Telepresence of Touch: Workshop 1: Local Device
Paul Mirel, MICA Engineer in Residence 20200927

To participate in this workshop, you will need:

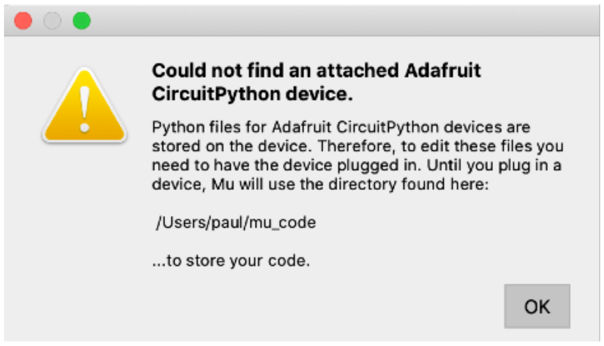
A. A computer that is running Apple Macintosh OS, or Windows OS (Operating System). The work can be done on a Linux machine, but I have no experience with this, so I cannot provide support. The software we will be using (the Mu editor) will not run on a Chromebook, or on smart phone or tablet. *You need administrator privilege on your computer to install Mu. If you install it without sufficient privilege, it will start and then close immediately, every time.*

B. A CircuitPlayground Express, or CircuitPlayground Bluefruit, and a USB cable to connect the CircuitPlayground to your computer.
https://www.adafruit.com/product/3333
https://www.adafruit.com/product/4333

Telepresence of Touch: Workshop 1: Local Device

1. Download and install the Mu editor, from https://codewith.mu

2. Run the Mu editor. It will complain that it can't find a CircuitPython Device. That's OK, we will connect one soon. Click OK.

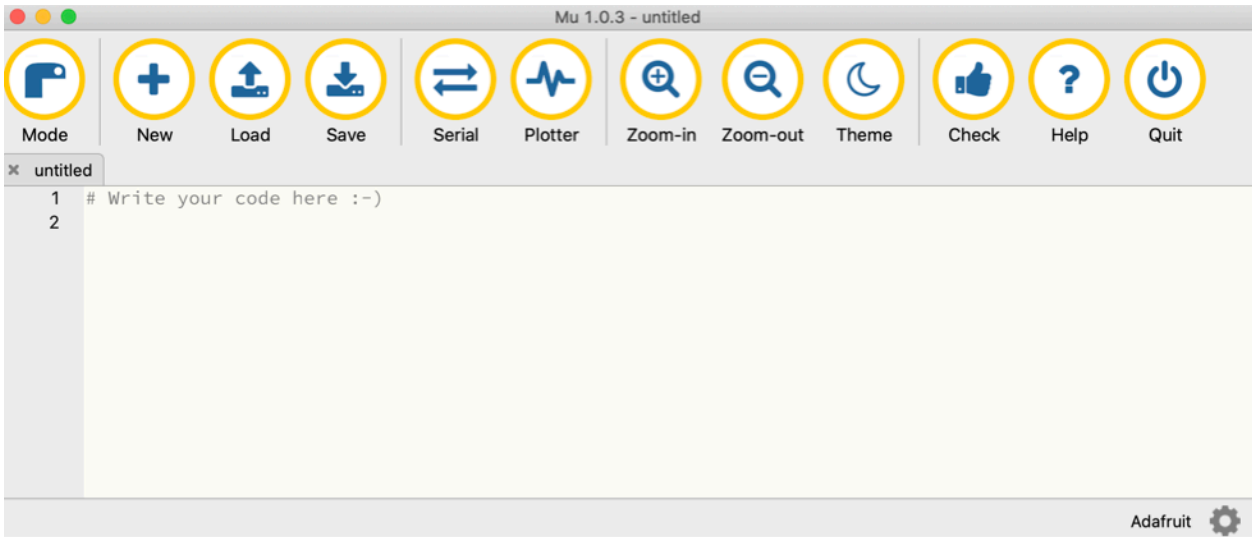

Could not find an attached Adafruit CircuitPython device.

Python files for Adafruit CircuitPython devices are stored on the device. Therefore, to edit these files you need to have the device plugged in. Until you plug in a device, Mu will use the directory found here:

/Users/paul/mu_code

...to store your code.

3. Once running the Mu editor should look something like this:

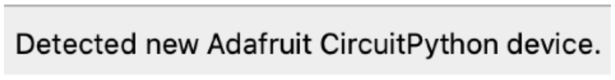

Mu 1.0.3 - untitled

```
# Write your code here :-)
```

4. Connect the CircuitPlayground device to your computer with a USB cable. *CAUTION: Do not place the CircuitPlayground device in contact with the body of your computer. The exposed electrical connections on the CircuitPlayground could*

make contact with the metal of the computer, which may damage the CircuitPlayground, or damage the computer.
If your computer asks if you want to set up a USB keyboard or mouse device, cancel that action.

5. Once the CircuitPlayground is connected, the Mu editor will display an alert that it has found the device:


Detected new Adafruit CircuitPython device.

6. Click on the Mode button, to see that the Mu editor is in Adafruit CircuitPython mode. Click OK to close. (If it is not in Adafruit CircuitPython mode, select that mode.)

7. Click on the Serial button. A Serial Dialogue Panel will open on the lower section of the Mu window, labelled Adafruit CircuitPython REPL. REPL is the Read, Evaluate, Print Loop. You can type individual commands in CircuitPython in this panel, and they will run on the CircuitPlayground device.

8. Click anywhere in the Serial Dialogue Panel to set the cursor in that panel.

9. Hold the ctrl key, and press the c key. We will call this ctrl-c from now on. Ctrl-c cancels whatever program is running. This is the same for Mac and Windows machines. *Do not use the Apple command key for this purpose.*

10. Press any key to enter the REPL mode. You will see a message like this, identifying which version of CircuitPython you are running, and what device you are running it on:
```
Adafruit CircuitPython 5.3.1 on 2020-07-13; Adafruit
CircuitPlayground Express with samd21g18
>>>
```

11. Let's download and install the latest version of CircuitPython. Go to: https://circuitpython.org/downloads

12. From the menu, select the device you are using to get the CircuitPython UF2 file that will run on your device. In this case, select either CircuitPlayground Express, or CircuitPlayground Bluefruit. UF2 is a USB Flashing Format file, abbreviated UFF, or "U F squared", written and generally pronounced as UF2.

13. Click on the DOWNLOAD .UF2 NOW button.

14. Locate the downloaded .uf2 file on your computer, probably in your Downloads folder.

15. Find the Reset button on the center of the CircuitPlayground. Press it twice, in a double-click type of timing. It's a bit sensitive to timing, so you may have to try more than once. When you are successful, all 10 lights will show green. PHOTO TBD

16. On your computer, there should be an external hard drive called CPLAYBOOT. Copy the .uf2 file, downloaded in step 13, to the CPLAYBOOT drive.
    The CPLAYBOOT drive will eject itself, to be replaced by a CIRCUITPY drive.

17. Click on the Serial button to close the Serial Dialogue Panel, which is now showing stale data from the previous connection. Click on the Serial button again to reopen the panel, connected to the updated CircuitPython device.

18. Click the Save button. Save the file with the name **code.py** onto the CIRCUITPY device. (To access the Save As function, you must double click on the file tab. This process is not documented. If you save your code to your computer, it will not run on your CircuitPlayground, so you must double-click the filename, and Save As, selecting the CIRCUITPY drive.)

19. In the code area, where it says, `# Write your code here :-)`, write the following code:

    print(8)

20. Then click the Save button to save the code and run it on the CircuitPlayground.

21. You should see this message appear in the Serial Dialogue Panel:

    ```
    code.py output:
    8
    ```

22. Change the message to something else, to prove that you can. If you want to print characters instead of numbers, you must use quotation marks.

    print("hello")

23. Let's program the device to print two things, with a time delay in between the two print statements.

    ```
    import time
    print( "hello" )
    time.sleep( 0.5 )
    print( "Paul" )
    ```

24. Let's make a program that prints those two things over and over and over as long as the CircuitPlayground has power.

    ```
    import time

    while True:
       print( "hello" )
       time.sleep( 0.5 )
       print( "Paul" )
    ```

25. Python programs are Spaceholder Character Sensitive. To indent the three lines, I selected them all, and then pressed the Tab key. *To unindent, select some lines and press Shift-Tab.*

26. Cancel the program by clicking in the Serial Dialogue Panel, then press ctrl-c. Restart the program by pressing ctrl-d (which I think of as "decancel").

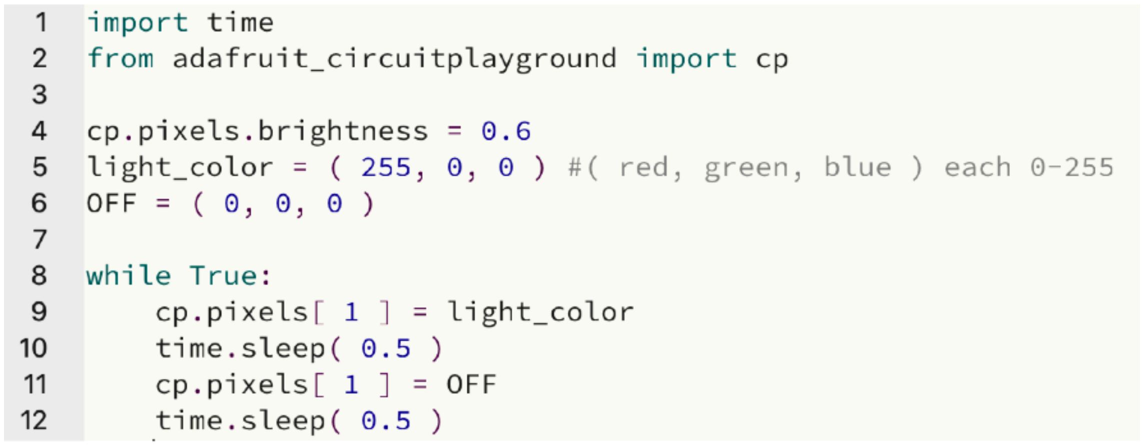
27. Programming: Libraries: Libraries are codes other people have written for you. The libraries we will be using are included as part of the CircuitPython code, but you still must import them into your program. Other libraries, for many uses, must be downloaded, and placed in a folder named **lib** on the CIRCUITPY drive.

28. Programming: Conditionals: Evaluate a condition and react. If True, do this. If False, do this other thing.

    while creates a repeated action. *while this condition is true, continue to do this thing over and over. when the condition is false, go on to the rest of the program.*

    if creates a single action. *if this condition is true, do this thing. if false, don't do it.*
    else creates a single action, when the if condition is false.

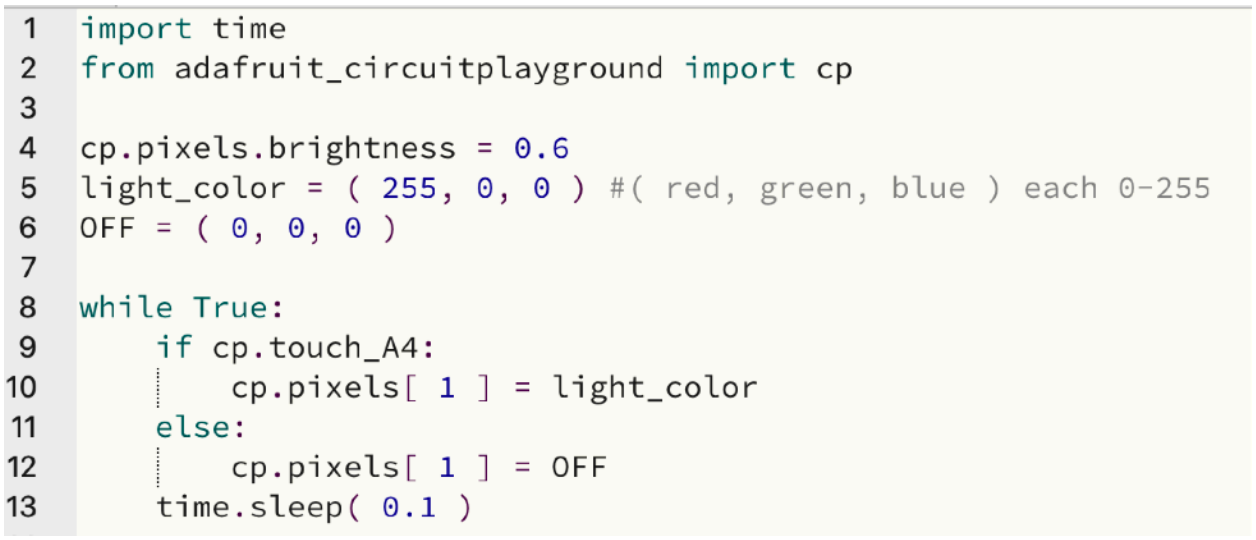
29. Blink an indicator light.



```
import time
from adafruit_circuitplayground import cp

cp.pixels.brightness = 0.6
light_color = ( 255, 0, 0 ) #( red, green, blue ) each 0-255
OFF = ( 0, 0, 0 )

while True:
    cp.pixels[ 1 ] = light_color
    time.sleep( 0.5 )
    cp.pixels[ 1 ] = OFF
    time.sleep( 0.5 )
```

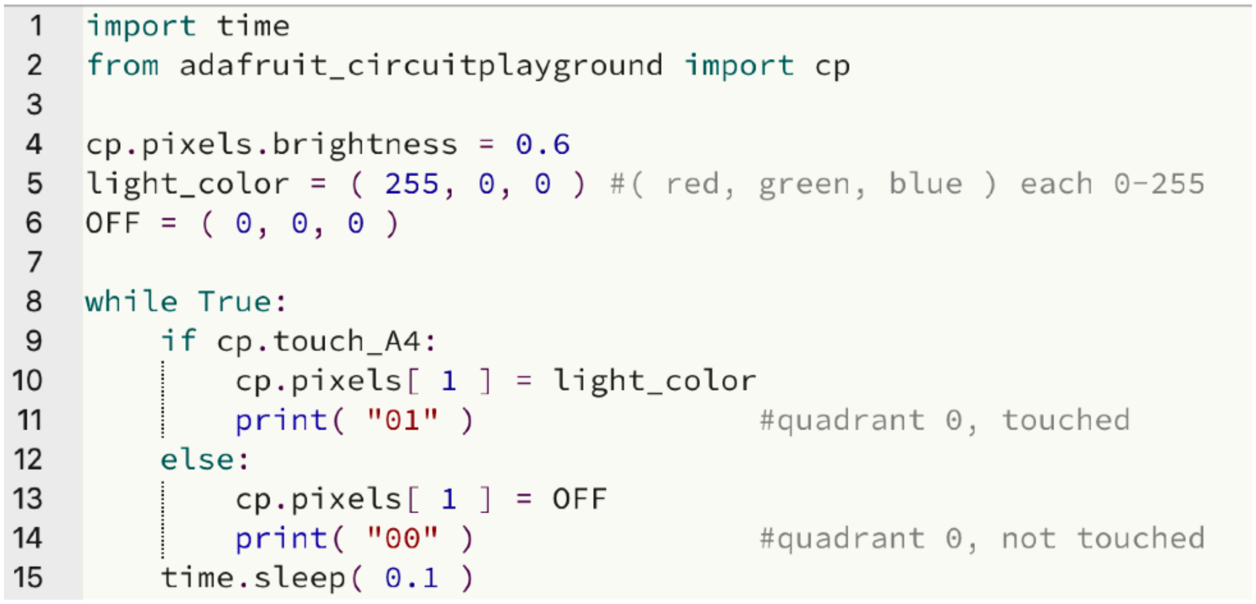
30. Save point codes can be found here, listed by the step number in these instructions: https://github.com/paulmirel/telepresence_of_touch/tree/master/circuit-playground/telepresence_of_touch_code_local/save_points/

31. Sense touch, show a local indicator.



```python
import time
from adafruit_circuitplayground import cp

cp.pixels.brightness = 0.6
light_color = ( 255, 0, 0 ) #( red, green, blue ) each 0-255
OFF = ( 0, 0, 0 )

while True:
    if cp.touch_A4:
        cp.pixels[ 1 ] = light_color
    else:
        cp.pixels[ 1 ] = OFF
    time.sleep( 0.1 )
```

32. Sense touch, show a local indicator, and send a report.



```python
import time
from adafruit_circuitplayground import cp

cp.pixels.brightness = 0.6
light_color = ( 255, 0, 0 ) #( red, green, blue ) each 0-255
OFF = ( 0, 0, 0 )

while True:
    if cp.touch_A4:
        cp.pixels[ 1 ] = light_color
        print( "01" )                #quadrant 0, touched
    else:
        cp.pixels[ 1 ] = OFF
        print( "00" )                #quadrant 0, not touched
    time.sleep( 0.1 )
```

33. Receive input.

```
value = input()
print( value )
```

34. Receive input, verbose echo.

```
value = input()
print("Received: {}".format(value))
```

35. Receive input, light a telepresent touch indicator.

```
import time
from adafruit_circuitplayground import cp

cp.pixels.brightness = 0.6
light_color = ( 255, 0, 0 )
OFF = ( 0, 0, 0 )

while True:
    value = input().strip()
    # Sometimes Windows sends an extra (or missing) newline
    # strip them off the input to ignore them
    print("Received: {}".format(value))
    if value == "01":
        cp.pixels[ 0 ] = light_color
    if value == "00":
        cp.pixels[ 0 ] = OFF
    time.sleep( 0.1 ) #pause to be able to hear a ctrl-c
```

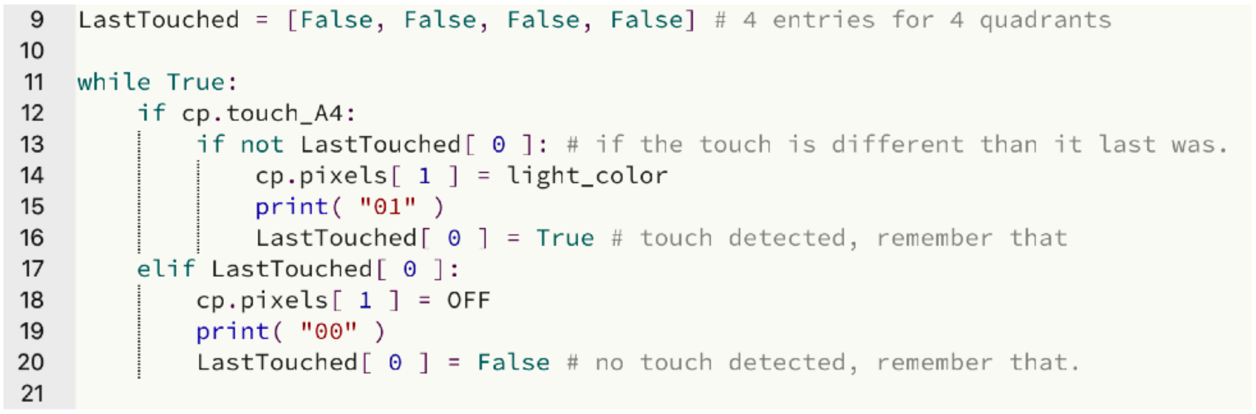
36. Check for input. Don't wait if there isn't any input.

```
1  import time
2  import supervisor
3  from adafruit_circuitplayground import cp
4
5  cp.pixels.brightness = 0.6
6  light_color = ( 255, 0, 0 )
7  OFF = ( 0, 0, 0 )
8
9  while True:
10     if supervisor.runtime.serial_bytes_available:
11         value = input().strip()
12         # Sometimes Windows sends an extra (or missing) newline
13         # strip them off the input to ignore them
14         print("Received: {}".format(value))
15         if value == "01":
16             cp.pixels[ 0 ] = light_color
17         if value == "00":
18             cp.pixels[ 0 ] = OFF
19     time.sleep( 0.1 ) #pause to be able to hear a ctrl-c
```

37. Full duplex: sense, indicate, report local touch AND indicate telepresent touch.
Lines 1-8 do not change and are not shown.

```
9  while True:
10     if cp.touch_A4:
11         cp.pixels[ 1 ] = light_color
12         print( "01" )
13     else:
14         cp.pixels[ 1 ] = OFF
15         print( "00" )
16
17     if supervisor.runtime.serial_bytes_available:
18         value = input().strip()
19         # Sometimes Windows sends an extra (or missing) newline
20         # strip them off the input to ignore them
21         print("Received: {}".format(value))
22         if value == "01":
23             cp.pixels[ 0 ] = light_color
24         if value == "00":
25             cp.pixels[ 0 ] = OFF
26     time.sleep( 0.1 ) #pause to be able to hear a ctrl-c
```

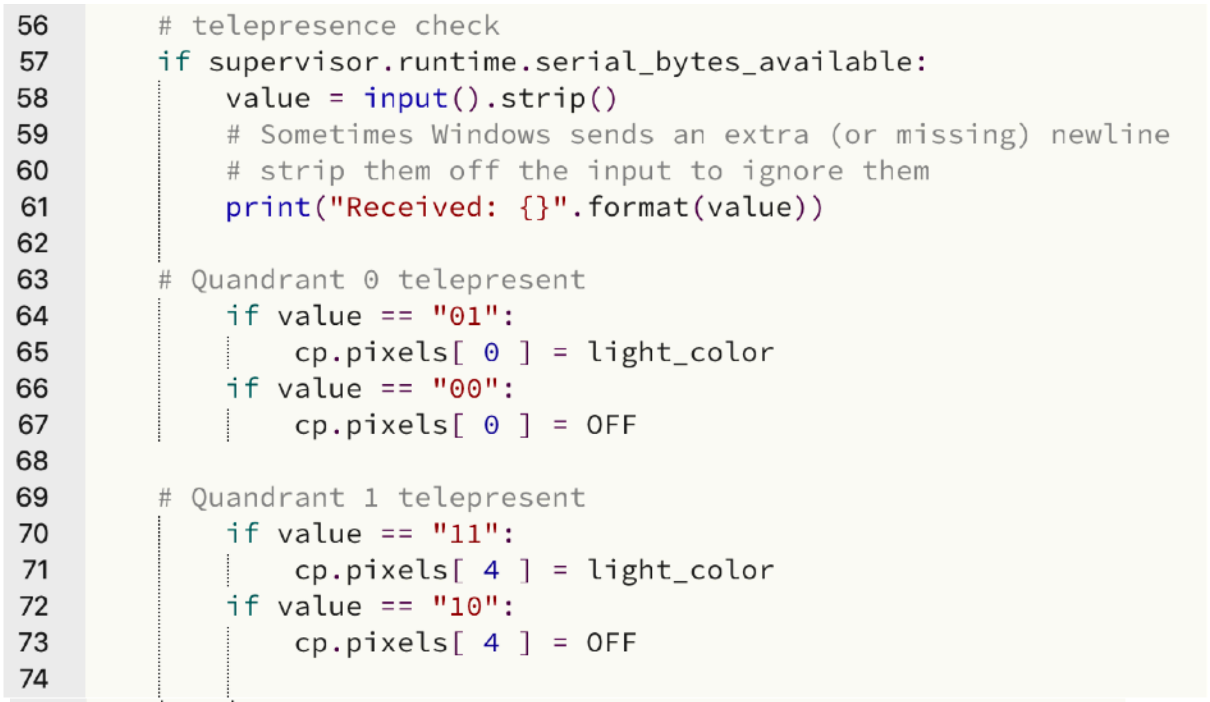
38. Report change only.



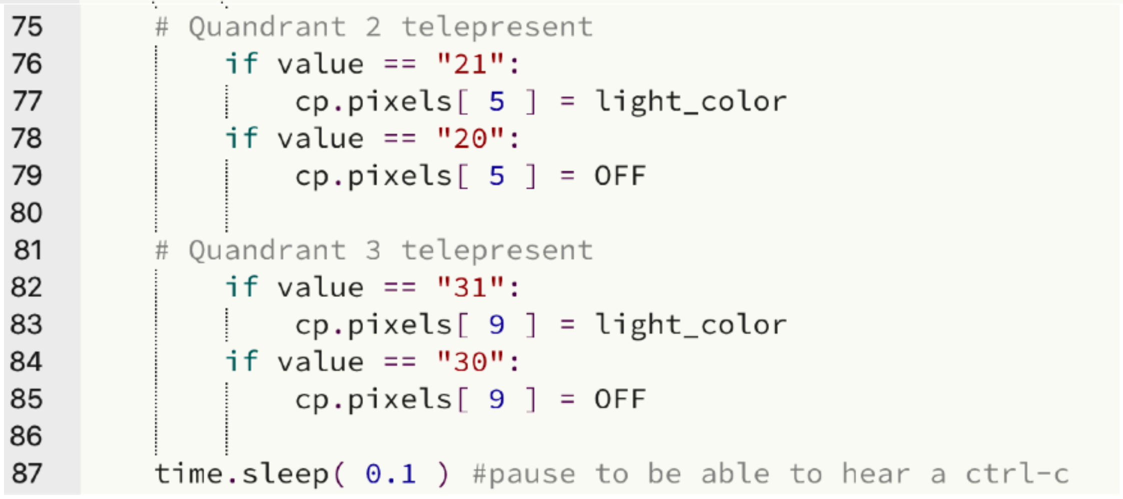
```
LastTouched = [False, False, False, False] # 4 entries for 4 quadrants

while True:
    if cp.touch_A4:
        if not LastTouched[ 0 ]: # if the touch is different than it last was.
            cp.pixels[ 1 ] = light_color
            print( "01" )
            LastTouched[ 0 ] = True # touch detected, remember that
    elif LastTouched[ 0 ]:
        cp.pixels[ 1 ] = OFF
        print( "00" )
        LastTouched[ 0 ] = False # no touch detected, remember that.

```

39. All four quadrants. Done.
    Code follows on pages 9 and 10

```
import time
import supervisor
from adafruit_circuitplayground import cp

cp.pixels.brightness = 0.6
light_color = ( 255, 0, 0 )
OFF = ( 0, 0, 0 )

LastTouched = [False, False, False, False] # 4 entries for 4 quadrants

while True:
    # Quandrant 0 local
    if cp.touch_A4:
        if not LastTouched[ 0 ]: # if the touch is different than it last was.
            cp.pixels[ 1 ] = light_color
            print( "01" )
            LastTouched[ 0 ] = True # touch detected, remember that
    elif LastTouched[ 0 ]:
        cp.pixels[ 1 ] = OFF
        print( "00" )
        LastTouched[ 0 ] = False # no touch detected, remember that.

    # Quandrant 1 local
    if cp.touch_A6:
        if not LastTouched[ 1 ]: # if the touch is different than it last was.
            cp.pixels[ 3 ] = light_color
            print( "11" )
            LastTouched[ 1 ] = True # touch detected, remember that
    elif LastTouched[ 1 ]:
        cp.pixels[ 3 ] = OFF
        print( "10" )
        LastTouched[ 1 ] = False # no touch detected, remember that.

    # Quandrant 2 local
    if cp.touch_A1:
        if not LastTouched[ 2 ]: # if the touch is different than it last was.
            cp.pixels[ 6 ] = light_color
            print( "21" )
            LastTouched[ 2 ] = True # touch detected, remember that
    elif LastTouched[ 2 ]:
        cp.pixels[ 6 ] = OFF
        print( "20" )
        LastTouched[ 2 ] = False # no touch detected, remember that.

    # Quandrant 3 local
    if cp.touch_A3:
        if not LastTouched[ 3 ]: # if the touch is different than it last was.
            cp.pixels[ 8 ] = light_color
            print( "31" )
            LastTouched[ 3 ] = True # touch detected, remember that
    elif LastTouched[ 3 ]:
        cp.pixels[ 8 ] = OFF
        print( "30" )
        LastTouched[ 3 ] = False # no touch detected, remember that.

```



```
    # telepresence check
    if supervisor.runtime.serial_bytes_available:
        value = input().strip()
        # Sometimes Windows sends an extra (or missing) newline
        # strip them off the input to ignore them
        print("Received: {}".format(value))

    # Quandrant 0 telepresent
        if value == "01":
            cp.pixels[ 0 ] = light_color
        if value == "00":
            cp.pixels[ 0 ] = OFF

    # Quandrant 1 telepresent
        if value == "11":
            cp.pixels[ 4 ] = light_color
        if value == "10":
            cp.pixels[ 4 ] = OFF

```



```
    # Quandrant 2 telepresent
        if value == "21":
            cp.pixels[ 5 ] = light_color
        if value == "20":
            cp.pixels[ 5 ] = OFF

    # Quandrant 3 telepresent
        if value == "31":
            cp.pixels[ 9 ] = light_color
        if value == "30":
            cp.pixels[ 9 ] = OFF

    time.sleep( 0.1 ) #pause to be able to hear a ctrl-c
```