

DEVELOP A PYTHON SCRIPT

<i>Team ID</i>	<i>PNT2022TMID05128</i>
<i>Project Name</i>	<i>Smart waste management system for metropolitan cities</i>

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 = "4yi0vc"
devicType = "BIN1"
deviceId = "BIN1ID"
authMethod= "token"
authToken= "123456789"
```

```

#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:

    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:' 'Dumpster poundage getting high, Time to collect :) 90 %'
elif distance < 40 and distance >16:
    dist = 'Risk warning:' 'dumpster is 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 :' ' Dumpster poundage getting high, Time to collect :)'

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

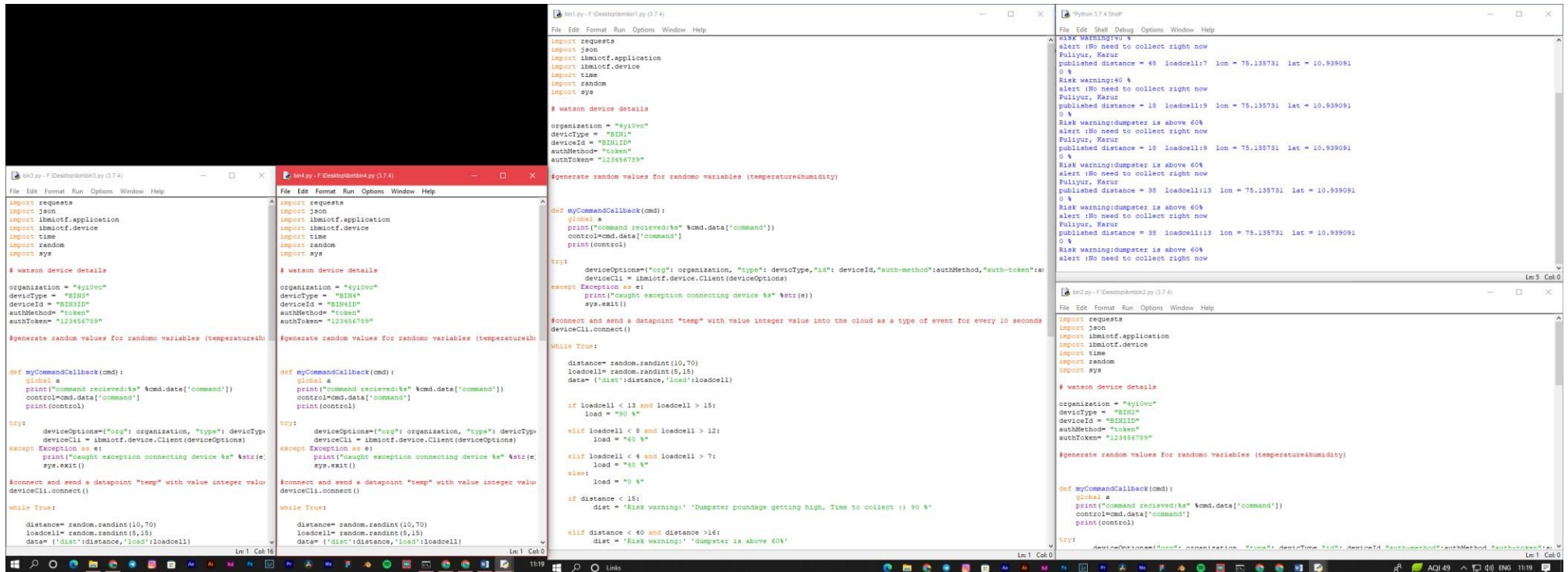
    warn = 'alert :' 'dumpster is above 60%'else :

    warn = 'alert :' 'No need to collect right now '
def myOnPublishCallback(lat=10.678991,long=78.177731):
    print("Gandigramam, Karur")
    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=myCommandCallback
#disconnect the device
deviceCli.disconnect

```

Screenshots Python script:



The image displays four screenshots of a Python script, likely for an IoT application using IBM Watson IoT. The script is written in Python 3.7.4 and is titled 'b04.py - F:\Desktop\bin\py (3.7.4)'. The code is organized into several sections: imports, device details, random variable generation, and a main loop for sending data points.

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

# Watson device details
organization = "4y10vc"
deviceType = "BIN1"
deviceId = "BIN1ID"
authMethod = "token"
authToken = "123456789"

#Generate random values for random variables (temperature, humidity)

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

try:
    deviceOptions={"org": organization, "type": deviceType, "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:
    distance= random.randint(10,70)
    loadcell= random.randint(5,15)
    data= {'dist': distance, 'load': loadcell}
```

The script also includes a section for generating random values for variables like temperature and humidity, and a main loop that sends data points to the cloud. The data points are generated using random values for distance and load cell. The script also includes a section for generating random values for variables like temperature and humidity, and a main loop that sends data points to the cloud. The data points are generated using random values for distance and load cell.

The script also includes a section for generating random values for variables like temperature and humidity, and a main loop that sends data points to the cloud. The data points are generated using random values for distance and load cell. The script also includes a section for generating random values for variables like temperature and humidity, and a main loop that sends data points to the cloud. The data points are generated using random values for distance and load cell.