

Introduction:

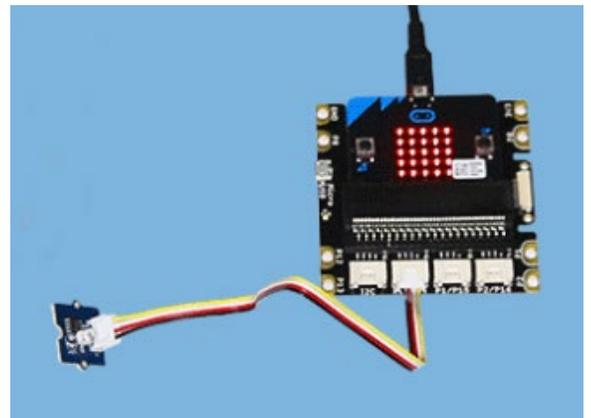
The micro:bit has an internal temperature sensor in its CPU. In this activity you will use an external temperature sensor to monitor the temperature away from the micro:bit itself to get a more accurate reading. The activity is divided into two parts:

Part 1: the Weather Face

Part 2: Tracking Temperature

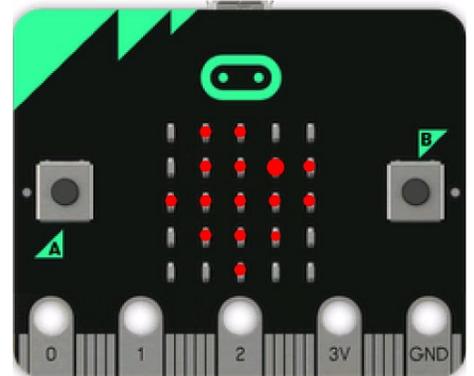
These activities assume you have completed the *first* set of micro:bit activities and will be using an external Grove temperature sensor.

1. Before you begin, be sure that:
 - your **micro:bit** is connected to your TI-Nspire CX II
 - your **micro:bit** is inserted in the **expansion board**.
(*Grove shield shown here*)
 - the Grove temperature sensor is attached to pin0.



2. Your micro:bit should look like this when it has power from the TI-Nspire:
Recall that the display on the **micro:bit** is showing the TI logo, an icon of the state of Texas with a bright spot near Dallas, the home of **Texas Instruments, Inc.**

This activity has two parts: a temperature monitoring application and a temperature tracking program.



10 Minutes of Code: Python Modules

MICRO:BIT: WEATHER FACE

TI-NSPIRE™ CX II

- Use a Grove temperature sensor: Connect the sensor board to the port labeled **P0/P14** (pin0) on the expansion board using the included cable. You can use a different port if you also change the pin variable in the code to match.



4. Part 1: WeatherFace

Some indoor/outdoor thermometers have an image of a person indicating the current weather conditions. This program will display an appropriate 'face' on the micro:bit display that will depend on the current temperature.

In the Python Editor use [menu] > **More Modules** > **BBC micro:bit** to select the **import** statement at the top of the menu items:

from microbit import *

Tip: If the message 'micro:bit not connected' ever appears, just unplug the micro:bit and plug it in again (reset).

- Write our favorite loop:

while get_key() != "esc":

It's found in

[menu] > **More Modules** > **BBC micro:bit** > **Commands** >

- In the loop body, read the Grove temperature sensor using the function found on

[menu] > **More Modules** > **BBC micro:bit** > **Grove devices** > **Input**

var = grove.read_temperature(pin)

Use any variable on the left of the = sign for your temperature variable. For the (pin) argument, select **pin0** from the pop-up list (or whatever socket your sensor is plugged into on the expansion board).

```
1.1 | *Doc | RAD | [Battery Icon] | X  
greetings.py saved successfully  
from microbit import *  
|
```

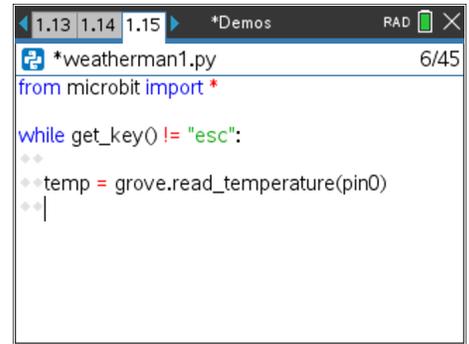
```
1.13 | 1.14 | 1.15 | *Demos | RAD | [Battery Icon] | X | 4/38  
*weatherman1.py  
from microbit import *  
while get_key() != "esc":  
  |
```

```
1.5 | 1.6 | 1.7 | *Python...OS6 | RAD | [Battery Icon] | X | 5/5  
*weatherman1.py  
from microbit import *  
while get_key() != "esc":  
  |  
  var = grove.read_temperature(pin)
```

10 Minutes of Code: Python Modules

TI-NSPIRE™ CX II

- In the image to the right, we used the variable **temp** to hold the temperature and **pin0** for the argument.



```
1.13 1.14 1.15 *Demos RAD 6/45
*weatherman1.py
from microbit import *

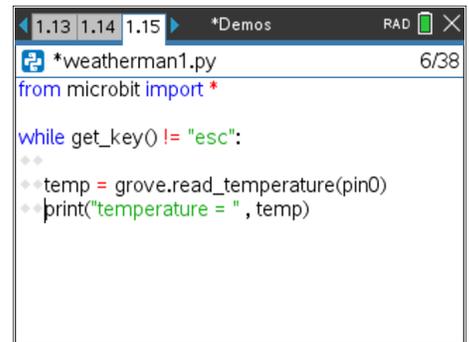
while get_key() != "esc":
    temp = grove.read_temperature(pin0)
```

- Add a **print** statement to display the temperature to be sure that everything is working:

print("temperature = ",temp)

Run the program now to see temperature values displayed on the TI-Nspire™.

If you think the numbers displayed are strange, keep in mind that the temperature is displayed in the Celsius scale. If you prefer to work with the Fahrenheit scale, then you will have to do the conversion yourself in the program.



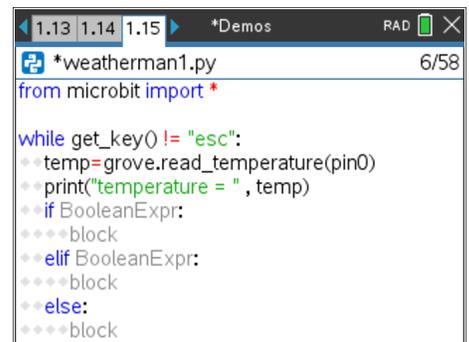
```
1.13 1.14 1.15 *Demos RAD 6/38
*weatherman1.py
from microbit import *

while get_key() != "esc":
    temp = grove.read_temperature(pin0)
    print("temperature = ", temp)
```

- Our program will display three different faces depending on the temperature (cold, mild, or hot). This is an ideal place for an **if...elif...else** structure. Get this structure from

[menu] > Built-ins > Control

All three components are inserted into your code, and you must replace the placeholders with conditions (**BooleanExp**) and actions (**block**).



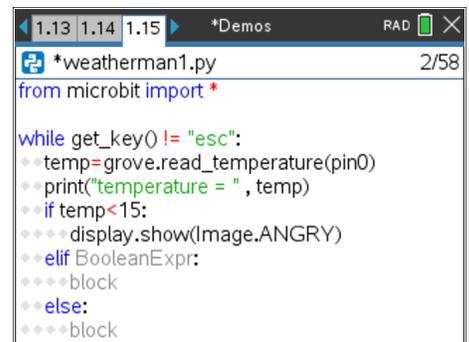
```
1.13 1.14 1.15 *Demos RAD 6/58
*weatherman1.py
from microbit import *

while get_key() != "esc":
    temp=grove.read_temperature(pin0)
    print("temperature = ", temp)
    if BooleanExp:
        block
    elif BooleanExp:
        block
    else:
        block
```

- The **if... block** will handle the 'cold' temperature. When it's cold, we'll display the ANGRY face:

if temp < 15:
display.show(Image.ANGRY)

Find **display.show()** on the **BBC microbit > Display** menu and select **ANGRY** from the pop-up list of images.



```
1.13 1.14 1.15 *Demos RAD 2/58
*weatherman1.py
from microbit import *

while get_key() != "esc":
    temp=grove.read_temperature(pin0)
    print("temperature = ", temp)
    if temp<15:
        display.show(Image.ANGRY)
    elif BooleanExp:
        block
    else:
        block
```

11. The **elif** : block also requires a condition since it stands for 'else if...'. This time we check for 'comfortable' temperatures between 15 and 27 degrees Celsius. But the first condition took care of the lower bound so we only need to check the upper bound:

elif temp < 27 :

and we can display the HAPPY face:

display.show(Image.HAPPY)

12. What face would you like to show when it's 'hot'? Add the statement to display another face in the **else: block**.

To test your program, you will have to subject the temperature sensor to a range of temperatures. You can place it in a refrigerator for a cool temperature but be careful when looking for a warm temperature. A hot day outside is OK but do not place the sensor in an oven or microwave and *do not get it wet*. Try rubbing your hands together (feel the heat?) and then place the sensor between your hands.

Can you see all three faces on the micro:bit?

Now try this...

13. **Part 2: Tracking Temperature**

Let's monitor the temperature for a while. Use two lists to collect time and temperature data and transfer the lists to the TI-Nspire CX for further study.

Make a copy of your weatheman1.py program (or whatever you called it).

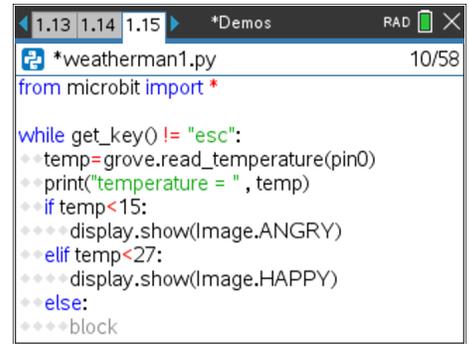
14. In your new program, before the **while** loop, create two empty lists

times = []

temps = []

and a timer variable. Call it **tim** so that the variable does not interfere with the **time** module:

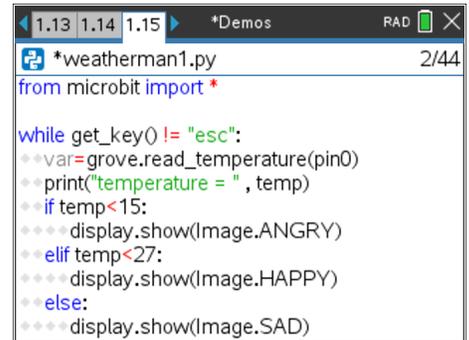
tim = 0



```

1.13 1.14 1.15 *Demos RAD X
*weatheman1.py 10/58
from microbit import *

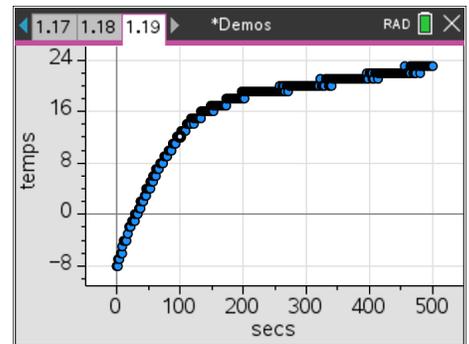
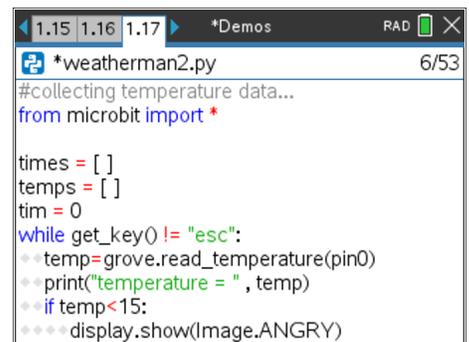
while get_key() != "esc":
    temp=grove.read_temperature(pin0)
    print("temperature = ", temp)
    if temp<15:
        display.show(Image.ANGRY)
    elif temp<27:
        display.show(Image.HAPPY)
    else:
        block
    
```



```

1.13 1.14 1.15 *Demos RAD X
*weatheman1.py 2/44
from microbit import *

while get_key() != "esc":
    var=grove.read_temperature(pin0)
    print("temperature = ", temp)
    if temp<15:
        display.show(Image.ANGRY)
    elif temp<27:
        display.show(Image.HAPPY)
    else:
        display.show(Image.SAD)
    
```

```

1.15 1.16 1.17 *Demos RAD X
*weatheman2.py 6/53
#collecting temperature data...
from microbit import *

times = [ ]
temps = [ ]
tim = 0
while get_key() != "esc":
    temp=grove.read_temperature(pin0)
    print("temperature = ", temp)
    if temp<15:
        display.show(Image.ANGRY)
    
```

10 Minutes of Code: Python Modules

TI-NSPIRE™ CX II

15. Below the **print** statement, add (append) the current values of **tim** and **temp** to their respective lists:

```
times.append(tim)  
temps.append(temp)
```

16. At the bottom of the while loop (after the else: block) add two statements:

```
sleep(1000)    optional!  
tim += 1      increment the tim variable representing one sample.
```

These two statements are indented to be part of the while loop block.

Note about timing: since it takes time to read the temperature sensor, to print the temperature, and to display the face image, the sleep(1000) pause is too large. So the tim variable is really just a sample counter. If you want the tim variable to actually represent seconds there are more accurate methods using the internal clock (time functions).

17. When the **while** loop ends, transfer the two lists to two TI-Nspire™ lists using the function **store_list()**. This function is included in the BBC micro:bit module and can be found on the > **BBC micro:bit >**

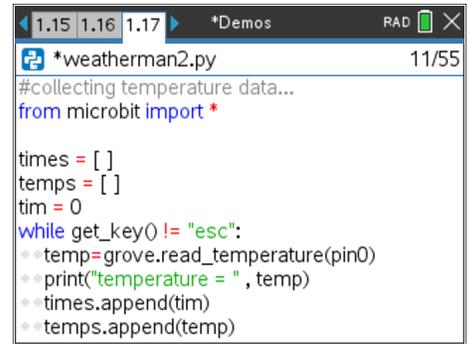
Commands menu.

You need two **store_list(,)** functions:

```
store_list("times",times)  
store_list("temps", temps)
```

We can use the same list names as the python variable names but the TI-nspire™ names must be in quotes.

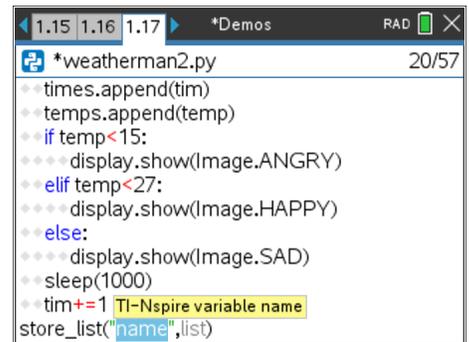
MICRO:BIT: WEATHER FACE



```
1.15 1.16 1.17 *Demos RAD X  
*weatherman2.py 11/55  
#collecting temperature data...  
from microbit import *  
  
times = []  
temps = []  
tim = 0  
while get_key() != "esc":  
    temp=grove.read_temperature(pin0)  
    print("temperature = ", temp)  
    times.append(tim)  
    temps.append(temp)
```



```
1.15 1.16 1.17 *Demos RAD X  
*weatherman2.py 19/57  
    print("temperature = ", temp)  
    times.append(tim)  
    temps.append(temp)  
    if temp<15:  
        display.show(Image.ANGRY)  
    elif temp<27:  
        display.show(Image.HAPPY)  
    else:  
        display.show(Image.SAD)  
    sleep(1000)  
    tim+=1
```



```
1.15 1.16 1.17 *Demos RAD X  
*weatherman2.py 20/57  
    times.append(tim)  
    temps.append(temp)  
    if temp<15:  
        display.show(Image.ANGRY)  
    elif temp<27:  
        display.show(Image.HAPPY)  
    else:  
        display.show(Image.SAD)  
    sleep(1000)  
    tim+=1 TI-Nspire variable name  
store_list("name",list)
```

10 Minutes of Code: Python Modules

TI-NSPIRE™ CX II

18. After running your program for a while, when pressing **[esc]**, the two lists are stored in the TI-Nspire™. Add a Data & Statistics app to your document and set up a scatter plot of the two lists by clicking the prompts on the bottom and left side of the screen. You will also have to adjust the viewing window, too.

The plot seen here was created by putting the temperature sensor in a freezer for a few minutes to get it really cold (-8C) then, taking the sensor out of the freezer, running the program to watch it warm up to room temperature (24C).

How do you explain this behavior?

19. **Challenge:** you can also use `tiplotlib` in your program to plot the data as it is being collected. If you have not yet used `tiplotlib`, see the `tiplotlib` activities in the **[+] Python Modules** section of this site: <https://education.ti.com/en/activities/ti-codes/python/ti-nspire-cx-ii/python-modules>

The plot seen here was created by placing the sensor in a freezer for a while.

MICRO:BIT: WEATHER FACE

