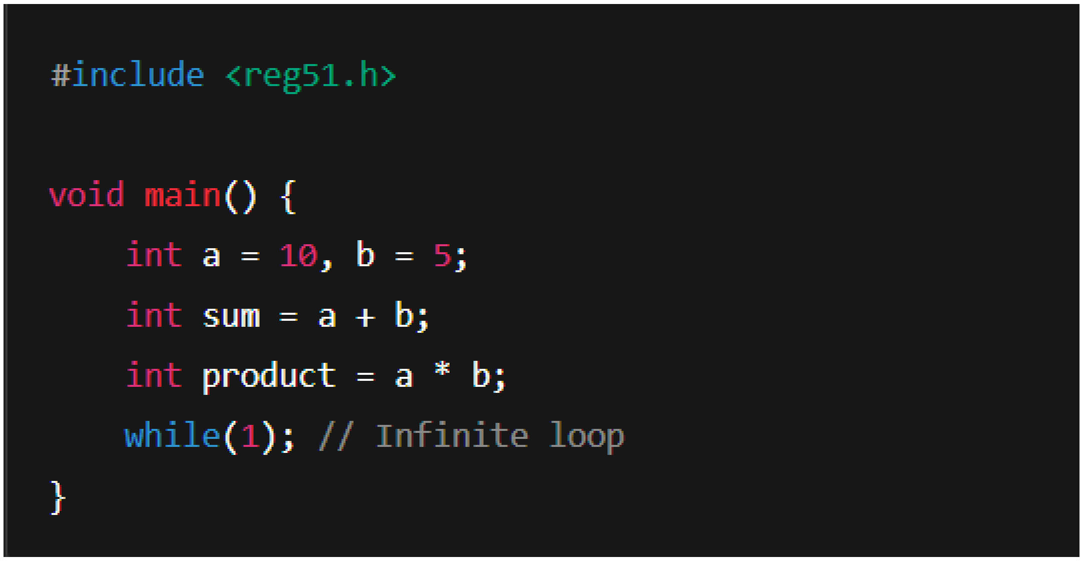
1. Addition and Multiplication using Embedded C

Aim:
To perform addition and multiplication of two numbers using Embedded C.

Procedure:

1. Open Keil IDE and create a new project.
2. Include the `reg51.h` header for 8051 microcontroller.
3. Declare two integer variables `a` and `b`.
4. Perform addition and multiplication operations.
5. Store the results in separate variables.

Program:



```
#include <reg51.h>

void main() {
    int a = 10, b = 5;
    int sum = a + b;
    int product = a * b;
    while(1); // Infinite loop
}
```

Output:
Sum = 15, Product = 50

Result:
Addition and multiplication performed successfully.

---

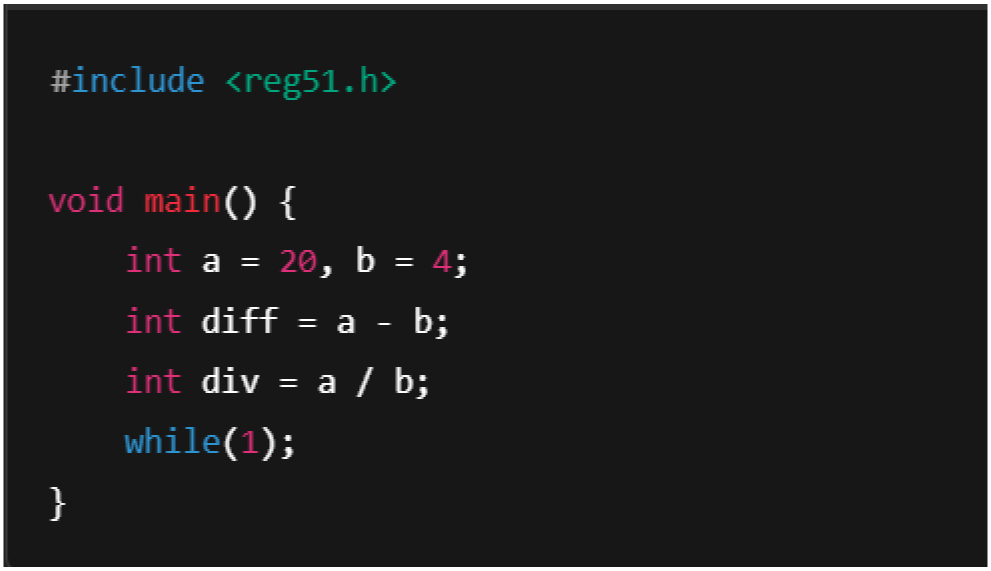
2. Subtraction and Division using Embedded C

Aim:
To perform subtraction and division using Embedded C.

Procedure:

1. Open Keil IDE and start a new C project.
2. Include the necessary header file.
3. Declare two integers and initialize them.
4. Use `-` for subtraction and `/` for division.
5. Store and monitor the result.

Program:



```c
#include <reg51.h>

void main() {
    int a = 20, b = 4;
    int diff = a - b;
    int div = a / b;
    while(1);
}
```

Output:
Difference = 16, Division = 5

Result:
Subtraction and division performed successfully.

---

3. ALU Operations using Embedded C

Aim:
To implement basic ALU operations (Add, Sub, And, Or) in Embedded C.

Procedure:

1. Start a new project in Keil IDE.

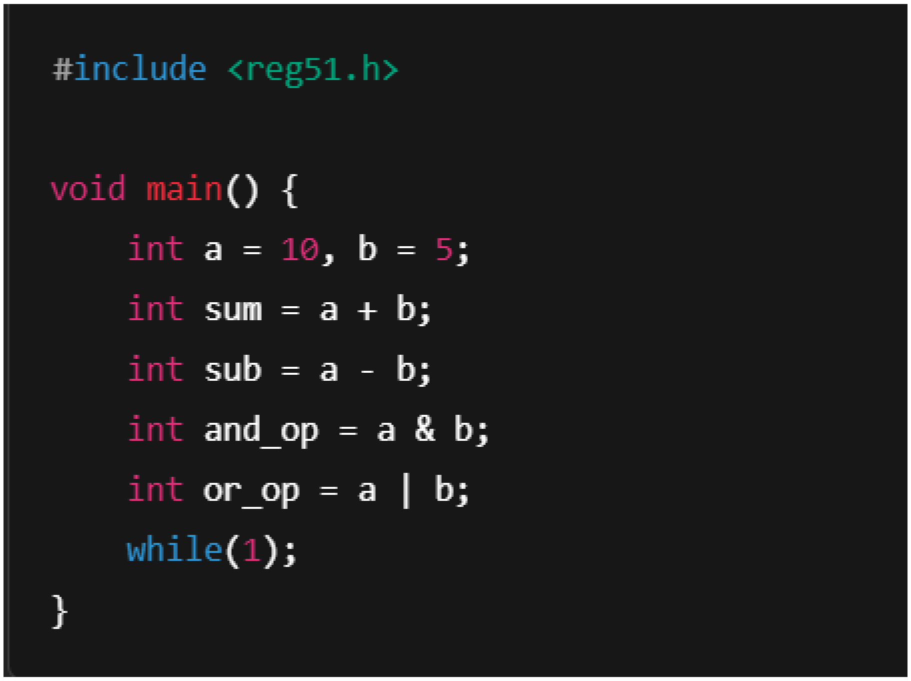
2. Include `reg51.h`.

3. Initialize variables `a` and `b`.

4. Perform +, -, &, | operations.

5. Save results in separate variables.

Program:



```c
#include <reg51.h>

void main() {
    int a = 10, b = 5;
    int sum = a + b;
    int sub = a - b;
    int and_op = a & b;
    int or_op = a | b;
    while(1);
}
```

Output:

Addition = 15, Sub = 5, AND = 0, OR = 15

Result:

ALU operations done using Embedded C.

---

4. Logical Operations using Embedded C

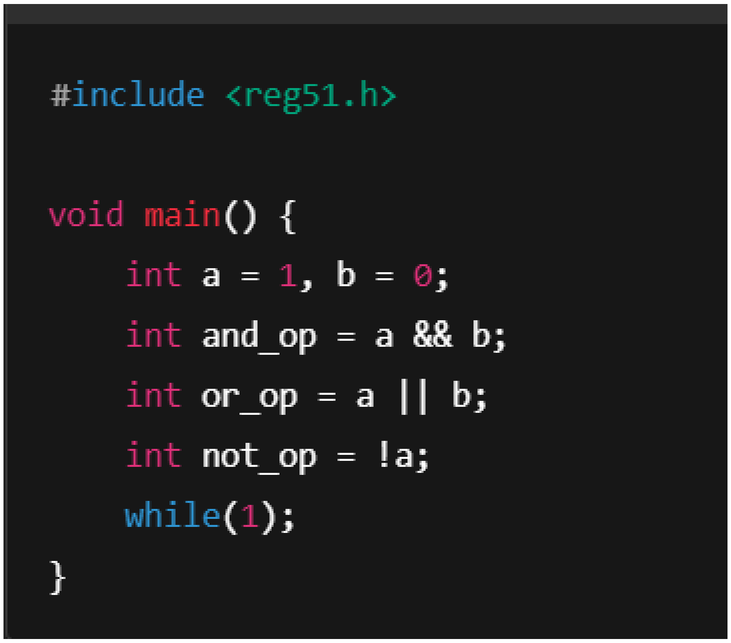
Aim:

To perform logical operations (AND, OR, NOT) using Embedded C.

Procedure:

1. Create a new C project in Keil IDE.
2. Include the standard 8051 header file.
3. Declare two logic variables.
4. Perform AND, OR, and NOT operations.
5. Store output in variables

Program:



```c
#include <reg51.h>

void main() {
    int a = 1, b = 0;
    int and_op = a && b;
    int or_op = a || b;
    int not_op = !a;
    while(1);
}
```

Output:

AND = 0, OR = 1, NOT = 0

Result:

Logical operations done successfully.

---

5. Comparison Operations using Embedded C

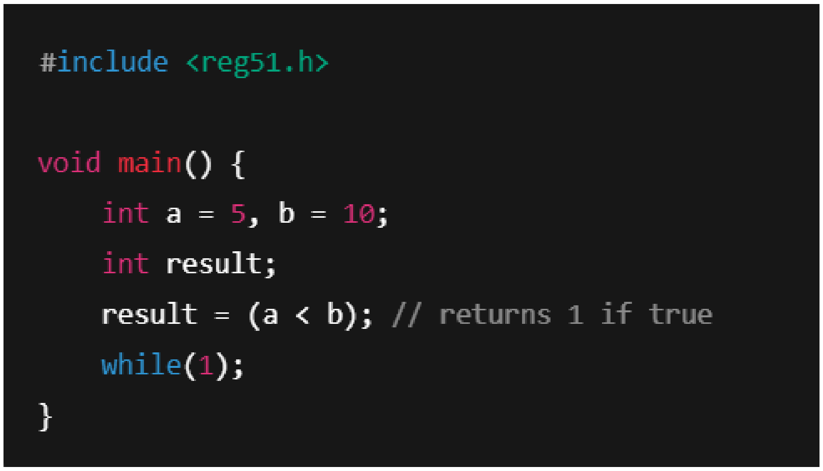
Aim:

To compare two numbers using relational operators.

Procedure:

1. Create a new Embedded C program.
2. Define and initialize two numbers.
3. Use relational operators like `<`, `>`, `==`.
4. Store result in a variable or use `if` statements.
5. Check output using a debugger or simulator.

Program:



```c
#include <reg51.h>

void main() {
    int a = 5, b = 10;
    int result;
    result = (a < b); // returns 1 if true
    while(1);
}
```

Output:

Result = 1 (true)

Result:

Comparison done successfully.

6. Arithmetic Operations in 8051 using Simulator

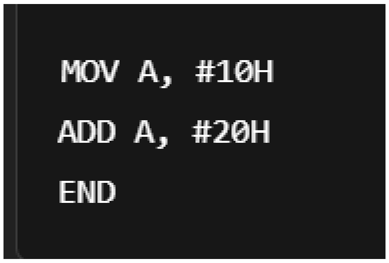
Aim:

To perform arithmetic operations on 8051 microcontroller using a simulator.

Procedure:

1. Open Keil IDE with 8051 configuration.
2. Write assembly code to add two numbers.
3. Use `MOV` to load values and `ADD` to perform operation.
4. Use `A` register to store the result.
5. Simulate to view the result in registers.

Program (Assembly - Addition):



```
MOV A, #10H
ADD A, #20H
END
```

Output:
A = 30H

Result:
Arithmetic operation performed in 8051.

---

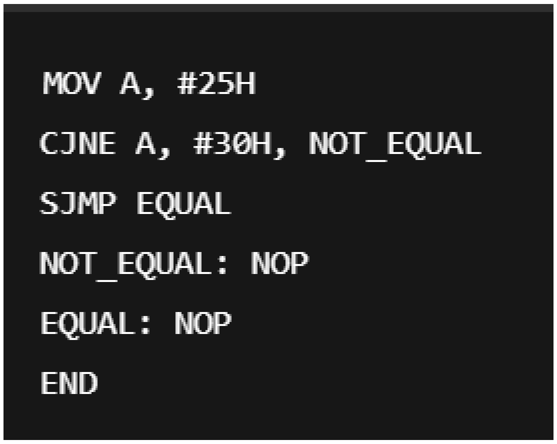
7. Compare Two Numbers in 8051 using Simulator

Aim:
To compare two numbers in 8051 and set flags accordingly.

Procedure:

1. Open Keil and create an 8051 assembly file.
2. Load first number into `A`.
3. Use `CJNE` instruction to compare.
4. Jump to corresponding label if not equal.
5. Use `NOP` or label to verify output in simulation.

Program (Assembly):



```
MOV A, #25H
CJNE A, #30H, NOT_EQUAL
SJMP EQUAL
NOT_EQUAL: NOP
EQUAL: NOP
END
```

Output:
A ≠ 30H → Jumps to NOT_EQUAL

Result:
Comparison executed correctly.

---

8. Transfer Data Between Two Registers

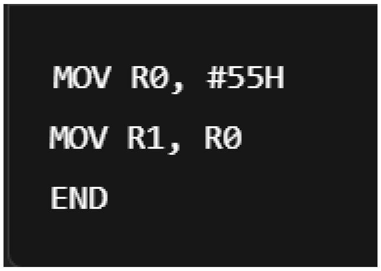
Aim:
To transfer data between two registers in 8051.

Procedure:

1. Write a program in 8051 assembly.
2. Use `MOV` to load data into R0.
3. Transfer data from R0 to R1.
4. Observe registers in simulator.

5. Ensure both values match

Program (Assembly):



```
MOV R0, #55H
MOV R1, R0
END
```

Output:
R1 = 55H

Result:
Data transferred between registers.

---

9. Transfer Data Between Memory and Register

Aim:
To move data between memory and a register.

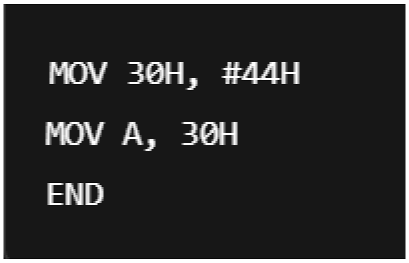
Procedure:

1. Open Keil and create a new 8051 assembly project.
2. Store a value at memory location 30H.
3. Use `MOV A, 30H` to load it into accumulator.

4. Verify memory and register values.

5. Simulate to check correctness

Program (Assembly):



```
MOV 30H, #44H
MOV A, 30H
END
```

Output:
A = 44H

Result:
Data moved between memory and register.

---

10. Arduino Platform and Sample Program

Aim:
To explain Arduino platform and demonstrate a simple LED blink program.

Procedure:

1. Install Arduino IDE on your computer.

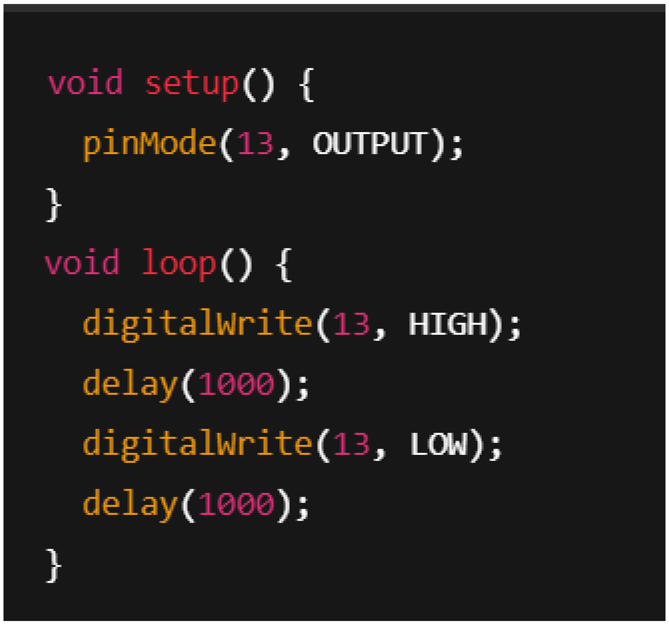
2. Connect Arduino board via USB.

3. Open IDE and write LED blink code.

4. Upload code using the upload button.

5. Observe LED blinking every second.

Program:



```
void setup() {
  pinMode(13, OUTPUT);
}
void loop() {
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

Output:
LED blinks every second.

Result:
Arduino platform and program working successfully.

---

11. Python Program for Data Transfer with Arduino

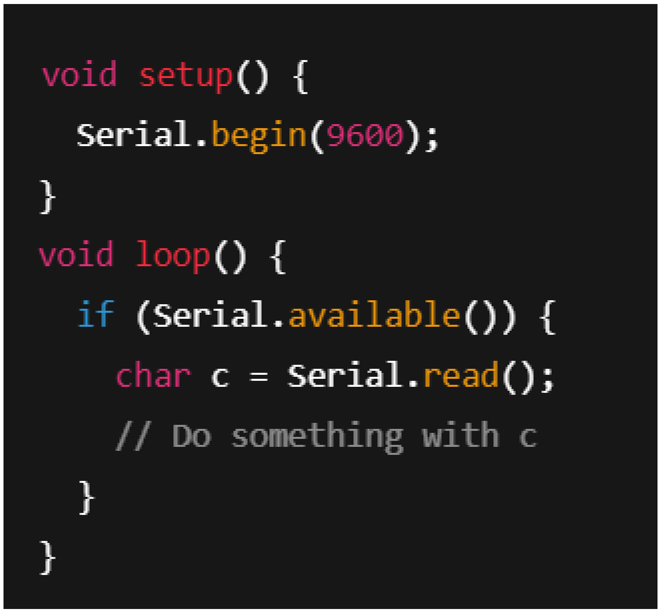
Aim:

To send data from Python to Arduino using serial communication.

Procedure:

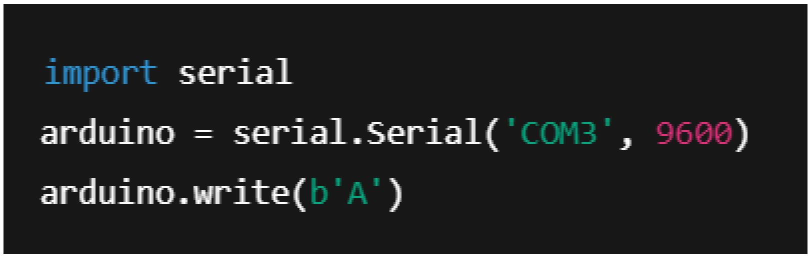
1. Connect Arduino to PC via USB.
2. Upload serial reading code to Arduino.
3. Install `pyserial` library in Python.
4. Open serial port using `serial.Serial()`.
5. Send data from Python to Arduino.

Arduino Code:



```
void setup() {
  Serial.begin(9600);
}
void loop() {
  if (Serial.available()) {
    char c = Serial.read();
    // Do something with c
  }
}
```

Python Code:



```python
import serial
arduino = serial.Serial('COM3', 9600)
arduino.write(b'A')
```

Output:
Character sent to Arduino.

Result:
Data transfer successful.

---

12. Arduino-Raspberry Pi Bluetooth Communication

Aim:
To establish Bluetooth communication between Arduino and Raspberry Pi.

Procedure:

1. Connect HC-05 Bluetooth module to Arduino.

2. Pair Bluetooth module with Raspberry Pi.

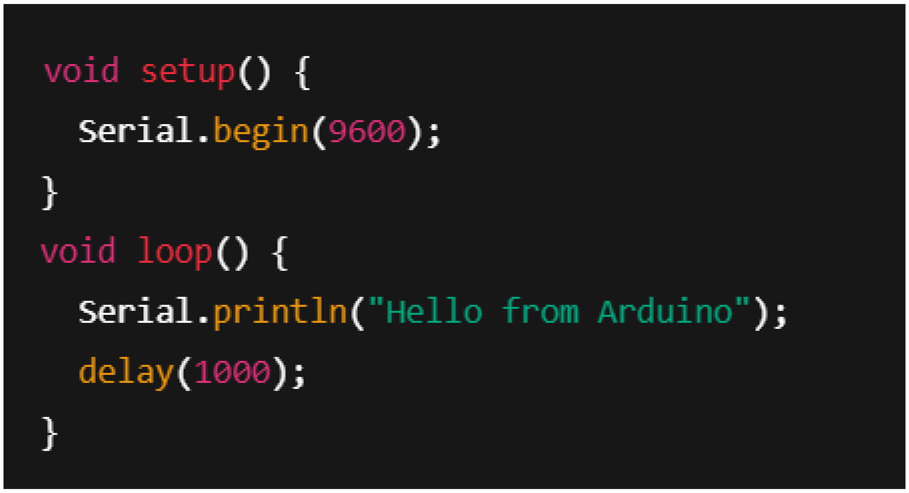
3. Upload serial print code to Arduino.

4. Use Python on Pi to read from `/dev/rfcomm0`.

5. Display received data on Pi terminal.

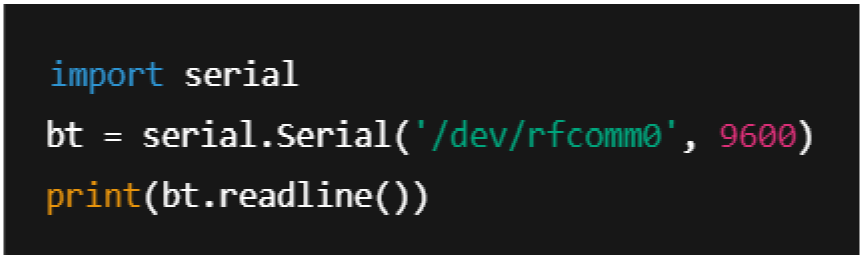
.

Arduino Code:



```
void setup() {
  Serial.begin(9600);
}
void loop() {
  Serial.println("Hello from Arduino");
  delay(1000);
}
```

Raspberry Pi Python Code:



```
import serial
bt = serial.Serial('/dev/rfcomm0', 9600)
print(bt.readline())
```

Output:
Pi receives message: *Hello from Arduino*

Result:
Bluetooth communication successful.

---

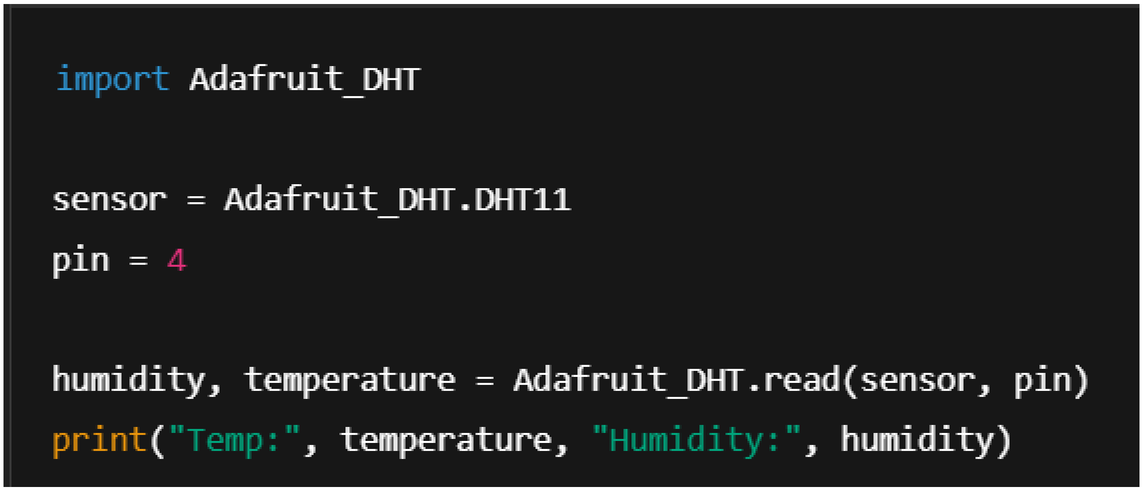
13. Sensor Interfacing with Raspberry Pi

Aim:
To interface a DHT11 temperature sensor with Raspberry Pi.

Procedure:

1. Connect DHT11 sensor to GPIO pin.
2. Install `Adafruit_DHT` library.
3. Write Python script to read from the sensor.
4. Run script using terminal.
5. Print temperature and humidity

Python Code:



```
import Adafruit_DHT

sensor = Adafruit_DHT.DHT11
pin = 4

humidity, temperature = Adafruit_DHT.read(sensor, pin)
print("Temp:", temperature, "Humidity:", humidity)
```

Output:
Displays temperature and humidity.

Result:
Sensor interfaced with Raspberry Pi successfully.

---

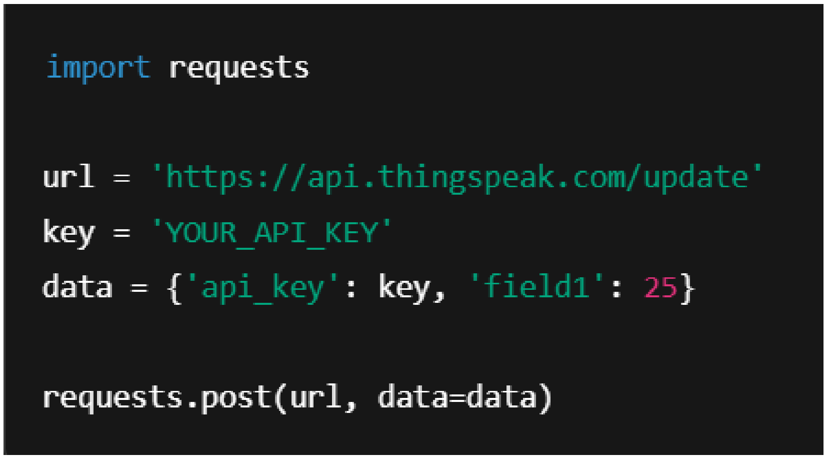
14. Setup Cloud Platform to Log Data

Aim:
To log sensor data to a cloud platform (like ThingSpeak).

Procedure:

1. Create a ThingSpeak account and get an API key.
2. Write a Python script to send data using POST.
3. Use `requests` module to call API.
4. Send sensor or dummy data.
5. View data on ThingSpeak channel.

Python Code (with sensor):



```
import requests

url = 'https://api.thingspeak.com/update'
key = 'YOUR_API_KEY'
data = {'api_key': key, 'field1': 25}

requests.post(url, data=data)
```

Output:
Data uploaded to cloud.

Result:
Cloud platform setup and data logged.

---

15. Log Data on Raspberry Pi and Upload to Cloud

Aim:
To collect sensor data on Pi and send it to the cloud.

Procedure:

1. Read data from sensor using Python.

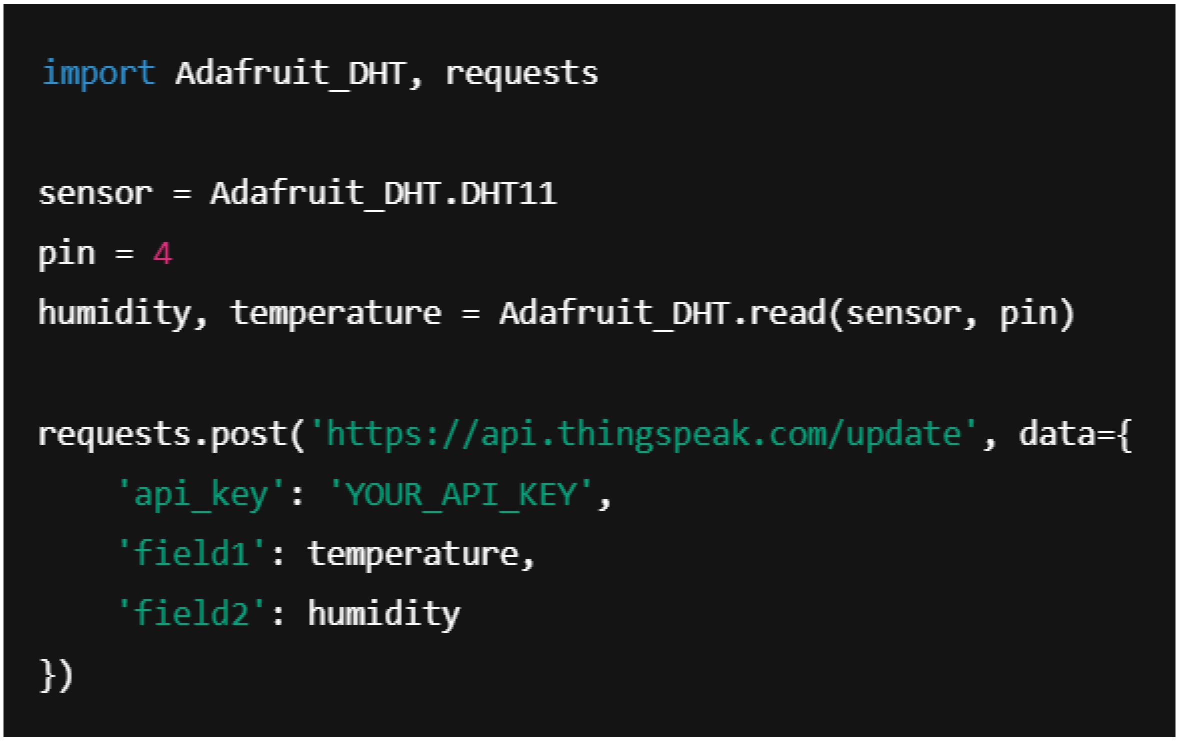
2. Store it in variables.

3. Use `requests.post()` to send to cloud.

4. Include API key and field data.

5. Monitor data on cloud platform..

Python Code:



```
import Adafruit_DHT, requests

sensor = Adafruit_DHT.DHT11
pin = 4
humidity, temperature = Adafruit_DHT.read(sensor, pin)

requests.post('https://api.thingspeak.com/update', data={
    'api_key': 'YOUR_API_KEY',
    'field1': temperature,
    'field2': humidity
})
```

Output:
Data logged and uploaded.

Result:

Data logged on Pi and sent to cloud.