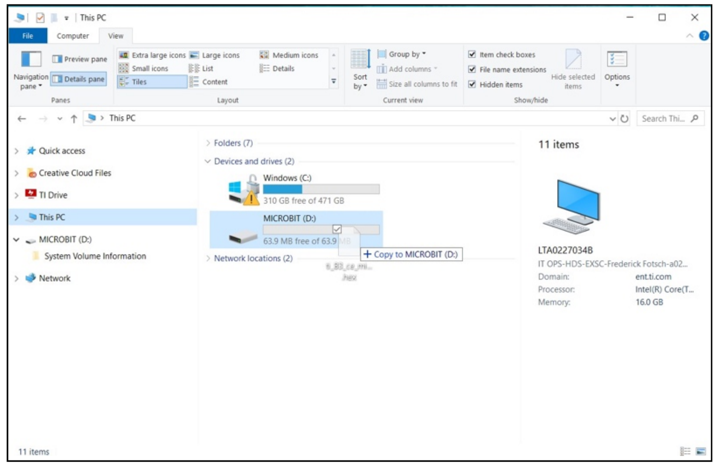
Getting Started Guide

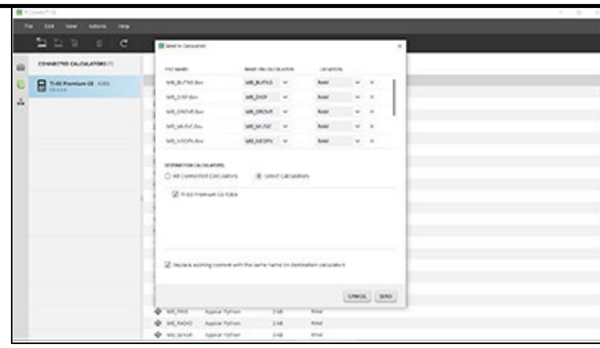
Installation

1. Use a USB cable to connect the micro:bit to a computer. The micro:bit will appear as a drive on your computer. Drag and drop the TI_Runtime_for_microbit.hex file to the micro:bit drive. The file will copy to the micro:bit and provide enhanced functionality between the TI-84 Plus CE-T PYTHON EDITION and the micro:bit.

   The runtime installation is an onetime process. Ifthe micro:bit is connected back to the PC and programmed in a different language other than python, such as MakeCode, JavaScript, C++, or Scratch, the ti_ce_runtime.hex will need to be installed again. See additional information at the end of the document.

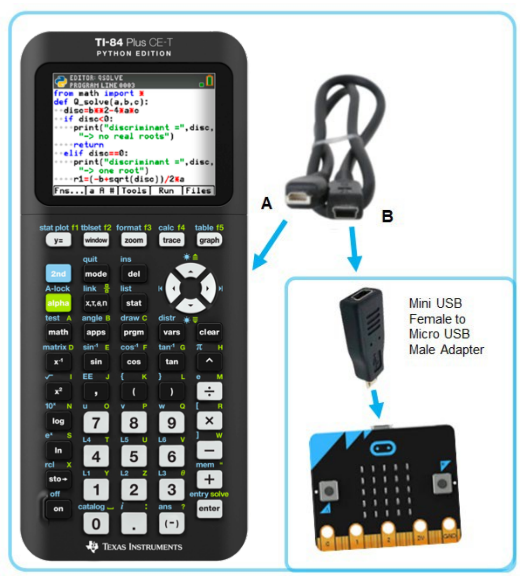


2. Use TI-Connect to transfer all of the micro:bit modules from your computer to the calculator.

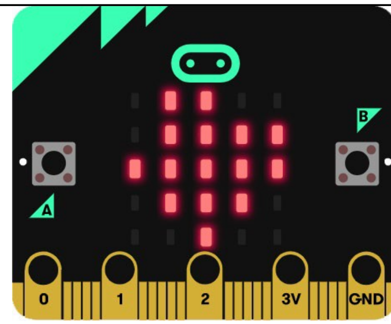


3. Connect the micro:bit to the TI-84 Plus CE-T PYTHON EDITION calculator using the unit-to-unit cableand a mini USB female to micro USB male adapter.

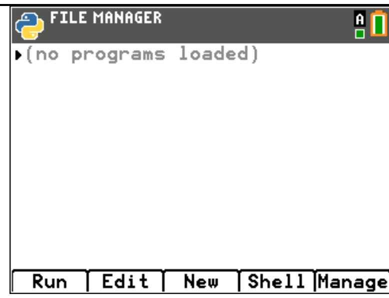
   Connect the adapter to the B-end of the cable, and plug the A-end into the calculator's USB port.

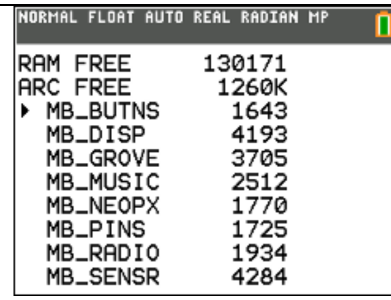


4. Press the reset button on the micro:bit.
   If the runtime has loaded successfully, the message TI-84 Plus CE-T PYTHON EDITION will scroll across the micro:bit LED display, followed by the TI logo.



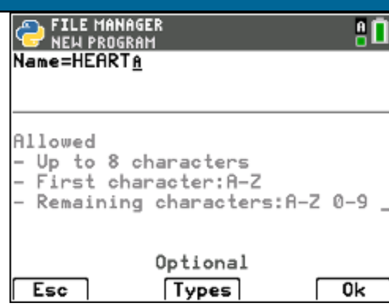
5. Open the python application. The micro:bit modules do not appear in the file manager; they are stored in archive memory.



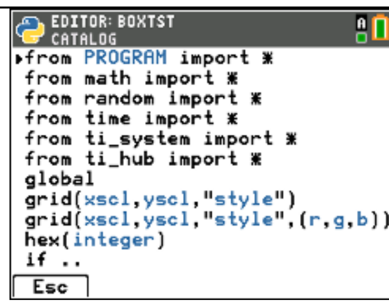


Creating Your First micro:bit Program

1. Select [New] to create a script with an appropriate name.

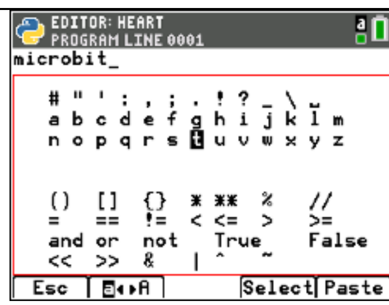


2. Select [2nd][catalog] and choose **from PROGAM import ***

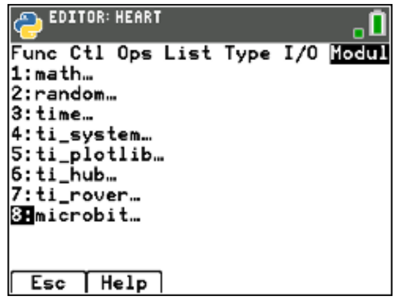


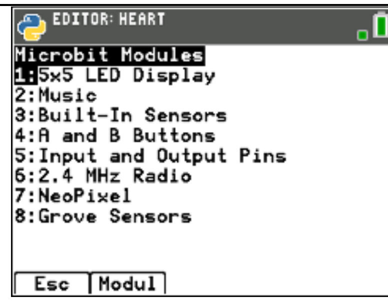
3. Position cursor after **from** and enter microbit.
   Select [a A #] and select then paste **microbit** into the **from _ import *** statement.

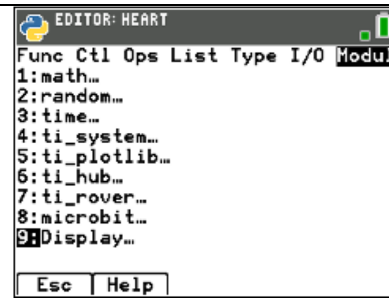
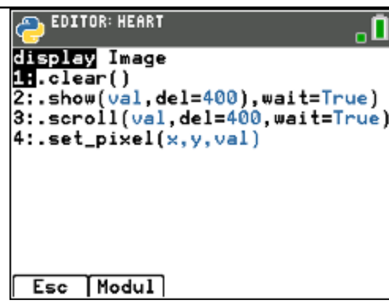
   Alternatively, use the alpha keys.

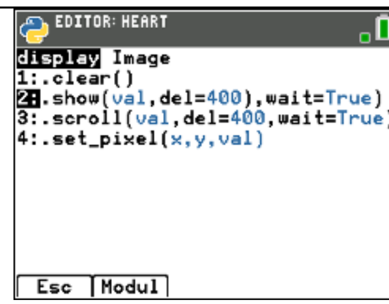


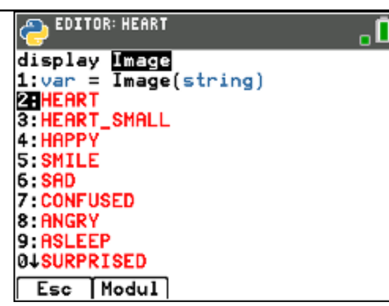
<table>
<tr>
<td>4. Select [enter] to go to the next line in the script after the import statement. This action will import the module into the editor and add the microbit menu option to the bottom of the module list.

Select[Fns...] and select Modul tab and press [enter]. The menu will display all of the microbit modules.</td>
<td>
</td>
</tr>
<tr>
<td>5. Select the 5x5 LED Display module needed for the new script. The import for the selected module is added to the script. If additional modules are required, return to this microbit menu to add them.

Choose only modules that are essential to the script to conserve RAM. You may choose other modules to include except the TI-Innovator Hub.</td>
<td>
</td>
</tr>
<tr>
<td>6. The module named mb_disp is added to the script. Select the [Fns] key and select Modul. Notice the new Display... menu item is now at the bottom of the list. All of the other microbit modules work in this way.</td>
<td>


</td>
</tr>
<tr>
<td>7. Select Display... and then
display.show(val,del=400,wait=True)
Ensure the cursor is positioned to the left of the comma in the command.
display.show(_ ,del=400,wait=True)</td>
<td>

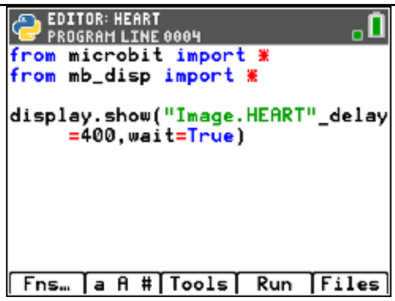

</td>
</tr>
<tr>
<td>8. Select the [Fns] key and select Modul. Select the Display... and then Image.
Select HEART to paste “Image.HEART” into the script.</td>
<td>

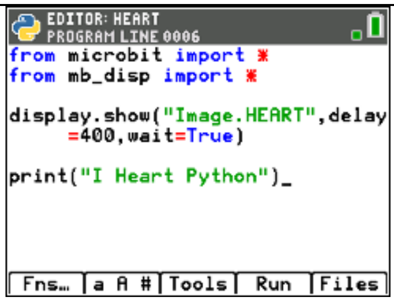

</td>
</tr>
</table>

Getting Started Guide

9. Select [Fns...], and I/O then print(). Type in "I Heart Python".

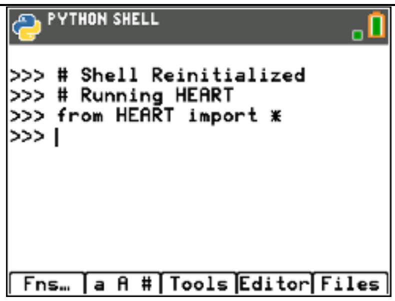
   The print() statement illustrates how microbitstatements are used along with standard or TImodule python statements within the same script.

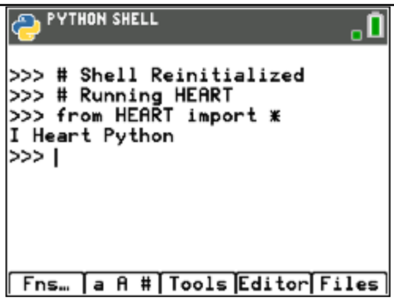


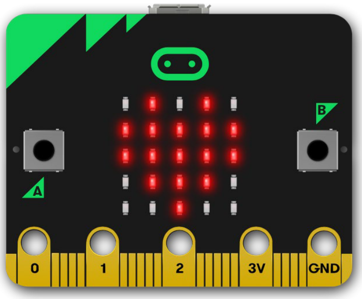


10. Be sure the micro:bit is connected to the calculator using the unit-to-unit cable and mini->micro adapter and then select [Run].

    Congratulations! You have programed the micro:bit with the TI 84 Plus CE-T PYTHON EDITION!



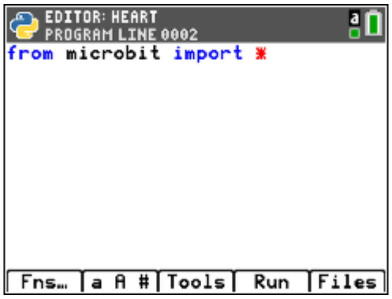


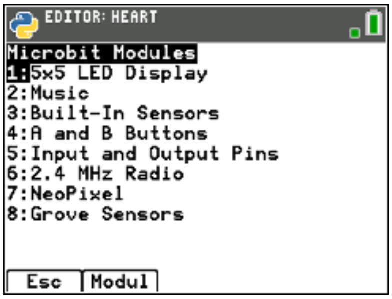


Modules and Methods

Microbit Modules

5x5 LED Display - imports **mb_music**
Music - imports **mb_music**
Built-In Sensors - imports **mb_sensr**
A and B Buttons - imports **mb_butns**
Input and Output Pins - imports **mb_pins**
2.4 MHz Radio - imports **mb_radio**
NeoPixel - imports **mb_neopx**
Grove Sensors - imports **mb_grove**





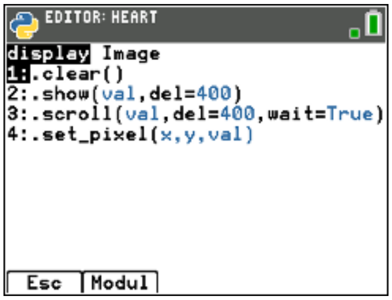
5x5 LED Display Module

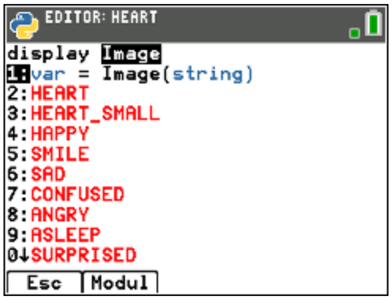
display

display.clear()
display.show(value,delay=400,wait=True)
display.scroll(value,delay=400,wait=True)
display.setpixel(x,y,value)

Image

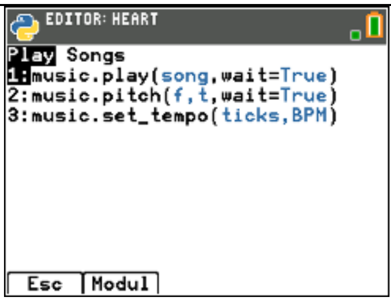
var=Image(string)
Image.NAME





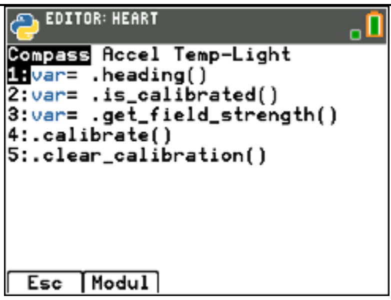
Music Module

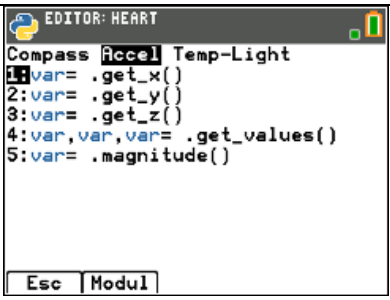
music.play(value)
music.pitch(frequency, duration, wait = 400)
music.set_tempo(ticks,BPM)

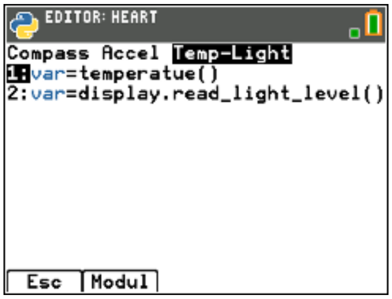


Built-In Sensors Module

var = accelerometer.get_x()
var = accelerometer.get_y()
var = accelerometer.get_z()
var,var,var = accelerometer.get_values()
var = accelerometer.magnitude()
var = compass.heading()
var = compass.get_field_strength()
var = compass.is_calibrated()
compass.clear_calibration()
compass.calibrate()
Follow instruction on micro:bit screen to complete calibration.
var = temperature()
sensor on front of board
var = display.read_light_level()
sensor on back of board

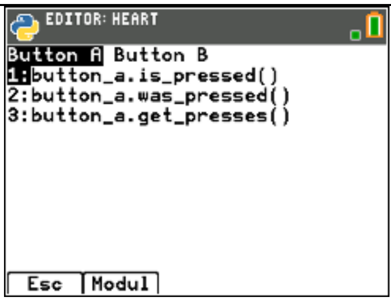


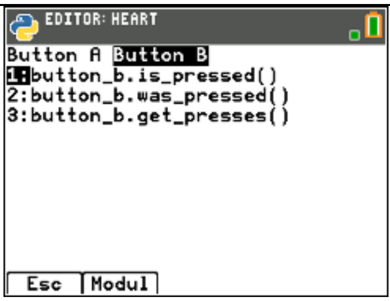




A and B Buttons Module

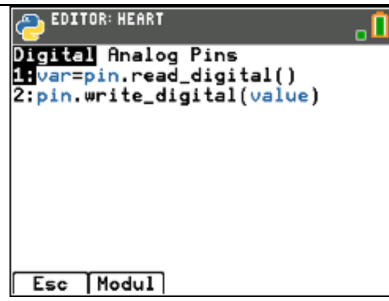
var = button_a.is_pressed()
var = button_a.was_pressed()
var = button_a.get_presses()
var = button_b.is_pressed()
var = button_b.was_pressed()
var = button_b.get_presses()

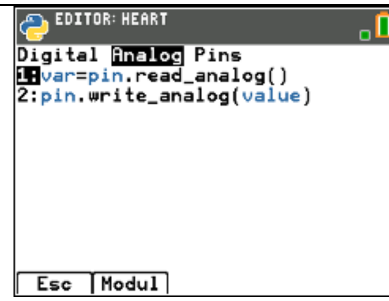


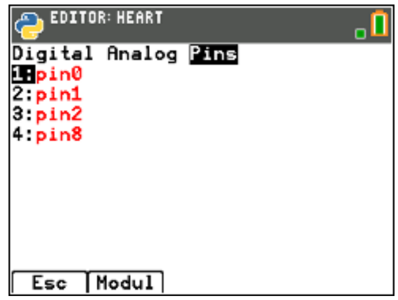


Input and Output Pins Module

var = pin0.read_digital()
var = pin0.read_analog()
pin0.write_digital(value)
pin0.write_analog(value)
var = pin1.read_digital()
var = pin1.read_analog()
pin1.write_digital(value)
pin1.write_analog(value)
var = pin2.read_digital()
var = pin2.read_analog()
pin2.write_digital(value)
pin2.write_analog(value)

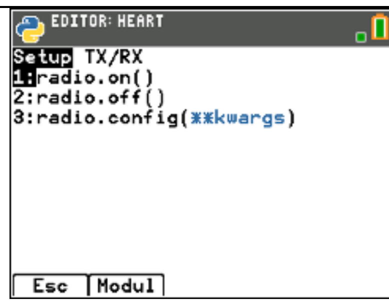


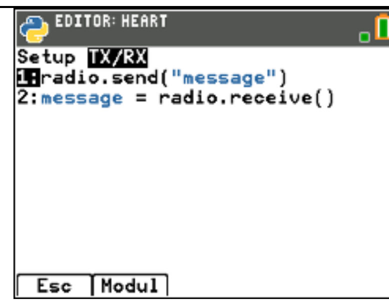




2.4 MHz Radio Module

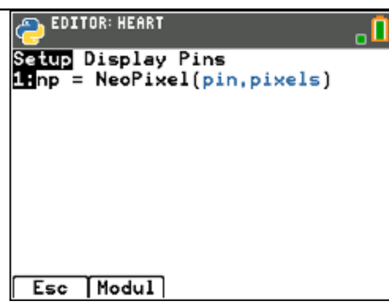
radio.on()
radio.off()
radio.config(channel=7,power=6,group=0)
Two radios must share channel and group to communicate.
radio.sent(message)
var = radio.receive()

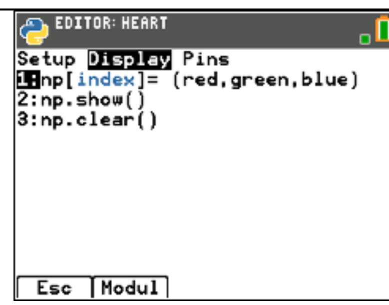


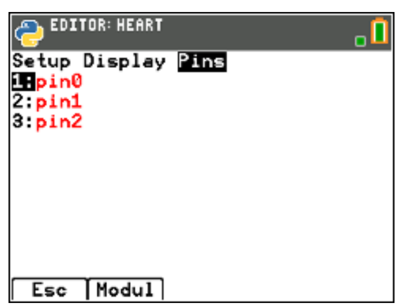


NeoPixel Module

np = NeoPixel(pin = pin0, pixels = 16)
This constructor is optional to chane pin and/or number of pixels on device. Maximum is 16. Current is regulated to not exceed 90mA.
np[index] = (red,green,blue)np.show()
np.clear()



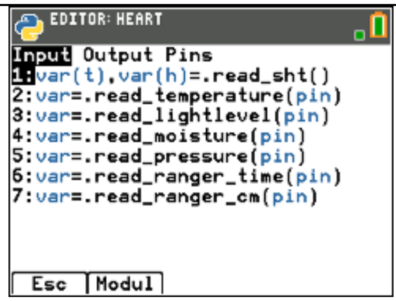


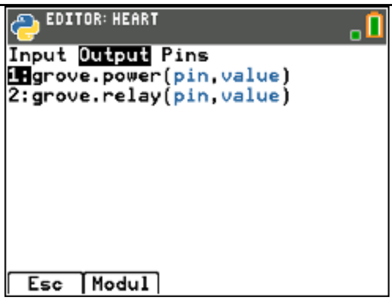


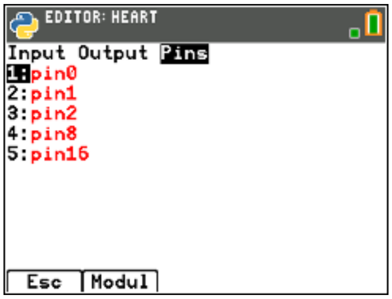
Getting Started Guide

Grove Sensors Module

var = grove.read_sht35()
var = grove.read_temperature(pin)
var = grove.read_lightlevel(pin)
var = grove.read_temperature(pin)
var = grove.read_moisture(pin)
var = grove.read_pressure(pin)
var = grove.read_ranger_time(pin)
var = grove.read_ranger_cm(pin)
grove.set_power(pin,pwr)
grove.set_relay(pin,state)
pin = pin0, pin1, pin2, pin8, pin16



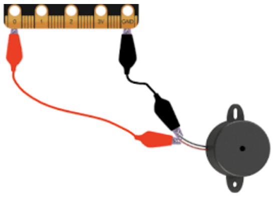




Example Test Programs

DISPTEST.8XV: Demonstrates all of the 5x5 matrix commands.

MUSCTEST.8XV: Demonstrates all of the music and tone commands. A speaker must be connected to pin0 and ground.



SNSRTEST.8XV: Demonstrates the compass, accelerometer, temperature, and light sensors. Follow the directions on the microbit to calibrate the compass.

BUTNTEST.8XV: Demonstrates all of the A and B button commands.

PINSTEST.8XV: Demonstrates the analog and digital inputs. Use an alligator clip to connect 3V or Gnd alternately to the pins 0,1, and 2. Connect an LED to Gnd and alternately to pins 0,1,2 for analog and digital outputs.

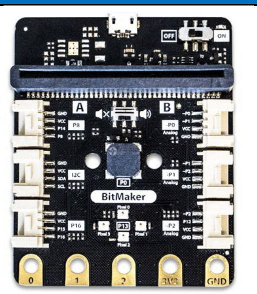
RADITEST.8XV: Demonstrates the radio commands. Two or more calculators with micro:bits attached must run this program concurrently for the demonstration.

NPTEST.8XV: Demonstrates the NeoPixel commands. To use TI-RGB array with this test, make the following connections:

Microbit	TI-RGB Array
3V	5V (Red)
GND	⏚ (BLACK)
Pin0 (can be changed in the constructor)	BB2(YELLOW)
pin1.read_analog()	BB5(BLUE)

GROVTEST.8XV: Demonstration requires an expansion board with Grove receptacles and the corresponding Grove sensors. The SHT35 Temperature and Humidity sensor must be plugged into an I2C port. Check the program for other sensor locations.

Resources

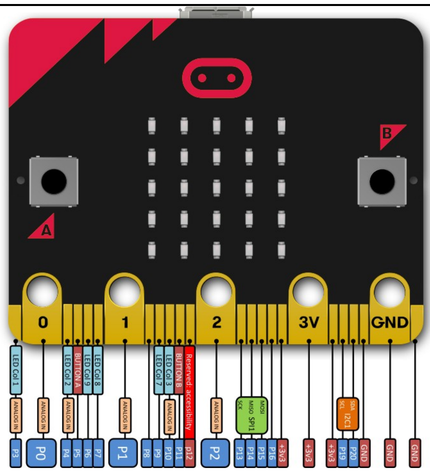


Expansion board with Grove ports
https://www.seeedstudio.com/BitMaker-p-4353.html

Speaker with PCB mount pins
https://www.jameco.com/webapp/wcs/stores/servlet/ProductDisplay?langId=-1&storeId=10001&catalogId=10001&productId=135722

USB Mini to USB Micro adapter
https://www.sfcable.com/usb-micro-male-to-mini-5pin-female-adapter.html?gclid=CjwKCAjw1v_0BRAkEiwALFkj5pILcMI-Jl_cWTZPnnjpWpM2Tinpp59tJSMTVIpMszd2ZwiErYtgiRoCRXwQAvD_BwE

Alligator Clips
https://www.amazon.com/WGGE-WG-026-Pieces-Colors-Alligator/dp/B06XX25HFX/ref=sr_1_1_sspa?dchild=1&keywords=alligator+clips&qid=1590053249&sr=8-1-spons&psc=1&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUEzSkNYOVUzQ1QyUjM5JmVuY3J5cHRlZElkPUExMDI2NDI3QlhTVTFCQjZKNlFXJmVuY3J5cHRlZEFkSWQ9QTAxOTMyMzVPSVAyNlNPUEs2OUgmd2lkZ2V0TmFtZT1zcF9hdGYmYWN0aW9uPWNsaWNrUmVkaXJlY3QmZG9Ob3RMb2dDbGljaz10cnVl



micro:bit Application Programming Interface (API)
The TI microbit module is aligned to the standard micro:bit API. Please use thisreference as you include micro:bit statements into your TI-python programs. https://microbit-micropython.readthedocs.io/en/v1.0.1/

micro:bit Let' Code
If the micro:bit is programed with another language such as MakeCode blocks, JavaScript, or C++, a different runtime .hex file is loaded onto the micro:bit. To restore the calculator's communication functionality, the TI-84 Plus CE-T PYTHON EDITION runtime must be reinstalled, as directed in the first step of this document.
Switching runtimes does not harm the micro:bit. This step is essential for micropython to be loaded on the micro:bit and enable python programming.
https://microbit.org/code/

Micro:bit runtime
https://lancaster-university.github.io/microbit-docs/

Micro:bt Micropython
https://tech.microbit.org/software/micropython/

Example Programs

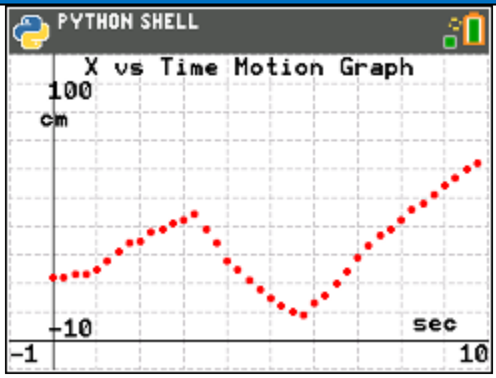
Program 1

```
import ti_plotlib as plt
from ti_system import *
from microbit import *
from mb_grove import *

plt.cls()
plt.window(-1,10,-10,100)
plt.grid(1,10,"dash")
plt.axes("on")
plt.labels("sec "," cm",11,3)
plt.title("X vs Time Motion Graph")
plt.color(255,0,0)

initilize=grove.read_ranger_cm(pin0)

for i in range(40):
  echo=grove.read_ranger_time(pin0)
  d=echo/2*34000
  t=i*.25
sleep(250)
plt.plot(t,d,"o")
plt.show_plot()
```



Program 2

```
import ti_plotlib as plt
from ti_system import *
from microbit import *
from mb_sensr import *

plt.window(-1800,1800,-1200,1200)
x = 318/2
y = 30+212/2
r = 981*318/(plt.xmax-plt.xmin)
plt.color(240,240,240)
plt.cls()
plt.gr.fillCircle(x,y,r)
plt.grid(200,200,"solid")
plt.color(0,0,0)
plt.gr.drawArc(x-r,y-r,2*r,2*r,0,3600)
plt.axes("on")
plt.labels("x (mg) "," y (mg)",8,1)
plt.gr.drawString("981 mg",x+15,y-15)
plt.color(255,0,0)

while not escape():
  accx = accelerometer.get_x()
  accy = accelerometer.get_y()
  plt.plot(accx,accy,"o")
plt.show_plot()
```