Unit 6: micro:bit with python — Skill Builder 1: The display

In this lesson, you will write your first python programs to control the micro:bit **display** in different ways. This lesson has two parts:

Part 1: alien encounter
Part 2: displaying images

Objectives:

- Control the display on the micro:bit board using **.show**( ), **.scroll**( ) and **.show**(image)
- Control the speed of the display using **sleep**(*ms*)

---

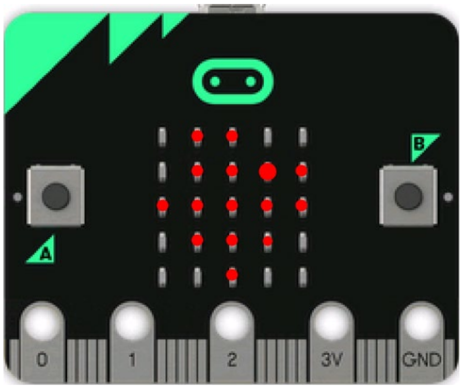
1. Before you begin, be sure that:
   - your TI-Nspire CX II has OS **5.3** or higher
   - You are comfortable programming in python and/or you have completed Units 1 through 5.
   - your **micro:bit** is connected to your TI-Nspire CX II
   - You have followed the set-up directions and file transfers in the micro:bit **Getting Started Guide**:

     https://education.ti.com/en/teachers/microbit

     This setup process should only have to be done once, but keep informed periodically about updates/upgrades.

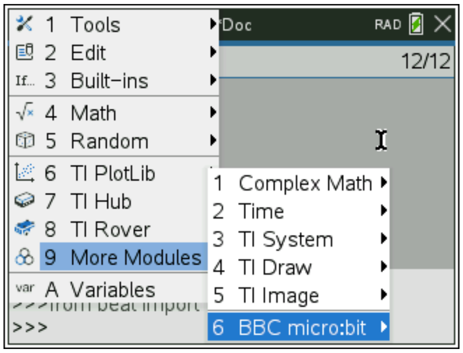
2. If all is good and the setup has been done correctly, your micro:bit should look like this when it has power from the TI-Nspire:

   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**.



3. and finally…

   The micro:bit module is installed to your Python Library. In a Python Editor, press **[menu] > More Modules** and see that **BBC micro:bit** is listed beneath the TI modules.



```
>>>from beat import
>>>
```

Note: In OS 5.3 and above, python modules that are stored in your **Pylib** *folder on your device are shown on this menu in addition to the ti_ modules. Your list may differ from the one shown here. The modules are listed alphabetically by filemane, so* ***BBC micro:bit*** *is listed among the "m's" in the list, not the "B's".*

Teacher Tip: Getting Started with micro:bit

This Unit assumes that the micro:bit and the TI-Nspire handheld (or computer software) are 'ready to go'. Make sure you (or your students) have completed the setup steps on their individual calculators and micro:bits. Be sure to read and perform the setup instructions in the 'Getting Started' guide that is included in the download folder from the TI Education website. https://education.ti.com/en/teachers/microbit. As with all new software and hardware, be sure to stay informed about upgrades and updates.

There is also an introductory TI-Nspire document, **my first program.tns**, that makes sure that the micro:bit is connected and working properly.

The BBC micro:bit module (microbit.tns) is a special module provided by Texas Instruments that is installed in the Pylib folder of each handheld and in the TI-Nspire CX II Computer Software and so it appears on **[menu] > More Modules > as BBC micro:bit.**

The first five units of the Python: 10 Minutes of Code curriculum should be completed prior to this Unit. The coding presented here is not complicated, but there should be a comfort level with python coding on the TI-Nspire CX II.

This Unit is designed to work with micro:bit versions 1 and 2 but does not address features that are unique to version 2. The BBC micro:bit module does support those features.

All the micro:bit projects found online (including micro:bit version 2 projects) can be developed on the TI-Nspire CX II using python and an attached micro:bit. But there are two important distinctions:

- You cannot disconnect the micro:bit from the calculator while the program is running even if a battery is attached to the micro:bit. The micro:bit runtime (.hex file) is configured to listen to the calculator for its instructions. The calculator is powering and controlling the micro:bit.
- Many of the demo programs found online use While True: to create an infinite loop. This loop runs directly on the micro:bit board until the program is replaced with another using a downloaded .hex file. When using the TI-Nspire, the calculator is controlling the micro:bit so the loop most commonly used is

While get_key() != 'esc':

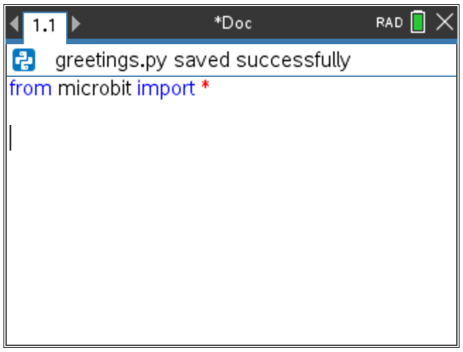
which allows the user to press the [esc] key to terminate the program.

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

4. **Part 1: alien encounter:** Of course, as with every other first programming experience, you will start with displaying a message on the micro:bit display.
Start a new TI-Nspire document and select **Add Python > New** to begin a new program (blank program), named '**greetings**'. 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).



greetings.py saved successfully

```
from microbit import *
```

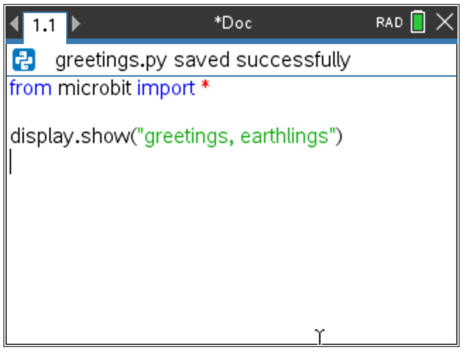
Teacher Tip: If a micro:bit is not attached, then the import statement itself will report an error when the program is run. Since python is a 'modular' language, supplemental features are added as needed. This import statement loads specific functions/methods required to operate the micro:bit using the TI-Nspire CX II. It also loads some useful methods from other standard modules and ti_ modules**.**

5. To display a text message on the micro:bit display, use the statement:

display.show(image or text**)**

This statement is found on:

[menu] > More Modules > BBC micro:bit > Display > Methods



greetings.py saved successfully

```
from microbit import *

display.show("greetings, earthlings")
```

The statement is inserted as **display.show(**image or text**)**, but (image or text) is just a placeholder that must be replaced with something. Inside the parentheses, replace image or text by typing your message string in quotes:

"greetings, earthlings"

When you run this program by pressing **[ctrl] [R]** you will see the letters of your message appear, one letter at a time, on the display. The lowercase letters 'e' do appear twice but you cannot distinguish two of them.

If you make a mistake…. go back to page 1.1 to edit your program, and then run the program again. Did you forget to add quotes around the text you wanted to display?

6. A better method for displaying messages is:

   display.scroll(“greetings, earthlings”)

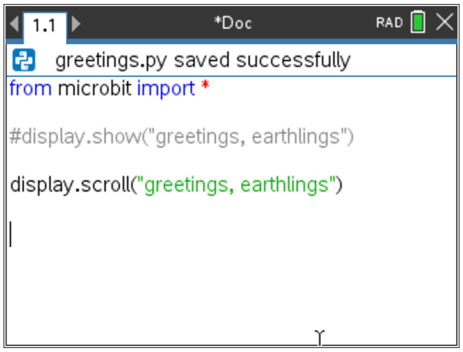
   which is also found on

   [menu] > More Modules > BBC micro:bit > Display > Methods

   *To complete the **.scroll( )** statement you can copy/paste the string from the **.show** statement.*

   Make the previous **.show()** statement a #comment (place the cursor on that line and press **[ctrl] [T]**) to disable it and then run the program again.

   *Yes, you can also simply change .**show** to .**scroll** by typing.*



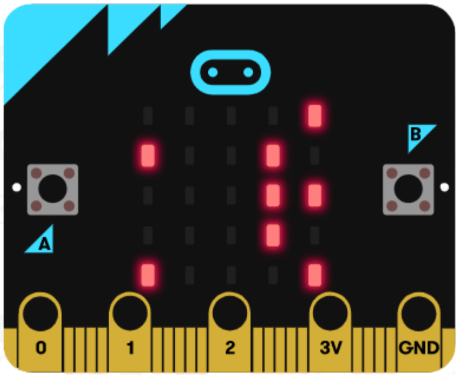
greetings.py saved successfully

```
from microbit import *

#display.show("greetings, earthlings")

display.scroll("greetings, earthlings")
```

7. The **.scroll()** method causes the message to scroll (move) from right to left like a banner across the display making it easier to read.
   You can control the speed of the scrolling by adding the **delay=** parameter:

   display.scroll(“greetings, earthlings”, delay = 200)

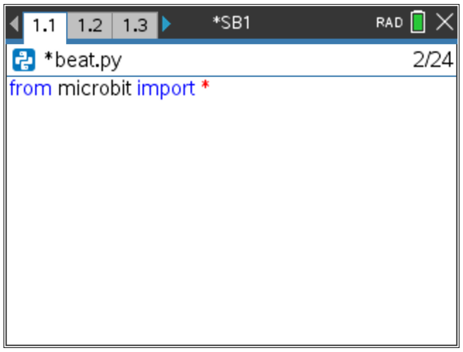
   which uses a 200 millisecond (0.2 second) delay in the scrolling. Try other delay values, too.



<greetings, earthlings.gif>

8. **Part 2: Be.Still.My.Beating.Heart** press **[ctrl] [doc]** to insert a page and select **Add Python > New** to add a new Python program to your document (ours is named ‘**beat**’).
   In the Python Editor, use **[menu] > More Modules > BBC micro:bit** and select the **import** statement at the top of the list:

   from microbit import *



*beat.py 2/24

```
from microbit import *
```

9. To display the HEART symbol on the micro:bit display, use the statement:

display.show( )

This statement is found on:

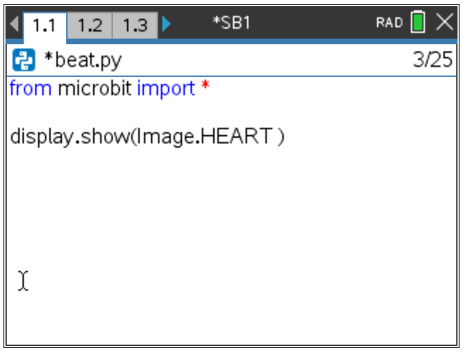
[menu] > More Modules > BBC micro:bit > Display > Methods

Inside the parentheses, replace the prompt by selecting:

Image.HEART

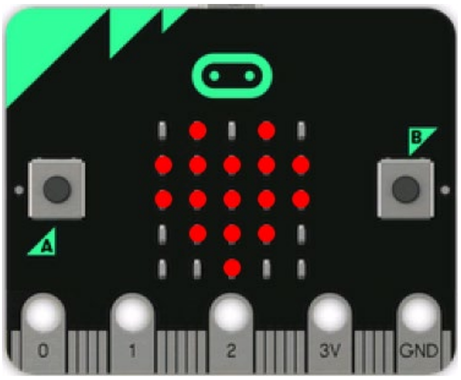
from

[menu] > More Modules > BBC micro:bit > Display > Images > Set 1> Heart



```
from microbit import *

display.show(Image.HEART )
```

10. Run the program (press **[ctrl] [R]**) to see the HEART icon displayed on the 5x5 LED grid of the micro:bit. This display remains until something takes its place, even after the program is done.



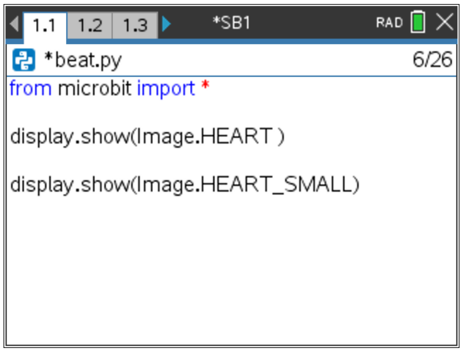
11. Go back to the Program Editor on the previous page, and add another display statement to show the small heart:

display.show(Image.HEART_SMALL)

You can find this image on the same **Images** menu:

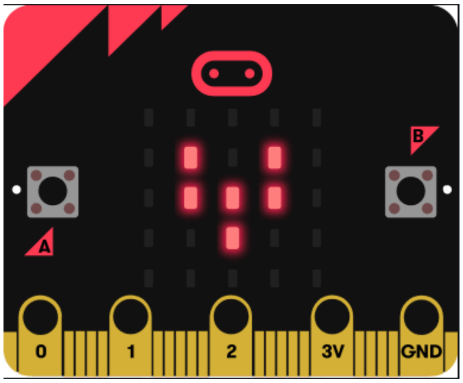
[menu] > More Modules > BBC micro:bit > Display > Images > Set 1

Tip: you can also copy/paste the first display statement and edit (type the **_SMALL**)*. It must have the underscore _ and be uppercase.*



```
from microbit import *

display.show(Image.HEART )

display.show(Image.HEART_SMALL)
```

12. Run the program again. It quickly displays the large heart and then displays the small heart that looks like this.

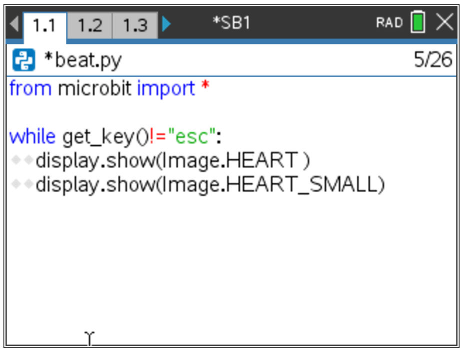


13. **Make a loop**: To get the two hearts to blink repeatedly ('beat'), enclose the two display statements in a loop. *Before* the two display statements insert:

    while get_key() != "esc":

    found on **[menu] > More Modules > BBC micro:bit > Commands** and <u>*indent*</u> the two display statements so that they form the loop body.

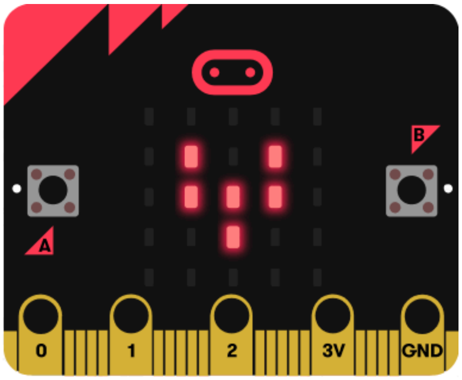
    Important Tip: **Indentation** *is critical in python programs. This is how python interprets loop blocks and if blocks. If the two display statements are not indented the same number of spaces then you will see a syntax error. Use the* **[space]** *key or the* **[tab]** *key to indent the two lines the same amount. Indentation spaces are indicated in this Editor as light gray diamond symbols (♦ ♦) to help with proper indentation.*



```
from microbit import *

while get_key()!="esc":
♦♦display.show(Image.HEART )
♦♦display.show(Image.HEART_SMALL)
```

14. Run your program again and watch the Beating Heart! Press the **[esc]** key to end the program.

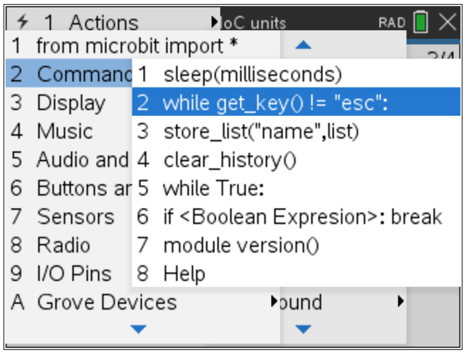
    Tip: if you ever think your program is stuck in an infinite loop press and hold the **[home/on]** key *on your TI-Nspire to 'break' the program. This could happen if you use* **while True:** *from the Commands menu improperly. These lessons avoid that type of structure.*



<beating_heart.gif>

15. *The micro:bit* **Commands** *menu contains some useful python commands that are also found on other menus. The micro:bit module imports these python commands for you.*

    You can these and any other python commands from other menus. You are not limited to just using the BBC micro:bit menu but you may need to provide the proper import commands.

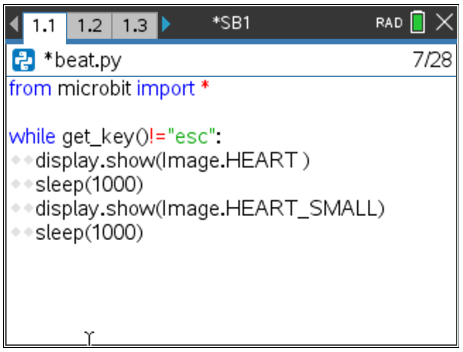


16. To control the beating heart rate, add two **sleep()** statements, one after each **display** statement:

    sleep(1000) means 1000 milliseconds or a 1 second delay.

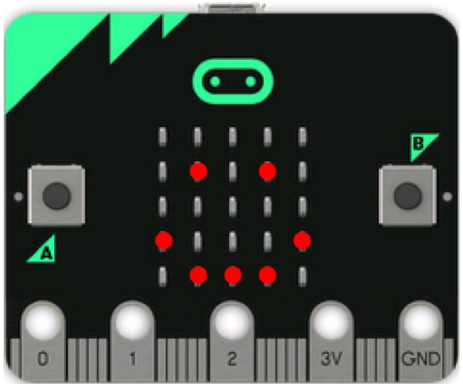
    Also found on **[menu] > More Modules > BBC micro:bit > Commands**

    Tip: watch the indentation!



```
from microbit import *

while get_key()!="esc":
  display.show(Image.HEART )
  sleep(1000)
  display.show(Image.HEART_SMALL)
  sleep(1000)
```

17. **Extension:** Try ‘**Making faces**’. Use a similar program structure as ‘**Beating Heart**’ but use the face images found in Set 1 instead.



Teacher Tip: The online micro:bit lessons typically use a **While True: loop in python and** a forever: loop in MakeCode**.** These ‘infinite loops’ run directly on the micro:bit until it is turned off or replaced by another program (.hex file). When using the TI-Nspire to control the micro:bit, the infinite loop is not necessary since the calculator is in full control. **While True: is on the Commands menu but should be used in conjunction with if (Boolean Expression): break to get out of the infinite loop.**

If a student gets stuck in an infinite loop on the handheld, to ‘break’ the program press and hold the [on] key. See the ‘Getting Started’ guide (PDF) that came with the micro:bit software from TI for information about breaking python programs on computers.

The **‘Commands’** menu contains a collection of common, frequently used python commands that come in handy when coding for micro:bit. The micro:bit module imports these commands from other modules where needed. But all python commands are available from other menus and can be used in micro:bit projects. Using the proper import statements is important, though.

Optional: To get the display back its ‘TI’ state (the TI logo of Texas) at the end of a program, add the statement

display.show(ti) *simply type the letters ti (lowercase)*

after the end of the While loop (dedented).

This is not necessary but it helps to indicate that the program is finished on the micro:bit display as well as on the screen.

At the beginning of a micro:bit program (before the while *loop begins) it might help to print something on the calculator screen, such as* print('running…').

About sleep( ): The one python function that is *significantly* modified for the micro:bit is the sleep( ) statement found in the time module. Normally, the argument value represents 'seconds', but the micro:bit module revises the sleep method so that the value represents milliseconds (*thousandths* of a second). After importing the micro:bit module, sleep(1000) represents a delay of one second. Note that sleep( ) controls the speed of the calculator program, not the micro:bit itself. Be sure that from microbit import * occurs *after* import time (or any module that might import time) for this reason.

The .show() method has two optional parameters: delay= and wait=

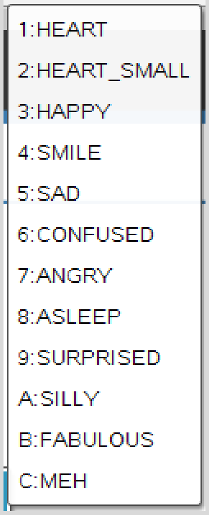
display.show(value, delay=*xxx* , wait=False/True)

delay value is in milliseconds and slows down the display.

wait=True will block processing until the animation is finished, otherwise the animation will happen in the background.

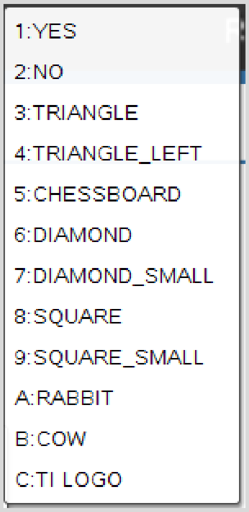
Here is the full list of images:

Set1



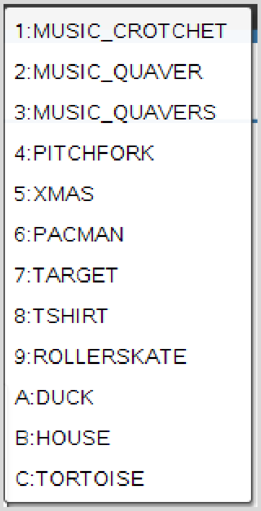
1:HEART
2:HEART_SMALL
3:HAPPY
4:SMILE
5:SAD
6:CONFUSED
7:ANGRY
8:ASLEEP
9:SURPRISED
A:SILLY
B:FABULOUS
C:MEH

Set 2



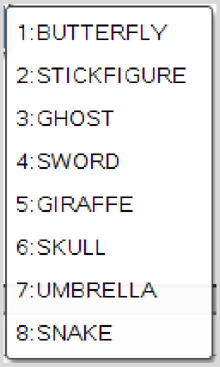
1:YES
2:NO
3:TRIANGLE
4:TRIANGLE_LEFT
5:CHESSBOARD
6:DIAMOND
7:DIAMOND_SMALL
8:SQUARE
9:SQUARE_SMALL
A:RABBIT
B:COW
C:TI LOGO

Set 3



1:MUSIC_CROTCHET
2:MUSIC_QUAVER
3:MUSIC_QUAVERS
4:PITCHFORK
5:XMAS
6:PACMAN
7:TARGET
8:TSHIRT
9:ROLLERSKATE
A:DUCK
B:HOUSE
C:TORTOISE

Set 4



1:BUTTERFLY
2:STICKFIGURE
3:GHOST
4:SWORD
5:GIRAFFE
6:SKULL
7:UMBRELLA
8:SNAKE

A similar project is 'Making Faces' using the same program structure as 'Beating Heart' but use the face icons in Set 1 instead.