

# Coding Fundamentals



## Micro:bit Python Programming Tug O' War

### Overview

In this lesson, students use variables and conditionals to create a Tug O' War game on the micro: bit using Python.

### Objectives

- Explain where for loops happen in real world situations
- Implement for and while loops into a program
- Program the micro:bit® to create 2-Player tug-o-war game

### Materials

- micro:bit and micro-USB cord
- Computer with access to the internet

### Approx. Time Required

1-2 hours

### Cyber Connections

- **Programming** – Students will program in Python.
- **Hardware and Software** – Students will utilize small electronics and learn how a computer is programmed while using micro-controllers.

**CYBER.ORG**  
THE ACADEMIC INITIATIVE OF THE CYBER INNOVATION CENTER

*This content is based upon work supported by the US Department of Homeland Security's Cybersecurity & Infrastructure Security Agency under the Cybersecurity Education Training and Assistance Program (CETAP).*

# Tug O' War

- This lesson will give students a tool just as powerful and useful as the **while** loop seen in previous lessons: the **for** loop. A **for** loop is very similar to the **while** loop in nature, as they are both tools that allow sections of a program to be looped or repeated over and over again. The key difference, however, is that a **for** loop is used to repeat a block of code a *set* or *exact* amount of times, specified by the programmer. A **while** loop on the other hand, loops until a certain condition is met. Direct students to the worksheet to see examples and practice using **while** and **for** loops.
- In Python, a **for** loop takes the following format:

```
for i in range(x,y,z):  
    #code to repeat here
```

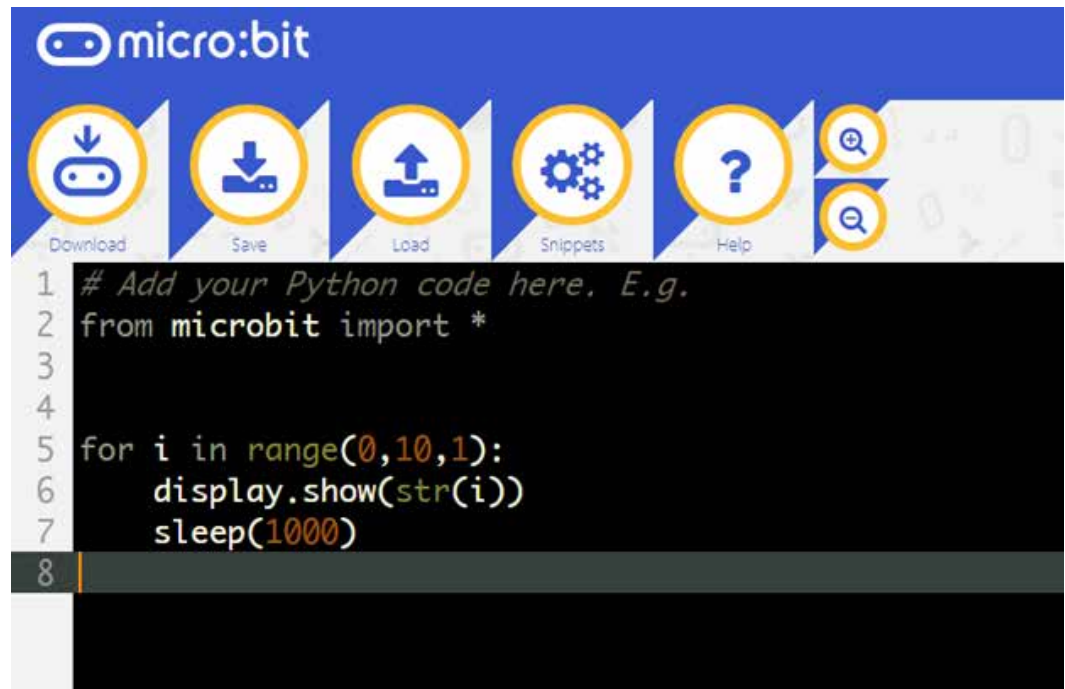
Where “i” is the variable name for the counter that counts up until the loop has been repeated the specified number of times. This can be named anything, as it is a variable. “i” is the standard naming, but it can be replaced with “counter” or even “blueberry” and still work.

“x” is the beginning number, or start place for your counter. This is where the variable above begins counting.

“y” is the final number, or ending place for your counter. This is where the variable finishes counting and the loop stops repeating.

“z” is the step size. This controls how big of a step the counter takes each time through the loop. 1 means one step, 2 would count by 2's, and -1 would count backwards by one each time through the loop. This can be set to any number.

- Below is an example of a **for** loop that works as a 10 second timer. It is also included on the worksheet for the students benefit. Encourage students to use this code and play around with different numbers to try to understand what they do. The code includes **str()** in the **display.show()** command. This piece of code is a command that converts numbers into something called a “string.” Essentially, this command is used to convert whatever number is inside its parentheses into an easily printable format.



```
1 # Add your Python code here, E.g.
2 from microbit import *
3
4
5 for i in range(0,10,1):
6     display.show(str(i))
7     sleep(1000)
8
```

- There are a few more commands and keywords that students will need to know to complete the project that goes along with this lesson. The first should look familiar and will be fairly obvious to grasp: **display.clear()**. This command clears the display by turning all LEDs off.
- Next is an incredibly useful command for the buttons on the micro:bit®: **button\_a.get\_presses()**. This command works for both buttons, and its function is to give you the number of presses the button has been pressed since you last used the command. For example, if you press A 5 times **button\_a.get\_presses()** will give you an answer of 5, but will reset the counter to 0.
- Finally, students will need to make use of the command: **break**. Break is used to “break out” of or stop a while loop. This is particularly useful when you don’t know exactly when you want a block of code to stop running. Its use will become clearer by examining the sample code below.
- The project for this lesson is to create a tug-o-war game. The game will begin with a dot in the middle of the display, and both players having a button to press. After counting down, players will begin to press the button as fast as they can to try to pull the dot to their side. If one side gets 3 or more button presses above the other player, they win. Students should try to use a for loop to make the game a “best two out of three” format. Below is a sample code of a working version of this project.

## Sample Code

```
# Add your Python code here. E.g.
from microbit import *

for counter in range(0,3,1):      #best 2 out of 3
    A_press = 0                   #keep track of a presses and set at 0
    B_press = 0                   #keep track of b presses and set at 0
    while True:
        sleep(100)               #give 0.1 seconds for presses to register
        A_press = A_press + button_a.get_presses() #Add new presses
        B_press = B_press + button_b.get_presses() #Add new presses

        if A_press == B_press:   #if they are equal
            display.clear()      #clear screen
            display.set_pixel(2,2,9) #dot in the middle

        elif A_press + 1 == B_press: #If B has 1 more press
            display.clear()      #clear screen
            display.set_pixel(3,2,9) #Dot 1 closer to B

        elif A_press - 1 == B_press: #If A has 1 more press
            display.clear()      #clear screen
            display.set_pixel(1,2,9) #Dot 1 closer to A

        elif A_press + 2 == B_press: #If B has 2 more
            display.clear()      #clear screen
            display.set_pixel(4,2,9) #Dot 2 closer to B

        elif A_press - 2 == B_press: #if A has 2 more
            display.clear()      #clear screen
            display.set_pixel(0,2,9) #Dot 2 closer to A

        elif A_press >= B_press + 3: #If A has 3 or more extra presses
            display.clear()      #clear screen
            display.show(Image.HAPPY) #show smiley face
            sleep(500)           #wait half a second
            display.scroll('A wins!') #show A wins
            break                #break while loop. Go to next for loop counter

        elif B_press >= A_press + 3: #If B has 3 or more extra presses
            display.clear()      #clear screen
            display.show(Image.HAPPY) #show smiley face
            sleep(500)           #wait half a second
            display.scroll('B wins!') #show B wins
            break                #break while loop, Go to next for loop counter
```

```
else:                                     #incase none of these if statements are met
    display.clear()                       #clear screen
    display.set_pixel(2,2,9)              #put dot in the middle
```

- Things to note about this program as students are trying to create their own:
  - a) The screen will need to be cleared first under any conditional statement, because otherwise the previous dots will remain on the display.
  - b) Students will need 2 separate variables to keep track of the number of times each button has been pressed. These will need to be reset to 0 at the beginning of each for loop. The `get_presses()` command should be added to these each time through the while loop. These variables are what students will use to compare to each other for who is winning the tug-o-war.
  - c) Students will need to use `break` to get out of the while loop and start the next time through of the for loop.