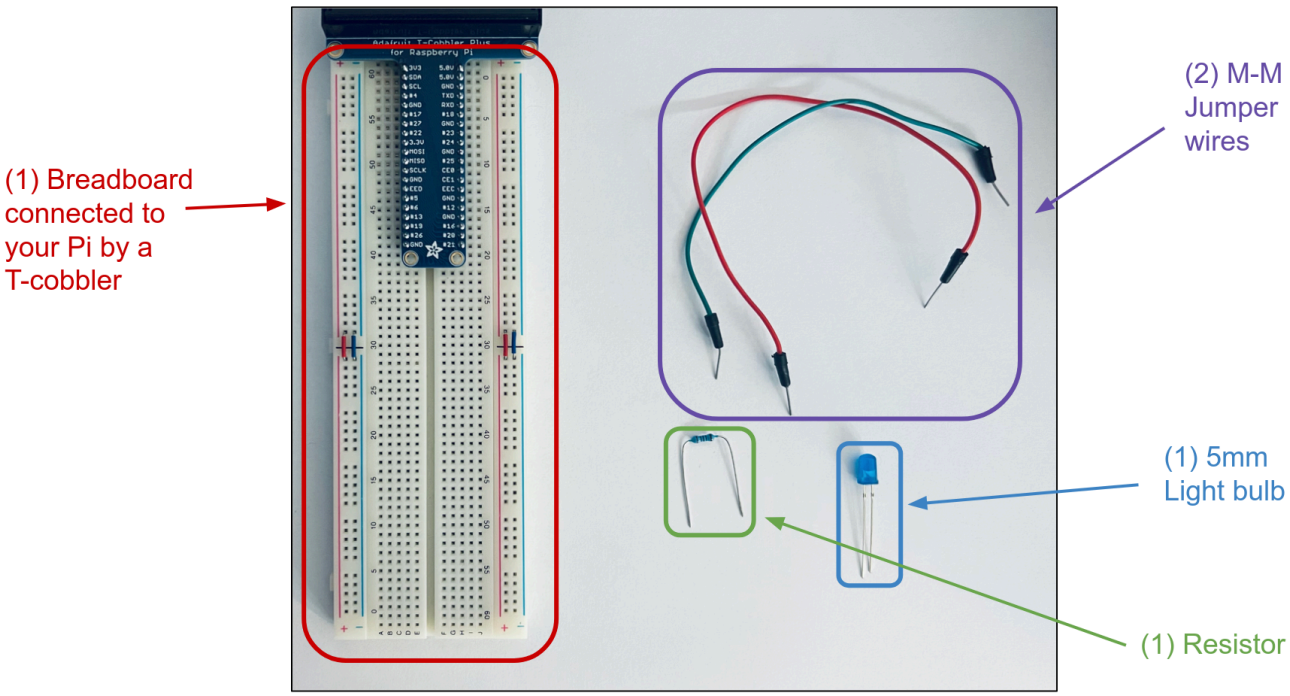
EPIC Showcase Night 2024

Part 1: Wiring Your Circuit

Gather Your Materials

You will need:

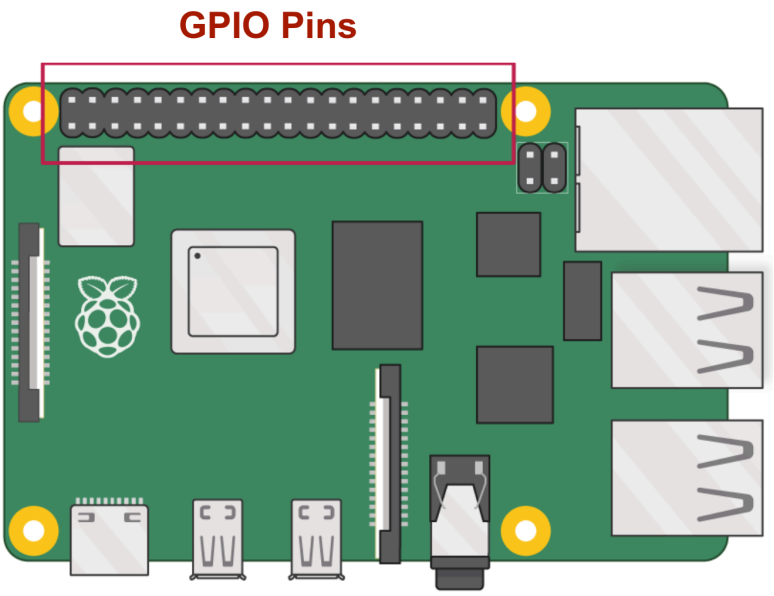


You will need:

- (1) Breadboard connected to your Pi by a T-cobbler
- (2) M-M jumper wires
- (1) 5mm Light Bulb
- (1) Resistor

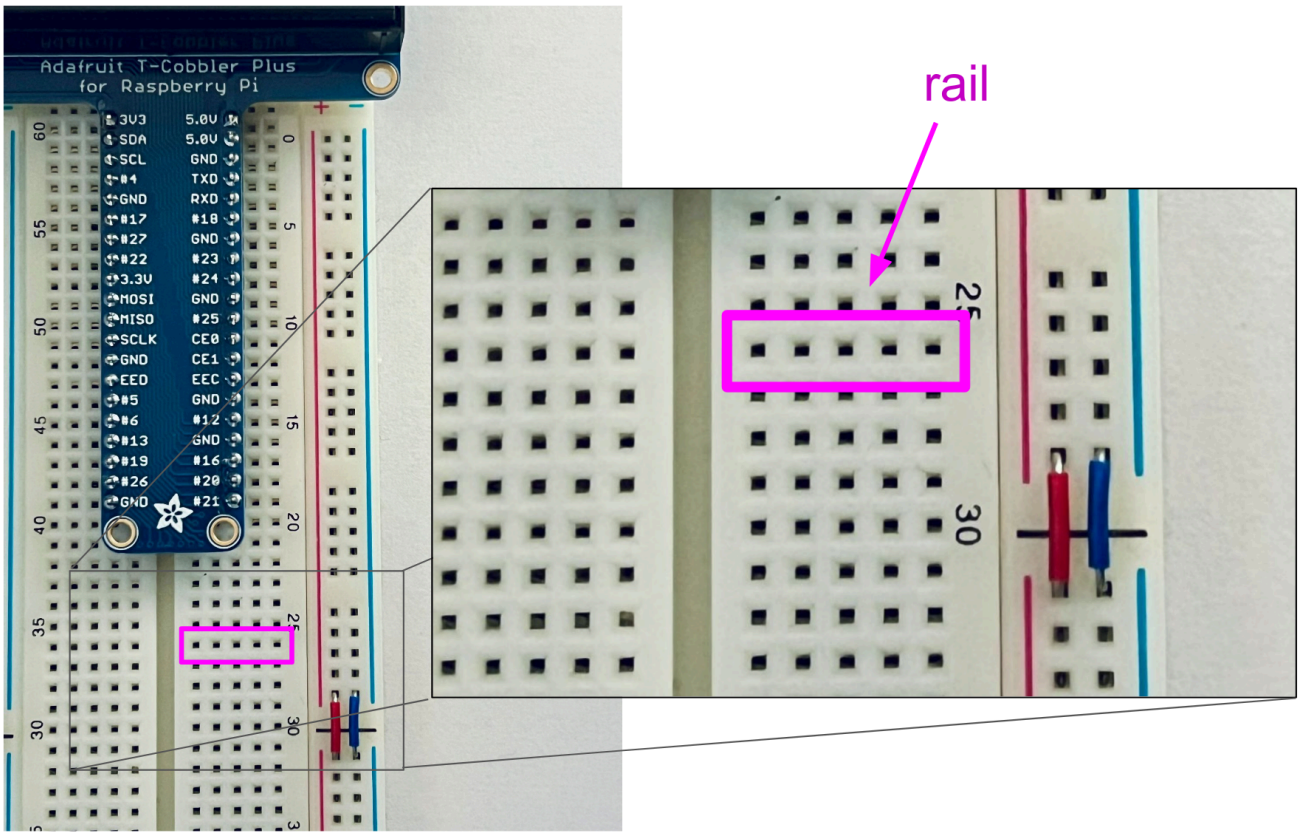
Create Your Circuit

Overview and Vocabulary



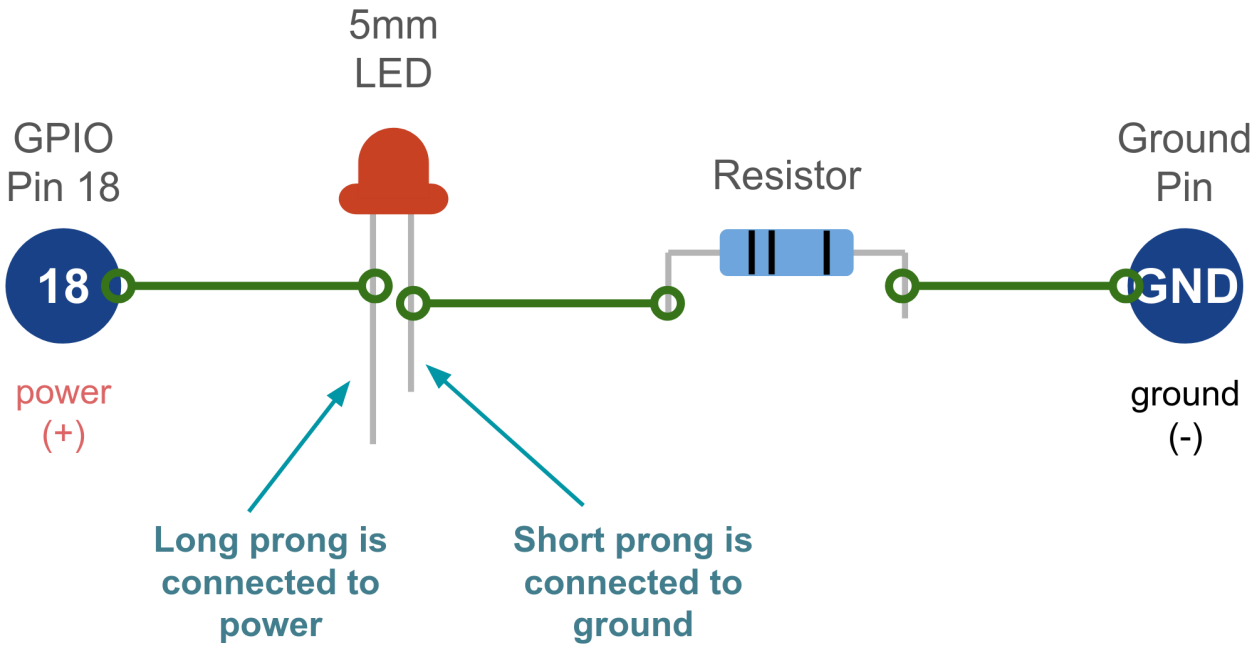
In order to turn on our light using Python, we will need to take advantage of our Raspberry Pi's **GPIO pins** or general purpose input-output pins. Some of the pins can be turned on and off using Python, which is how we will power our light.

Right now, your Pi already is connected to a **breadboard** using a **T-cobbler**. A breadboard allows us to quickly create circuits with M-M wires instead of soldering. The T-cobbler connects each of the Pi's pins to your breadboard . You can tell which pin you are using by reading the white text on the blue T-cobbler.



To connect a wire to another wire using a breadboard, plug the two wires into the same **rail** or horizontal row on the board. *Note: the horizontal rails of the breadboard are not connected across the gap in the middle or to either of the colored vertical rails on each side.*

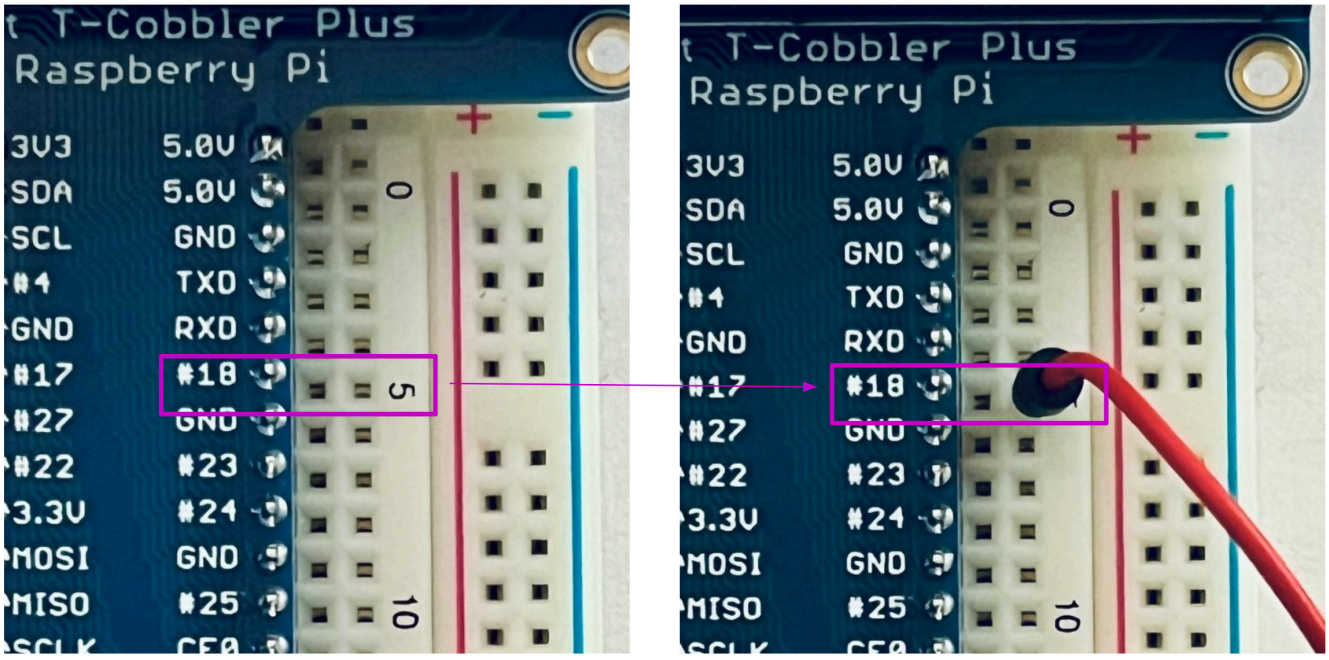
When our circuit is complete, we want it to look something like this:



Complete the following steps with the Pi off!

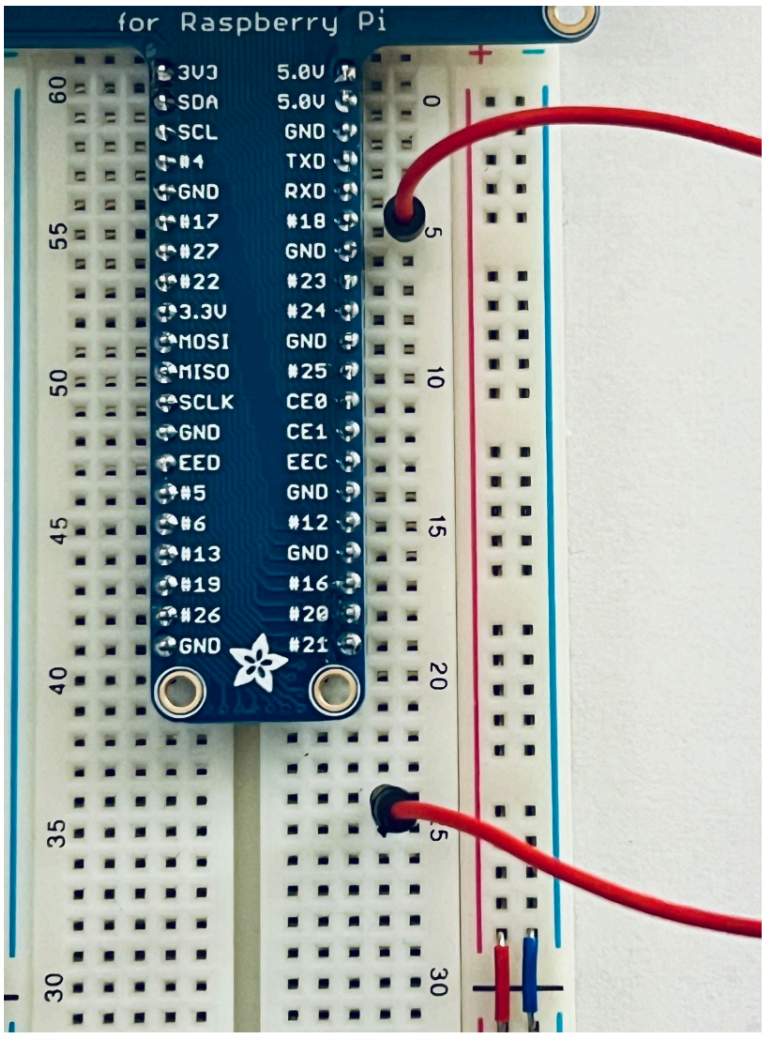
Step 1: Connect to GPIO Pin 18

1. Get one jumper wire and plug one end into the same row on your breadboard as pin 18 on your T-cobbler. *Use the numbering on the blue T-cobbler, not the breadboard.*



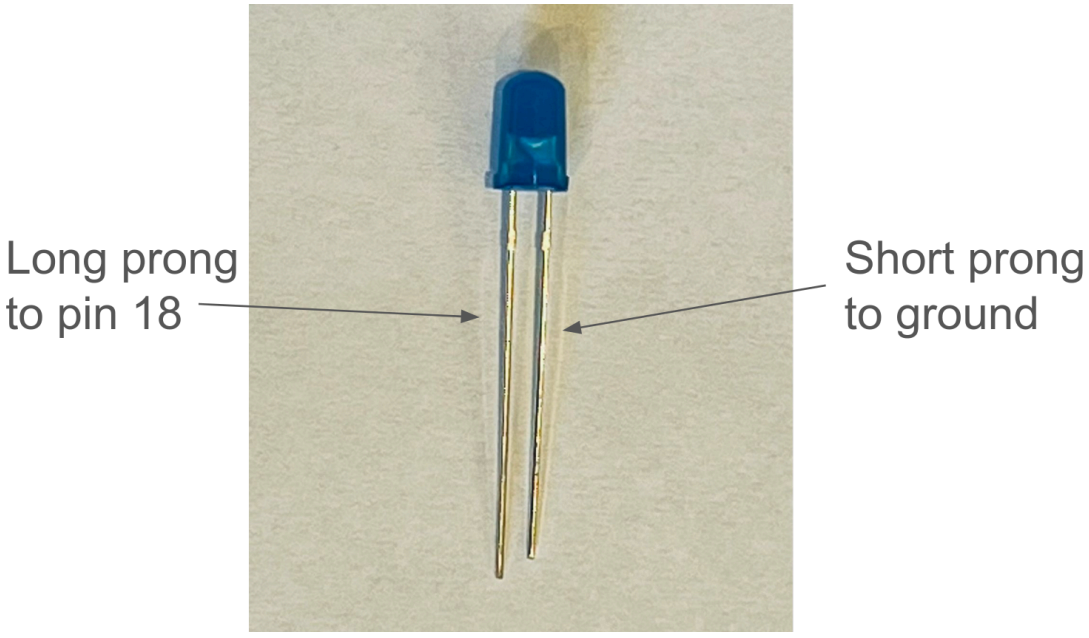
Step 2: Plug the Other End into an Empty Rail

1. Connect the other end of the wire you just connected into an empty rail of the breadboard. **DO NOT** put it in the same row as any other piece of your T-cobbler.

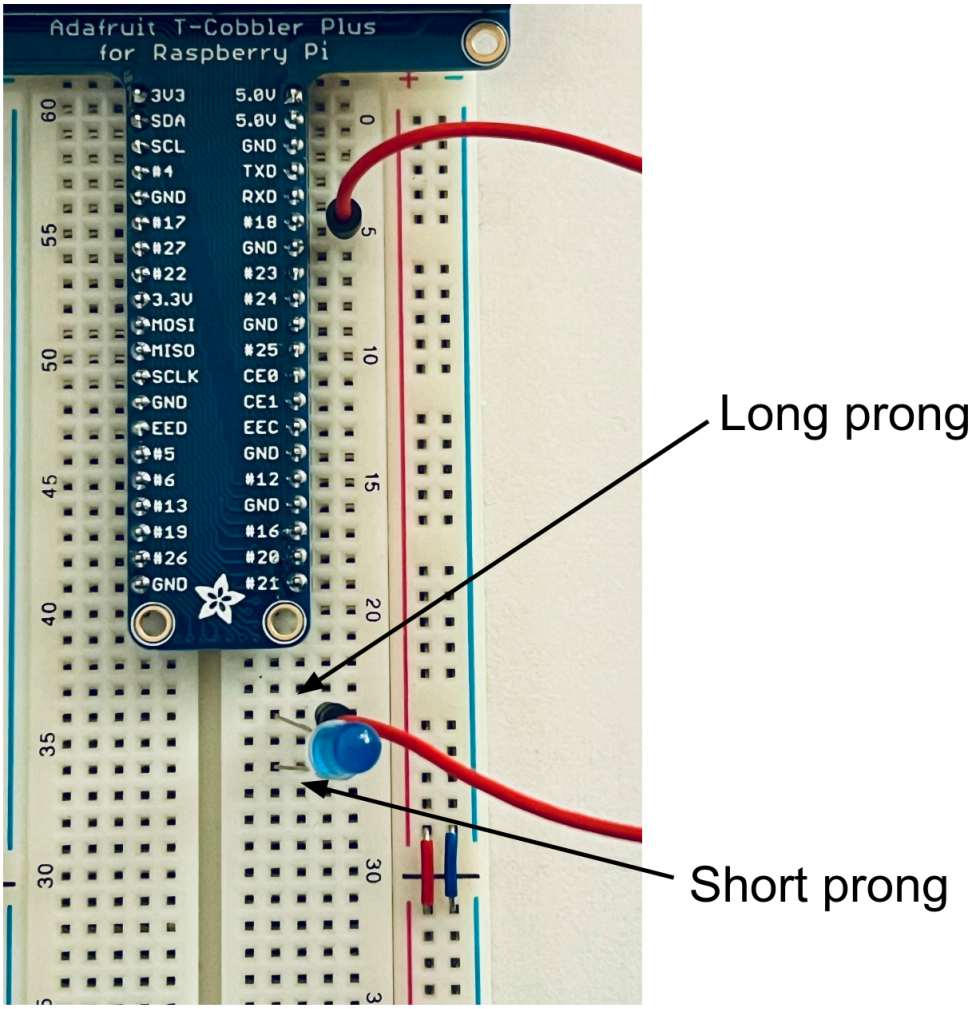


Step 3: Connect Your Light

1. Put the **long end of your light bulb** into the same row as the wire in step 2. If you connect the short end instead, your light will not turn on.



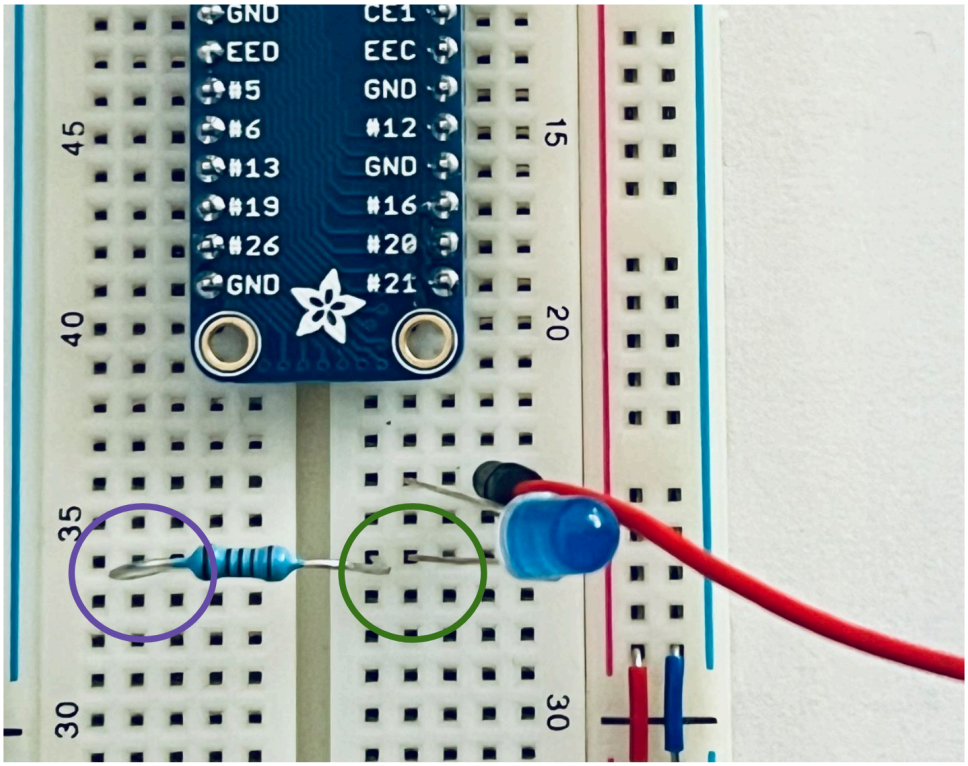
2. **DO NOT** put the short end of the light bulb in the same row, **instead** put it in an adjacent row like the picture below.



3. Push down on the light to make sure it is actually in the breadboard. You may need to apply a little bit of force.

Step 4: Connect the Resistor

1. Connect one end of the resistor to the same row as the short end of your light, as shown in green below.

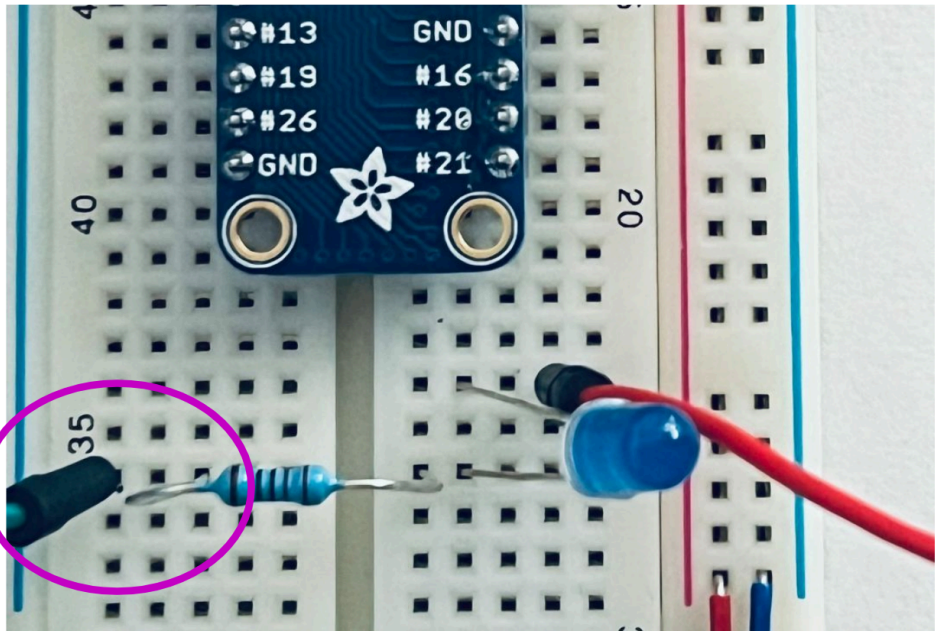


2. Connect the other end to an empty rail. This can be any empty rail; it does not have to be on the other side of the breadboard.

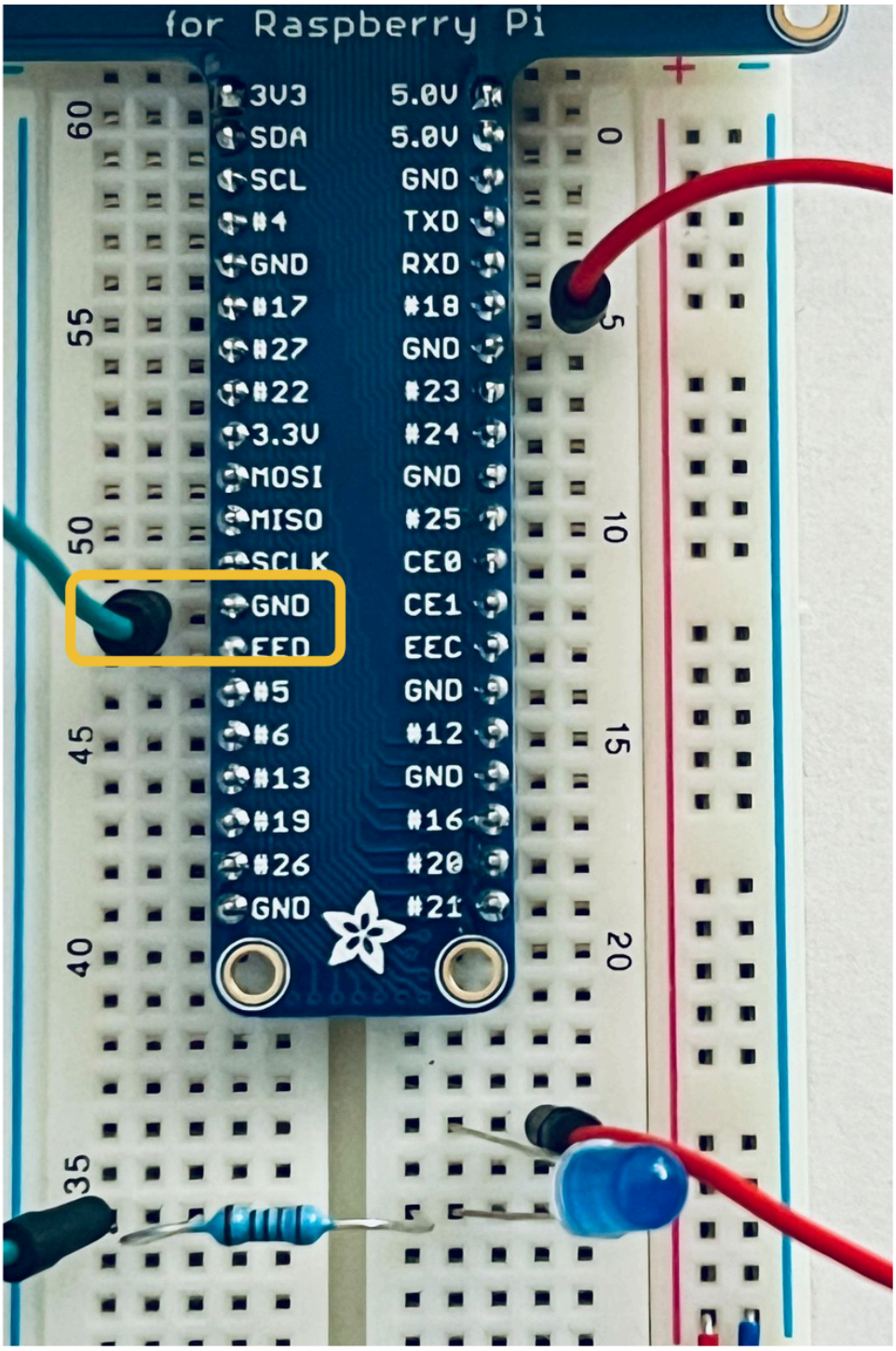
DO NOT PUT BOTH ENDS OF THE RESISTOR IN THE SAME RAIL AS YOUR LIGHT. Because all wires in the same rail are connected, electricity will not pass through the resistor. This can cause your light bulb to explode.

Step 5: Connect the Last Wire

1. Put one end of your last jumper wire into the same rail as the end of the resistor that is in an empty row.



2. Then put the other end next to any pin in your T-cobbler labelight **GND** for Ground.



Get your circuit checked before continuing!

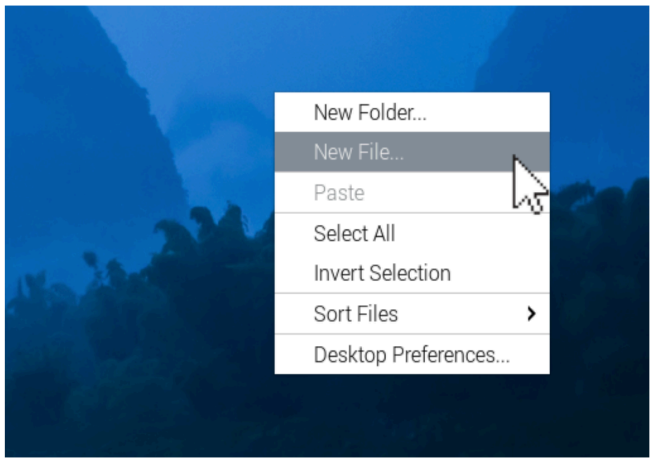
Part 2: Programming your Raspberry Pi

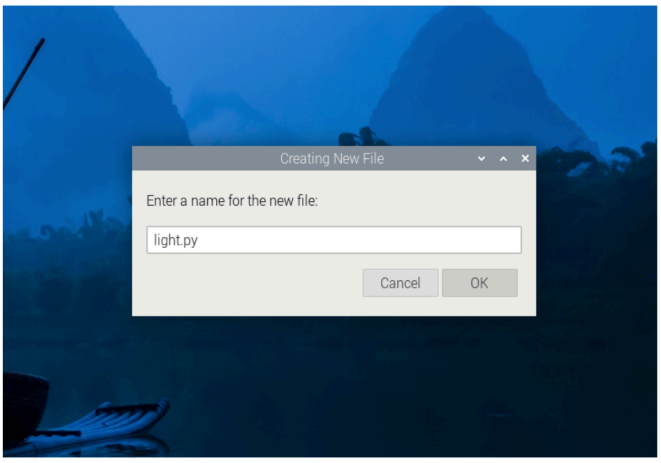
Now that your circuit is built, it is time to turn your Pi back on and get programming!

Step 0: Login to your Pi

1. Power on your Pi by plugging it back in and login.
2. The username is `pi` and the password is `raspberry`.
3. When you log in, it will warn you that you have an insecure password; you can ignore the warning.

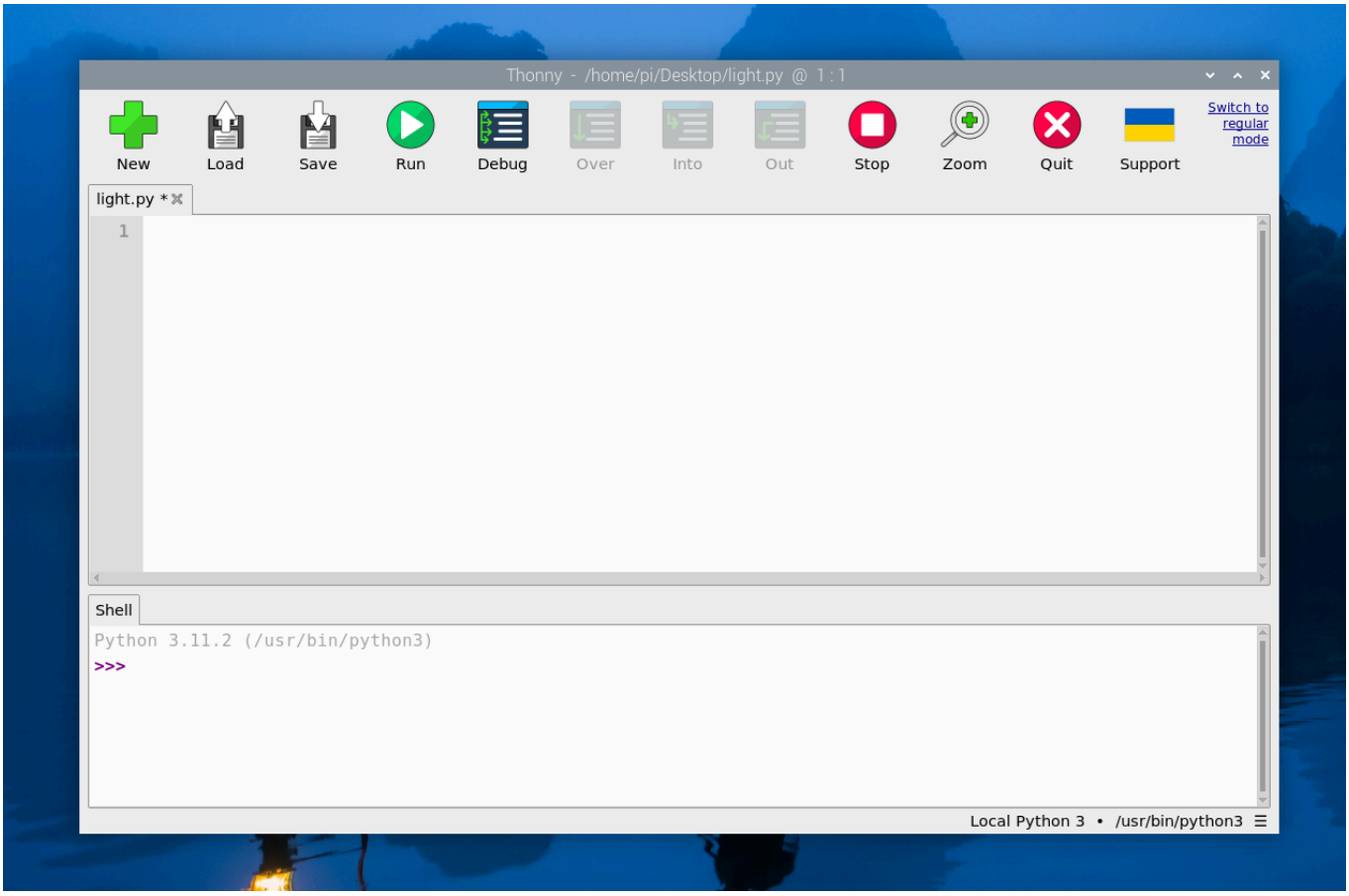
Step 1: Create a new file called `light.py`





1. Right click on the Desktop and select `New File`.
2. Create a new file called `light.py`.
    - The `py` extension is important here so we create a Python file instead of a regular text file.

Step 2: Edit your Python File



1. Right click on your file and open it in **Thonny**.

Programs in Python execute line-by-line starting from the top of the file to the bottom. We are going to write a program to set up our GPIO pins, turn on a light, wait 10 seconds, then turn it off again.

A few quick tips about programming in Python:

- If you want your program to stop running at any point, click on the shell and use `CTRL-C` to stop your program.
- `#` characters are used to create comments or non-executable lines. Anything after `#` on a line will be ignored by the interpreter, which is why programmers use them to leave notes for themselves in their code.

2. Set up your GPIO pins in your Python program. At the beginning of your program, add the following lines:

```
import RPI.GPIO as GPIO #Import the GPIO library so we can use the GPIO pins
import time #Import the time library so we can add time delays to our code

GPIO.setmode(GPIO.BCM) #Set the board mode of the GPIO pins so we can use the numbering on the
Pi
GPIO.setwarnngs(False) #Set all warnings False to start
```

3. Now, we are going to make our light bulb turn on using pin 18

```
GPIO.setup(18, GPIO.OUT) #Set 18 to be an output pin (as opposed to an input pin)

GPIO.output(18, GPIO.HIGH) #Set the output of 18 to high(turn on the pin)
time.sleep(5) #Wait for 5 seconds with the light on
GPIO.output(18, GPIO.LOW) #Set the output of 18 to low(turn off the pin)


GPIO.cleanup() #Turns all the pins off at the end
```

4. Save and run your program! Your light should turn on for 5 seconds and turn off again.

Troubleshooting

Fixing bugs is a huge part of computer science, so don't be discouraged if it doesn't work the first time!

If your program isn't working, here are some things to look for:

- *Did you spell everything correctly and use the correct syntax?* - If you misspell anything or use slightly different syntax, your program will not run.
- *Are any of your lines indented by mistake?* - Indentation has a specific purpose in Python, so we can't indent random lines. If any of your lines start with extra space(s), your program may error
- *Did you save your file?* - Thonny does not autosave your files, so you will need to save them manually every time you make changes. Use `CTRL-S` or the `Save` button to save your file.
- *Is your circuit correct?* - If your program is running but the light doesn't turn on, you may have an issue with your circuit.

If you are still having trouble, try running the `troubleshoot.py` file on the Desktop from Thonny.

Step 3: Have Some Fun!

Now that your program works, try modifying it! You could try making it blink a couple times or maybe even communicate a message in Morse code! Or if you want some practice with hardware, come get another light bulb to add a second light.