



# กิจกรรมค่ายสู่ฝันวันนักวิทย์

## MQTT Tic-Tac-Toe

ภาควิชาวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์

มหาวิทยาลัยศรีนครินทรวิโรฒ

30 ตุลาคม 2564



# การกำหนดกลุ่มของคำสั่ง (Indentation)

- Indentation คือการเว้นย่อหน้า ที่ส่วนต้นในแต่ละแถวของคำสั่ง
- ในภาษาไพทอนใช้การเว้นย่อหน้านี้ช่วยในการแบ่งกลุ่มของคำสั่ง (block of code)

```
1  for i in range(1,10):
2      print(i)
3      func1()
4  indent
5  ←→
6
7
8
9
```

กลุ่มคำสั่งเดียวกันจะต้องมี  
การเว้นย่อหน้าเท่ากัน

- หากเว้นย่อหน้าไม่เท่ากันจะเกิด indentation Error
- ในการย่อหน้าต้องใช้อักขระแบบเดียวกัน
  - 4 spaces ไม่เท่ากับ 1 Tab

```
1  L = list("python")
2  print(L[0])
3  print(L[1])
4  print(L[2])
5  print(L[3])
6  print(L[4])
7  print(L[5])
```

File "<ipython-input-4-be89818fb65a>", line 3  
print(L[1])  
^  
**IndentationError:** unexpected indent



# อักษรตัวเล็กและตัวใหญ่ค่าไม่เท่ากันในไพทอน(case sensitive)

- ต้องให้ความระมัดระวังเรื่องการใช้ตัวอักษรเล็กหรือใหญ่
- การเรียกชื่อตัวแปร การเรียกชื่อฟังก์ชัน ที่ไม่เหมือนกันถือว่าเป็นคนละตัวกัน

```
1 var1 = 10
2 print(var1)
3 print(Var1)
4 print(vaR1)
```

var1, Var1, vaR1 ถือว่าเป็นคนละตัวกัน

10

**NameError**

Traceback (most recent call last)

<ipython-input-5-2873241e0afa> in <module>

```
1 var1 = 10
2 print(var1)
----> 3 print(Var1)
4 print(vaR1)
```

**NameError**: name 'Var1' is not defined



# คอมเมนต์ (Comments)

- คอมเมนต์คือส่วนของโค้ดที่ไพทอนจะไม่สนใจ
- คอมเมนต์ถูกใช้ในการอธิบายโค้ดเพื่อเพิ่มความเข้าใจ
- ในบางครั้งเราสามารถใช้เพื่อยกเลิกการทำงานของคำสั่ง
- บรรทัดที่เป็นคอมเมนต์จะขึ้นต้นด้วย **#**

```
1  #This is a comment
2  print("Hello, Python1")
3
4  print("Hello, Python2") #This is a comment
5
6  #print("Hello, Python3")
```

Hello, Python1  
Hello, Python2



# ตัวแปร (Variable)

- ตัวแปรถูกสร้างขึ้นเพื่อเอาไว้เก็บข้อมูล เพื่อการประมวลผลต่างๆ
- ตัวแปรจะถูกสร้างขึ้นทันทีที่มีการกำหนดค่าให้กับมันในครั้งแรก

```
1 x = 5
2 y = 0.5
3 z = "Python"
4 print(type(x))
5 print(type(y))
6 print(type(z))
```

```
<class 'int'>
<class 'float'>
<class 'str'>
```

ตัวแปรจะมีชนิดตามข้อมูลที่มันเก็บ  
x เป็นชนิดเลขจำนวนเต็ม (integer)  
y เป็นชนิดเลขทศนิยม (floating point)  
z เป็นชนิดข้อความ (string)



# โครงสร้างการเขียนโปรแกรม

ส่วน A

```
1 import random
2 import time
3 import json
4 from enum import IntEnum
```

สำหรับอ้างอิงโมดูลที่ต้องการใช้งาน

ส่วน B

```
5
6 department = "Computer Science"
7 university = "SWU"
```

กลุ่มตัวแปรที่ประกาศไว้ใช้งาน ตัวแปรในส่วนนี้สามารถเข้าถึงได้จากทุกที่ในโปรแกรม (global variable)

ส่วน C

```
8
9 def show10student():
10     print("---print from function---")
11     for i in range(0,10):
12         std_id = "id6500"+str(random.randint(0, 1000))
13         student = std_id+':'+department+':'+university
14         print(student)
```

ประกาศฟังก์ชันต่างๆไว้ใช้งาน

ส่วน D

```
15
16 print("Start program")
17 show10student()
18 print("End program")
19
```

จุดเริ่มต้นกลุ่มคำสั่งหลักของโปรแกรม

ผลลัพธ์

Start program

---print from function---

id6500114:Computer Science:SWU

id6500117:Computer Science:SWU

id6500890:Computer Science:SWU

id6500261:Computer Science:SWU

id6500440:Computer Science:SWU

id6500336:Computer Science:SWU

id6500118:Computer Science:SWU

id6500796:Computer Science:SWU

id6500741:Computer Science:SWU

id6500276:Computer Science:SWU

End program



# การตรวจสอบเงื่อนไข

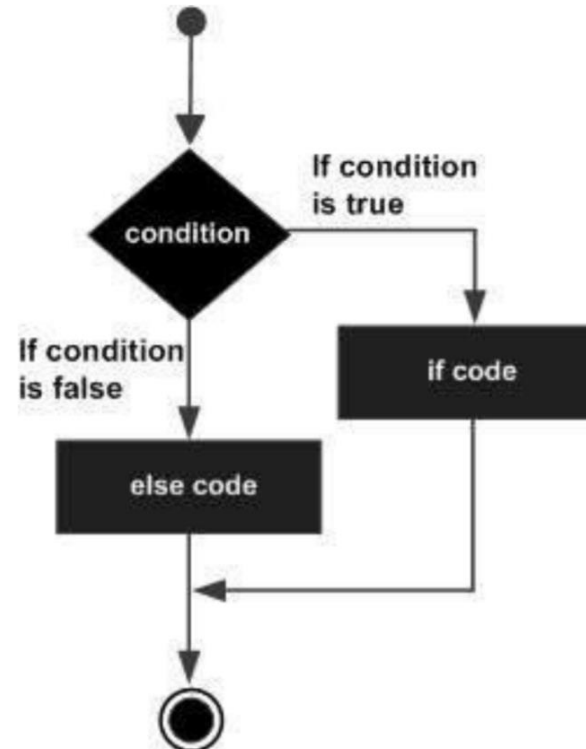
- คำสั่ง if (if statement) : หากเงื่อนไขเป็นจริงจะทำงานตามชุดคำสั่งในส่วนของ if หากไม่จริงจะทำชุดคำสั่งในส่วนของ else

If เงื่อนไข :

{ \_\_\_\_\_ ชุดคำสั่ง  
\_\_\_\_\_ ทำเมื่อตรงกับเงื่อนไข  
\_\_\_\_\_

else :

{ \_\_\_\_\_ ชุดคำสั่ง  
\_\_\_\_\_ ทำกรณีที่ไม่ตรงกับเงื่อนไข  
\_\_\_\_\_



ตารางสรุปการใช้เครื่องหมายโอเปอเรเตอร์เปรียบเทียบ

โอเปอเรเตอร์	ความหมาย	ตัวอย่าง
=	เท่ากับ	A == 10
>	มากกว่า	A > 3
>=	มากกว่าหรือเท่ากับ	A >= 3
<	น้อยกว่า	A < 2
<=	น้อยกว่าหรือเท่ากับ	A <= 2
!=	ไม่เท่ากับ	A != 10
not	ตรวจสอบค่าตรงกันข้ามระหว่าง True กับ False	Not A

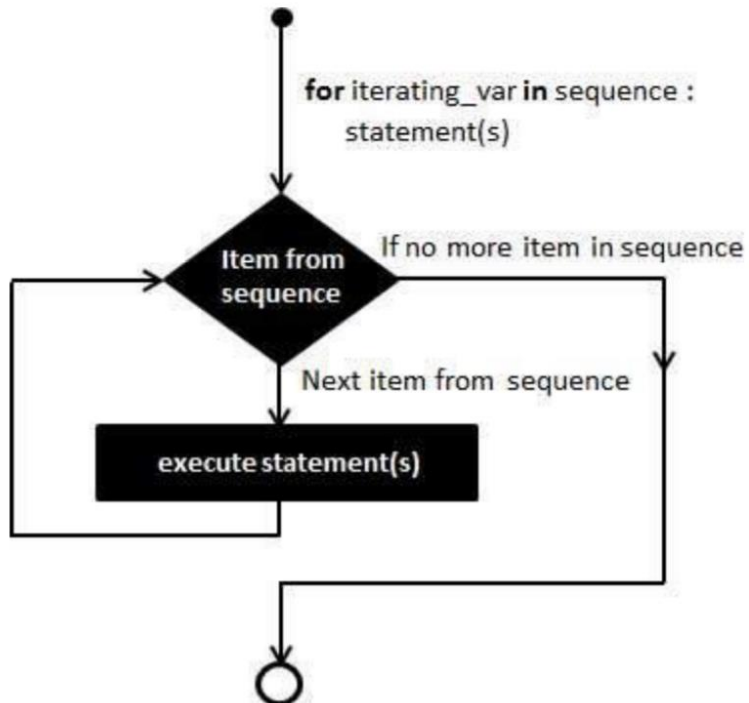


# คำสั่งการวนซ้ำ (Repetition, Loop)

- กำหนดให้ทำกลุ่มคำสั่งย่อยตามจำนวนรอบที่กำหนด หรือในกรณีที่เงื่อนไขเป็นจริง
- กลุ่มคำสั่งการวนซ้ำประกอบด้วยคำสั่ง **for** และคำสั่ง **while**

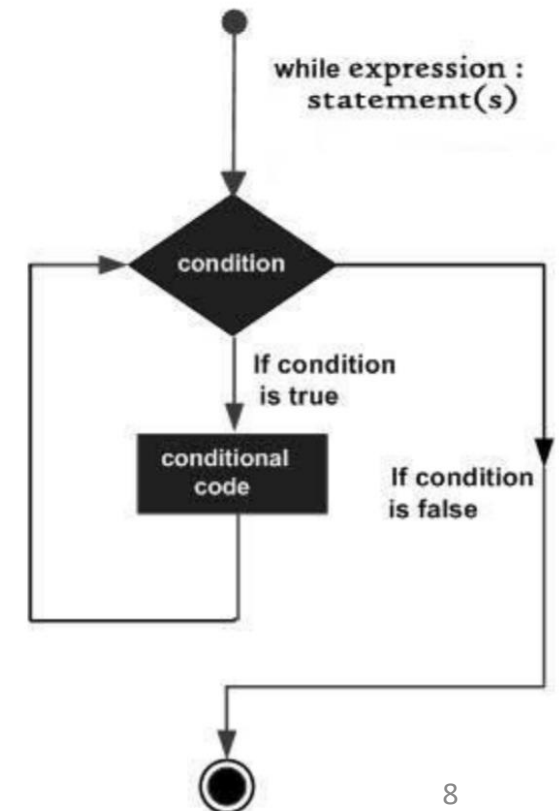
1	<b>for</b> i <b>in</b> [1,2,3,4]:
2	print(i)

1  
2  
3  
4



1	i = 1
2	<b>while</b> i < 5:
3	print(i)
4	i=i+1

1  
2  
3  
4







# คำสั่งวนทำซ้ำแบบไม่มีวันสิ้นสุด และการหยุดการวนทำซ้ำ

- การวนซ้ำแบบไม่มีวันสิ้นสุด (infinite loop)
- สามารถหยุดการวนทำซ้ำได้ด้วยคำสั่ง **break**

```
1 while True:
2     d = input()
3     if d == 'exit':
4         print("Bye Bye")
5         break
```

hello  
exit  
Bye Bye

```
1 while 1:
2     d = input()
3     if d == 'exit':
4         print("Bye Bye")
5         break
```

hello  
exit  
Bye Bye



# ฟังก์ชัน

- อาจเรียกว่า subprogram หรือ subroutine
- เป็นการแยกส่วนสิ่งที่ซ้ำๆ กัน หรือเข้าใจยาก ออกมาจากโปรแกรมหลัก
- ฟังก์ชันควรมีหน้าที่การทำงานชัดเจน เช่น ฟังก์ชันหาค่าเฉลี่ย

## องค์ประกอบของฟังก์ชัน

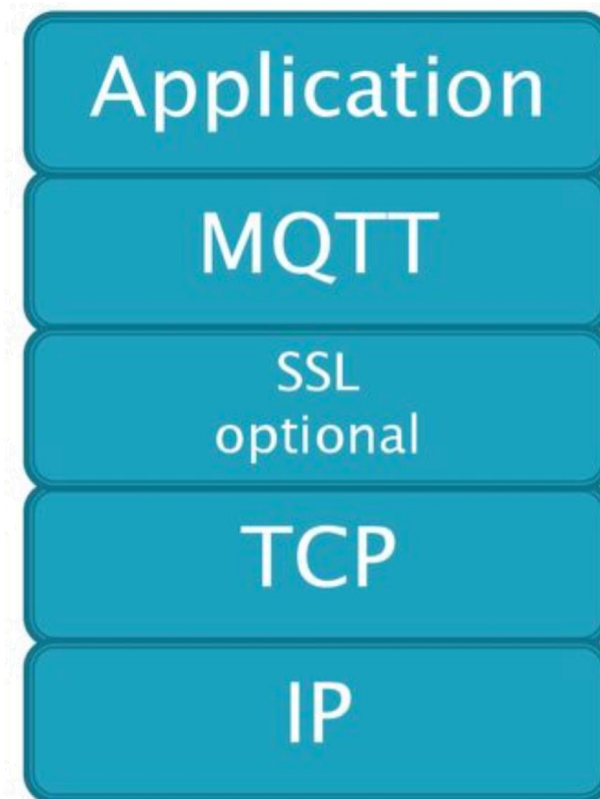
- เริ่มต้นด้วย def เสมอ
- ชื่อฟังก์ชัน มีข้อกำหนดเหมือนการตั้งชื่อตัวแปร
- ค่าที่รับเข้ามา หรือ "พารามิเตอร์" (ไม่จำเป็นต้องมีก็ได้)
- การคืนค่าจากฟังก์ชันด้วยคำสั่ง return (ไม่จำเป็นต้องมีก็ได้)

```
1  def average4(w,x,y,z):  
2      s = w+x+y+z  
3      return s/4
```



# MQTT Protocol

- MQTT เป็น Protocol ที่นิยมใช้ในงาน IoT
- ทำงานอยู่บน TCP/IP เหมือนกับ HTTP
- เนื่องจาก data packet ที่รับส่งมีขนาดเล็ก
- ลดการใช้งาน bandwidth – Small header
- ประหยัดพลังงานของอุปกรณ์ IoT โดยเฉพาะอุปกรณ์ที่ใช้แบตเตอรี่เป็นแหล่งพลังงาน
- เป็นการสื่อสารแบบ Asynchronous



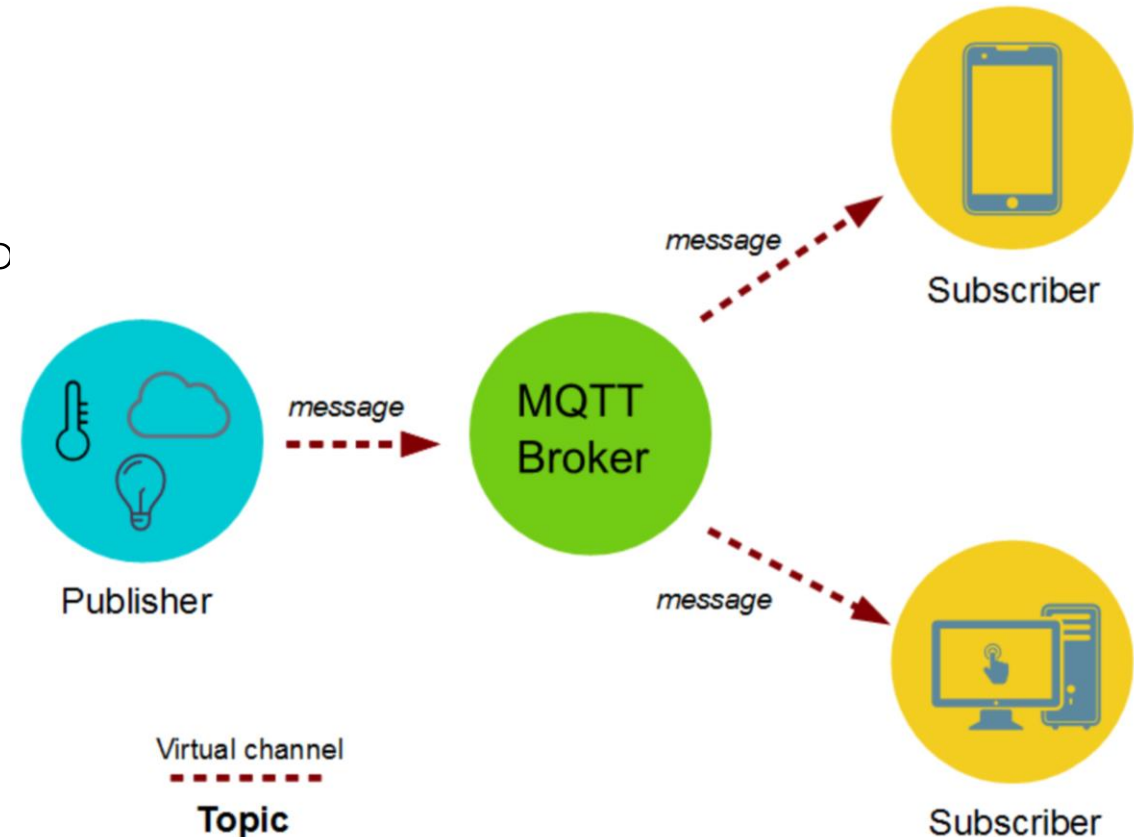
TCP/IP Port: 1883

When running over SSL, TCP/IP port 8883



# MQTT: Components

- **Broker**, which is the server that handles the data transmission between the clients.
- **A topic**, which is the place a device want to put or retrieve a message to/from.
- **The message**, which is the data that a device receives “when subscribing” from a topic or send “when publishing” to a topic.
- **Publish**, is the process a device does to send its message to the broker.
- **Subscribe**, where a device does to retrieve a message from the broker.



# Public MQTT Broker

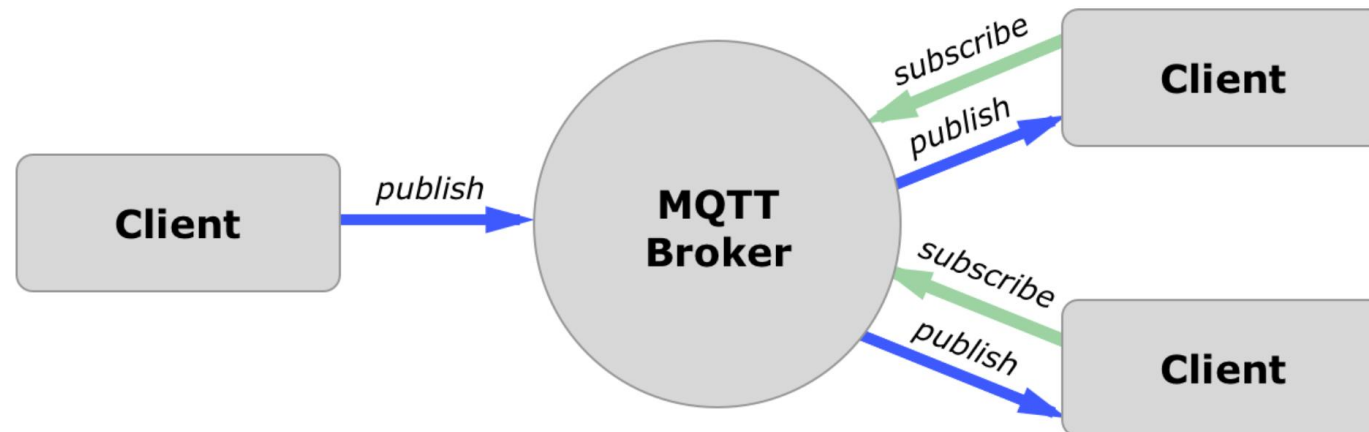
- Free service

Address	Port
lot.eclipse.org	1883(MQTT), 80(WebSockets), 443(WebSockets + SSL)
test.mosquitto.org	1883(MQTT), 8883(MQTT+SSL), 8884(MQTT+SSL), 80(WebSockets)
www.cloudmqtt.com	18443, 28443 (SSL)
mqtt.simpleml.com	1883(MQTT), 8883(MQTT+SSL), 80(REST), 80(WebSockets), 5683(CoAP)
broker.hivemq.com	1883(MQTT)
mqtt.swifitch.cz	1883(MQTT)



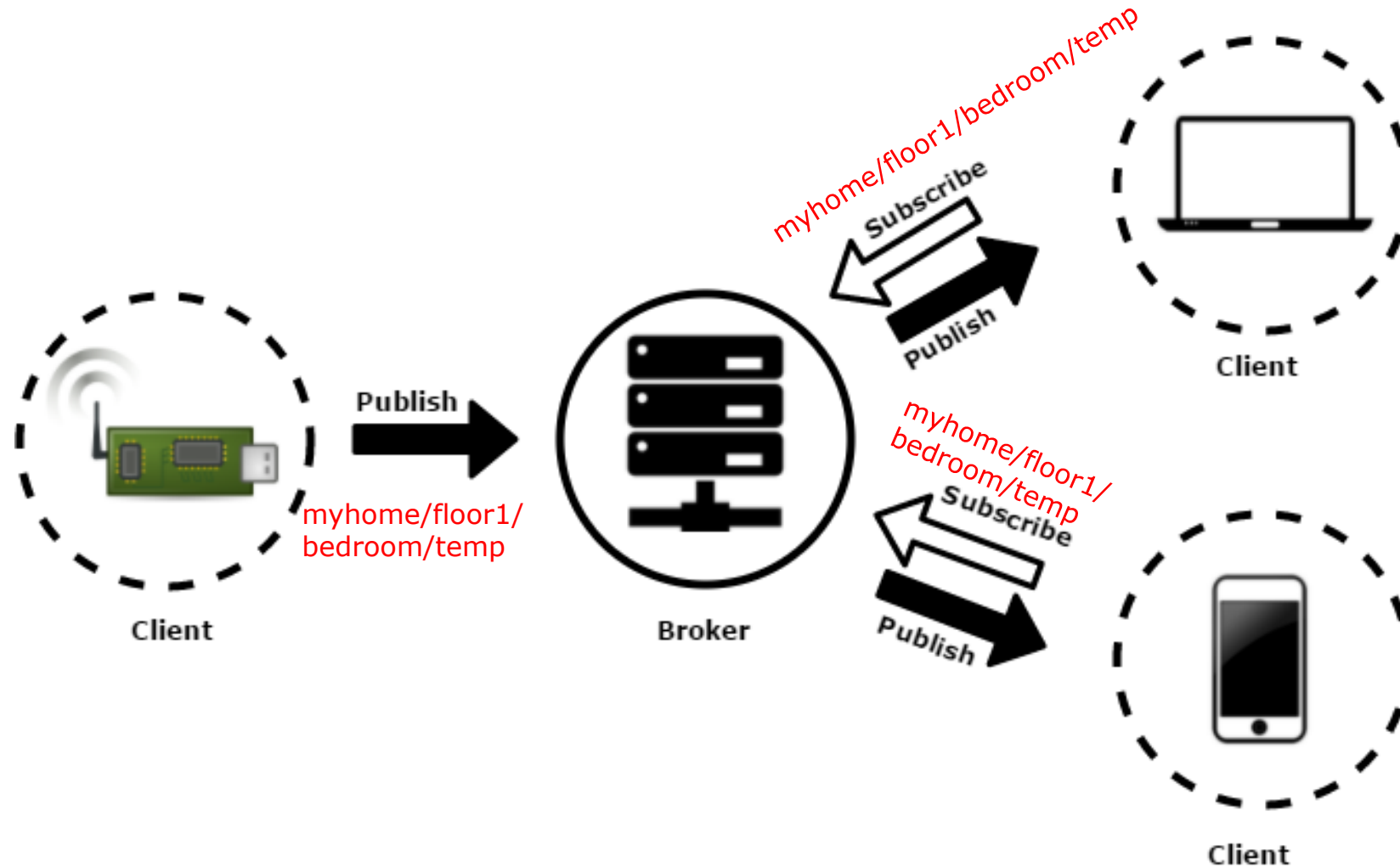
# How MQTT works

- MQTT is based on clients and a server.
- MQTT server is called a broker and the clients are simply the connected devices.
  - When a device (a client) wants to send data to the broker, we call this operation a “publish”.
  - When a device (a client) wants to receive data from the broker, we call this operation a “subscribe”.
- Clients are publishing and subscribing to topics. So, the broker here is the one that handles the publishing/subscribing actions to the target topics.





# MQTT Routing – Topic based





# Topics

- MQTT Topics are structured in a hierarchy similar to folders and files in a file system using the forward slash ( / ) as a delimiter.
- Allow to create a user friendly and self descriptive naming structures

- Topic names are:

- Case sensitive
- use UTF-8 strings.
- Must consist of at least one character to be valid.

Topic level separator

swumqtt/group1/player1

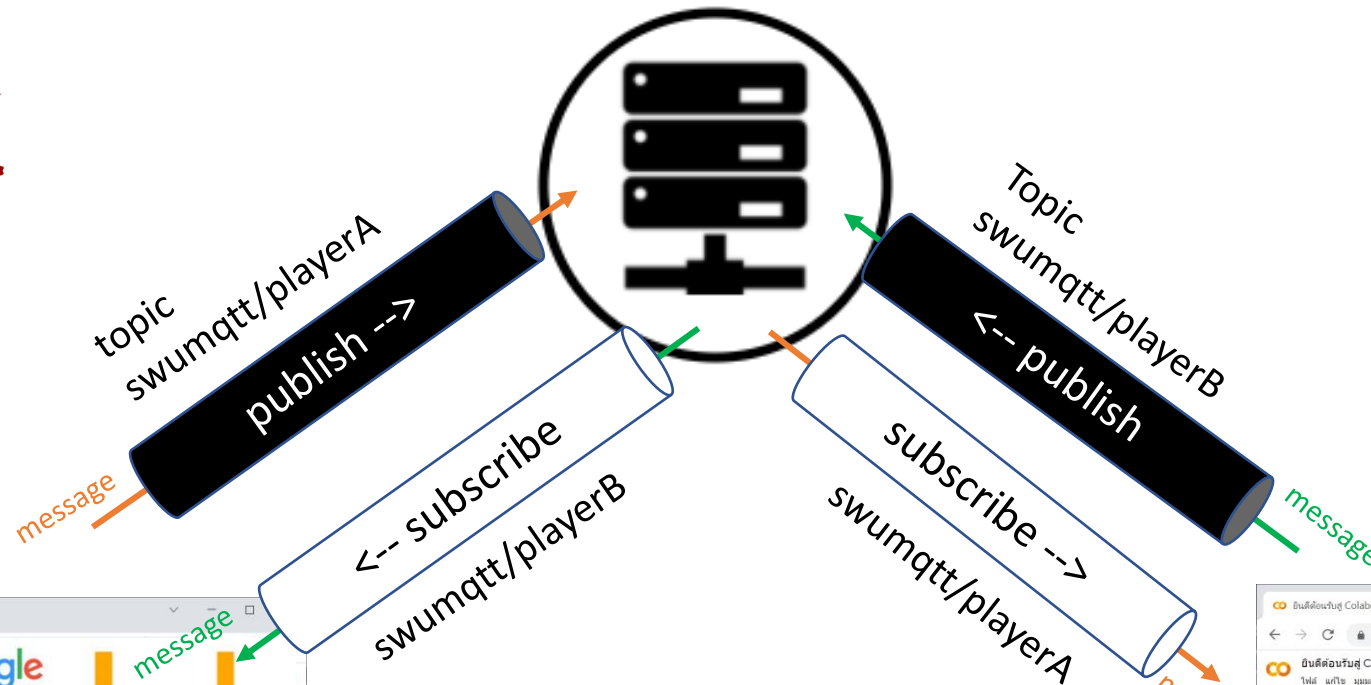
Topic level Topic level



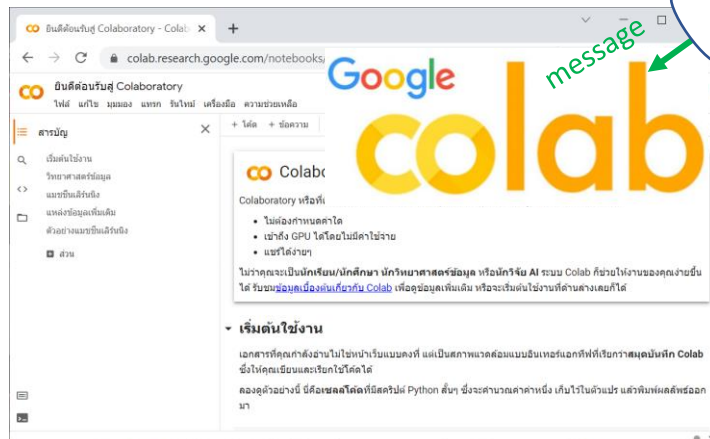


# MQTT Tic-Tac-Toe

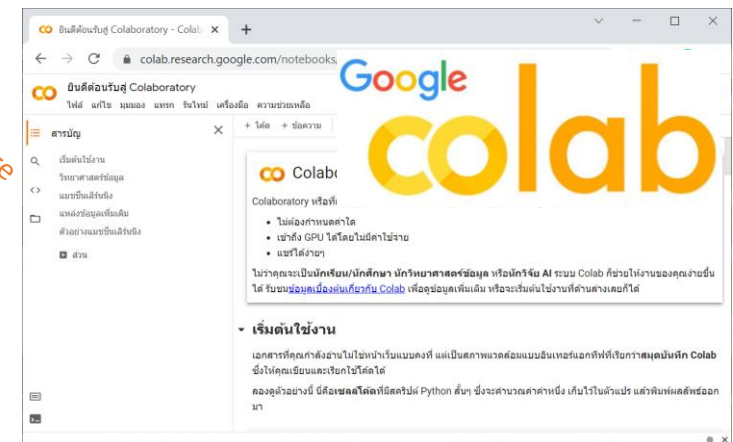
Broker: [mqtt.eclipseprojects.io](https://mqtt.eclipseprojects.io) port: 1883



Tic-Tac-Toe player A



Tic-Tac-Toe player B





# MQTT-Message

- ข้อมูลใน mqtt message มักอยู่ในรูปแบบ JSON
- JSON (JavaScript Object Notation) เป็นรูปแบบที่ใช้ในการแลกเปลี่ยนรับส่งข้อมูล
- เข้าใจง่าย, ขนาดเล็ก, สิ้นเปลือง Network Bandwidth น้อย
- ง่ายต่อเครื่องจักร (Machines, Computer) ในการวิเคราะห์(Parse) หรือสร้าง (Generate)
- โครงสร้างอยู่ในรูปแบบ **Key: Value**
- รูปแบบในการเขียน JSON
  - JSON Object: คล้ายกับข้อมูล 1 เรคอร์ด สามารถมีฟิลด์เดียว หรือหลายฟิลด์ก็ได้
  - JSON Array: เป็นข้อมูลหลายเรคอร์ด โครงสร้างคล้าย Array 2 มิติ



# JSON Syntax

- **JSON objects** start the object with “{” and end in with “}”
  - Members (properties), use pairs of “key : value”

ข้อมูล Key:Value อยู่ภายใน “{ Key:Value , Key:Value, ... ,Key:Value}”

Ex.

```
{ "title" : "เครื่องทำน้ำอุ่น" }  
  
{ "id" : 0,  
  "title" : "เครื่องทำน้ำอุ่น",  
  "Status" : "On" }
```

- **JSON arrays** put the arrays between “[ ]”
  - Elements put the values directly separated by commas

ข้อมูลเก็บอยู่ใน “[ value, value, ... ,value]”

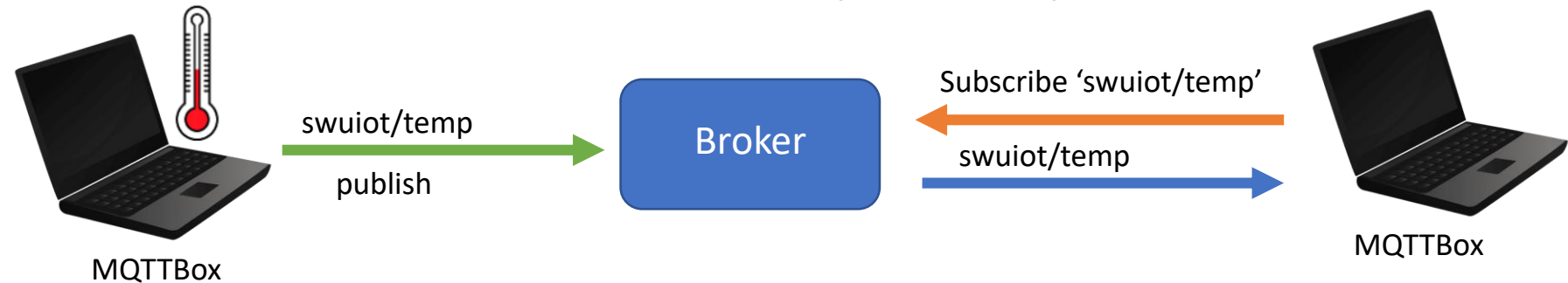
Ex.

```
[ { "id" : 0, "title" : "เครื่องทำน้ำอุ่น", "Status" : "On" },  
  { "id" : 1, "title" : "เครื่องกรองอากาศ", "Status" : "Off" },  
  { "id" : 2, "title" : "ไฟส่องสว่างชั้น1", "Status" : "On" } ]
```



# Workshop 1: ตั้งค่า MQTT Client ด้วย MQTTBox

- ติดตั้งโปรแกรม MQTTBox จำลองการทำงานเป็น Client ตัวส่ง (Publisher) และ Client ตัวรับ (Subscriber)



## Steps:

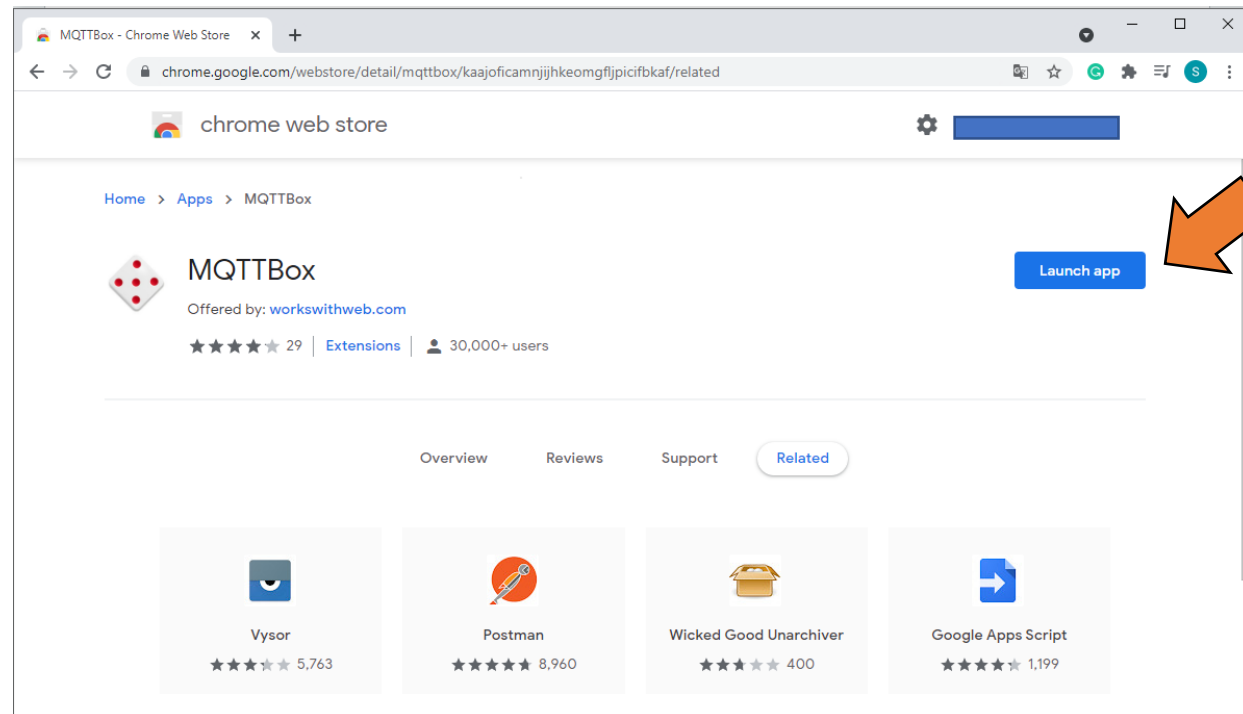
- ติดตั้ง MQTTBox Extension ให้กับ Google chrome  
<https://chrome.google.com/webstore/detail/mqttbox/kaajoficamnjijhkeomgfljpicifbkaf>
- Create MQTT Client และตั้งค่าเชื่อมต่อ Public Broker ในครั้งนี้ใช้ **test.mosquitto.org:1883**
- ตรวจสอบสถานะการเชื่อมต่อ



# Step1: Download MQTTBox

- **Chrome Extension:**

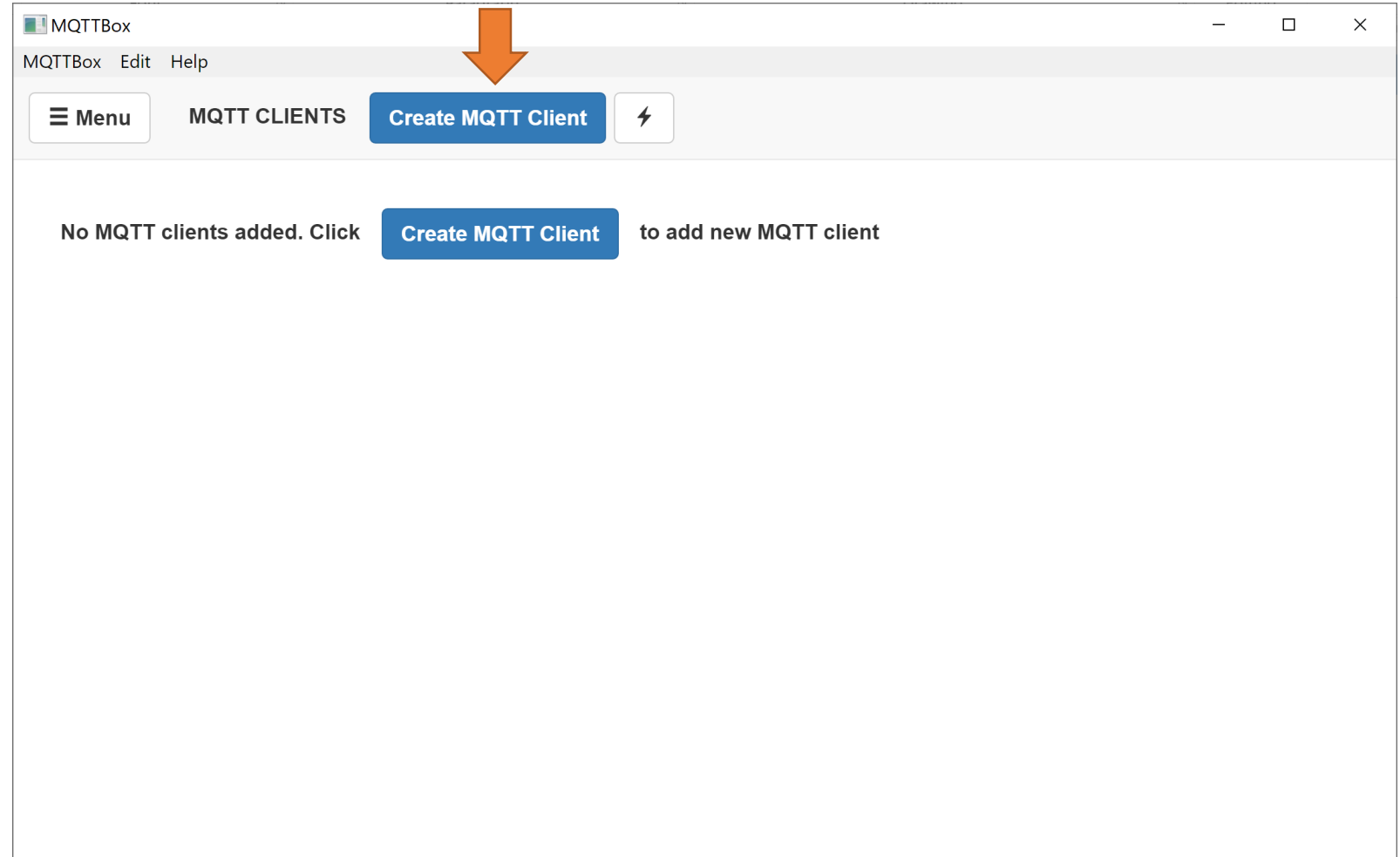
- <https://chrome.google.com/webstore/detail/mqttbox/kaajoficamnjjhkeomgfjpicifbkaf>





## Step2: ติดตั้งและเปิดโปรแกรม

1. หลังจากติดตั้งเสร็จเปิดโปรแกรม
2. คลิกปุ่ม Create MQTT Client เพื่อเข้าสู่หน้าจอตั้งค่า MQTT Client





## Step3: สร้าง MQTT Client

1. MQTT Client Name ตั้งชื่อตามต้องการ เช่น swumqbox
2. Protocol เลือก mqtt/tcp
3. Host คือ MQTT Broker ในครั้งนี้ใช้ test.mosquitto.org (อาจทดลองใช้ broker อื่นก็ได้เช่น iot.eclipse.org) หากไม่กำหนด port จะใช้เป็น default คือ 1883 ตัวอย่างการกำหนดค่า **test.mosquitto.org:1883**
4. กดปุ่ม Save เพื่อบันทึก client

### Note:

- ไม่จำเป็นต้องกำหนด username, password เนื่องจาก test.mosquitto.org ไม่ต้องการ
- MQTT Client id เป็นหมายเลข ID ของ client ซึ่งจะเป็นค่า Unique



## Step3: สร้าง MQTT Client

MQTTBox Edit Help

Menu MQTT CLIENT SETTINGS Client Settings Help

<b>MQTT Client Name</b>	<b>MQTT Client Id</b>	<b>Append timestamp to MQTT client id?</b>	<b>Broker is MQTT v3.1.1 compliant?</b>
swumqbox	a27423d2-d8e1-4aa8-ac2e	<input checked="" type="checkbox"/> Yes	<input checked="" type="checkbox"/> Yes
<b>Protocol</b>	<b>Host</b>	<b>Clean Session?</b>	<b>Auto connect on app launch?</b>
mqtt / tcp	test.mosquitto.org:1883	<input checked="" type="checkbox"/> Yes	<input checked="" type="checkbox"/> Yes
<b>Username</b>	<b>Password</b>	<b>Reschedule Pings?</b>	<b>Queue outgoing QoS zero messages?</b>
Username	Password	<input checked="" type="checkbox"/> Yes	<input checked="" type="checkbox"/> Yes
<b>Reconnect Period (milliseconds)</b>	<b>Connect Timeout (milliseconds)</b>	<b>KeepAlive (seconds)</b>	
1000	30000	10	
<b>Will - Topic</b>	<b>Will - QoS</b>	<b>Will - Retain</b>	<b>Will - Payload</b>
Will - Topic	0 - Almost Once	<input type="checkbox"/> No	
<b>Save</b>		<b>Delete</b>	





## Step4: ตรวจสอบสถานะ MQTT Client

- สถานะต้องเป็น Connected
- หากไม่ใช่ ทำการกดที่รูปเฟืองเพื่อทำการแก้ไขค่า
- ให้ตรวจสอบ URI และ port หากถูกต้องแล้ว
- หากกำหนดค่าถูกต้องแล้วแต่ยังไม่สามารถเชื่อมต่อได้ ให้ทดลองเปลี่ยนเป็น Broker อื่น ที่แสดงในตาราง Public MQTT Broker เช่น  
broker.hivemq.com,  
mqtt.simpleml.com, iot.eclipse.org



## Workshop 2: ทดลอง Publish และ Subscribe

- หลังจากเชื่อมต่อ Broker ได้แล้วจะพบ 2 หน้าต่างคือ Publisher สำหรับจำลองการทำงานเป็นผู้ส่ง และ Subscriber สำหรับการจำลองการทำงานเป็นผู้รับ

### Steps:

- ตั้งค่า Subscriber
- ตั้งค่า Publisher
- ทดสอบ publish ข้อมูล
- ทดสอบเพิ่ม Subscriber
- ทดสอบเปลี่ยนค่า topic ของ Publisher

The screenshot shows the MQTTBox web interface. At the top, there's a status bar with 'MQTTBox', 'Edit', and 'Help' menus. Below that, a 'Connected' status indicator is shown. The main interface is divided into two panels: 'Topic to publish' (left) and 'Topic to subscribe' (right). The 'Topic to publish' panel includes fields for 'Topic to publish', 'QoS' (set to '0 - Almost Once'), 'Retain' (unchecked), 'Payload Type' (set to 'Strings / JSON / XML / Characters'), and a 'Payload' text area. A 'Publish' button is at the bottom. The 'Topic to subscribe' panel includes a 'Topic to subscribe' field, 'QoS' (set to '0 - Almost Once'), and a 'Subscribe' button. The URL bar shows 'mqtt://test.mosquitto.org:1883'.



## Step1: ตั้งค่า Subscriber

1. กำหนดชื่อ Topic ที่ต้องการใช้งาน เช่น temp หรือ myhome/sensor/temp
2. คลิกปุ่ม Subscribe เพื่อรอรับข้อมูลใน Topic ที่ได้ลงทะเบียนไป
3. หน้าจอ Subscribe แสดง Topic ที่ลงทะเบียนไว้

×

Topic to subscribe

temp

QoS

0 - Almost Once

Subscribe

ลงทะเบียน topic: temp เรียบร้อยแล้ว



×

temp



## Step2,3: ตั้งค่า Publisher และ Publish

1. กำหนดชื่อ Topic ให้ตรงกับใน Step1
2. ตรวจสอบชนิดข้อมูลใน Payload Type ที่เลือกได้ และกำหนดค่า Payload Type ให้เป็น Strings/JSON/XML/Characters
3. ใส่ข้อมูลในช่อง Payload ด้วยข้อความที่ต้องการ เช่น
  - String: “ทดสอบTEST1234”
  - JSON: {“TEST”: “ทดสอบ”}
4. กดปุ่ม Publish และสังเกตผลลัพธ์ที่หน้าต่าง Subscriber

The screenshot shows the MQTT Publisher configuration window. The 'Topic to publish' field is set to 'temp'. The 'QoS' is set to '0 - Almost Once'. The 'Retain' checkbox is unchecked. The 'Payload Type' is set to 'Strings / JSON / XML / Characters'. The 'Payload' field contains the JSON string: `{"test": "ทดสอบ"}`. A blue 'Publish' button is visible. Below the main configuration, a preview shows the payload and metadata: `topic:temp, qos:0, retain:false`. An inset window titled 'Result shown in Subscriber window' displays the received message with the same JSON payload and a long hexadecimal 'Raw payload' string.

Topic to publish

temp

QoS

0 - Almost Once

Retain ☐

Payload Type

Strings / JSON / XML / Characters

e.g: {'hello':'world'}

Payload

`{"test": "ทดสอบ"}`

Publish

`topic:temp, qos:0, retain:false`

Result shown in Subscriber window

`temp`

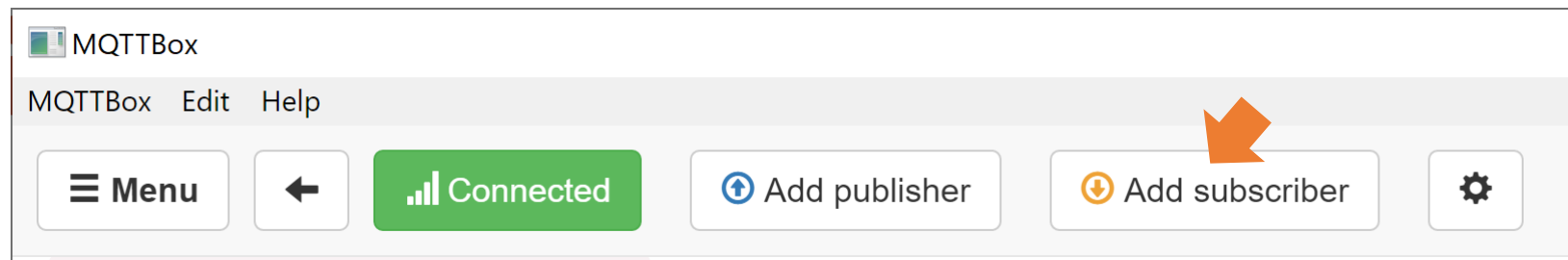
`{"test": "ทดสอบ"}`

`qos : 0, retain : false, cmd : publish, dup : false, topic : temp, mes  
sageld : , length : 32, Raw payload : 123341161011151163458342241  
8415122418414822418417022418417322418415434125`



## Step 4: ทดสอบเพิ่ม Subscriber และแก้ไขข้อมูล

1. เพิ่มตัวรับโดยกด Add subscriber แล้วระบุ Topic ให้ตรงกับใน Step 1
2. ที่หน้าต่างตัวส่ง Publisher พิมพ์ข้อความชุดใหม่ เช่น Hello 25
  - ให้นิสิตใช้ข้อมูลเป็น Hello “name” เช่น Hello Jim
3. หน้าต่างตัวรับ Subscriber ทั้งสองหน้าต่างจะได้รับข้อมูลเหมือนกัน





## Step 4: ทดสอบเพิ่ม Subscriber และแก้ไขข้อมูล

The screenshot displays the MQTTBox application interface. On the left, the 'Topic to publish' field is set to 'temp'. The 'QoS' is set to '0 - Almost Once'. The 'Retain' checkbox is unchecked. The 'Payload Type' is set to 'Strings / JSON / XML / Characters'. The 'Payload' field contains 'Hello 25'. A yellow box highlights the 'Hello 25' payload with the text 'Hello "name" เช่น Hello Jim'. The 'Publish' button is visible. Below the 'Publish' button, the published message is shown: 'Hello 25', 'topic:temp', 'qos:0', 'retain:false'.

On the right, two subscriber panels are shown, both with the topic 'temp'. The first panel shows the received message: 'Hello 25', 'qos : 0, retain : false, cmd : publish, dup : false, topic : temp, messageid : , length : 14, Raw payload : 72101108108111325053'. The second panel shows the received message: 'Hello 25', 'qos : 0, retain : false, cmd : publish, dup : false, topic : temp, messageid : , length : 14, Raw payload : 72101108108111325053'.



## Step 5: ทดสอบเปลี่ยนค่า topic ของ Publisher

1. ที่หน้าต่าง Publisher เปลี่ยนชื่อ Topic เป็น myhome/floor1/temp ให้นิสิตรเปลี่ยน myhome เป็น my+name เช่น myJim/floor1/temp
2. ทดสอบ Publish และสังเกตผลที่หน้าต่าง Subscriber ทั้งสองหน้าต่าง

**Question 1:** เพราะเหตุใดจึงไม่ปรากฏข้อมูลใดๆที่ Subscriber ทั้งสอง

**Exercise 1:** ปรับการตั้งค่าที่ Subscriber เพื่อให้สามารถรับข้อมูลได้ (คลิกที่หัวของหน้าต่าง Subscriber จะแสดงฟอร์มให้สามารถแก้ไขค่าได้)

The screenshot shows the MQTT Explorer interface. The 'Topic to publish' field is set to 'myhome/floor1/temp'. The 'QoS' is set to '0 - Almost Once'. The 'Retain' checkbox is unchecked. The 'Payload Type' is set to 'Strings / JSON / XML'. The 'Payload' field contains 'Hello 25'. A blue arrow points from the 'Hello 25' payload to the 'Hello 25' message in the subscriber window. The subscriber window shows a message with the following details: 'qos : 0, retain : false, cmd : publish, dup : false, topic : temp, message id : , length : 14, Raw payload : 72101108108111325053'. A text box above the subscriber window says 'ไม่ปรากฏข้อมูลใดๆที่ Subscriber ทั้งสอง' (No data appears in either subscriber). A text box below the subscriber window shows the message details: 'Hello 25', 'topic:myhome/floor1/temp, qos:0, retain:false'.



# อ้างอิง

- Python ๑๐๑ หนังสือสอนเขียนโปรแกรมภาษา Python  
<https://www.cp.eng.chula.ac.th/books/python101/>