

Develop a python script

Team ID	PNT2022TMID26515
Project Name	Smart waste management system for metropolitan cities

Step 1: Open python idle

Step2: Type the program Step

3: Then click on file and save the document

Step 4: Then click on Run then Run Module

Step 5: output will be appeared in the idle window

Python script

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

```
# watson device details
organization = "dr2sg4"
devicType = "PNT2022TMID26515"
```

```
deviceId = "963519104041"  
authMethod= "token"  
authToken= "963519104041"
```

```
#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,"authtoken":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:
    distance= random.randint(10,70)
    loadcell= random.randint(5,15)
    data= {'dist':distance,'load':loadcell}

    if loadcell < 13 and loadcell > 15:
        load = "90 %"

    elif loadcell < 8 and loadcell > 12:
        load = "60 %"

    elif loadcell < 4 and loadcell > 7:
        load = "40 %"

    else:
        load = "0 %"

    if distance < 15:
        dist = 'Risk warning:' 'Time to collect :)
90 %'

    elif distance < 40 and distance >16:
        dist = 'Risk warning:' 'above 60%'

    elif distance < 60 and distance > 41:
        dist = 'Risk warning:' '40 %'
```

else:

dist = 'Risk warning:' '17 %'

if load == "90 %" or distance == "90 %":

warn = 'alert :' 'Time to collect :')

elif load == "60 %" or distance == "60 %":

warn = 'alert :' 'above 60%'

else :

warn = 'alert :' 'No need to collect right
now '

def

myOnPublishCallback(lat=10.678991,long=7
8.177731):

print("Arunthenganvilai, kanyakumari")

print("published distance = %s "

%distance,"loadcell:%s " %loadcell,"lon = %s
" %long,"lat = %s" %lat)

print(load)

print(dist)

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=myCommandC  
allback
```

```
#disconnect the device
```

```
deviceCli.disconnect
```

Screenshots Python script:

The image displays three screenshots of a Python script, likely for simulating an IoT device and connecting it to a cloud platform like AWS IoT Core.

Top Left Screenshot: Shows the initial part of the script, including imports and the `deviceCli.connect()` function. The script defines variables for `distance`, `loadcell`, and `data`. It includes conditional logic for `load` and `distance` thresholds, and a `dist` variable for risk warnings.

Top Right Screenshot: Shows the `myCommandCallback(cmd)` function, which handles incoming commands. It includes a `try` block for processing the command and a `catch` block for handling exceptions. The script also defines `deviceOptions` and `deviceCli` for connecting to the cloud.

Bottom Left Screenshot: Shows the `myOnPublishCallback` function, which handles incoming data from the cloud. It includes a `try` block for processing the data and a `catch` block for handling exceptions. The script also includes a `time.sleep(10)` call and a `success` flag for the publish event.

Bottom Right Screenshot: Shows the `main` function, which calls `deviceCli.connect()` and `myCommandCallback`. It also includes a `time.sleep(10)` call and a `success` flag for the publish event.

