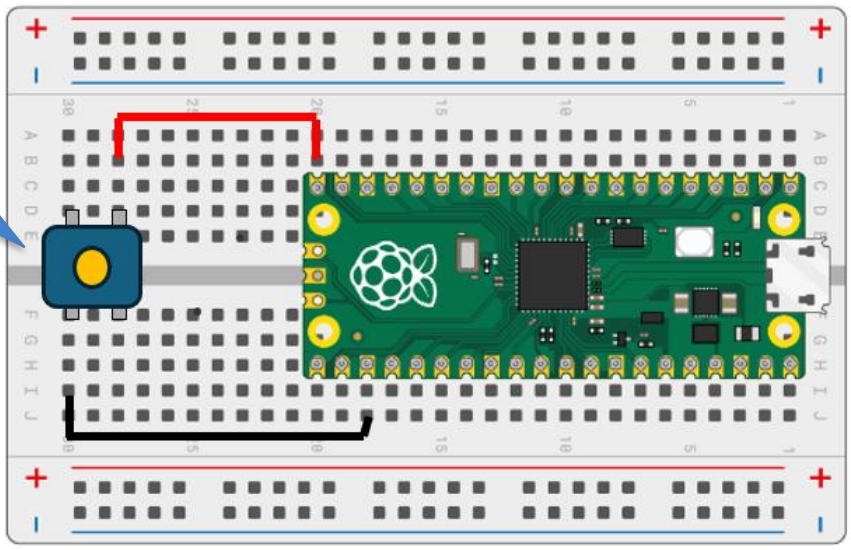
Pico 2.1 – Buttons Intro

In this tutorial we will connect a button to the the green on-board LED.
Later we can connect to the LED strip, but let's keep things simple for now.

Contents

You need a button on your breadboard for this project. Ask Mr Hughes if you do not have one in your kit



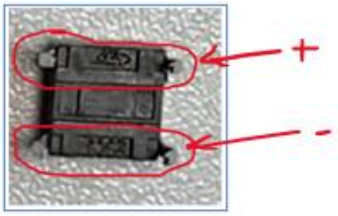
Buttons allow you to input commands.

This tutorial will show you how to attach a button to the breadboard.
We will then make the button turn the green on-board LED on and off.

Attaching a button to the Breadboard:

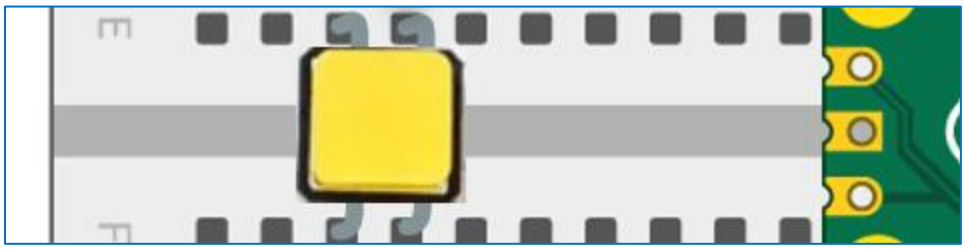
This button is a simple press to turn on, momentary button.
This button has 4 legs.
Even though it has 4 legs there are only two terminals.

This is a picture of a button, upside-down, showing the legs.



Here is the same button, viewed from a different angle:

The pins that curving towards each other = connected, treat both as the same terminal!
Attach your button to the breadboard.
Note how the curve of the button pins are positioned to straddle the breadboard centre line.

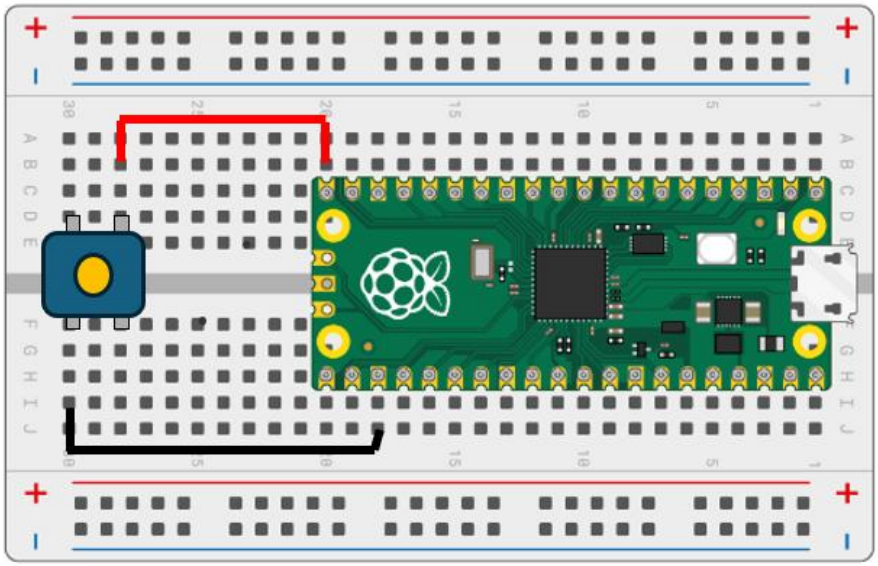


Wiring the button:

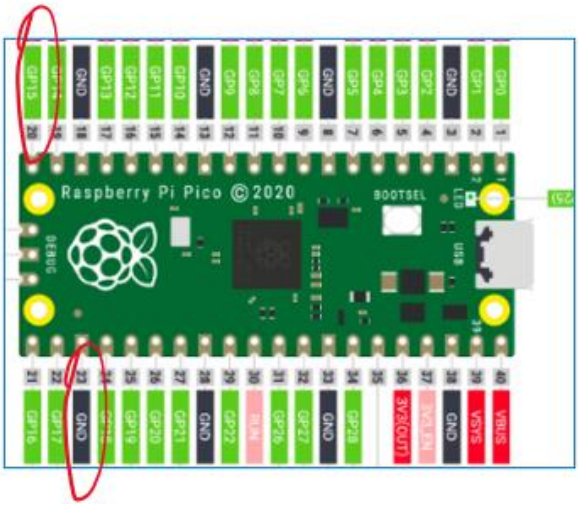
We now need to attach jumper wires to each terminal of the button.

Note that the positive and negative wires are *not*
on the same two terminals that curve towards each other.

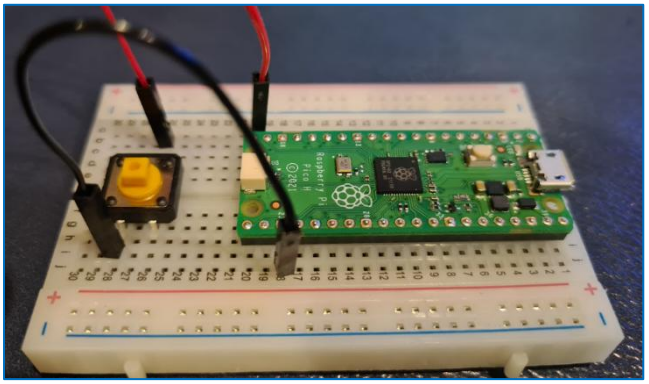
Here you can see I have connected the positive wire to pin 20 (GP15).
The black wire is connected to pin 23 (GND).



Here is a reminder of the PinOut diagram:



This is how your breadboard should look:



Test Code for Button Press:

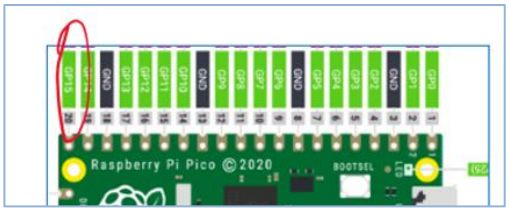
Before we go further, let's check our button is working as it should.

```
1  from machine import Pin
2  import time
3
4  #button variable
5  button = Pin(15, Pin.IN, Pin.PULL_UP)
```

Import the libraries we have used in previous tutorials.

This is referring to the GP15 Pin on the Pico. Our red button wire connects to this.

When the button is pressed it sends a signal to GP15 Pin on the Pico.



We need a while loop to constantly listen out for the button press.

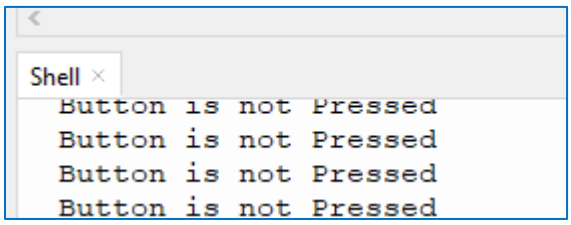
```
6
7  #commands
8  while True:
9      if button.value() == 0:
10         print("Button is Pressed")
11     else:
12         print("Button is not Pressed")
13     time.sleep(0.1)
14
```

This while loop is *listening* out for the button press.

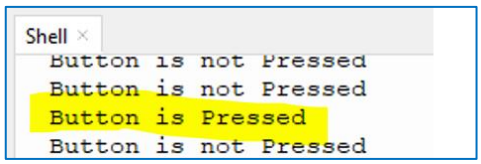
If the button IS pressed it will output "Button is Pressed".

Use this test code to see if your button is wired correctly.
Copy/paste it into Thonny.
Click the Thonny run button 

Look to the bottom of the Thonny App.
The shell should be showing this output:



Try pressing the button on the breadboard.
If it worked the output in the shell should change:



Ok, you should now have a working button.
Let's make the button press light up something.

Test Code:

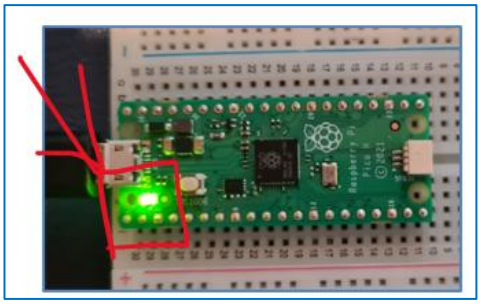
```
from machine import Pin
import time

#button variable
button = Pin(15, Pin.IN, Pin.PULL_UP)

#commands
while True:
  if button.value() == 0:
    print("Button is Pressed")
  else:
    print("Button is not Pressed")
  time.sleep(0.1)
```

Button Press and the On-Board LED

We are going to make the green on-board LED light up when the button is pressed.



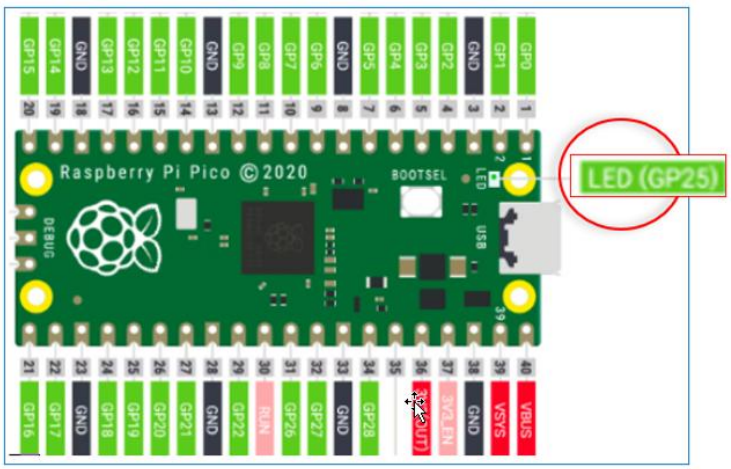
If you remember we used this green LED to test our connection in the 1.1 Quick Start guide:

We want to do something similar here, but make the LED light up when the button is pressed.

```
1  from machine import Pin
2  import time
3
4  #on board LED variable
5  led = Pin(25, Pin.OUT)
6  #button variable
7  button = Pin(15, Pin.IN, Pin.PULL_UP)
```

This new variable enables the on-board green LED (GP25)

This PinOut diagram indicates the name of the on-board LED.



We need to add commands to our while loop that turn the on-board LED on and off.

```
9  #commands
10 while True:
11     if button.value() == 0:
12         print("Button is Pressed")
13         led.value(1)
14     else:
15         print("Button is not Pressed")
16         led.value(0)
17     time.sleep(0.1)
```

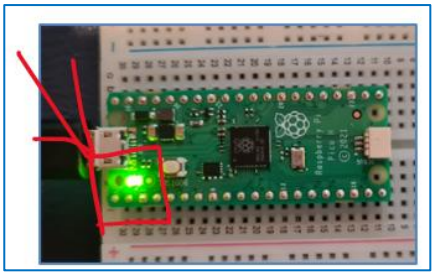
Test Code:

```
from machine import Pin
import time

#on board LED variable
led = Pin(25, Pin.OUT)
#button variable
button = Pin(15, Pin.IN, Pin.PULL_UP)

#commands
while True:
  if button.value() == 0:
     print("Button is Pressed")
     led.value(1)
  else:
     print("Button is not Pressed")
     led.value(0)
  time.sleep(0.1)
```

Did the green on-board LED light up when you pressed the button?



Challenge 1 – Make the onboard LED Flash:

- ✓ Can you make the on-board LED flash five times when the button is pressed?

You could do this as a sequence of instructions.
A more efficient way would be to use a For Loop.

Challenge 2 – Using 'Toggle':

- ✓ Can you use the 'toggle' command to make the LED change its state on button press (a YouTube video demonstrates this here)?

The **toggle** command switches the LED value from on to off, or from off to on. Whatever its current state, it will toggle it to the other.

```
9  #commands
10 while True:
11     if button.value() == 0:
12         led.toggle()
13     else:
14         print("Button is not Pressed")
15     time.sleep(0.1)
```

This is very useful if you have two buttons that are connected to the same output device. For example, I might have a light with a button by my desk, and another remote button by my door. I can use "Toggle" to make either button flip the switch from one state to another.

Challenge 3 – Counting clicks:

- ✓ Count button presses.
- ✓ Make the LED behave differently on each button press.

Can you add a new variable for **clicks.**

```
4  #counting button presses
5  clicks = 0
6  #on board LED variable
7  led = Pin(25, Pin.OUT)
```

This can be used to count button presses.

```
12 while True:
13     if button.value() == 0:
14         clicks = clicks + 1
15         if clicks == 1:
16             for x in range(10):
17                 led.toggle()
18                 time.sleep(0.3)
19         elif clicks == 2:
20             for x in range(6):
```

If the button is pressed once it will trigger one For Loop.
If it is pressed a second time another For Loop is triggered.

Test Code:

```
from machine import Pin
import time

#counting button presses
clicks = 0
#on board LED variable
led = Pin(25, Pin.OUT)
#button variable
button = Pin(15, Pin.IN, Pin.PULL_UP)

#commands
while True:
    if button.value() == 0:
        clicks = clicks + 1
        if clicks == 1:
            for x in range(10):
                led.toggle()
                time.sleep(0.3)
        elif clicks == 2:
            for x in range(6):
                led.toggle()
                time.sleep(1)
        elif clicks == 3:
            clicks = 0
        else:
            print("Button is not Pressed")
    time.sleep(0.1)
```

Challenge 4: Add more 'clicks'

Update the code so 3 clicks makes some other flashing sequence happen.

Challenge 5: Add a second button

Add a second button to your breadboard.
Use 'Toggle' to make it switch an output on and off e.g. the on-board green LED.
Make both the buttons toggle the green LED.