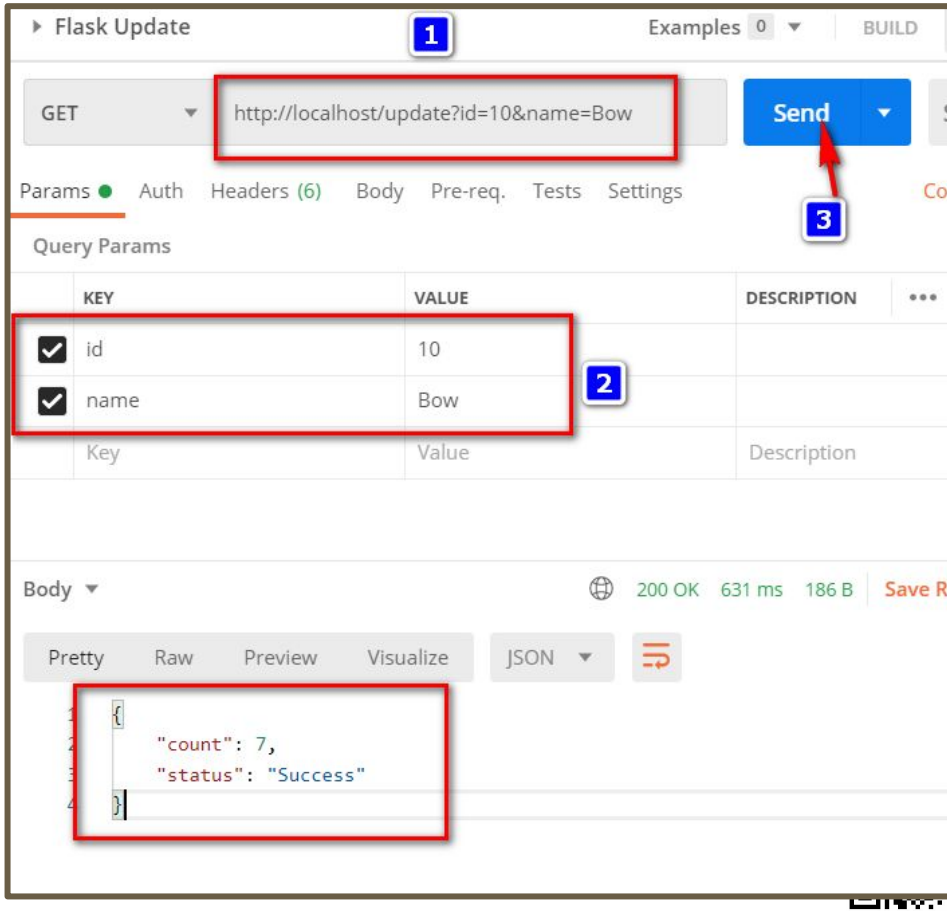


Fail



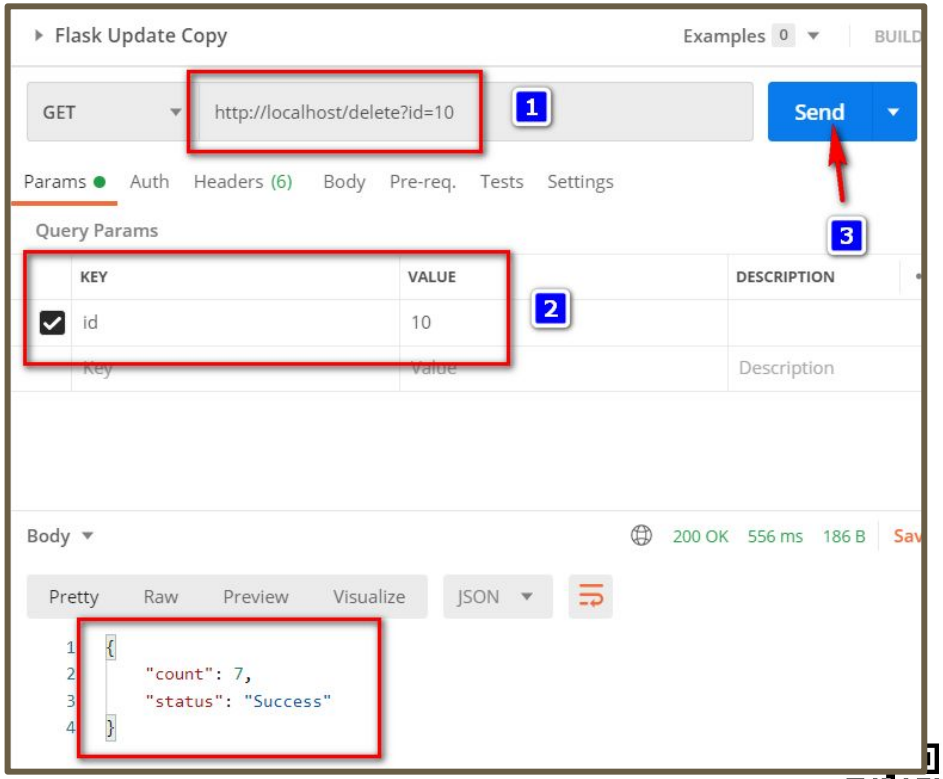
Update : Append to flask_api_mysql.py

```python
# 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
```



Delete : Append to flask_api_mysql.py

```python
# 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
```



Access our own API via Python

flask_api_call.py

Run with new Terminal!

```python
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': 'สมปอง'}, {'age': 22, 'gender': 'male', 'id': 2, 'name': 'สมชาย'}, {'age':
23, 'gender': 'female', 'id': 3, 'name': 'สมศรี'}, {'age': 24, 'gender': 'female', 'id': 4, 'name': 'สมสมร'}, {'age': 26, '
gender': 'male', 'id': 6, 'name': 'สมเจตน์'}]
------ 1 -------
{'age': 22, 'gender': 'male', 'id': 2, 'name': 'สมชาย'}
----- 2 --------
{'age': 23, 'gender': 'female', 'id': 3, 'name': 'สมศรี'}
23
สมศรี
----- Loop --------
{'age': 20, 'gender': 'male', 'id': 1, 'name': 'สมปอง'}
{'age': 22, 'gender': 'male', 'id': 2, 'name': 'สมชาย'}
{'age': 23, 'gender': 'female', 'id': 3, 'name': 'สมศรี'}
{'age': 24, 'gender': 'female', 'id': 4, 'name': 'สมสมร'}
{'age': 26, 'gender': 'male', 'id': 6, 'name': 'สมเจตน์'}
----------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