Unit 6: micro:bit | **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*) and **delay=**
- Writing a loop structure

Teacher Tip: : Getting Started with BBC micro:bit and TI-84 Plus CE Python

The four lessons in this Unit assume that the micro:bit and the **TI-84 Plus CE Python** handheld are 'ready to go'. Make sure you (and your students) are using **OS version 5.7** or higher (**[mem] About**), have completed the setup steps on their individual calculators and the 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.

The BBC micro:bit modules are special Python modules developed and provided by Texas Instruments that are installed in the calculator as AppVars.

The first five units of the CE 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-84, especially working with the unique python meu system.

This Unit is designed to work **with micro:bit versions 1 and 2** but does not address features that are unique to version 2 (audio, microphone, and logo touch). The micro:bit version 2 modules for CE Python do support those features and the coding process is similar to these activities. Just use the menus.

All the micro:bit projects found online (including micro:bit version 2 projects) can be developed on the TI-84 Plus CE Python and an attached micro:bit version 1 or version 2. 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-84, the calculator is controlling the micro:bit so the loop most commonly used is

  while not escape():

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

If the message 'micro:bit not connected' appears even though it IS connected, try unplugging the micro:bit, wait a few seconds and plug it in again (reset/reboot).

1. Before you begin, be sure that:
   - You are using a **TI-84 Plus CE Python with OS 5.7**
   - You are comfortable programming in python and/or you have already completed Units 1 through 5.
   - Your **micro:bit** is connected to your calculator and lit.
   - You have followed the set-up directions and file transfers in the micro:bit **Getting Started Guide**:

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

     and thus have the TI runtime file on the micro:bit and the latest microbit modules in your calculator

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

2. From the home screen (Quit the Python App) you can check your calculator to be sure that the appropriate files are installed: press **[2nd] [+]** for **[mem],** select **Mem Management > AppVars** and scroll to the 'M' section to find MICROBIT and the MB_* support files shown.

   Note: there are more than those displayed in this image: a total of 11 (version 1) or 13 (version 2) micro:bit AppVars.

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

   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**.

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. In the Python Editor start a <New> program (we name it GREETING).

   In OS version 5.7, when selecting **<Fns…> Modul**, a soft key appears called **<Add-On>** which gives access to additional TI-developed modules. (if you don't see this, you need to update to OS 5.7)

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

<Fns…> Modul screen

5. Select **<Add-On>** and see some additional modules that are available, some of them shown here. Your list may differ but should include **microbit**. Select **from microbit import *** to paste that code into your program.

6. In the Editor, the presence of this statement adds a new option to the bottom of both the **<Fns…> Modul** menu and the **[math]** menu: **Micro:bit…**

   As you add features from Micro:bit this menu will grow!

   Note: **[math]** *is a keypad shortcut for* **<Fns…> Modul** *but does not display the* **<Add-On>** *soft key*.

7. The **Micro:bit…** sub-menu contains all the tools necessary for programming the micro:bit in separate sub-**modules**.

   Each of these menu items will paste its own **import** *statement since the tools are all in separate Python modules (AppVars). If you are using a micro:bit version 1 there are fewer items on this screen.*

   Select the **Display** menu…

Teacher Tip: the micro:bit commands are broken up into separate modules to conserve memory.
The **Version** sub-menu contains the microbit module version and a function to determine the runtime version (.hex file) on an attached micro:bit. You can write a program to include the runtime_version() fiunction or jus texecture the function at the Shell prompt.

8. After selecting **Display**, you see a new import statement in your program:

   from mb_disp import *

9. Look at the **[math]** menu again: There is now a **display…** menu item added to the bottom of the list. Select **display…**

Teacher Tip: The *order* in which micro:bit modules appear on the **Modul** menu is determined by the order of the import statements in the code. The system 'looks' at the code and then displays the menu items accordingly. So the menu *name* is more important than the menu *number*.

10. The **display…** menu (shown) contains the functions that control the display on the micro:bit, the 5x5 grid of red LEDs in the center of the board and more.

    Select **.show(val)**

11. Your cursor is blinking inside the function's arguments (actually on the right parenthesis). Type "greetings, earthlings" in the parentheses, including the quotes:

display.show("greetings earthlings")

Press **[2nd] [alpha]** to turn alpha lock on and press **[+]** for a quotation mark. All letters typed will be lowercase. For uppercase, press **[alpha]**.

Note: There are two optional arguments that you can add:

display.show(___**, delay = 400, wait = True**)

delay= *(time in milliseconds) controls the speed of the display.*
wait= *(**True** or **False**) tells the micro:bit/calculator to finish displaying before moving on to do something else.*
Each parameter can and should be edited to see their impact and each may be specified using their keyword. This is recommended, but not required.

<Run> the program and watch the display on the micro:bit.
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.

Teacher Tip: for more information on micro:bit functions, see the micro:bit API at https://microbit-micropython.readthedocs.io/en/latest/microbit.html

12. A better method for displaying long messages is:

display.scroll("greetings earthlings")

which is also found on the **[math] display…** menu.

Make the previous **.show()** statement a #comment. Place the cursor at the beginning of that line and press **[2nd] [3]** to insert a '#' and then run the program again.

Yes, you can also simply change **.show** *to* **.scroll** *by typing.*
Note: .scroll() also supports the optional **delay=** *and* **wait=** *arguments.*

13. The **display.scroll( )** function causes the message to scroll smoothly from right to left like a banner. **delay** controls the speed of the scrolling. **wait** tells the calculator and micro:bit to wait until the scroll is done before moving on to another instruction. Try other **delay** values to see the effect on the scrolling.

display.scroll(____ , delay = 100, wait = True)

14. **Part 2: Displaying images- Be.Still.My.Beating.Heart**
This section shows you how to display images on the micro:bit.
Select **<Files>** and make a new program called HEART.
In the Editor, add the two import statements as before:

from microbit import *
from mb_disp import *

First, get the microbit module from **<Fns…> <Add-On>**
Then press **[math] Micro:bit**… *for* **display…**

15. To display the HEART image on the micro:bit display, use the statement:

display.show(…)

Again, this statement is found on:
[math] display…
Immediately, with your cursor still inside the parentheses, select:

HEART

from the **[math] display… Images** *sub-menu.*

Note: there are 36 Images to choose from and you can design your own using **var=Image(…)** *found on the Display sub-menu. See micro:bit documentation online for information on this feature.*

In case you have not noticed…to cut down on keypresses, the menu system allows for up-arrow and left-arrow 'wraparound', so, for example:
Select **[math]** *then up-arrow to* **display…**

16. The string **"Image.Heart"** is inserted into your code as shown in this screen.

17. **<Run>** the program. Do you see the heart ♥? This display remains on the micro:bit until something takes its place, even after the program is done. To restore the original TI icon, reset the micro:bit. There is a 'reset' button on the back of the micro:bit or you can unplug and re-plug from the calculator. Or just leave it alone.

18. Go back to the Editor and add another **display.show( )** statement to show the small heart:

    display.show("Image.HEART_SMALL", …)

    Be sure to get the **display.show( )** statement first, then paste the **"Image.HEART_SMALL"** string. You can find this 'small heart' string on the same **Image** sub-menu right below HEART.

19. **<Run>** the program again. It quickly displays the large heart and then displays the small heart that looks like this.

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

    while not escape( ):

    ♦ ♦

    found on **[math] ti_system…**
    and *indent* the two display statements so that they form the loop body.

    You also need to import **ti_system** in order to use this special **while** statement. Place that **import** statement at the top of your code.

    Important Tip: new to Python? **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 (**[alpha] [0]**) *or select* **<Tools> Indent►** *to indent both lines the same amount. Indentation spaces are indicated in this Editor as light gray diamond symbols (♦ ♦) to help with proper indentation.*

21. **<Run>** your program again and watch the Beating Heart! Press the **[clear]** key to end the program.

    Tip: if you ever think your program is stuck in an infinite loop, press and hold the **[on]** key *on your calculator to 'break' the program. This could happen if you use* **while True:**. *These lessons avoid that type of structure by using the* **while not escape():** *loop.*

22. To control the beating heartrate, add the **delay=** and, optionally **wait=,** argument and adjust the values in these two display commands as shown here. Remember that the delay value is in milliseconds.

    Note: some blank lines have been removed from the program in this screen to show the entire program.

    Change both **wait=** *parameters to* **False** *to see how fast the heart blinks.* **False** *is found on* **<a A #>.** *wait =* **True** *tells the micro:bit to complete the task before going to the next task (like 'pause' or 'sleep'). When* **False** *it goes immediately to the next task and ignores the delay= value.*

23. **<u>Optional:</u>** Try '**Making Faces**'. Use a similar program structure as '**Beating Heart**' but use the 'face' images instead

Teacher Tip: Many 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-84 Plus CE to control the micro:bit, this infinite loop is not necessary since the calculator is in full control.

If a student gets stuck in an infinite loop on the handheld, to 'break' the program press 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.

Python commands on other menus also come in handy when coding for micro:bit. The micro:bit module does not import these commands. Using the proper import statements is important, though, as in using **while not escape()** which requires the **ti_system** module.

Caution regarding the **time** module: If you need to use another function from the **time** module (besides **sleep( )**) in your program be aware that the micro:bit module <u>modifies</u> the **sleep( )** function to use milliseconds rather than seconds (it's a micro:bit 'feature') so be sure to place **from time import *** *above* **from microbit import *** in your code so that the **sleep( )** function operates on milliseconds rather than seconds. This is a micro:bit convention. The microbit module has the **sleep( )** function included.

Optional: To get the display back its 'TI' state (displaying the TI logo of Texas) at the end of a program, either

- *press the 'reset' button on the back of the micro:bit.*

or

- *restart the micro:bit by unplugging and re-plugging from the calculator.*