micro:bit miniPET

Course	Learn to Program with the BBC micro:bit and Python
Topic	Program Your Own miniPET
Author	Darrell Little
Grade level	Ages 13 and up
Time duration	One hour
Overview	Using built-in Images in the micro:bit Library, display and interact with your virtual micro:bit pet.
Materials	PC with web browser, micro:bit with USB cable, battery pack
Concepts	Import Code Libraries Variables While True Loop Built-in Images If/Else statements Button Press Pseudocode
Advanced Project	Add a Health Score and event timer

Preparation

- On your computer, open a web browser and got to https://python.microbit.org/v/2.0 or one of the other online code editors.
- For this workshop, we will be using this code environment which allows you to connect directly the micro:bit to transfer the completed code.
- During the process, you may download the code to save your work or transfer the code to the micro:bit to test your progress.
- Don't be worried about errors, they are expected and allow you to learn to troubleshoot your code.

Base Project

Using the Python programming language and several of the built-in images included in the micro:bit Library, create and virtual pet and interact with it using the buttons on the micro:bit.

There are several different built-in images to choose from for your pet:

```
Image.RABBIT

Image.COW

Image.DUCK

Image.TORTOISE

Image.BUTTERFLY

Image.GIRAFFE

Image.SNAKE
```

Other images that might be used in this project include:

```
Image.HAPPY

Image.SAD

Image.ANGRY

Image.ASLEEP

Image.HEART
```

Pseudocode is way to represent the rules and logic that will be used in a program. This can be an effective way to think through how the program should run and how the player will interact with the micro:bit.

The foundation of the miniPET program should be something like this:

```
Prompt Player to press Button A to feed the pet, B to play with
the pet or Both buttons to let the pet sleep

While Playing
Display Image of Pet
If Both A and B buttons are pressed, display the Asleep image
Else, If Button A pressed, display a Heart image for good health
Else, if Button B pressed, display a Happy image for fun
```

Code

1. Import the needed library. The "*" character means to include the entire contents in the library. All the functions used in this workshop will be imported into the program. Also, use the `display.scroll` function to display the instructions on the LED array to the user.

```
15 from microbit import *
16 display.scroll("PRESS A TO FEED")
17 display.scroll("PRESS B TO PLAY")
18 display.scroll("PRESS A + B TO SLEEP")
19
```

2. Another function very similar to `display.scroll` is used to show one of the built-in images included in the micro:bit library. Only using the display.show() function by itself will not have the desired effect (the image displays so quickly, the user does not see it). Adding the sleep() function slows things down enough so that the image can be seen. Inside the parenthesis, put the number of milliseconds (1 ms is 1/1000 of a second). To pause 2 seconds, you'll use 2000 ms.

```
19
20 display.show(Image.RABBIT)
21 sleep(2000)
22
```

3. Next, we need to watch for a button press. Using technique called a "while loop" the program will continuously repeat a block of code depending on a condition. The code `while True:` loops the block of code that follows it until the condition is not True. Since nothing in the code block changes True to False, it is infinite.

4. There are special functions in the library that are used to detect button presses: `button_a.is_pressed()` and `button_a.is_pressed()`. Looking at the pseudocode, there are three different actions we want to take: Feed, Play and Sleep, each with an associated image.

5. To determine which action to take based on the button(s) pressed, the If, Else If and Else logical statements are used. In Python these commands are `if elif else`

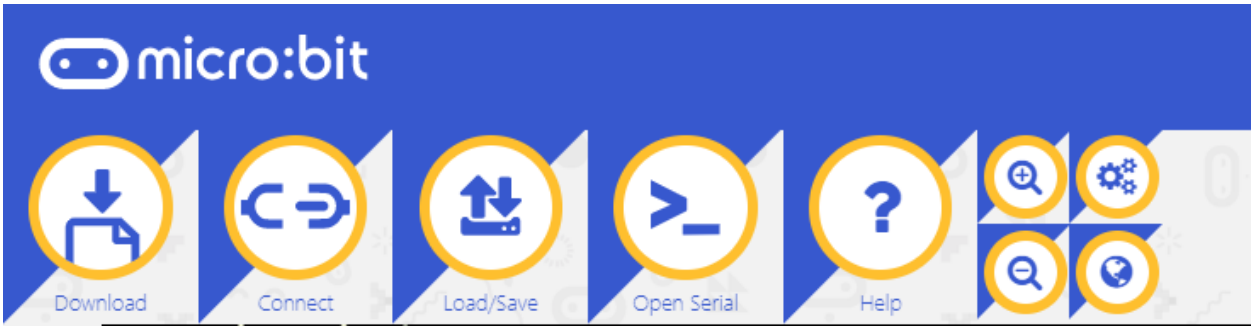
   First, look for the A + B combination, then B, then A. If none of the buttons are pressed, display the image for your pet. The code block should look like this:

```
23 while True:
24     if button_a.is_pressed() and button_b.is_pressed():
25         display.show(Image.ASLEEP)
26         sleep(2000)
27     elif button_b.is_pressed():
28         display.show(Image.HAPPY)
29         sleep(2000)
30     elif button_a.is_pressed():
31         display.show(Image.HEART)
32         sleep(2000)
33     else:
34         display.show(Image.RABBIT)
35
```

The entire program should look like this:

```
15 from microbit import *
16 display.scroll("PRESS A TO FEED")
17 display.scroll("PRESS B TO PLAY")
18 display.scroll("PRESS A + B TO SLEEP")
19
20 display.show(Image.RABBIT)
21 sleep(2000)
22
23 while True:
24     if button_a.is_pressed() and button_b.is_pressed():
25         display.show(Image.ASLEEP)
26         sleep(2000)
27     elif button_b.is_pressed():
28         display.show(Image.HAPPY)
29         sleep(2000)
30     elif button_a.is_pressed():
31         display.show(Image.HEART)
32         sleep(2000)
33     else:
34         display.show(Image.RABBIT)
35
```

Using the Download button, save the HEX file to your PC (on the Desktop or other folder you choose) and with the micro:bit plugged in with the USB cable, drag the HEX file to the micro:bit (usually shows as drive D).



Advanced

By adding a variable for `health` and `timer` – think about how you could add to this program to make it more challenging. For each interaction with your pet, add points for health. If too much time goes by without an action (ignoring your pet), points are subtracted from health. When health reaches zero, the game is over.

Sketch out your **pseudocode** below: