MQTT LAB

Prerequisite: try... except

Exceptions are errors that Python may encounter during the execution of your program

```
>>> # Exemple classique : test d'une division par zéro
>>> variable = 1/0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: int division or modulo by zero
```

try ...except : Allows you to put the instructions you want to test in a first block and the instructions to execute in case of error in another block.

```
try:
    # Bloc à essayer
except:
    # Bloc qui sera exécuté en cas d'erreur
```

Exercise 1: Local broker from Debian with mosquitto sub/pub

In your bash or on google, use the "man" command to read the command line manual:

- mosquitto_pub
- mosquitto_sub

From the Debian server (10.34.161.21 with port: 22), and using two PUTTY terminals (or SSH on MAC) use the two commands above to transmit a "Hello world" message via the MQTT topic: /UserXX/HW of the local broker (127.0.0.1)

Exercise 2: Local broker from Raspbian with a python script

2-a) Do the same thing as in exercise 1 from the Raspbian server (10.34.161.22 with port: 22) but use a python script instead of the mosquitto_pub command to send the message.

In a second terminal (always from your Raspbian server), you continue to use mosquitto_sub to retrieve the sent message.

Vincent LEFEVERE et Jad NASSAR

Example of a python script that publishes the message "toto" on the topic 'test' :

```
import paho.mqtt.client as mqtt
"""
The publish() function being asynchronous, it ends before the actual
sending of the message. So we have to use loop_forever() to allow
the program to run forever (to handle MQTT events). loop_forever()
stops when we execute the loop_stop() function. In this example, we
create a function on_pub() which will be called as soon as the
publication event takes place (i.e. client.on_publish = on_pub)
"""
def on_pub(client,userdata,result):
      print("message published")
      client.loop_stop()
      client.disconnect()


client = mqtt.Client()
client.on_publish = on_pub
client.connect("127.0.0.1", 1883)
client.publish("test", "toto",2) #2 corresponds to the maximum QoS
client.loop_forever()
```

2-b) In a terminal (always from your Raspbian server), use mosquitto_pub to send a "Hello World" message.

Then in a second step, do the same thing as exercise 1 from the Raspbian server (10.34.161.22 with port: 22) but using a python script instead of the mosquitto_sub command to do the subscription and display the received messages.

Example of a python script that subscribes to the test topic of the local broker:

```
"""
The subscribe() function indicates to which topic we want to
subscribe, then via loop_forever() we wait and process the MQTT
events.
In this example, we create a function on_mes() which will be called
as soon as the message reception event takes place (i.e.
client.on_message = on_mes). In the function after decoding the
message and displaying it, we stop the event processing loop and
disconnect client.loop_stop() and client.disconnect() (like in the
first example)
"""
import paho.mqtt.client as mqtt
def on_mes(client, userdata, mes):
      msg=mes.payload.decode("utf-8")
      print("Received ",msg," on topic ", mes.topic)
      client.loop_stop()
      client.disconnect()
client = mqtt.Client()
client.on_message = on_mes
client.connect("127.0.0.1", 1883)
client.subscribe("#",2)
client.loop_forever()
```

Exercise 3: Free broker with 1 Debian client and 1 Raspbian client with 2 equivalent sub/pub scripts

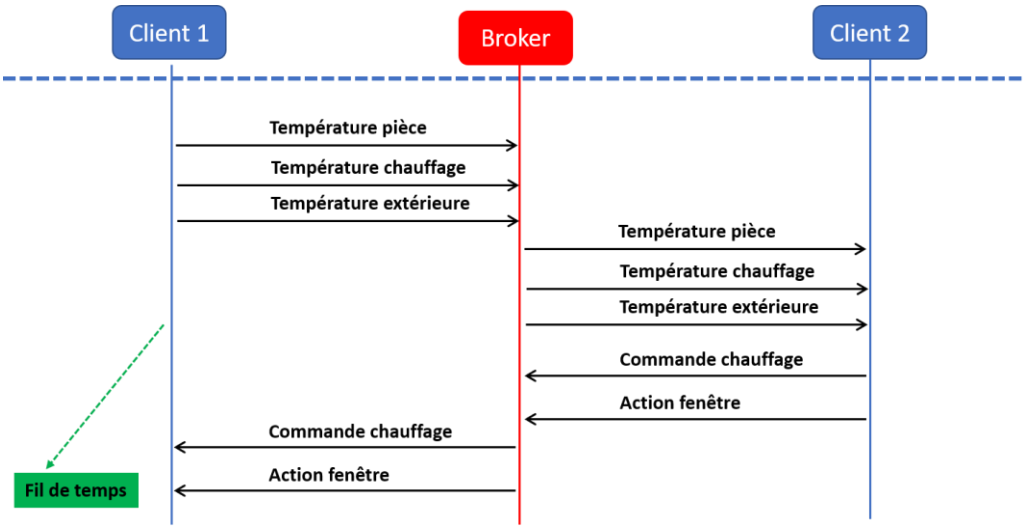
Using SCP on MAC and WINSCP on Windows transfer the script from exercise 2-b to the Debian server. The operation is done in two times.

For this exercise, we don't use the local broker (127.0.0.1), we have to use the free public MQTT broker (test.mosquitto.org).

Use 2 scripts : one on the Debian server to publish the "Hello World" message (on the test.mosquitto.org broker), and the other one on the Raspbian server to get it from the broker and display it.

Exercice 4:

The goal of this exercise is to ensure a 2-way communication with 2 equivalent sub/pub scripts in MQTT.



The client 1 will send 3 temperatures: room, heating, outside.

The client 2 subscribed to the corresponding topics will decide to turn off the heating if the temperature exceeds 25 degrees and/or open the window if the outside temperature is higher than the inside temperature (if it is warmer outside).

Use 3 topics for temperatures and 2 topics for actions (heating/window) of the following form:

- /Junia/Userxx/temp_piece
- /Junia/Userxx/temp_chauff
- /Junia/Userxx/temp_ext
- /Junia/Userxx/comm_chauff
- /Junia/Userxx/act_fen

For this exercise, we don't use the local broker (127.0.0.1) anymore, we have to use the public and free MQTT broker (test.mosquitto.org).

The client 1 will be on the Debian server and the client 2 on the Raspbian server.

Exercise 5: Use JSON with Ex 4 with serialized data in 1 single topic for the 3 temperatures and 1 topic for the heating and window action commands.

Use topics of the following form:

- /Junia/Userxx/temp
- /Junia/Userxx/cmd_act