

Develop the Python Script

(Publish data to IBM cloud)

| | |
|--------------|--|
| Team ID | PNT2022TMID12811 |
| Project Name | Industry-specific intelligent fire management system |

Industry-specific intelligent fire management system



The screenshot shows a Python script in a text editor and its execution in a terminal. The script, named `publish.py`, is designed to publish data to an MQTT broker. It includes a comment: `#Through python coding we are going to access the subscriber`. The script imports `paho.mqtt.client as paho`, `time`, and `random`. It defines a function `on_publish` that prints "Publish the data". The main logic creates a `paho.Client`, connects to `broker.mqttdashboard.com` on port 1883, and enters a `while True` loop. In the loop, it generates a random integer between 1 and 30, publishes it to the topic `iottopic` with a QoS of 1, prints the value, and sleeps for 10 seconds.

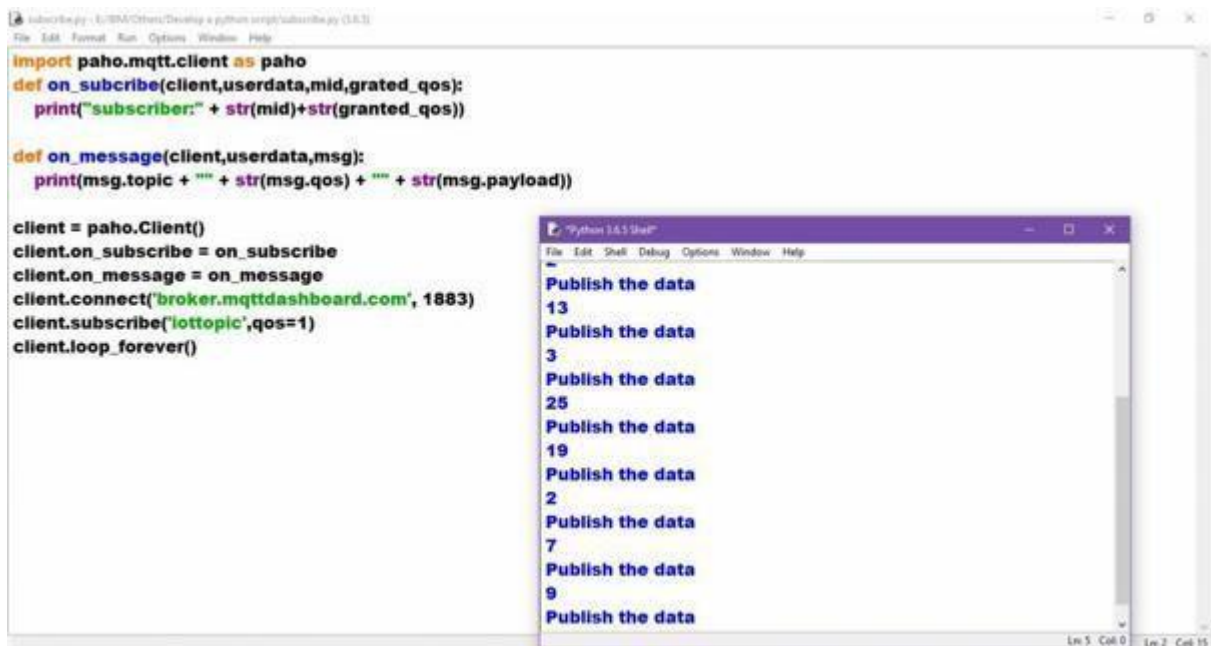
```
#Through python coding we are going to access the subscriber
import paho.mqtt.client as paho
import time
import random

def on_publish(client, userdata, mid):
    print("Publish the data ")

client = paho.Client()
client.on_publish = on_publish
client.connect('broker.mqttdashboard.com', 1883)
client.loop_start()
while True:
    temp = random.randint(1,30)
    (re,mid) = client.publish('iottopic',str(temp),qos=1)
    print(temp)
    time.sleep(10)
```

The terminal output shows the following sequence of events:

```
Python 3.6.5 Shell
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 17:00:18) [MS
C v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more informatio
n.
>>>
===== RESTART: E:\IBM\Others\Develop a python script\
publish.py =====
7
Publish the data
19
Publish the data
10
Publish the data
```



The screenshot shows a Python script in a text editor and its execution in a terminal. The script, named `subscribe.py`, is designed to subscribe to an MQTT broker. It imports `paho.mqtt.client as paho`. It defines two functions: `on_subscribe`, which prints the subscriber ID and granted QoS, and `on_message`, which prints the message topic, QoS, and payload. The main logic creates a `paho.Client`, connects to `broker.mqttdashboard.com` on port 1883, subscribes to the topic `iottopic` with a QoS of 1, and enters a `client.loop_forever()` loop.

```
import paho.mqtt.client as paho
def on_subscribe(client,userdata,mid,grated_qos):
    print("subscriber:" + str(mid)+str(granted_qos))

def on_message(client,userdata,msg):
    print(msg.topic + " " + str(msg.qos) + " " + str(msg.payload))

client = paho.Client()
client.on_subscribe = on_subscribe
client.on_message = on_message
client.connect('broker.mqttdashboard.com', 1883)
client.subscribe('iottopic',qos=1)
client.loop_forever()
```

The terminal output shows the following sequence of events:

```
Python 3.6.5 Shell
Publish the data
13
Publish the data
3
Publish the data
25
Publish the data
19
Publish the data
2
Publish the data
7
Publish the data
9
Publish the data
```

IBM Watson IoT Platform

hariharan07ananth@psnacet.edu.in
ID: kkfe0q

Browse Action Device Types Interfaces

Add Device +

Identity Device Information **Recent Events** State Logs

The recent events listed show the live stream of data that is coming and going from this device.

| Event | Value | Format | Last Received |
|---------|----------------------------------|--------|-------------------|
| event_1 | {"Temperature":94,"Humidity":95} | json | a few seconds ago |
| event_1 | {"Temperature":73,"Humidity":43} | json | a few seconds ago |
| event_1 | {"Temperature":72,"Humidity":22} | json | a few seconds ago |
| event_1 | {"Temperature":57,"Humidity":55} | json | a few seconds ago |
| event_1 | {"Temperature":64,"Humidity":53} | json | a few seconds ago |

1 Simulation running

IBM Watson IoT Platform

hariharan07ananth@psna...
ID: (select org)

Buildings

Collect data from

and make value from it

Cookie Preferences

Show all

Program :

```
#IBM Watson IOT Platform
#pip install wiotp-sdk
import wiotp.sdk.device
import time
import random
```

```

myConfig = {"identity":
{
    "orgId": "hj5fmy",
    "typeId": "NodeMCU",
    "deviceId": "12345" },
    "auth": { "token": "12345678" }
}

def myCommandCallback(cmd):
    print("Message received from IBM IoT Platform: %s" % cmd.data['command'])
    m=cmd.data['command']

client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
client.connect()

while True:
    temp=random.randint(-20,125)
    hum=random.randint(0,100)
    myData={'temperature':temp, 'humidity':hum}
    client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0,
onPublish=None)
    print("Published data Successfully: %s", myData)
    client.commandCallback = myCommandCallback
    time.sleep(2)
    client.disconnect()

```