

Python Code Development

Team ID	PNT2022TMID02550
Project Name	Smart waste management system for metropolitan cities

Python Script

```
import requests
import json
#import ibmiotf.application
import ibmiotf.device
import time
import random
import sys

# watson device details

organization = "nafgr4"
devicType = "RaspberryPi"
deviceId = "12345"
authMethod= "token"
authToken= "12345678"

#generate random values for randomo variables (temperature&humidity)

def myCommandCallback(cmd):
    global a
    print("command recieved:%s" %cmd.data['command'])
    control=cmd.data['command']
    print(control)

try:
    deviceOptions={"org": organization, "type": devicType,"id":
deviceId,"auth-method":authMethod,"auth-token":authToken}
    deviceCli = ibmiotf.device.Client(deviceOptions)
except Exception as e:
    print("caught exception connecting device %s" %str(e))
    sys.exit()

#connect and send a datapoint "temp" with value integer value into the cloud
as a type of event for every 10 seconds
deviceCli.connect()
while True:

    distance1=random.randint(10,80)
```

```

distance2=random.randint(10,80)
data= {'dist':distance1,'dist2':distance2}

if distance1 < 15 and distance2<15:
    warn = 'Risk warning:' 'Dumpster poundage getting high, Time to
collect :) 90 %'

elif distance1 >40 and distance2 >40:
    warn = 'Risk warning:' 'dumpster is above 50%'

else :
    warn = 'alert :' 'No need to collect right now '
def myOnPublishCallback(lat=13.0827,long=80.2707):
    print("Chennai")
    print("published distance1 = %s " %distance1,"distance2 = %s "
%distance2,"lon = %s " %long,"lat = %s" %lat)
    print(warn)

time.sleep(10)

success=deviceCli.publishEvent ("IoTSensor","json",warn,qos=0,on_publish=
myOnPublishCallback)

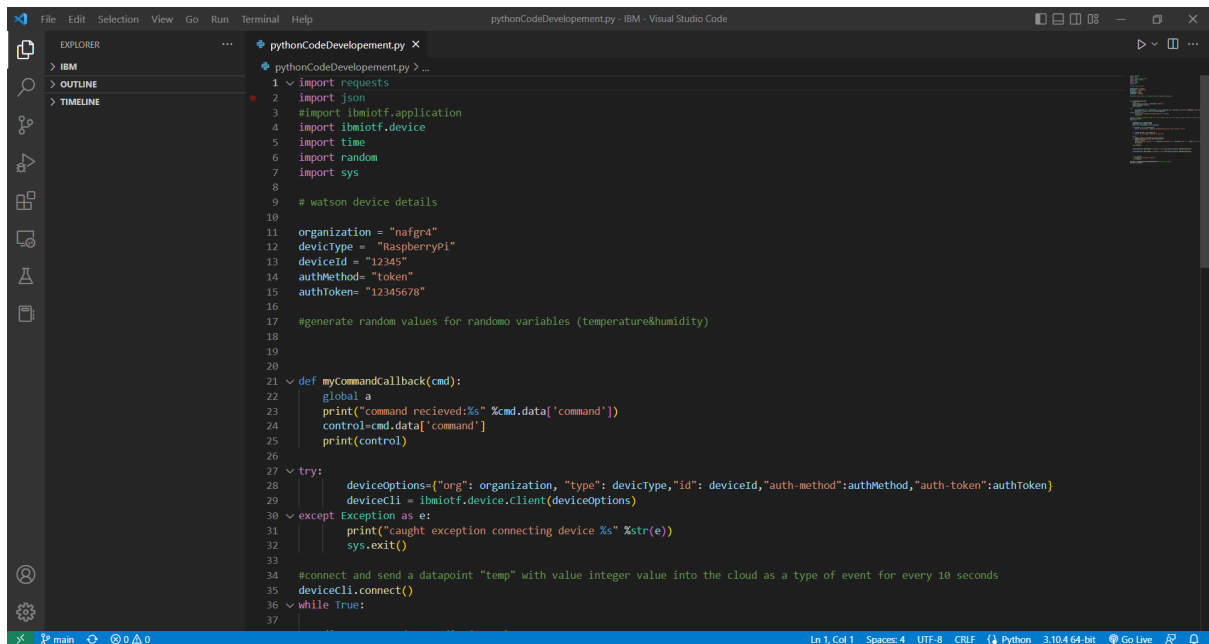
success=deviceCli.publishEvent ("IoTSensor","json",data,qos=0,on_publish=
myOnPublishCallback)

if not success:
    print("not connected to ibmiot")
time.sleep(30)

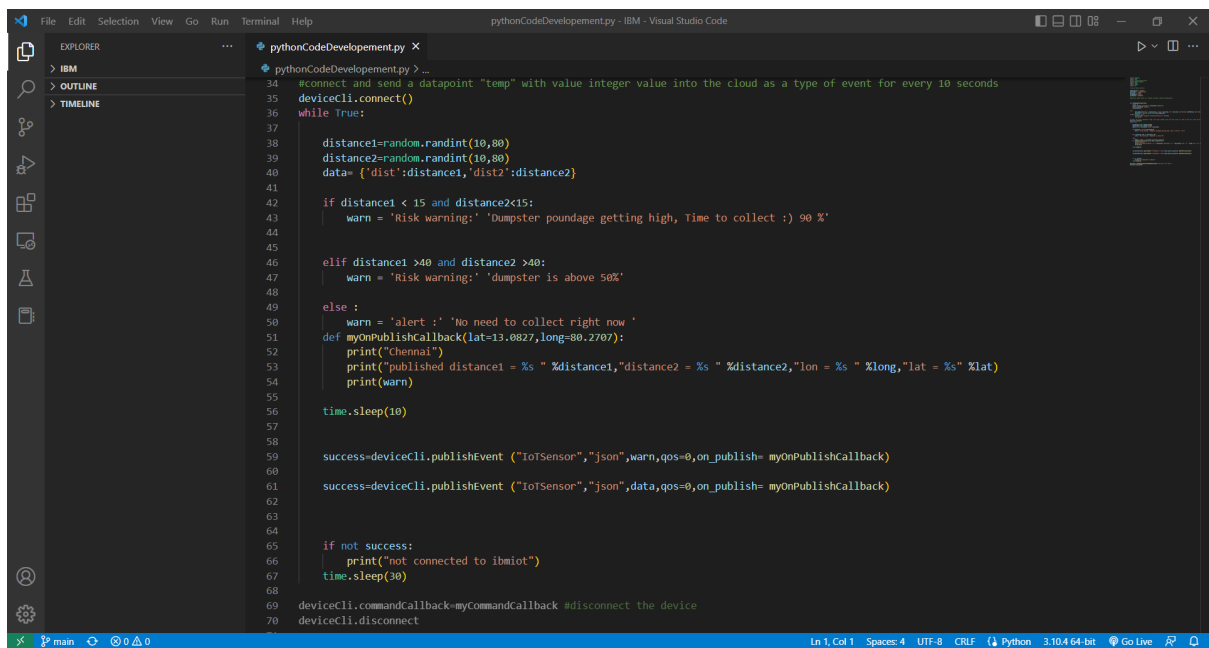
deviceCli.commandCallback=myCommandCallback #disconnect the device
deviceCli.disconnect

```

Code:

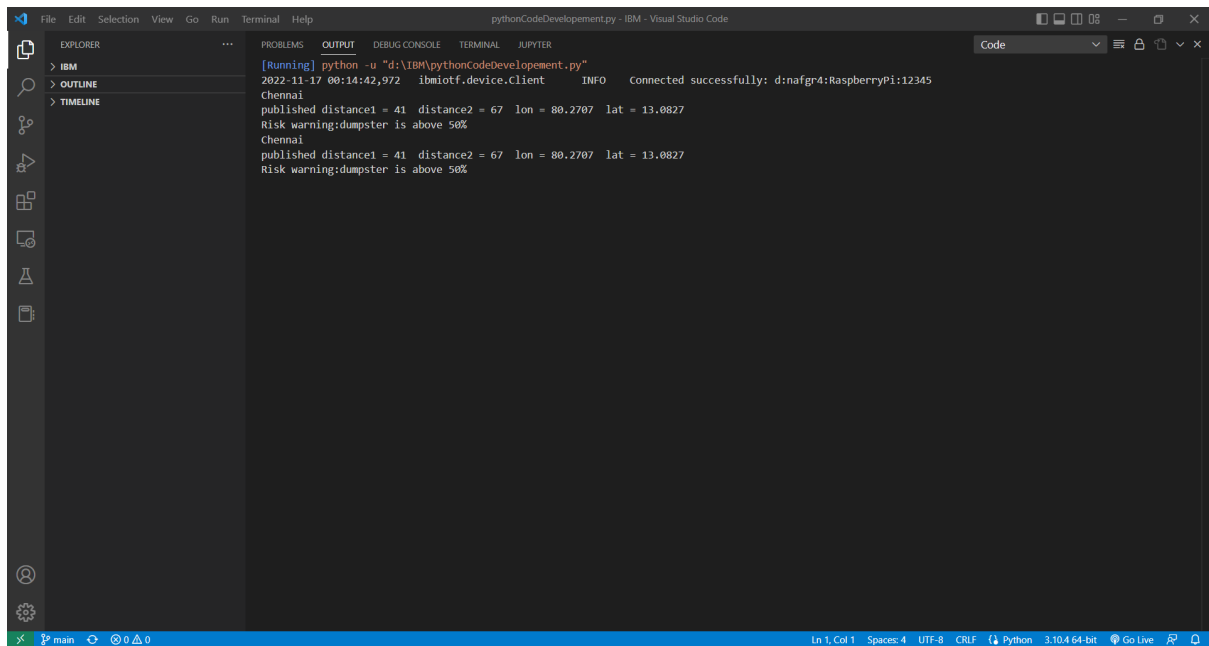


```
pythonCodeDevelopment.py X
pythonCodeDevelopment.py > ...
1 import requests
2 import json
3 #import ibmiotf.application
4 import ibmiotf.device
5 import time
6 import random
7 import sys
8
9 # watson device details
10
11 organization = "nafgr4"
12 deviceType = "RaspberryPi"
13 deviceId = "12345"
14 authMethod = "token"
15 authToken = "12345678"
16
17 #generate random values for random variables (temperature&humidity)
18
19
20
21 def myCommandCallback(cmd):
22     global a
23     print("command recieved:%s" %cmd.data['command'])
24     control=cmd.data['command']
25     print(control)
26
27 try:
28     deviceOptions={"org": organization, "type": deviceType, "id": deviceId, "auth-method":authMethod, "auth-token":authToken}
29     deviceCli = ibmiotf.device.Client(deviceOptions)
30 except Exception as e:
31     print("caught exception connecting device %s" %str(e))
32     sys.exit()
33
34 #connect and send a datapoint "temp" with value integer value into the cloud as a type of event for every 10 seconds
35 deviceCli.connect()
36 while True:
37
```



```
pythonCodeDevelopment.py X
pythonCodeDevelopment.py > ...
34 #connect and send a datapoint "temp" with value integer value into the cloud as a type of event for every 10 seconds
35 deviceCli.connect()
36 while True:
37
38     distance1=random.randint(10,80)
39     distance2=random.randint(10,80)
40     data= {'dist':distance1,'dist2':distance2}
41
42     if distance1 < 15 and distance2<15:
43         warn = 'Risk warning:' 'Dumpster poundage getting high, Time to collect :) 90 %'
44
45
46     elif distance1 >40 and distance2 >40:
47         warn = 'Risk warning:' 'dumpster is above 50%'
48
49     else :
50         warn = 'alert :' 'No need to collect right now '
51
52     def myOnPublishcallback(lat=13.0827,lon=80.2707):
53         print("Chennai")
54         print("published distance1 = %s " %distance1,"distance2 = %s " %distance2,"lon = %s " %long,"lat = %s" %lat)
55         print(warn)
56
57     time.sleep(10)
58
59     success=deviceCli.publishEvent ("IoTSensor","json",warn,qos=0,on_publish= myOnPublishCallback)
60
61     success=deviceCli.publishEvent ("IoTSensor","json",data,qos=0,on_publish= myOnPublishCallback)
62
63
64
65     if not success:
66         print("not connected to ibmiot")
67         time.sleep(30)
68
69 deviceCli.commandCallback=myCommandCallback #disconnect the device
70 deviceCli.disconnect
```

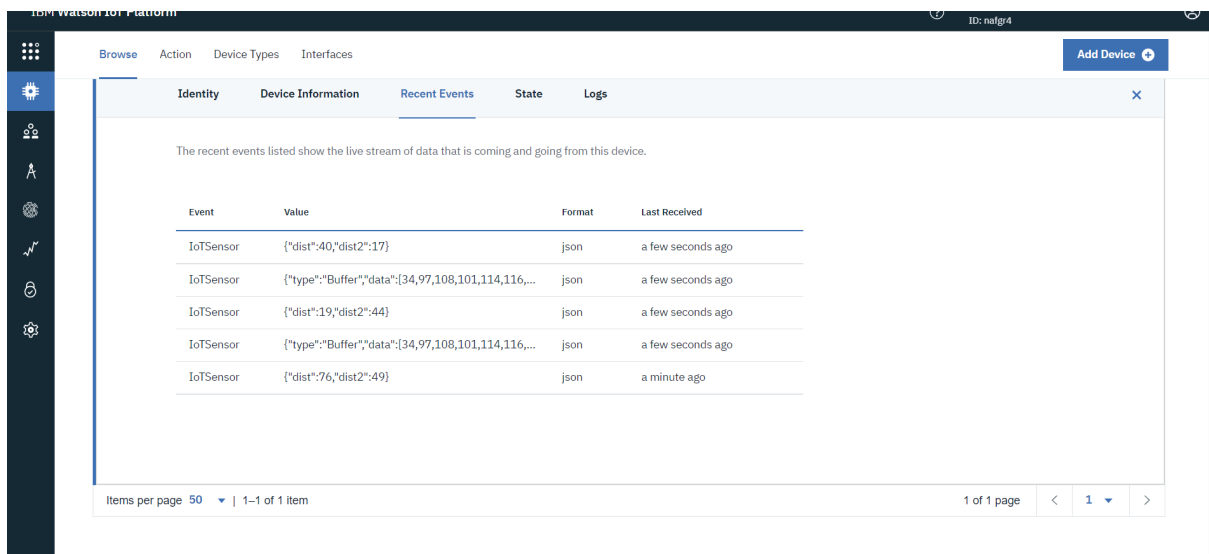
Connection To IBM Watson



The screenshot shows the Visual Studio Code interface with a terminal window open. The terminal output indicates a successful connection to the IBM Watson IoT Platform. The output is as follows:

```
[Running] python -u "d:\IBM\pythonCodeDevelopment.py"
2022-11-17 00:14:42,972 ibmiotf.device.Client INFO Connected successfully: d:nafgr4:RaspberryPi:12345
Chennai
published distance1 = 41 distance2 = 67 lon = 80.2707 lat = 13.0827
Risk warning:dumpster is above 50%
Chennai
published distance1 = 41 distance2 = 67 lon = 80.2707 lat = 13.0827
Risk warning:dumpster is above 50%
```

IBM Watson



The screenshot shows the IBM Watson IoT Platform interface. The 'Recent Events' tab is selected, displaying a table of events for the device 'nafgr4'. The table has four columns: Event, Value, Format, and Last Received. The events are listed as follows:

Event	Value	Format	Last Received
IoTSensor	{"dist":40,"dist2":17}	json	a few seconds ago
IoTSensor	{"type":"Buffer","data":[34,97,108,101,114,116,...]}	json	a few seconds ago
IoTSensor	{"dist":19,"dist2":44}	json	a few seconds ago
IoTSensor	{"type":"Buffer","data":[34,97,108,101,114,116,...]}	json	a few seconds ago
IoTSensor	{"dist":76,"dist2":49}	json	a minute ago

At the bottom of the table, there is a pagination control showing 'Items per page 50' and '1-1 of 1 item'.

Node Red Output

The screenshot shows the Node-RED web interface. On the left, a sidebar contains a 'filter nodes' search bar and a list of node categories: common, function, network, input, output, sequence, parser, storage, and IBM Watson. The main workspace, titled 'Flow 1', contains a flow with three nodes: 'Hello Node-RED!' (function), 'IBM IoT' (input, with a 'connected' status), and 'msg.payload' (output). The 'IBM IoT' node is connected to the 'Hello Node-RED!' node, and the 'Hello Node-RED!' node is connected to the 'msg.payload' node. On the right, a 'debug' console displays a series of messages. The messages are JSON objects with a 'dist' property and a 'dist2' property. The messages are as follows:

```
{ dist: 40, dist2: 17 }
11/17/2022, 12:14:53 AM node: 020649a.0d0d98
{ type: 'RaspberryPiId12345ev1IoT Sensor', msg: { payload: 'string[34]' } }
"Risk warning: dumpster is above 50%"
11/17/2022, 12:14:54 AM node: 020649a.0d0d98
{ type: 'RaspberryPiId12345ev1IoT Sensor', msg: { payload: 'Object' } }
{ dist: 41, dist2: 67 }
11/17/2022, 12:15:33 AM node: 020649a.0d0d98
{ type: 'RaspberryPiId12345ev1IoT Sensor', msg: { payload: 'string[36]' } }
"alert :No need to collect right now "
11/17/2022, 12:15:34 AM node: 020649a.0d0d98
{ type: 'RaspberryPiId12345ev1IoT Sensor', msg: { payload: 'Object' } }
{ dist: 46, dist2: 24 }
11/17/2022, 12:16:13 AM node: 020649a.0d0d98
{ type: 'RaspberryPiId12345ev1IoT Sensor', msg: { payload: 'string[34]' } }
"Risk warning: dumpster is above 50%"
11/17/2022, 12:16:14 AM node: 020649a.0d0d98
{ type: 'RaspberryPiId12345ev1IoT Sensor', msg: { payload: 'Object' } }
{ dist: 79, dist2: 56 }
```