

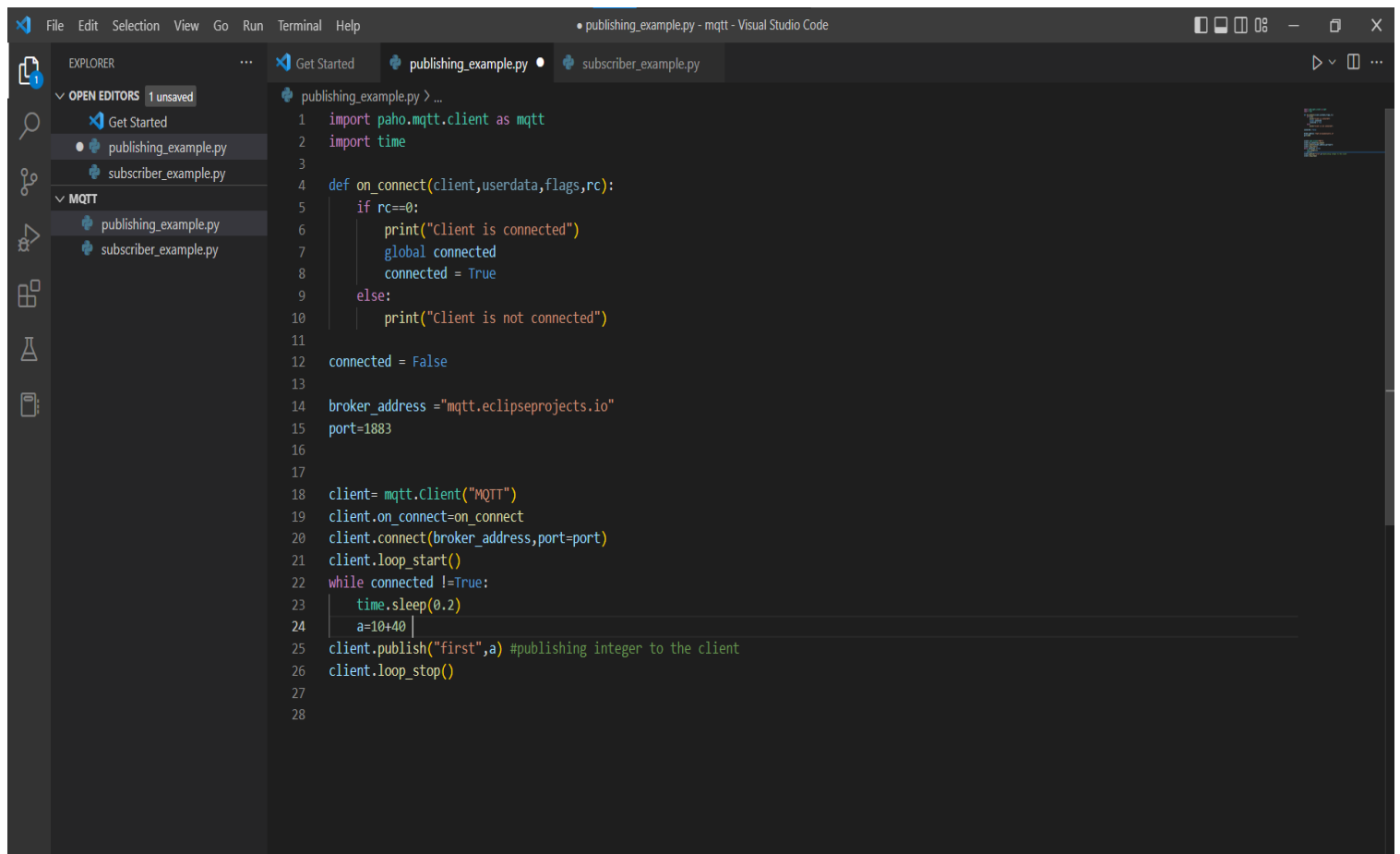
Task 2 – Basics of MQTT

name - Kathit Bhongale

In this task, a free online MQTT server is used to host the service

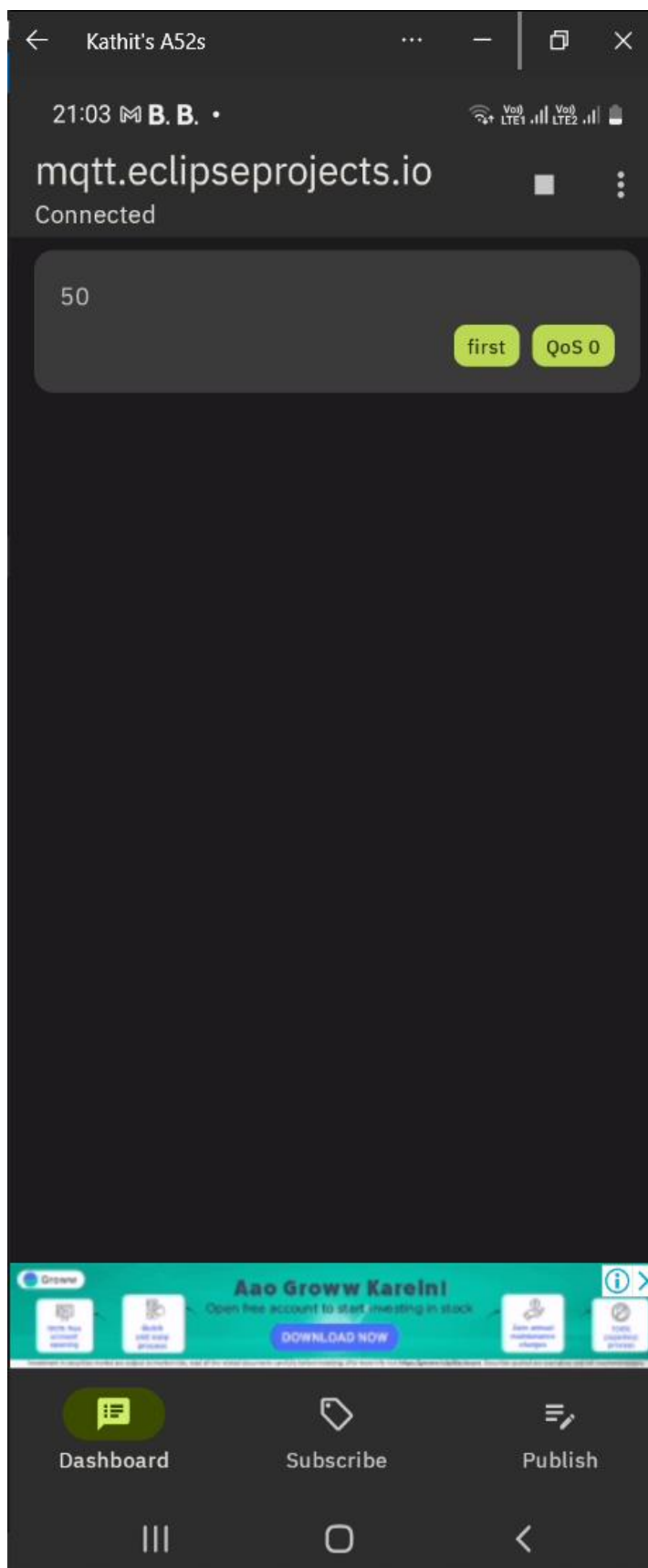
Two devices (a laptop and an android phone) are used for subscribing and publishing messages

1) Publishing a message (integer)

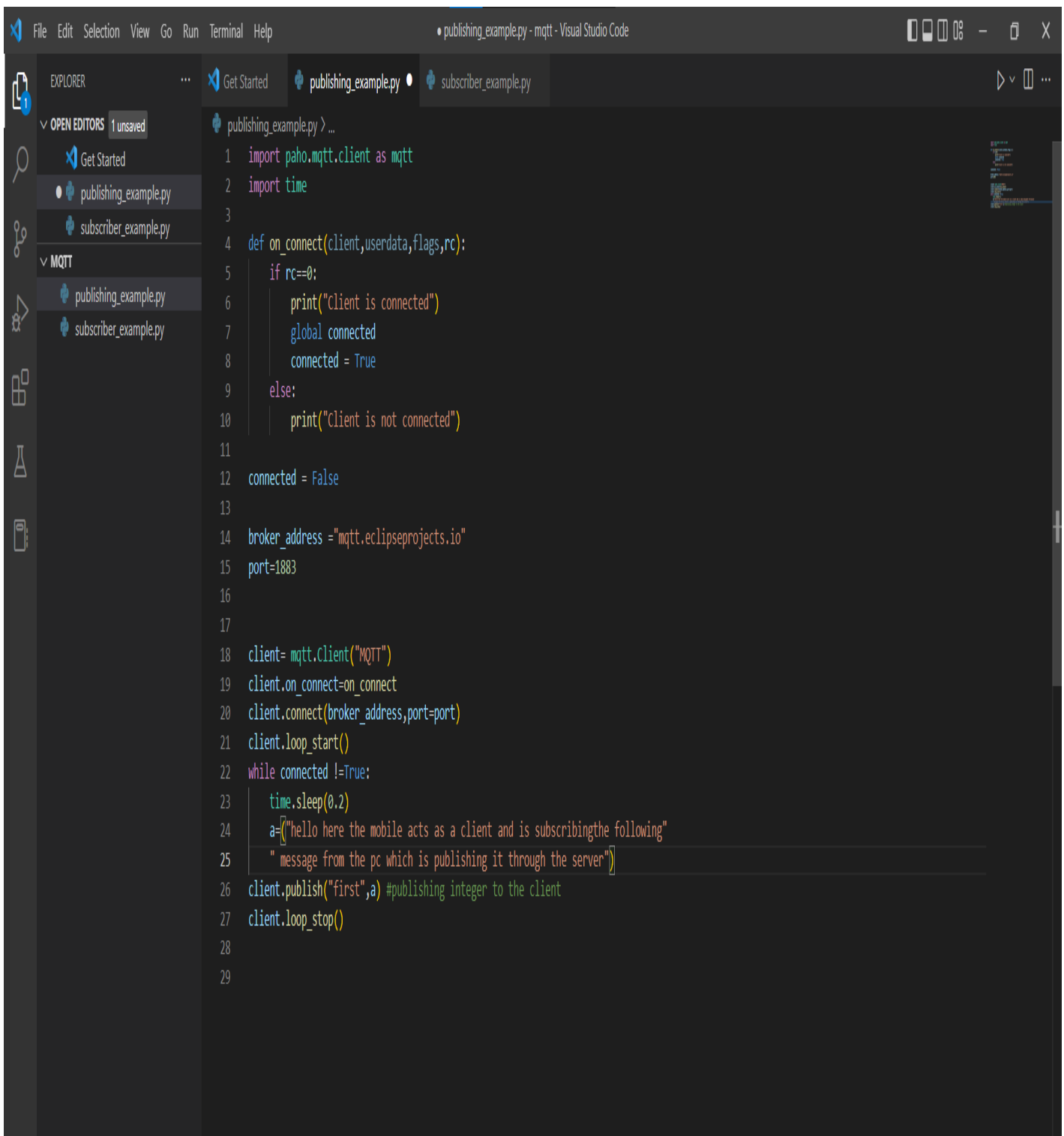
A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows a project structure with 'MQTT' as a folder containing 'publishing_example.py' and 'subscriber_example.py'. The main editor window displays the code for 'publishing_example.py'. The code imports 'paho.mqtt.client as mqtt' and 'time'. It defines an 'on_connect' function that prints 'Client is connected' if the connection is successful and 'Client is not connected' otherwise. A global 'connected' variable is set to False. The broker address is 'mqtt.eclipseprojects.io' and the port is 1883. A client is created with the name 'MQTT', and its 'on_connect' method is set to the 'on_connect' function. The client connects to the broker and starts its loop. A while loop runs as long as 'connected' is not True, with a 0.2-second sleep and a counter 'a' that increments by 10 each iteration. Finally, the client publishes the string 'first' with the integer value of 'a' and then stops the loop.

```
1 import paho.mqtt.client as mqtt
2 import time
3
4 def on_connect(client,userdata,flags,rc):
5     if rc==0:
6         print("Client is connected")
7         global connected
8         connected = True
9     else:
10        print("Client is not connected")
11
12 connected = False
13
14 broker_address ="mqtt.eclipseprojects.io"
15 port=1883
16
17
18 client= mqtt.Client("MQTT")
19 client.on_connect=on_connect
20 client.connect(broker_address,port=port)
21 client.loop_start()
22 while connected !=True:
23     time.sleep(0.2)
24     a=10+40
25 client.publish("first",a) #publishing integer to the client
26 client.loop_stop()
27
28
```

Published message in the client



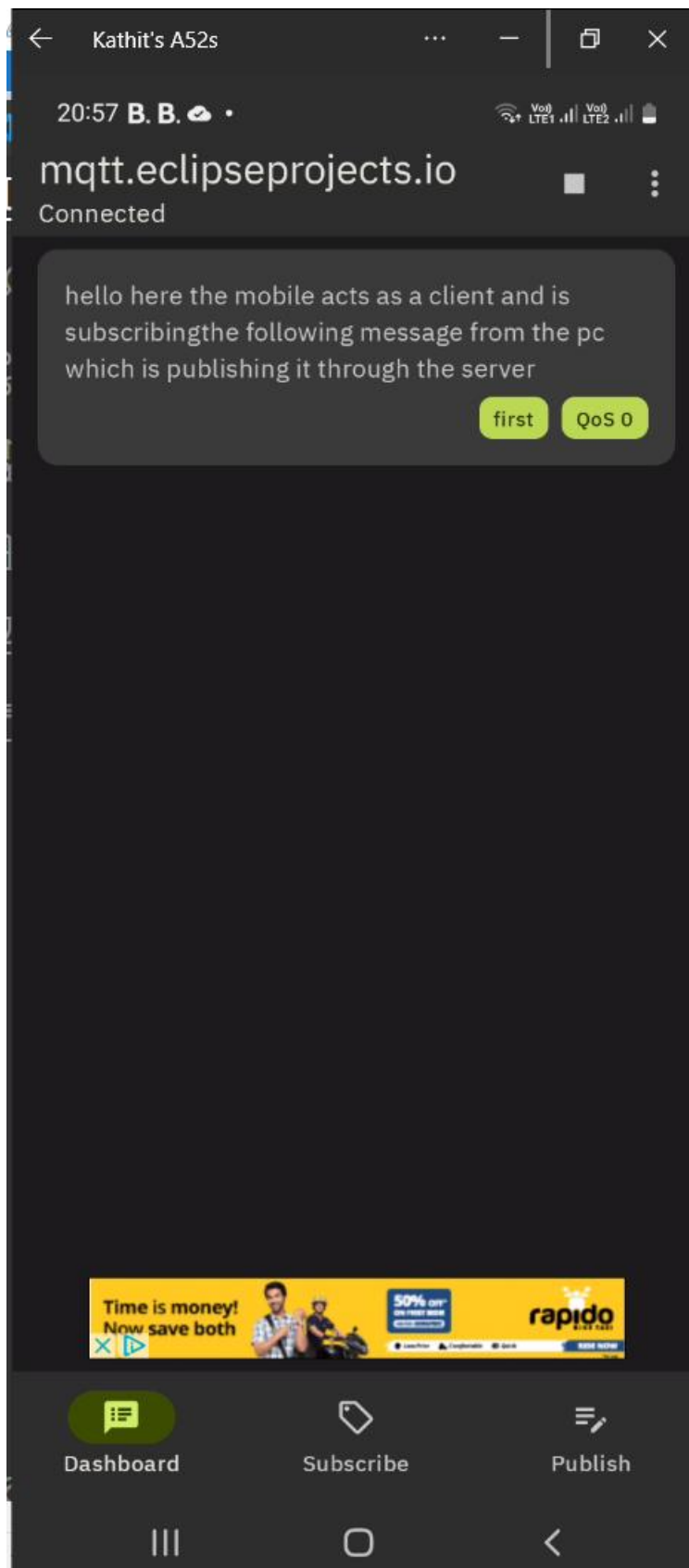
2) Publishing a message (string)



The screenshot shows the Visual Studio Code interface with a Python file named `publishing_example.py` open. The Explorer sidebar on the left shows the file structure with `publishing_example.py` and `subscriber_example.py` under an `MQTT` folder. The main editor displays the following Python code:

```
1 import paho.mqtt.client as mqtt
2 import time
3
4 def on_connect(client,userdata,flags,rc):
5     if rc==0:
6         print("Client is connected")
7         global connected
8         connected = True
9     else:
10        print("Client is not connected")
11
12 connected = False
13
14 broker_address = "mqtt.eclipseprojects.io"
15 port=1883
16
17
18 client= mqtt.Client("MQTT")
19 client.on_connect=on_connect
20 client.connect(broker_address,port=port)
21 client.loop_start()
22 while connected !=True:
23     time.sleep(0.2)
24     a=["hello here the mobile acts as a client and is subscribingthe following"
25       " message from the pc which is publishing it through the server"]
26 client.publish("first",a) #publishing integer to the client
27 client.loop_stop()
28
29
```

Published message in the client device



3) Publishing the message from mobile (integer)

The screenshot shows a mobile application window titled "Kathit's A52s". The status bar at the top displays the time 22:35, battery level, and network status (VoLTE1, VoLTE2). The main screen has a dark background with the word "Publish" at the top. Below it are two text input fields: "Topic" with the value "second" and "Message" with the value "456". At the bottom right of the input area are two green buttons labeled "Save" and "Publish". A yellow banner advertisement for "rapido" is visible above the bottom navigation bar. The navigation bar at the very bottom contains three icons with labels: "Dashboard", "Subscribe", and "Publish" (which is highlighted with a green background).

Kathit's A52s

22:35

VoLTE1 VoLTE2

Publish

Topic

second

Message

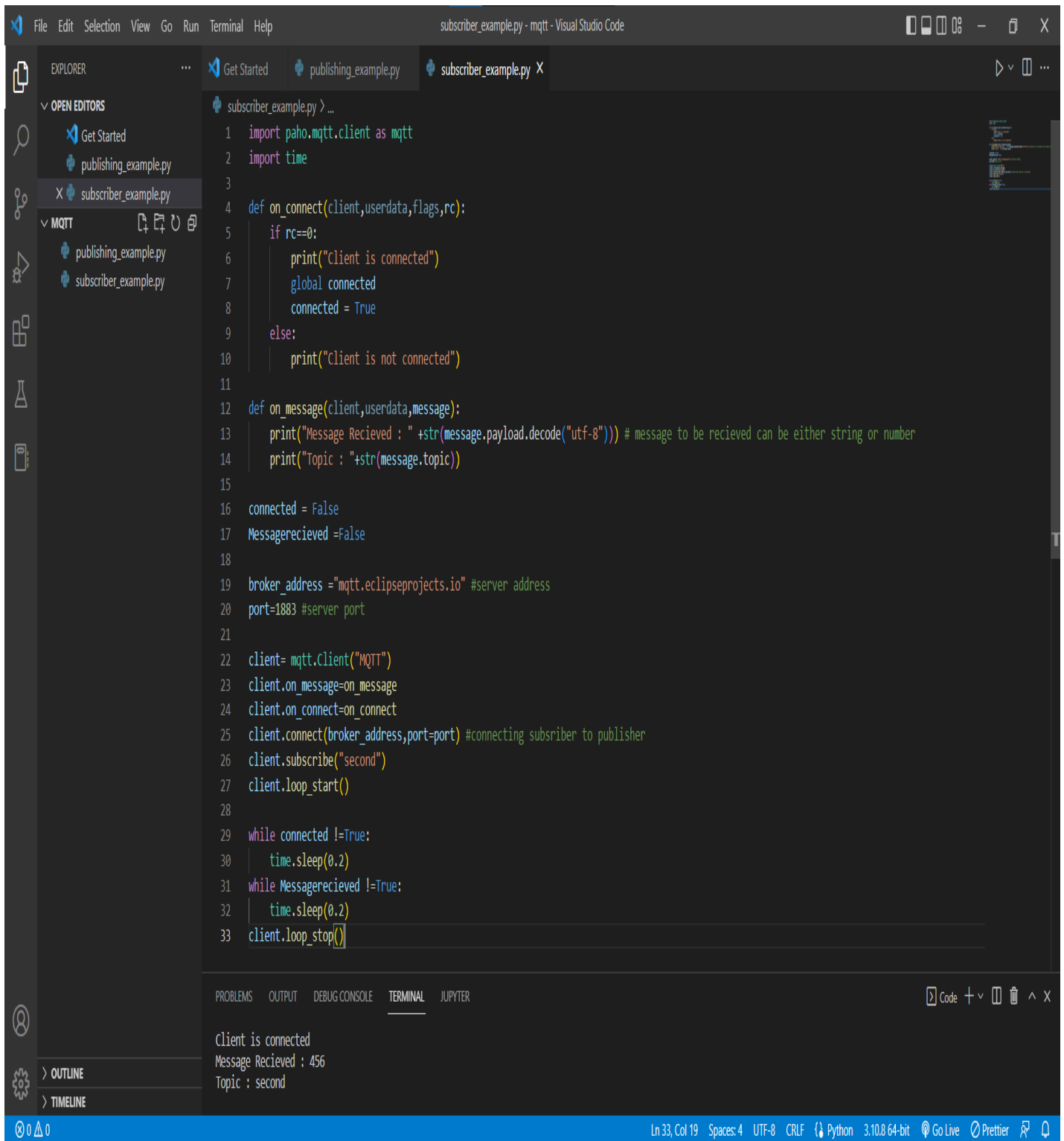
456

Save Publish

Time is money! Now save both 50% off on your first order rapido

Dashboard Subscribe Publish

Output in the client (laptop acting as a subscriber)



The screenshot displays the Visual Studio Code interface with a Python script named `subscriber_example.py` open in the editor. The script is configured to connect to an MQTT broker at `mqtt.eclipseprojects.io` on port 1883 and subscribe to the `second` topic. The terminal window at the bottom shows the execution output, indicating a successful connection and the receipt of a message.

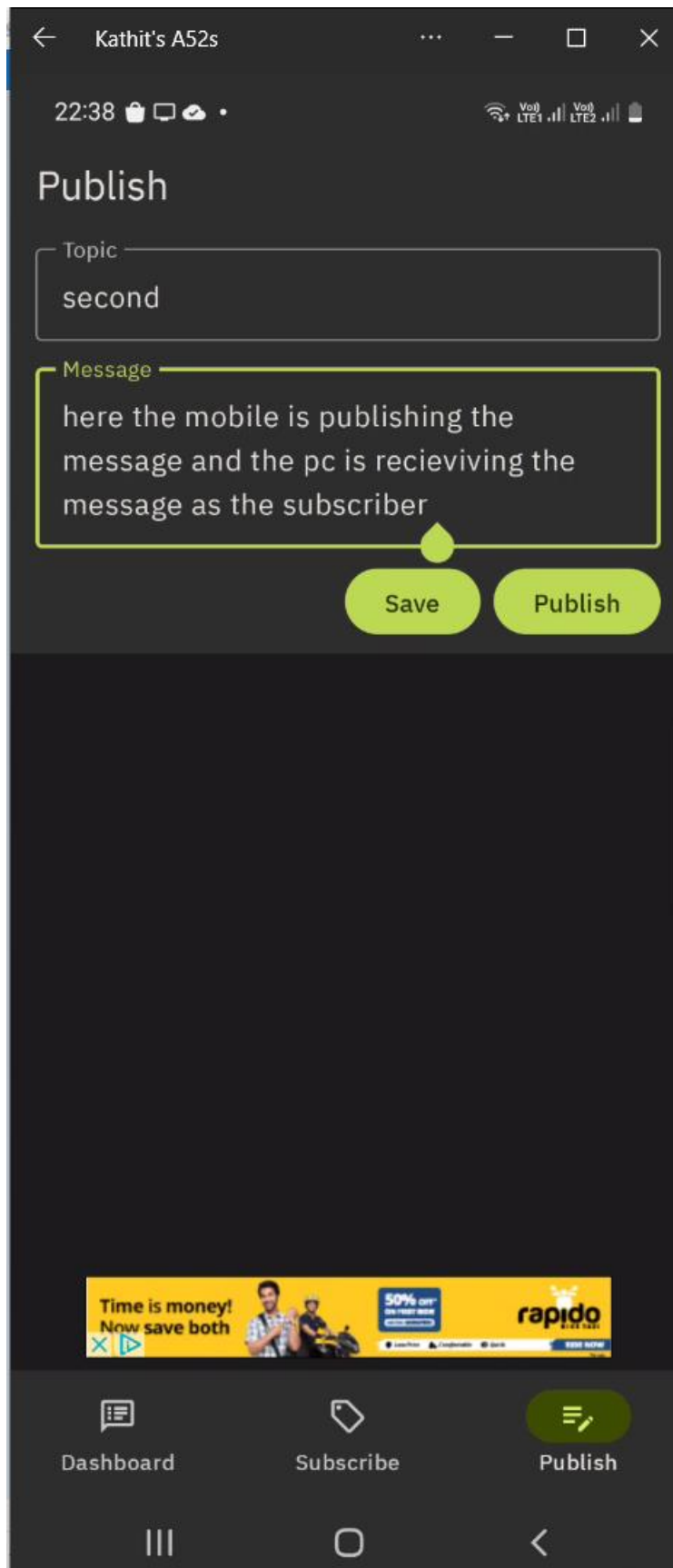
```
1 import paho.mqtt.client as mqtt
2 import time
3
4 def on_connect(client,userdata,flags,rc):
5     if rc==0:
6         print("Client is connected")
7         global connected
8         connected = True
9     else:
10        print("client is not connected")
11
12 def on_message(client,userdata,message):
13     print("Message Recieved : "+str(message.payload.decode("utf-8"))) # message to be recieved can be either string or number
14     print("Topic : "+str(message.topic))
15
16 connected = False
17 Messagerecieved =False
18
19 broker_address ="mqtt.eclipseprojects.io" #server address
20 port=1883 #server port
21
22 client= mqtt.Client("MQTT")
23 client.on_message=on_message
24 client.on_connect=on_connect
25 client.connect(broker_address,port=port) #connecting subscriber to publisher
26 client.subscribe("second")
27 client.loop_start()
28
29 while connected !=True:
30     time.sleep(0.2)
31 while Messagerecieved !=True:
32     time.sleep(0.2)
33 client.loop_stop()
```

Client is connected
Message Recieved : 456
Topic : second

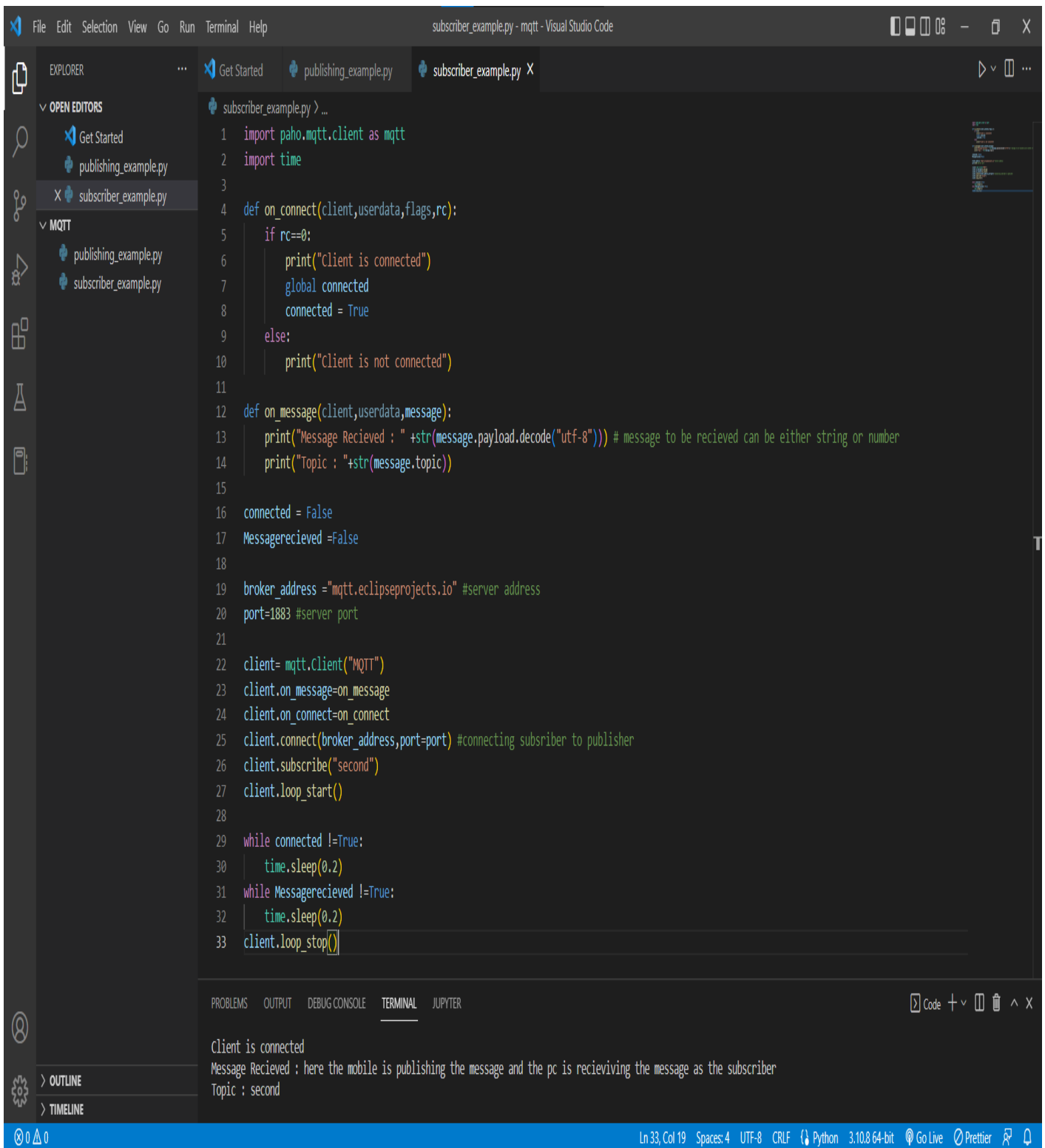
Message received in the terminal



4) Publishing message from the mobile (string)



Output in the client (laptop acting as a subscriber)



The screenshot displays the Visual Studio Code interface with a Python script named `subscriber_example.py` open in the editor. The script is an MQTT client that connects to a broker, subscribes to a topic, and prints received messages. The terminal at the bottom shows the execution output.

```
1 import paho.mqtt.client as mqtt
2 import time
3
4 def on_connect(client, userdata, flags, rc):
5     if rc==0:
6         print("Client is connected")
7         global connected
8         connected = True
9     else:
10        print("Client is not connected")
11
12 def on_message(client, userdata, message):
13     print("Message Recieved : " +str(message.payload.decode("utf-8"))) # message to be recieved can be either string or number
14     print("Topic : "+str(message.topic))
15
16 connected = False
17 Messagerecieved =False
18
19 broker_address ="mqtt.eclipseprojects.io" #server address
20 port=1883 #server port
21
22 client= mqtt.Client("MQTT")
23 client.on_message=on_message
24 client.on_connect=on_connect
25 client.connect(broker_address,port=port) #connecting subscriber to publisher
26 client.subscribe("second")
27 client.loop_start()
28
29 while connected !=True:
30     time.sleep(0.2)
31 while Messagerecieved !=True:
32     time.sleep(0.2)
33 client.loop_stop()
```

The terminal output shows the following messages:

```
Client is connected
Message Recieved : here the mobile is publishing the message and the pc is recieving the message as the subscriber
Topic : second
```

Message Received in the terminal

