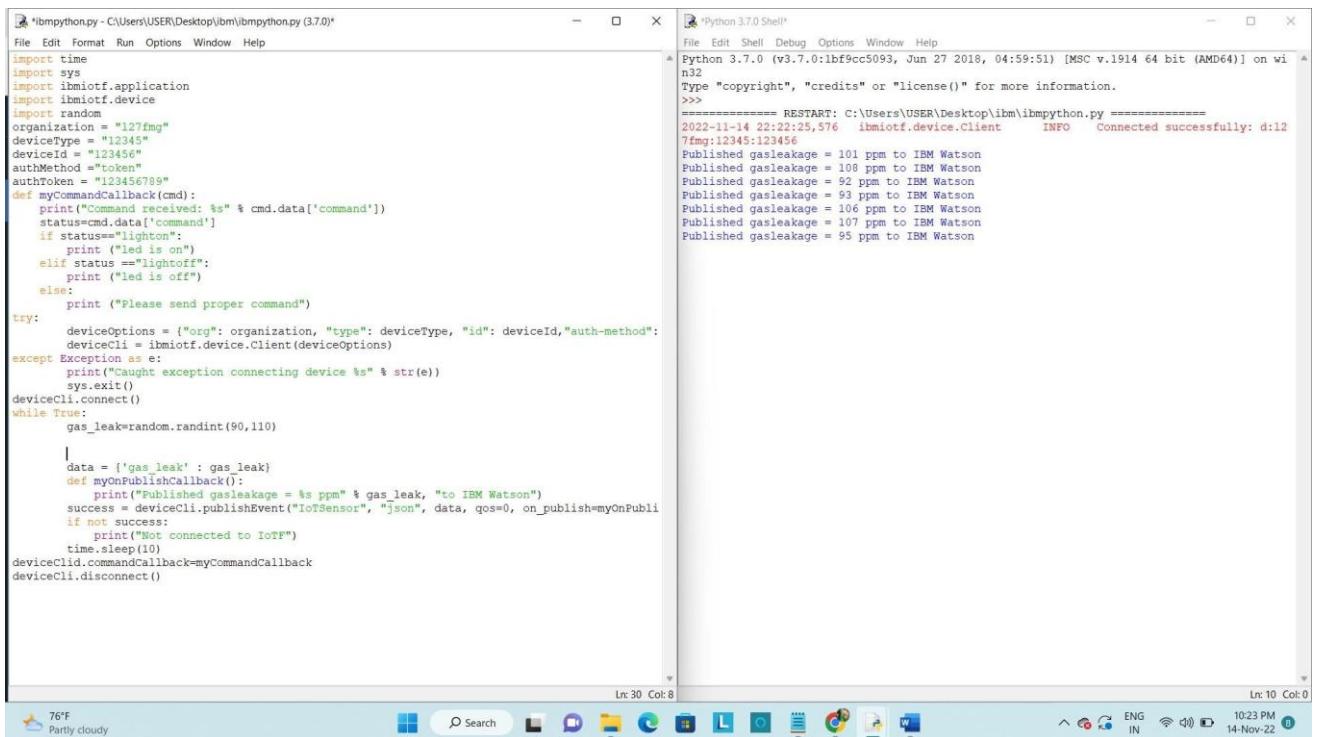


## PUBLISH DATA TO PYTHON CODE

Date	17 November 2022
Team ID	PNT2022TMID53681
Project Name	Gas Leakage Monitoring & Alerting System for Industries
Maximum Marks	4 Marks

### Gas Leakage Monitoring & Alerting System for Industries

Python code :



The image shows a screenshot of a Windows desktop with two windows open. The left window is a text editor showing a Python script named 'ibmpython.py'. The script imports 'time', 'sys', 'ibmiotf.application', 'ibmiotf.device', and 'random'. It defines variables for 'organization', 'deviceType', 'deviceId', 'authMethod', and 'authToken'. A function 'myCommandCallback(cmd)' is defined to handle incoming commands like 'lighton', 'lightoff', and 'led is on/off'. The main part of the script connects to the IBM IoT Platform using 'deviceCli.connect()' and enters a loop where it generates random gas leak data and publishes it to IBM Watson using 'deviceCli.publishEvent()'. The right window is a 'Python 3.7.0 Shell' showing the execution output. It displays the restart command, connection status, and a series of published gas leakage values (e.g., 101 ppm, 108 ppm, 92 ppm, etc.) to IBM Watson.

```
File Edit Format Run Options Window Help
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random

organization = "127fmg"
deviceType = "12345"
deviceId = "123456"
authMethod = "token"
authToken = "123456789"

def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="lighton":
        print ("led is on")
    elif status == "lightoff":
        print ("led is off")
    else:
        print ("Please send proper command")

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

deviceCli.connect()
while True:
    gas_leak=random.randint(90,110)

    data = {'gas leak' : gas_leak}
    def myOnPublishCallback():
        print("Published gasleakage = %s ppm" % gas_leak, "to IBM Watson")
    success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0, on_publish=myOnPubli
    if not success:
        print("Not connected to IoT")
    time.sleep(10)
deviceCli.commandCallback=myCommandCallback
deviceCli.disconnect()
```

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on wi
n32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\USER\Desktop\ibm\ibmpython.py =====
2022-11-14 22:22:25,576 ibmiotf.device.Client INFO Connected successfully: d:12
7fmg:12345:123456
Published gasleakage = 101 ppm to IBM Watson
Published gasleakage = 108 ppm to IBM Watson
Published gasleakage = 92 ppm to IBM Watson
Published gasleakage = 93 ppm to IBM Watson
Published gasleakage = 106 ppm to IBM Watson
Published gasleakage = 107 ppm to IBM Watson
Published gasleakage = 95 ppm to IBM Watson
```

### IBM Watson Connection :

The screenshot displays the IBM Watson IoT Platform interface. At the top, there's a navigation bar with tabs for 'Browse', 'Action', 'Device Types', and 'Interfaces'. The 'Recent Events' tab is selected, showing a table of device events. The table has four columns: 'Event', 'Value', 'Format', and 'Last Received'. It lists four events from an 'IoTSensor' device, each with a unique value and a timestamp of 'a few seconds ago'. The interface also includes a top search bar, a left sidebar with navigation icons, and a bottom status bar showing the date and time.

Event	Value	Format	Last Received
IoTSensor	{"gas_leak":106}	json	a few seconds ago
IoTSensor	{"gas_leak":93}	json	a few seconds ago
IoTSensor	{"gas_leak":92}	json	a few seconds ago
IoTSensor	{"gas_leak":108}	json	a few seconds ago



