Project title: environmental monitoring

**TEAM NAME:** Proj\_228486\_Team\_1

**STUDENT NAME:** R.PRIYADHARSHINI

**INTRODUCTION:**

Python scripts play a pivotal role in the realm of Environmental Monitoring in the Internet of Things (IoT). In this context, Python serves as a powerful and flexible tool for collecting, processing, and analyzing data from various sensors and devices.Python scripts are instrumental in interfacing with sensors, aggregating data from them, and subsequently transmitting it for analysis and visualization. With its ease of use, a wide range of libraries, and extensive community support, Python empowers developers to create sophisticated IoT solutions that aid in tracking, understanding, and responding to environmental changes, contributing to the betterment of our planet. This introduction sets the stage for exploring how Python scripts can be employed to build efficient and sustainable IoT systems for environmental monitoring.

**CODE:**

Creating a complete Python script for IoT devices to send real-time environmental data to a monitoring platform is a complex task, and it typically depends on the specific IoT hardware and platform you’re using. However, I can provide you with a high-level example script that you can adapt to your specific hardware and platform.

```python

Import time

Import random

Import requests

# Replace these with your IoT device’s credentials and endpoint

Device\_id = “your\_device\_id”

Api\_key = “your\_api\_key”

Monitoring\_endpoint = <https://your-monitoring-platform.com/api/data>

Def read\_environmental\_data():

# Replace this with your code to read environmental data from sensors

Temperature = random.uniform(20.0, 30.0)

Humidity = random.uniform(40.0, 60.0)

Return {“temperature”: temperature, “humidity”: humidity}

Def send\_data\_to\_monitoring\_platform(data):

Headers = {

“Authorization”: f”Bearer {api\_key}”

}

Try:

Response = requests.post(monitoring\_endpoint, json=data, headers=headers)

If response.status\_code == 200:

Print(“Data sent successfully”)

Else:

Print(f”Failed to send data. Status Code: {response.status\_code}”)

Except requests.exceptions.RequestException as e:

Print(f”Failed to send data. Error: {e}”)

If \_\_name\_\_ == “\_\_main\_\_”:

While True:

Environmental\_data = read\_environmental\_data()

Send\_data\_to\_monitoring\_platform({“device\_id”: device\_id, “data”: environmental\_data})

Time.sleep(60) # Adjust the interval as needed

```

Please note that you’ll need to replace the placeholder values for `device\_id`, `api\_key`, and `monitoring\_endpoint` with you’re actual IoT device’s credentials and the URL of your monitoring platform’s API.

This script generates random environmental data (temperature and humidity) for demonstration purposes. You should modify the `read\_environmental\_data` function to read data from your actual sensors.

**CONCLUSION:**

In conclusion, the Python script designed for environmental monitoring in IoT offers a powerful and versatile solution for gathering, processing, and analyzing data from various sensors and devices. This script not only provides real-time insights into environmental conditions but also facilitates data-driven decision-making for improved sustainability and resource management. Its flexibility, scalability, and ease of integration with IoT devices make it a valuable tool for monitoring air quality, temperature, humidity, and other environmental parameters. As the world continues to prioritize environmental conservation, this Python script plays a crucial role in harnessing the potential of IoT technology to create a greener, more sustainable future.