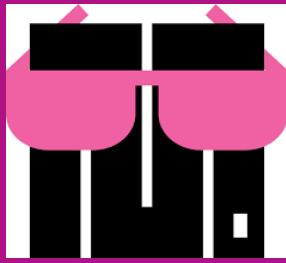
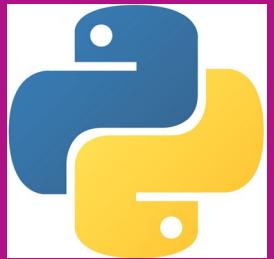
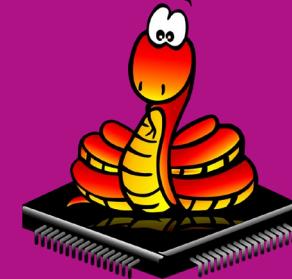


INTERNET OF THINGS (IOT) PROJECTS USING PYTHON

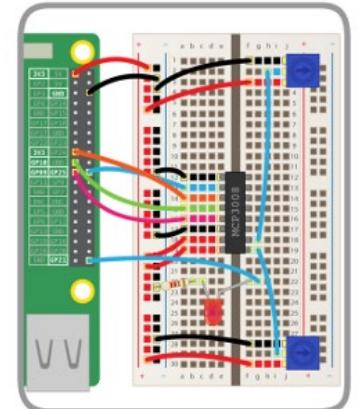
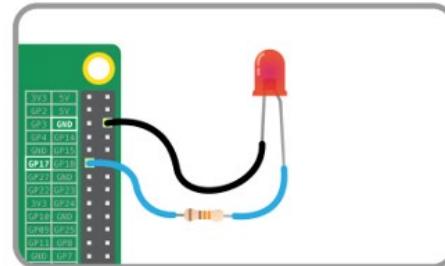
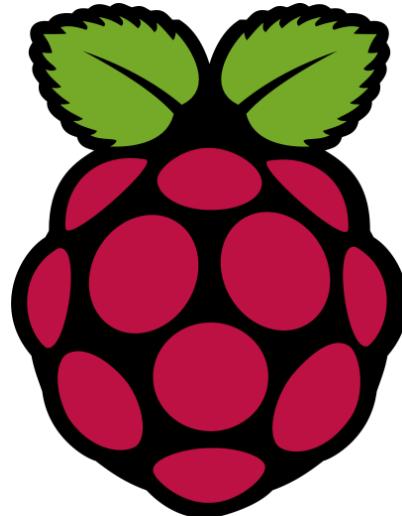


(CSE 4110)

(LECTURE – 4)



T_h



Getting Started with Raspberry Pi Pico

Thonny Python IDE - Windows

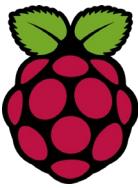
Configure and install
Raspberry Pi
Pico MicroPython
interpreter in Thonny IDE.



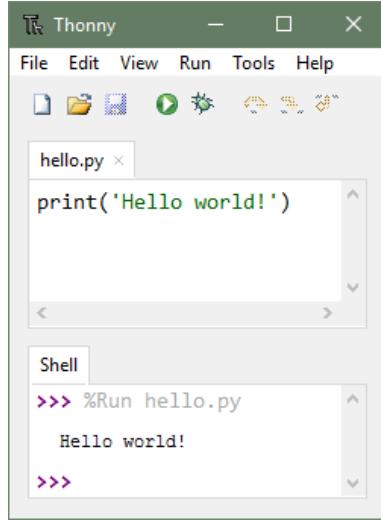
Latest Thonny IDE
comes with preinstalled
Micropython interpreter
for Raspberry Pi Pico



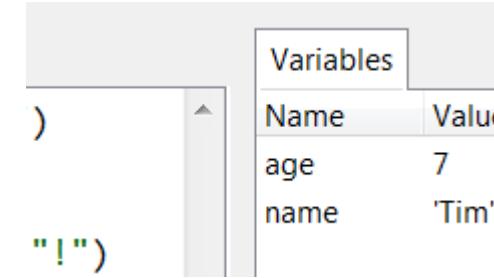
Why Thonny? Python IDE for beginners <https://thonny.org/>



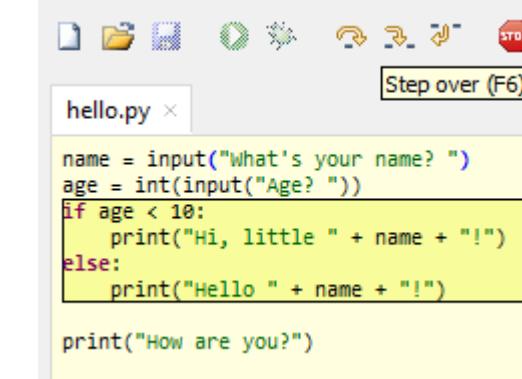
1. Easy to get



2. No-hassle variables.



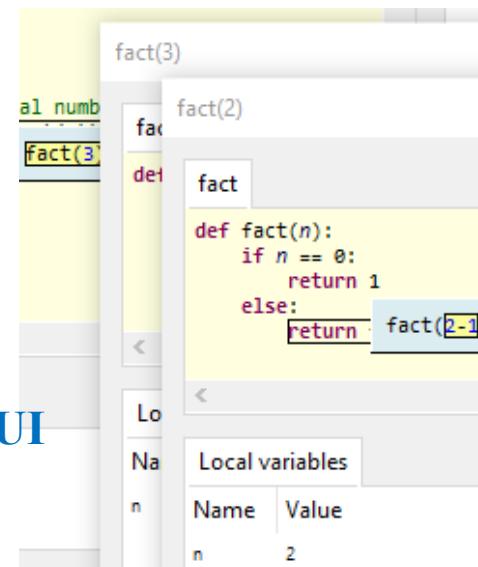
3. Simple debugger.



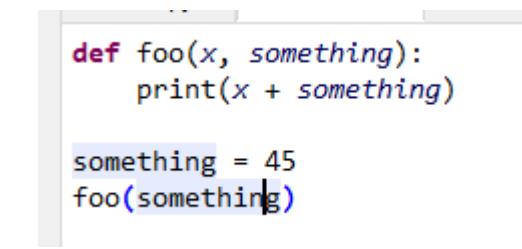
4. Step through expression evaluation

```
name = input("What's your name? ")
age = int(input("Age? "))
if age < 10:
    print('Hi, little ' + name + "!")
else:
    print("Hello " + name + "!")
print("How are you?")
```

6. Faithful representation of function calls



7. Explains scopes.



8. Mode for explaining references.

Variables	
Name	Value ID
a	0x94A440
b	0x94A440

Heap	
ID	Value
0x94A440	[1, 2, 3]

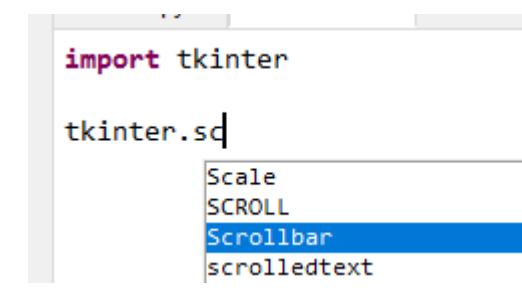
5. Highlights syntax errors.

```
first_name = "Albus"
last_name = "Dumbledore"

result = math.pi * (34 + 12)
```

10. Simple and clean pip GUI

11. Code completion



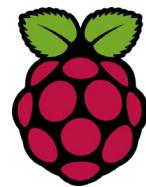
12. Beginner friendly system shell.

Thonny Micro-Python IDE Installation

Method - 1



Installing Thonny and Python separately (windows)



Open cmd, then type `pip install thonnyapp`

`pip install -U thonnyapp`

OR

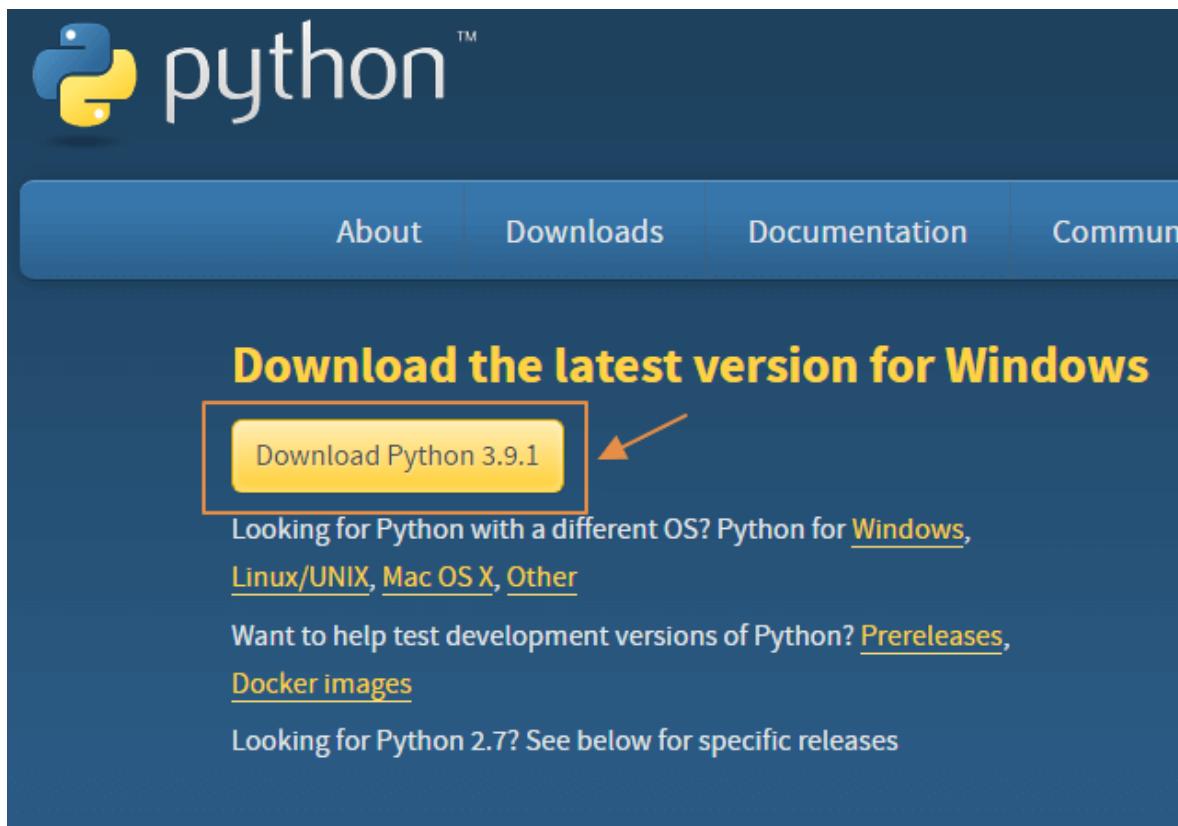
Download and Install Thonny IDE.

Thonny
Python IDE for beginners

 Download version [4.0.1](#) for
Windows • Mac • Linux

Step 1: Download and install Python

Visit the python website and download the latest version of python for Windows. Install both and we can proceed to the next step.



The screenshot shows the Python website's "Downloads" section. A yellow call-to-action button labeled "Download Python 3.9.1" is highlighted with a red arrow pointing to it. Below the button, text links to "Python for Windows", "Linux/UNIX", "Mac OS X", and "Other". Further down, links for "Prereleases" and "Docker images" are shown, along with a note for Python 2.7 users.

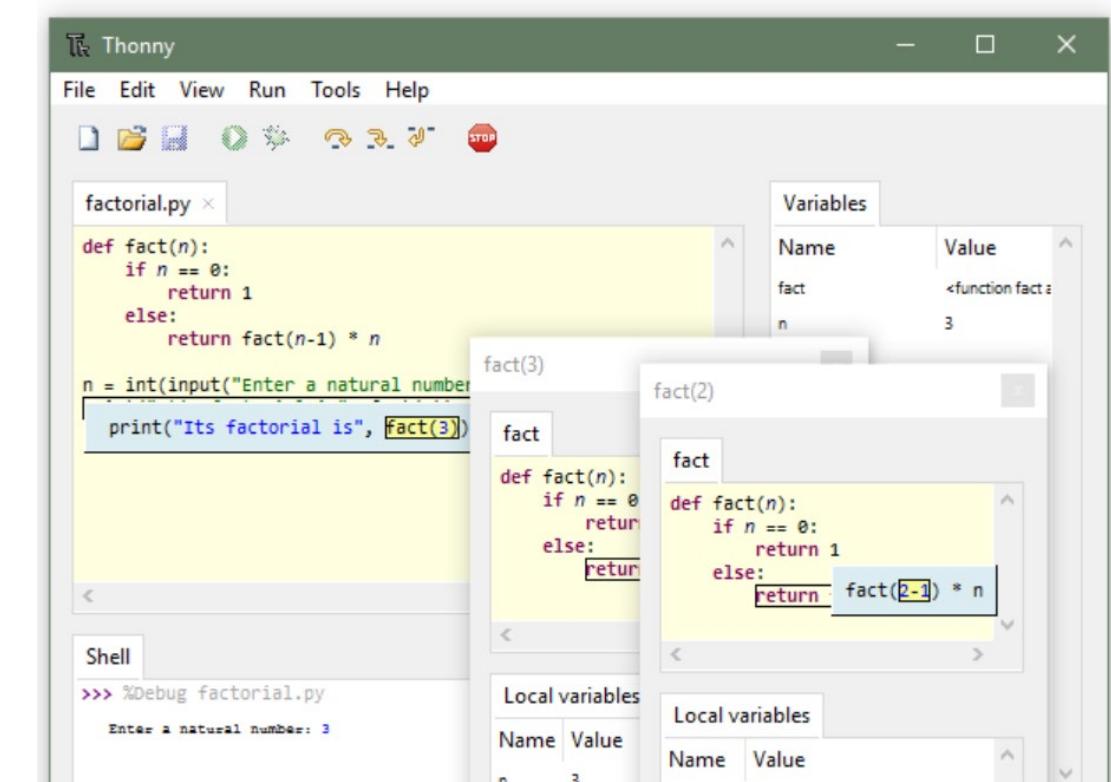
Download the latest version for Windows

Download Python 3.9.1

Looking for Python with a different OS? Python for [Windows](#), [Linux/UNIX](#), [Mac OS X](#), [Other](#)

Want to help test development versions of Python? [Prereleases](#), [Docker images](#)

Looking for Python 2.7? See below for specific releases



The screenshot shows the Thonny IDE interface. On the left, a code editor window titled "factorial.py" contains the following Python code:

```
def fact(n):
    if n == 0:
        return 1
    else:
        return fact(n-1) * n

n = int(input("Enter a natural number"))
print("Its factorial is", fact(n))
```

In the center, a terminal window titled "Shell" shows the command `>>> %Debug factorial.py` and the user input `Enter a natural number: 3`. To the right, several floating windows show the execution stack of the factorial function, with local variable values for `n` and `fact` at each level of recursion.

Installing Thonny and Python separately (windows)

After installing Python, open a Windows command prompt and enter the following command (assuming you installed Python to C:Python39) and hit ENTER:

pip install thonnyapp

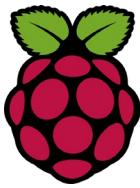
This command installs the latest thonny and thonnyapp packages to your Python and also creates shortcuts on the Desktop and in the Start menu.

If you subsequently want to update Thonny, then open the command prompt again and enter:

pip install -U thonnyapp

Thonny Micro-Python IDE Installation

Method - 2



Micro-Python Installation

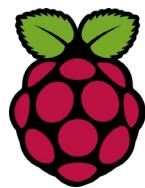
Step 2: Now Press the BOOTSEL pin on your Pico and Connect the Raspberry Pi Pico and you can able to see the Pico Mounted as a Flash Drive. Open the Index.html file and it will head over to raspberry pi pico page. Here we need to download the UF2 File to your computer (for now).

The screenshot shows the official Raspberry Pi Pico documentation. At the top, there are three tabs: 'Board specifications', 'Getting started with MicroPython' (which is highlighted with an orange border and has an arrow pointing to it from the text above), and 'Getting started with C/C++'. Below the tabs, the 'Getting started with MicroPython' section is expanded. It contains a large red '1' icon and the heading 'Drag and drop MicroPython'. The text explains that you can program your Pico by connecting it to a computer via USB and dragging and dropping a file onto it. It also mentions a 'downloadable UF2 file' for installation. To the right of this text is a photograph of a green Raspberry Pi Pico board with its pins visible. At the bottom right of the expanded section is a green button with white text that says 'Download UF2 file' with a right-pointing arrow.

1. Download the MicroPython UF2 file by clicking the button below.
2. Push and hold the BOOTSEL button and plug your Pico into the USB port of your Raspberry Pi or other computer. Release the BOOTSEL button after your Pico is connected.
3. It will mount as a Mass Storage Device called RPI-RP2.
4. Drag and drop the MicroPython UF2 file onto the RPI-RP2 volume. Your Pico will reboot. You are now running MicroPython.

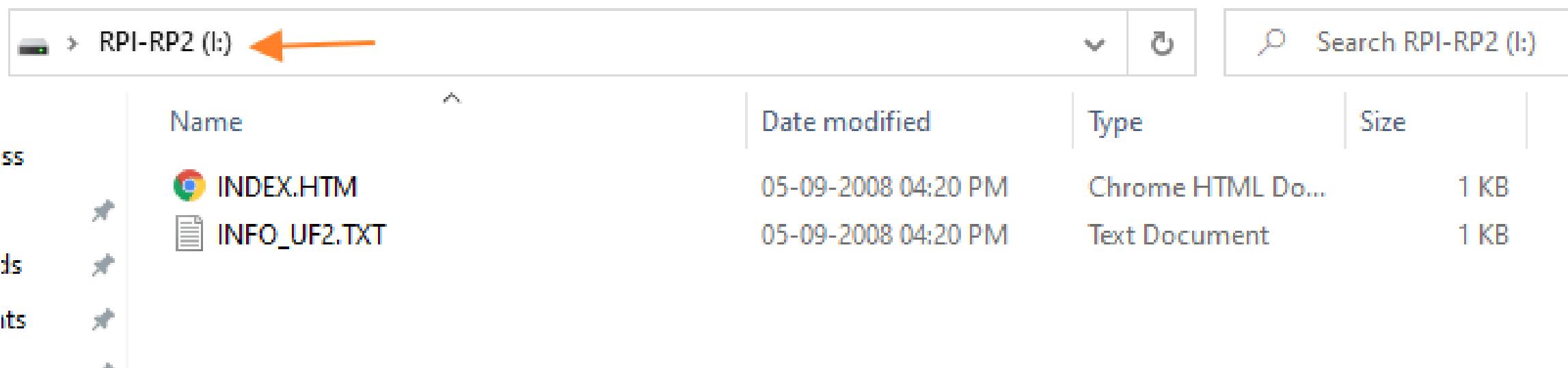
You can access the REPL via USB Serial. Our [MicroPython documentation](#) contains step-by-step instructions for connecting to your Pico and programming it in MicroPython.

[Download UF2 file →](#)

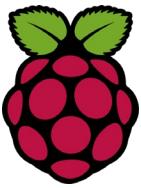


Micro-Python Installation

Step 3: Drag and Drop or Copy the UF2 File that we downloaded on your computer to the Pico. Now the pico will eject itself. Now the micropython is running in our Raspberry Pi Pico. We can able to see that pico device on our Thonny IDE

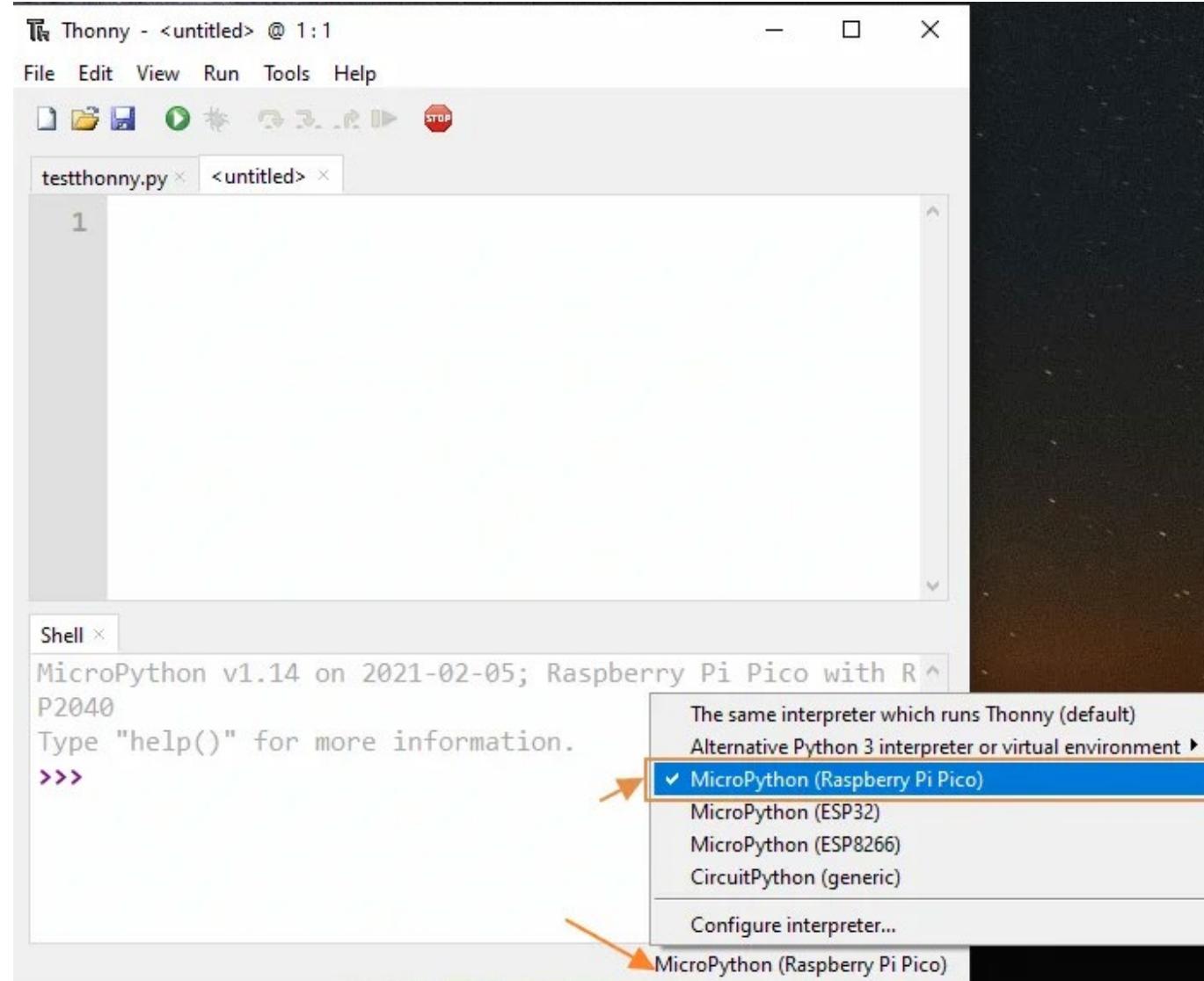


	Name	Date modified	Type	Size
ss	INDEX.HTM	05-09-2008 04:20 PM	Chrome HTML Do...	1 KB
ds	INFO_UF2.TXT	05-09-2008 04:20 PM	Text Document	1 KB
its				



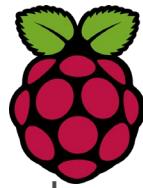
Thonny IDE Configuration

Step 4: Open Thonny IDE, and at the right bottom of the window click the button and you can able to see 'MicroPython (Raspberry Pi Pico)' select it. Now we are ready to code.



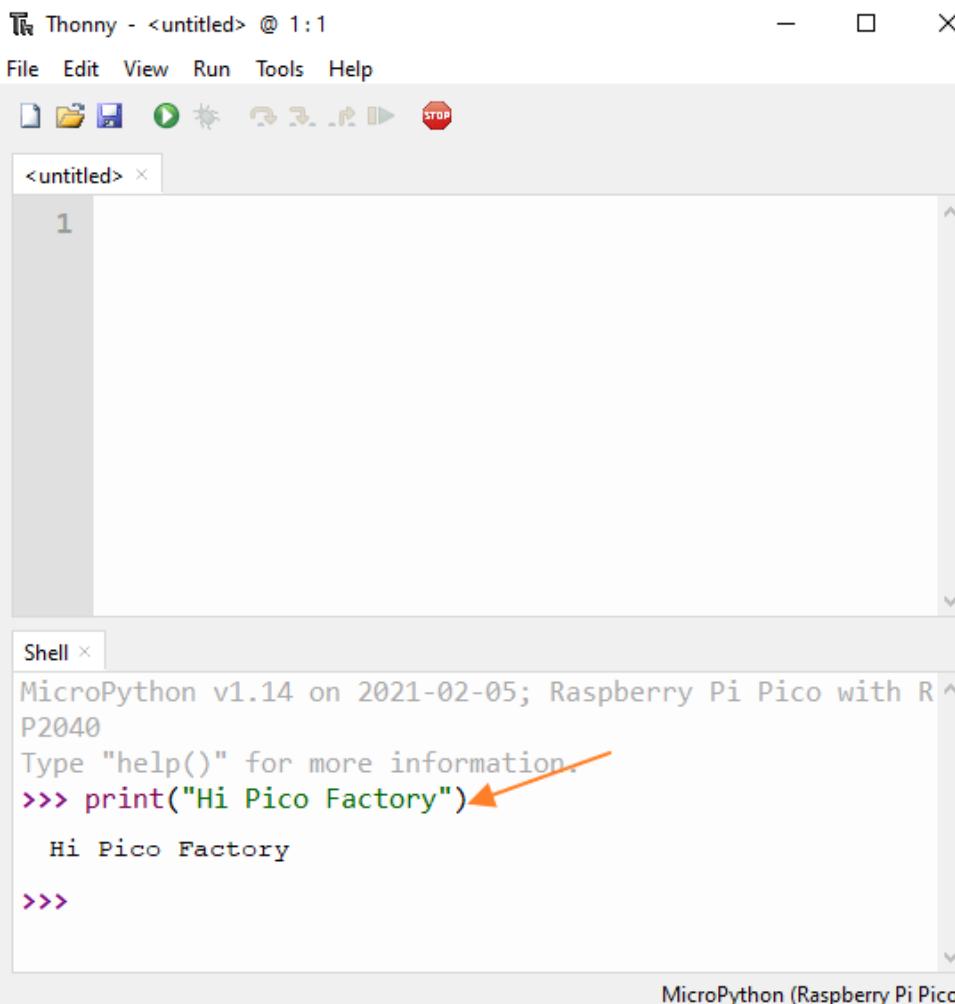


Writing our First Program using Raspberry Pi



Step 5: We can use shell or the python file to code our raspberry pi. Using shell you can test line by line code. Using python file, you can write complete code, then run it using the Pico.

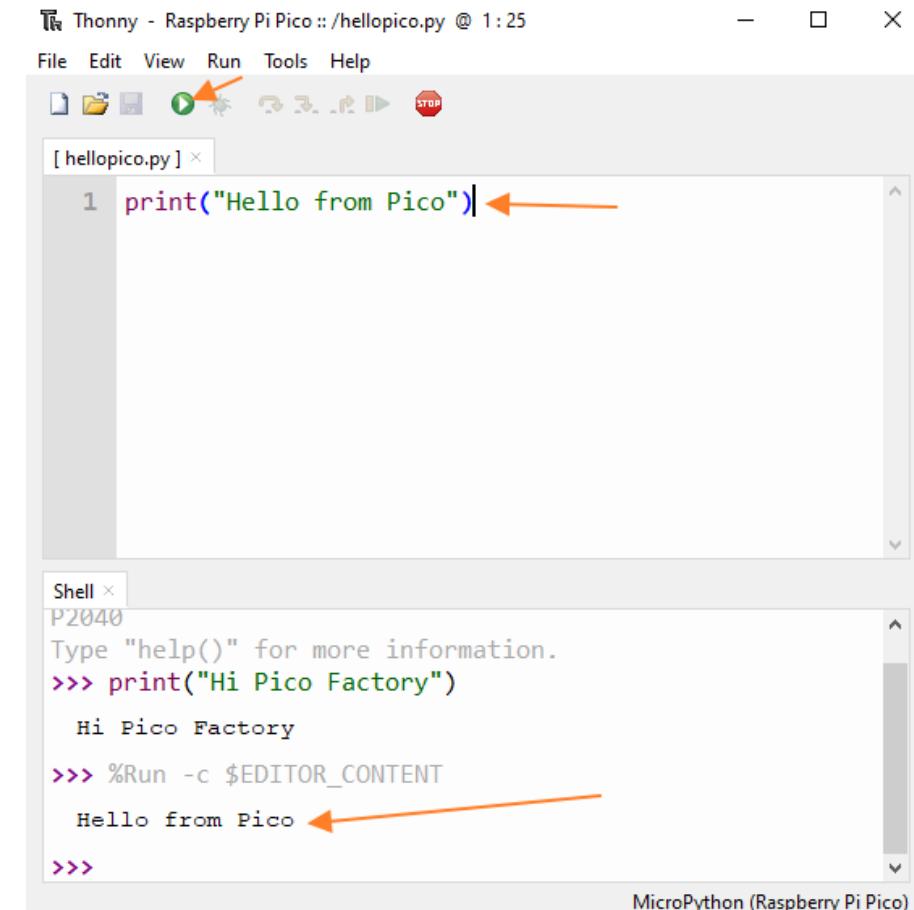
Printing 'Hi Pico Factory' on Shell



```
Thonny - <untitled> @ 1:1
File Edit View Run Tools Help
<untitled> x
1

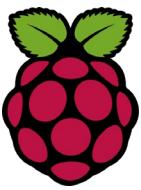
Shell x
MicroPython v1.14 on 2021-02-05; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> print("Hi Pico Factory")
Hi Pico Factory
>>>
MicroPython (Raspberry Pi Pico)
```

Let's try to run this on our Pico. Write 'Hello from Pico' on the code area and click the Play Icon in the IDE. Now the result will be printed on the output.



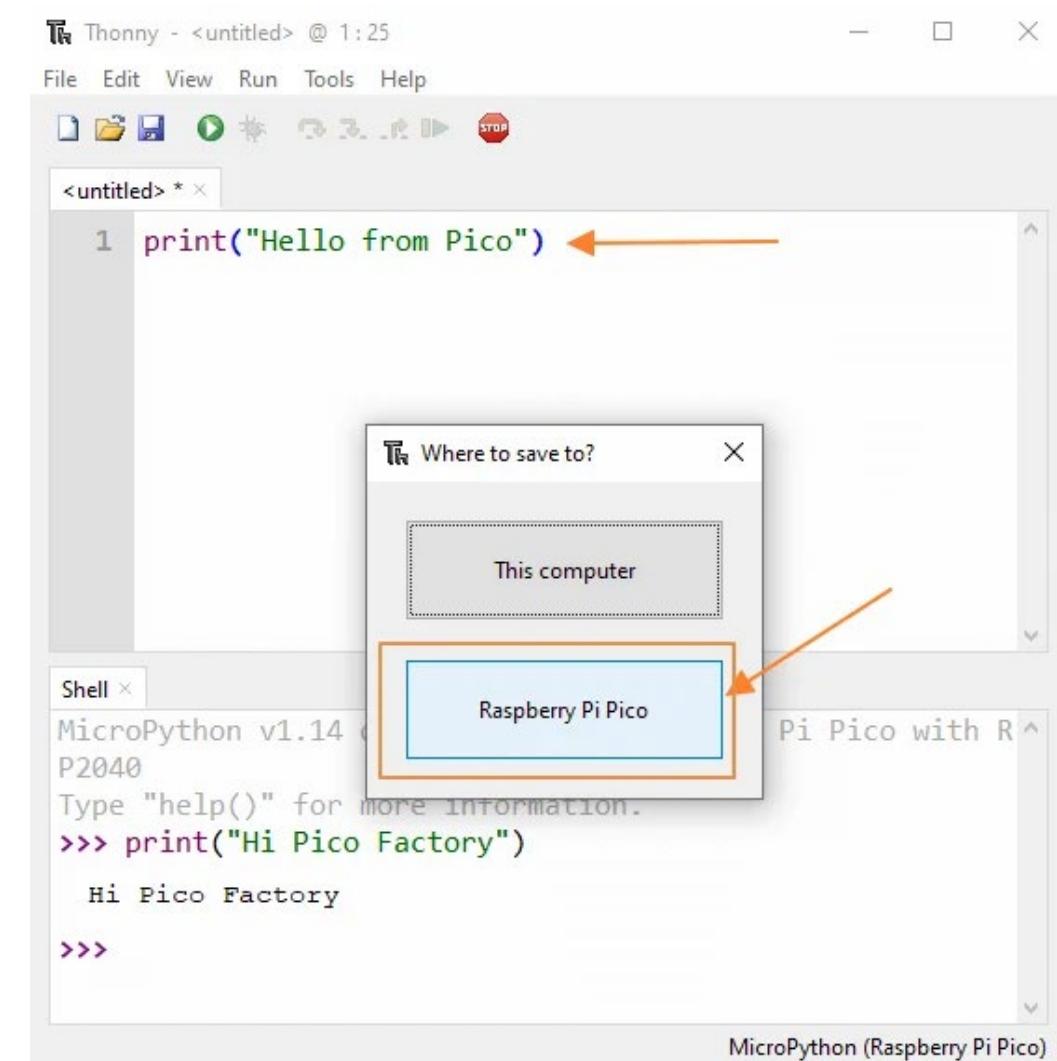
```
Thonny - Raspberry Pi Pico :: /hellopico.py @ 1:25
File Edit View Run Tools Help
[ hellopico.py ] x
1 print("Hello from Pico") x

Shell x
P2040
Type "help()" for more information.
>>> print("Hi Pico Factory")
Hi Pico Factory
>>> %Run -c $EDITOR_CONTENT
Hello from Pico
>>>
MicroPython (Raspberry Pi Pico)
```



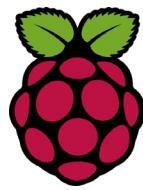
Saving python File

If you click save, it will ask where to save the file. Choosing 'Raspberry Pi Pico'saves the python file directly to the Pico.





Testing with On-board LED



Step 6: Testing with On-board LED.

Thonny - Raspberry Pi Pico :: /hellopico.py @ 4:19

File Edit View Run Tools Help



```
[ hellopico.py ] × testthonny.py ×
1 from machine import Pin
2 ledpin = Pin(25, Pin.OUT)
3 ledpin.value(1)
4 print("LED is ON")|
```

Thonny - Raspberry Pi Pico :: /hellopico.py @ 4:19

File Edit View Run Tools Help



```
[ hellopico.py ] × testthonny.py ×
1 from machine import Pin
2 ledpin = Pin(25, Pin.OUT)
3 ledpin.value(1)
4 print("LED is ON")|
```

Shell ×

```
>>> %Run -c $EDITOR_CONTENT
```

```
Hello Pico  
Bye Pico
```

```
>>> %Run -c $EDITOR_CONTENT
```

```
>>> %Run -c $EDITOR_CONTENT
```

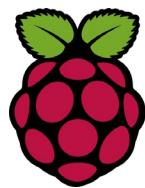
```
LED is ON
```

```
>>>
```

MicroPython (Raspberry Pi Pico)

'LED is ON' is printed only once and not printing continuously like in Arduino c language.

the ledpin.value(0) turns OFF the LED.



LED Blinking with utimer

```
1. from machine import Pin  
2. import utime  
3.  
4. ledpin = Pin(25, Pin.OUT)  
5.  
6. while True:  
7.     ledpin.value(1)  
8.     print("LED ON")  
9.     utime.sleep(1)  
10.    ledpin.value(0)  
11.    print("LED OFF")  
12.    utime.sleep(1)
```

Thonny - Raspberry Pi Pico :: /hellopico.py @ 11:19

File Edit View Run Tools Help

[hellopico.py] testthonny.py * x

```
import utime  
  
ledpin = Pin(25, Pin.OUT)  
  
while True:  
    ledpin.value(1)  
    print("LED ON")  
    utime.sleep(1)  
    ledpin.value(0)  
    print("LED OFF")  
    utime.sleep(1)
```

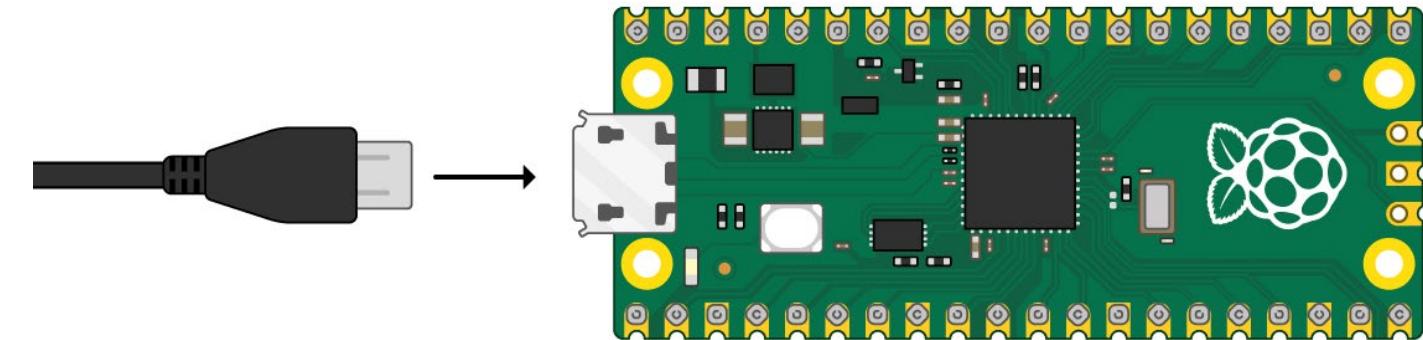
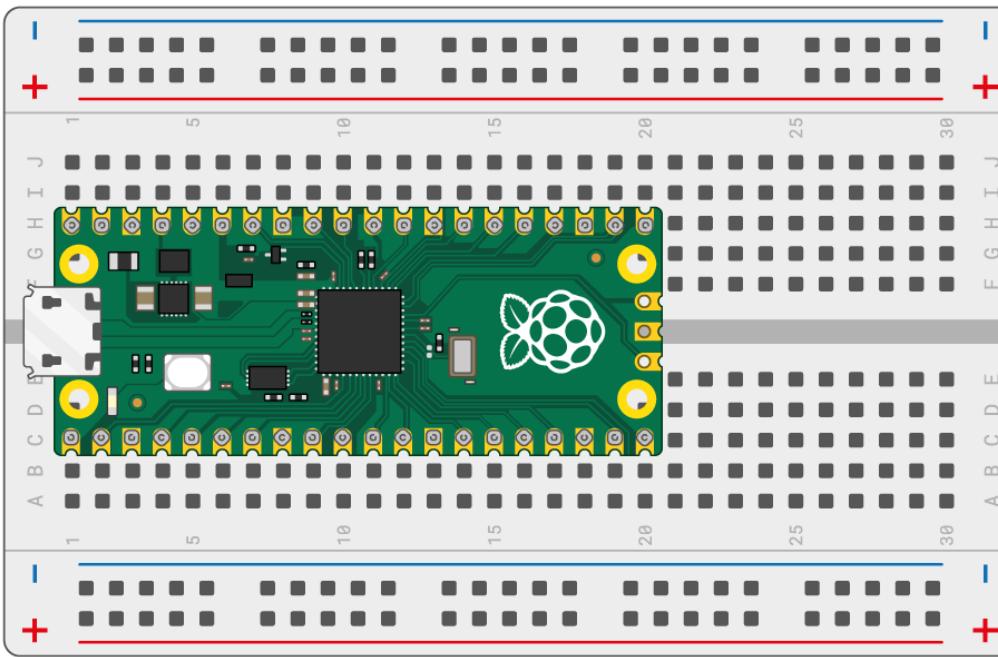
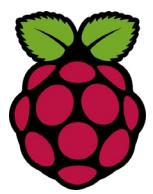
Shell x

```
LED ON  
LED OFF  
LED ON  
LED OFF  
LED ON  
LED OFF  
LED ON  
LED OFF
```

MicroPython (Raspberry Pi Pico)

Thonny Micro-Python IDE Installation

Method - 3



Install Thonny



 Download version 3.3.3 for
[Windows](#) • [Mac](#) • [Linux](#)

NB! Windows installer is signed with new identity and you may receive a warning dialog from Defender until it gains more reputation.

Just click "More info" and "Run anyway".

The screenshot shows the Thonny Python IDE interface with the following details:

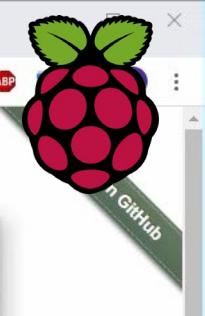
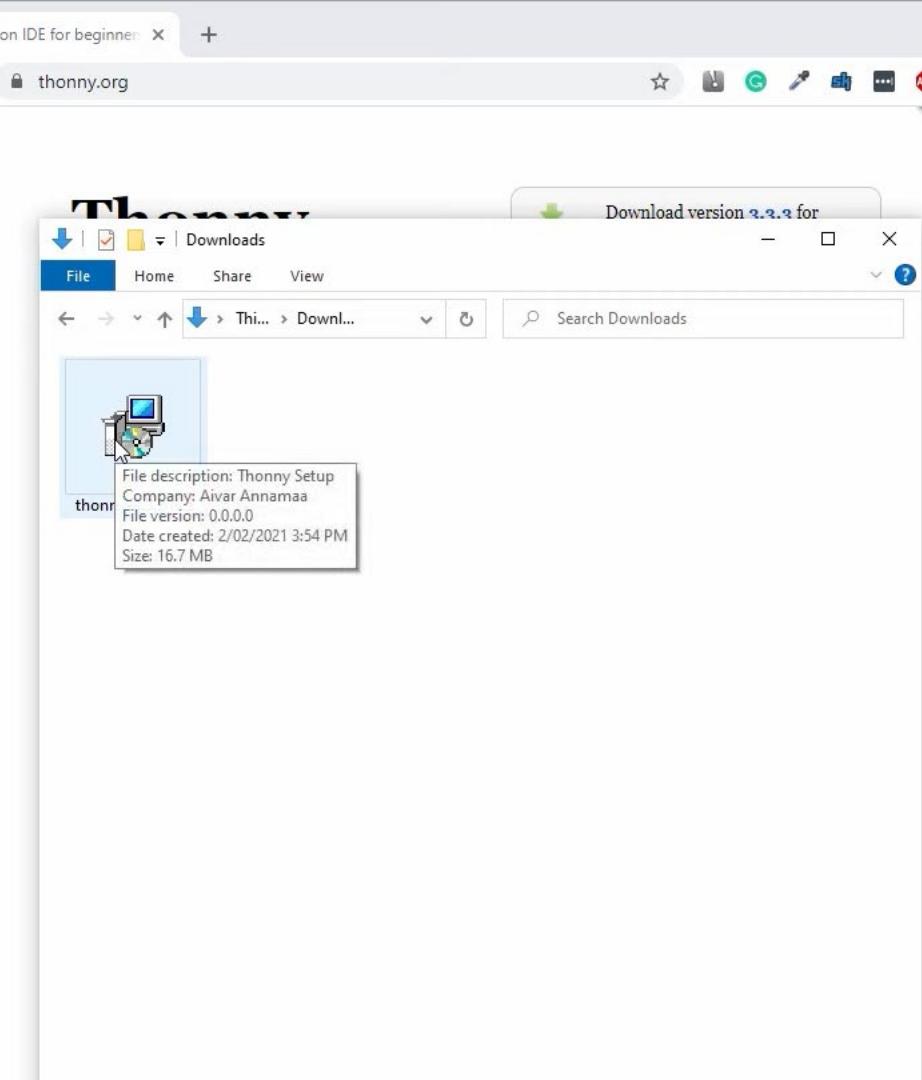
- File Menu:** File, Edit, View, Run, Tools, Help.
- Toolbar:** Contains icons for Open, Save, Run, Stop, and others.
- Code Editor (factorial.py):**

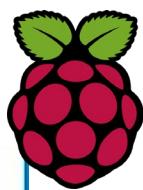
```
def fact(n):
    if n == 0:
        return 1
    else:
        return fact(n-1) * n

n = int(input("Enter a natural number: "))
print("Its factorial is", fact(3))
```
- Variables:** Shows the variable `fact` as a function and `n` as 3.
- Call Stack:** Displays two frames:
 - fact(3):** The current frame, showing the call `fact(3)`.
 - fact(2):** The previous frame, showing the call `fact(2)`.
- Shell:** Shows the command `>>> %Debug factorial.py` and the user input `Enter a natural number: 3`.
- Local variables:** Shows the variable `n` with value 3 in the current frame and `n` with value 2 in the previous frame.

Features

Easy to get started. Thonny comes with Python 3.7 built in, so just one simple installer is needed and you're ready to learn programming. (You can also use a separate [Python distribution](#) if you prefer.) The initial





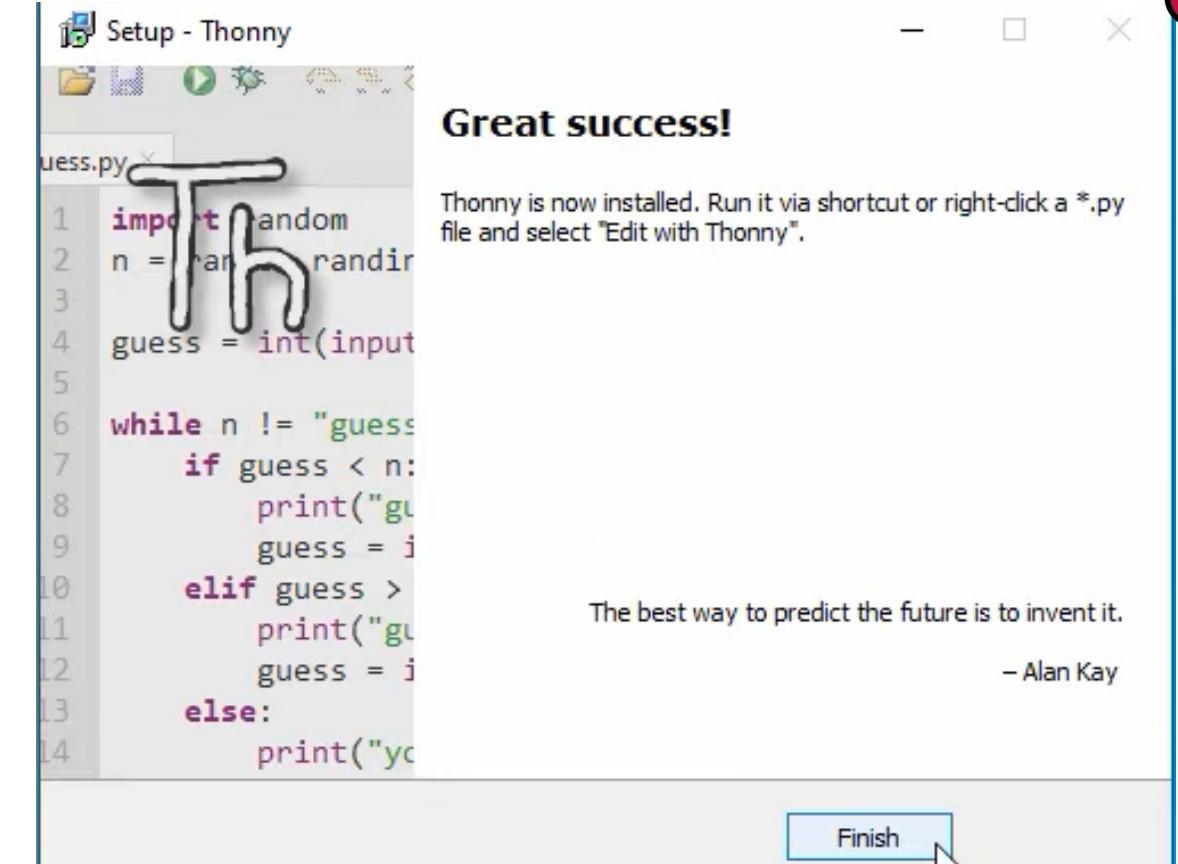
Installing

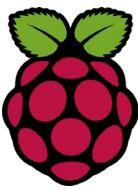
Please wait while Setup installs Thonny on your computer.

Extracting files...

Th

Cancel





Set up Thonny

- Hold the **BOOTSEL** button on your Pico, and connect your Pico to your computer via USB.
- Go to **Run > Select interpreter** and choose **MicroPython (Raspberry Pi Pico)**

Run

- Select interpreter...
- Run current script F5
 - Debug current script (nicer) Ctrl+F5
 - Debug current script (faster) Shift+F5
 - Debug current script (birdseye) Ctrl+Shift+B
- Step over F6
- Step into F7
- Step out
- Resume F8
- Run to cursor Ctrl+F8
- Step back Ctrl+B

Run current script in terminal Ctrl+T

Dock user windows

Pygame Zero mode

Stop/Restart backend Ctrl+F2

- Interrupt execution Ctrl+C
- Send EOF / Soft reboot Ctrl+D
- Disconnect

Shell

Couldn't find the device automatically.
Check the connection (making sure the device is not in bootloader mode) or choose "Configure interpreter" in the interpreter menu (bottom-right corner of the window) to select specific port or another interpreter.

MicroPython (Raspberry Pi Pico)

Thonny options

General Interpreter Editor Theme & Font Run & Debug Terminal Shell Assistant

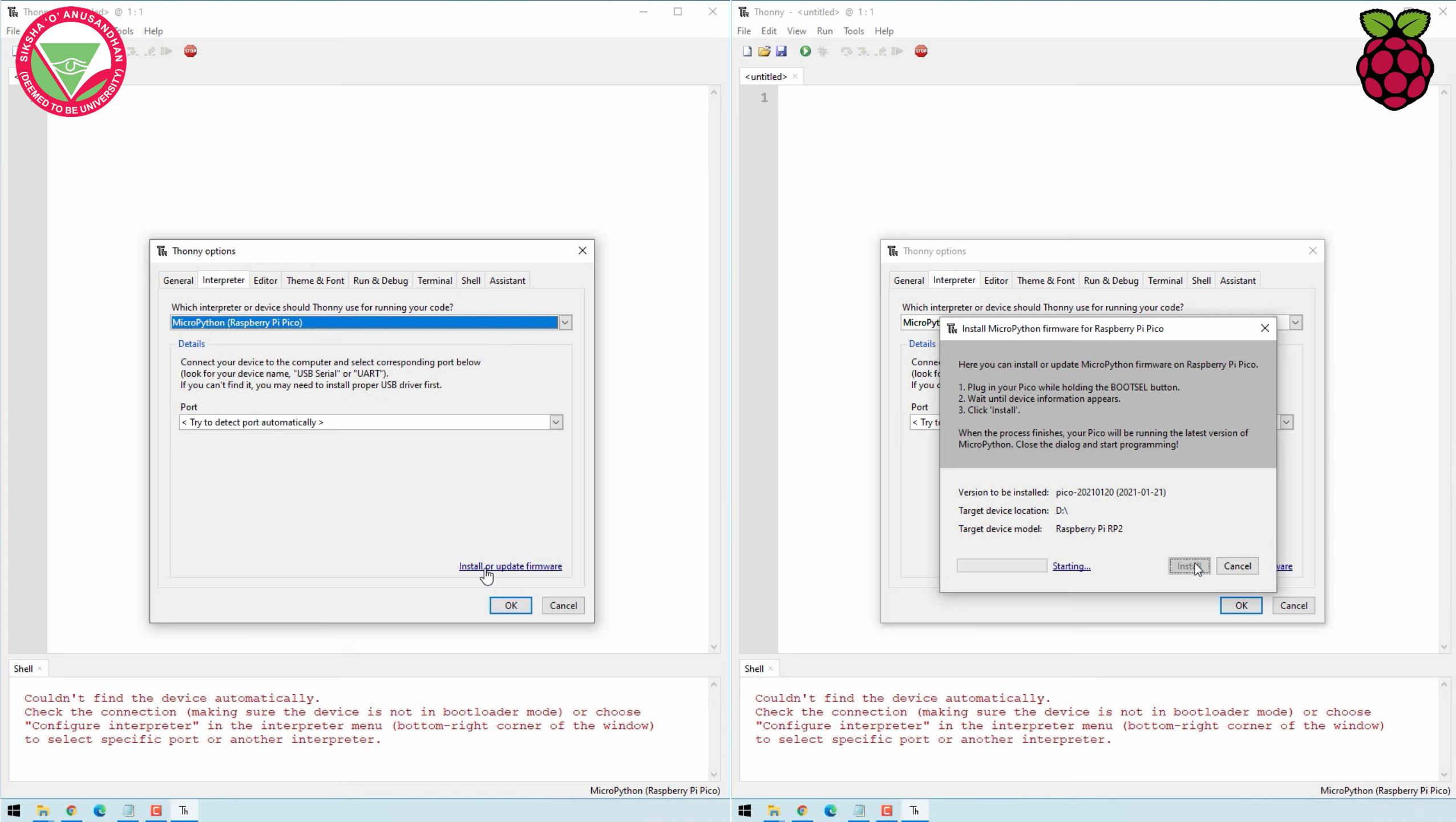
Which interpreter or device should Thonny use for running your code?
MicroPython (Raspberry Pi Pico)

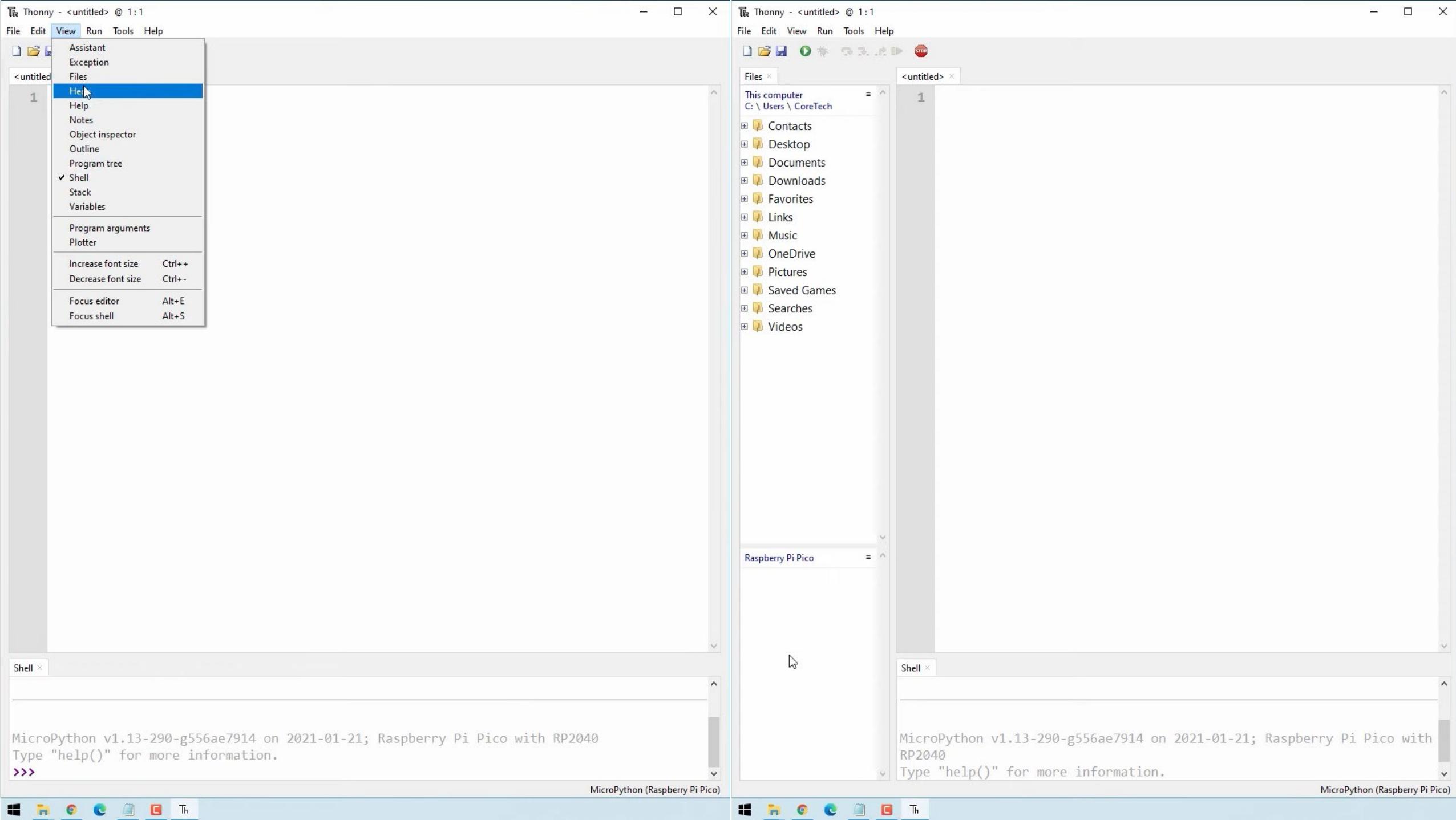
The same interpreter which runs Thonny (default)
Alternative Python 3 interpreter or virtual environment
Remote Python 3 (SSH)
MicroPython (SSH)
MicroPython (BBC micro:bit)
MicroPython (Raspberry Pi Pico)
MicroPython (ESP32)
MicroPython (ESP8266)
MicroPython (generic)
CircuitPython (generic)
A special virtual environment (deprecated)

Install or update firmware

OK Cancel

MicroPython (Raspberry Pi Pico)





The image shows two side-by-side instances of the Thonny Python IDE running on a Windows operating system. Both windows have the title "Thonny - <untitled> @ 8:14".

Left Window:

- File Explorer:** Shows standard Windows folders like This computer, Contacts, Desktop, Documents, Downloads, Favorites, Links, Music, OneDrive, Pictures, Saved Games, Searches, and Videos.
- Code Editor:** An untitled file containing the following Python code:

```
1 from machine import Pin
2 from time import sleep
3
4 led = Pin(25, Pin.OUT)
5
6 while True:
7     led.toggle()
8     sleep(0.5)
```
- Shell:** Displays the MicroPython prompt and a "hello world" print statement.

```
Raspberry Pi Pico
MicroPython v1.13-290-g556ae7914 on 2021-01-21; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> print('hello world')
hello world
>>>
```

Right Window:

- File Explorer:** Shows the same Windows folder structure.
- Code Editor:** An untitled file containing the same Python code as the left window.
- Save Dialog:** A modal window titled "Where to save to?" with two options: "This computer" and "Raspberry Pi Pico". The "Raspberry Pi Pico" option is selected.
- Shell:** Displays the MicroPython prompt and a "hello world" print statement, identical to the left window's shell output.

Taskbar: At the bottom of the screen, the taskbar shows the Windows Start button, several pinned application icons (including File Explorer, Edge, and File History), and the Thonny icon.

This computer
C:\ Users \ CoreTech

Contacts
Desktop
Documents
Downloads
Favorites
Links
Music
OneDrive
Pictures
Saved Games
Searches
Videos

Save to Raspberry Pi Pico

Raspberry Pi Pico

Name

Size (bytes)

File name: main.py OK Cancel

Shell

```
SELECT THE CONNECTION (RUNNING DAEMON AND ADVICE IS NOT IN DOWNLOAD MODE) OR CHOOSE "CONFIGURE INTERPRETER" IN THE INTERPRETER MENU (BOTTOM-RIGHT CORNER OF THE WINDOW) TO SELECT SPECIFIC PORT OR ANOTHER INTERPRETER.
```

MicroPython v1.13-290-g556ae7914 on 2021-01-21; Raspberry Pi Pico with RP2040
Type "help()" for more information.
->>> print('hello world')
hello world
->>>

This computer
C:\ Users \ CoreTech

Contacts
Desktop
Documents
Downloads
Favorites
Links
Music
OneDrive
Pictures
Saved Games
Searches
Videos

[main.py]

from machine import Pin
from time import sleep

led = Pin(25, Pin.OUT)

while True:
 led.toggle()
 sleep(0.5)

Raspberry Pi Pico

main.py

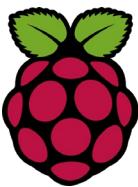
Shell

```
SELECT THE CONNECTION (RUNNING DAEMON AND ADVICE IS NOT IN DOWNLOAD MODE) OR CHOOSE "CONFIGURE INTERPRETER" IN THE INTERPRETER MENU (BOTTOM-RIGHT CORNER OF THE WINDOW) TO SELECT SPECIFIC PORT OR ANOTHER INTERPRETER.
```

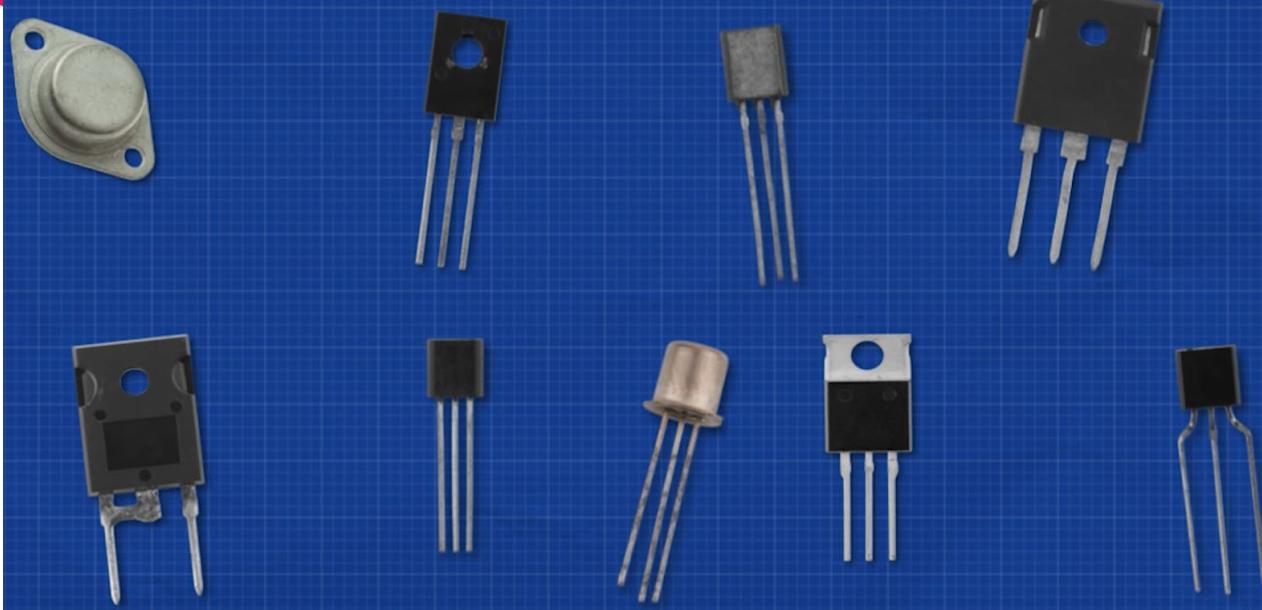
MicroPython v1.13-290-g556ae7914 on 2021-01-21; Raspberry Pi Pico with RP2040
Type "help()" for more information.
->>> print('hello world')
hello world
->>>

press Ctrl+D (or use the Stop/Restart button) to restart your Pico.

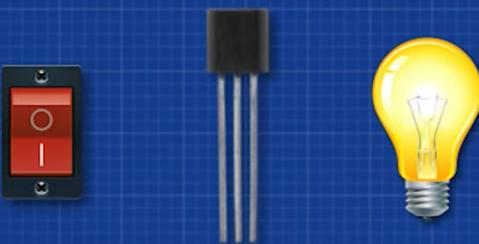
TRANSISTORS



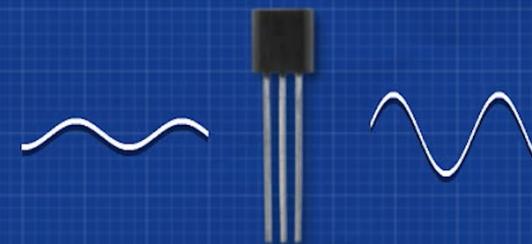
Transistors



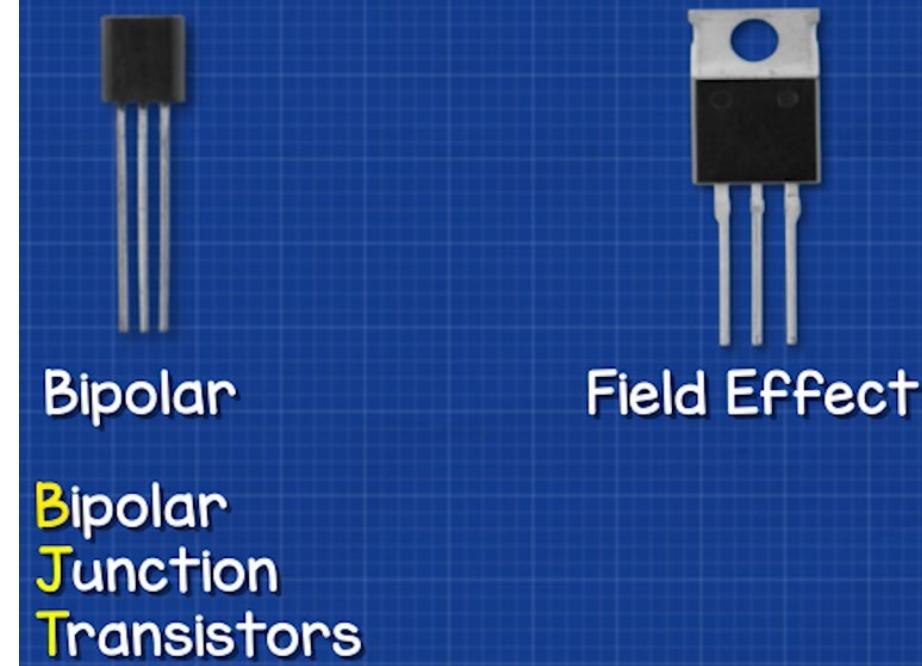
Transistors are small electronic components with two main functions.

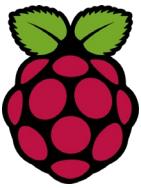


Switch circuits
on and off



Amplify signals





Transistors

Resin case



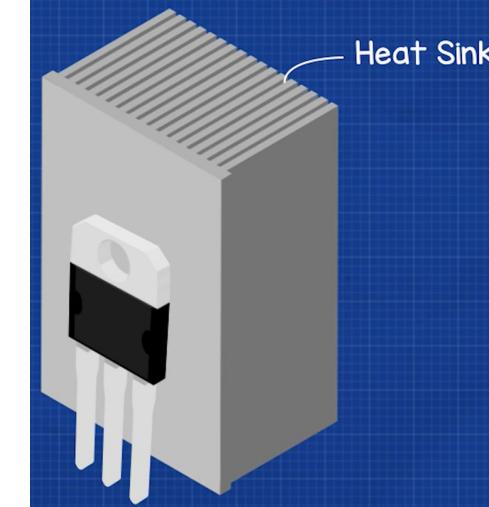
Small, low power, transistors have a resin case

Resin case



Higher power transistors have a partly metal case

Metal

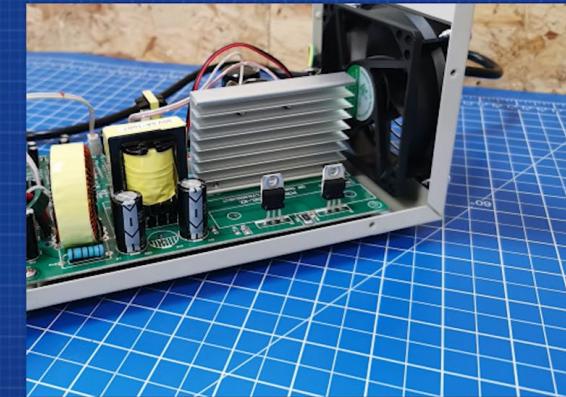


Heat Sink

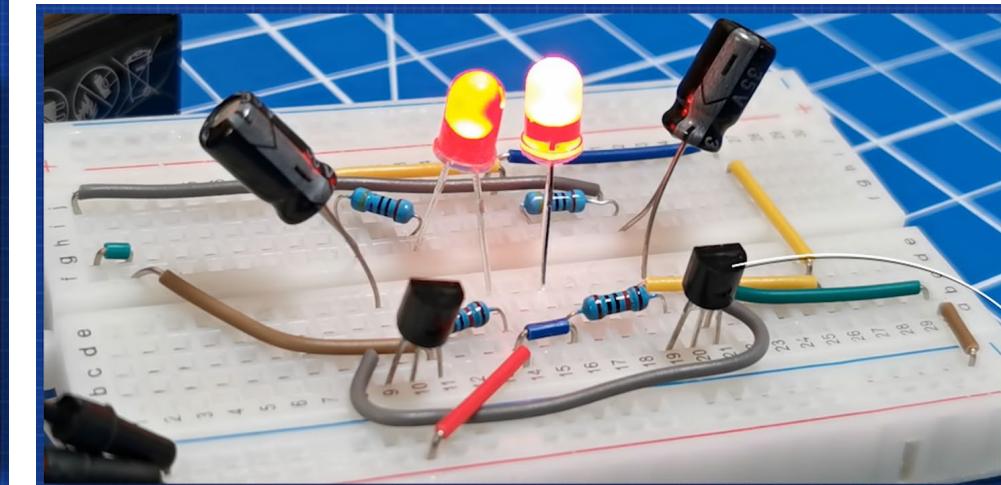
Thermal Image of MOSFET



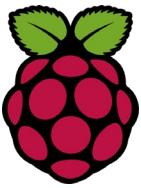
Standard Image of MOSFET



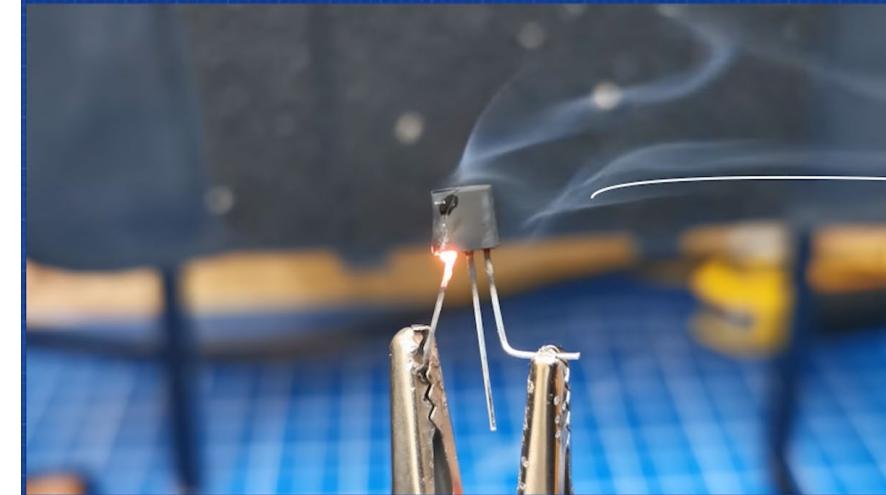
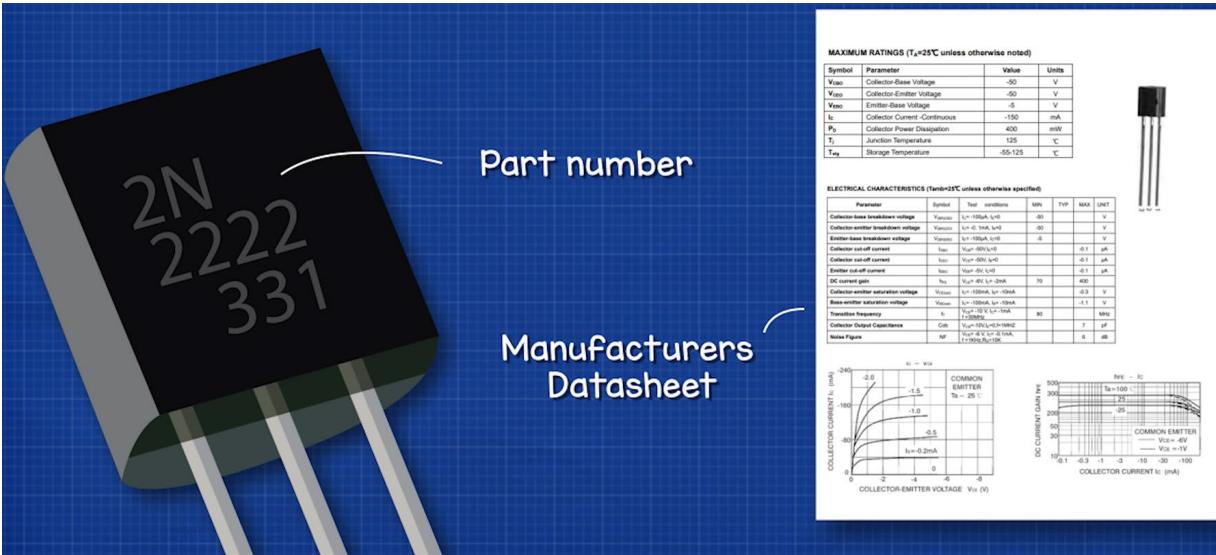
Example: Mosfet reached 45°C (113°F) at 1.2A



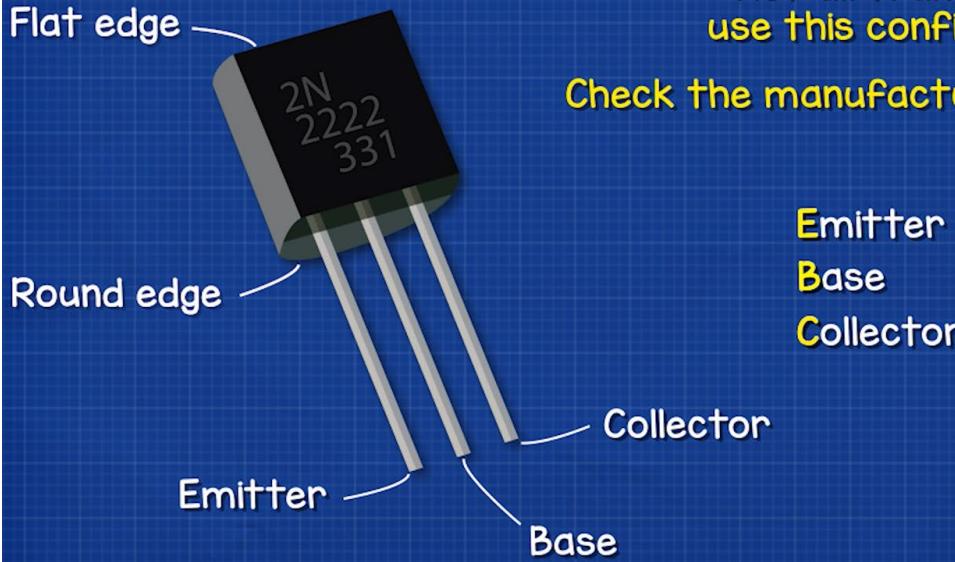
No heat sink required



Transistors



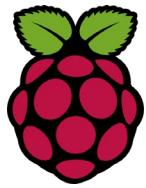
Typical configuration



Not all transistors use this configuration

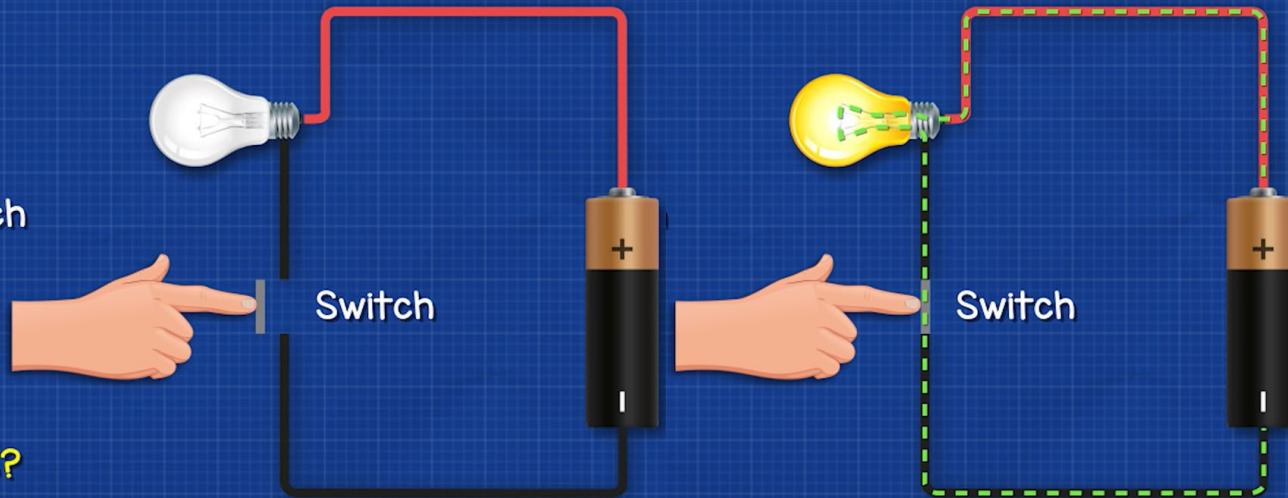
Check the manufacturers datasheet

Emitter
Base
Collector



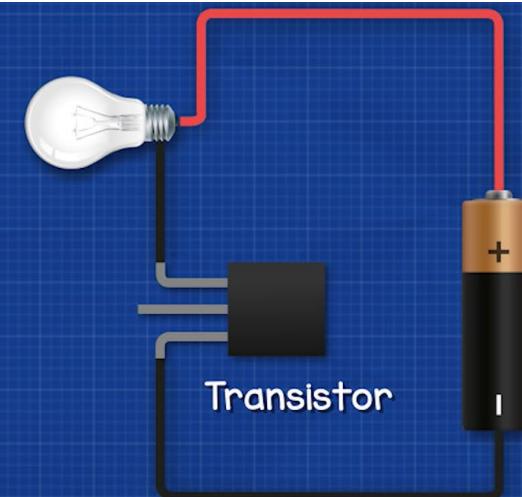
Transistors Application

Requires a human
to manually control the switch

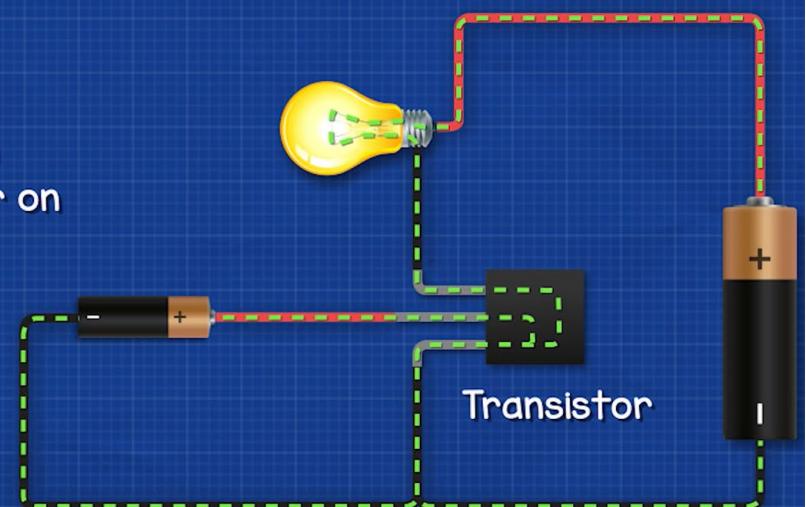


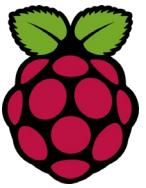
How can we automate this?

Transistor is blocking the current
so light is off

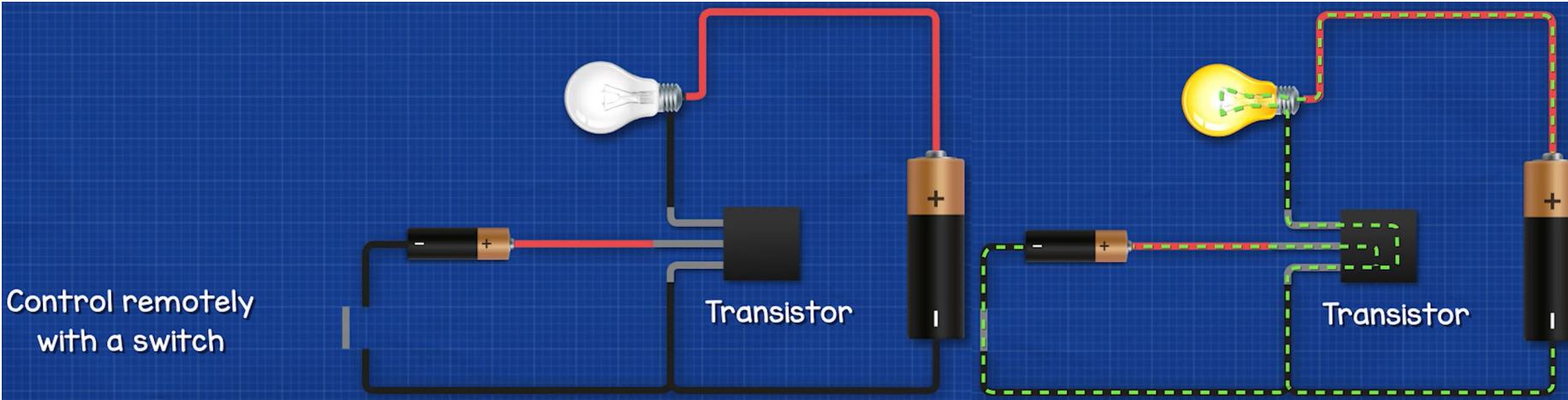


A small voltage
to the centre pin
turns the transistor on



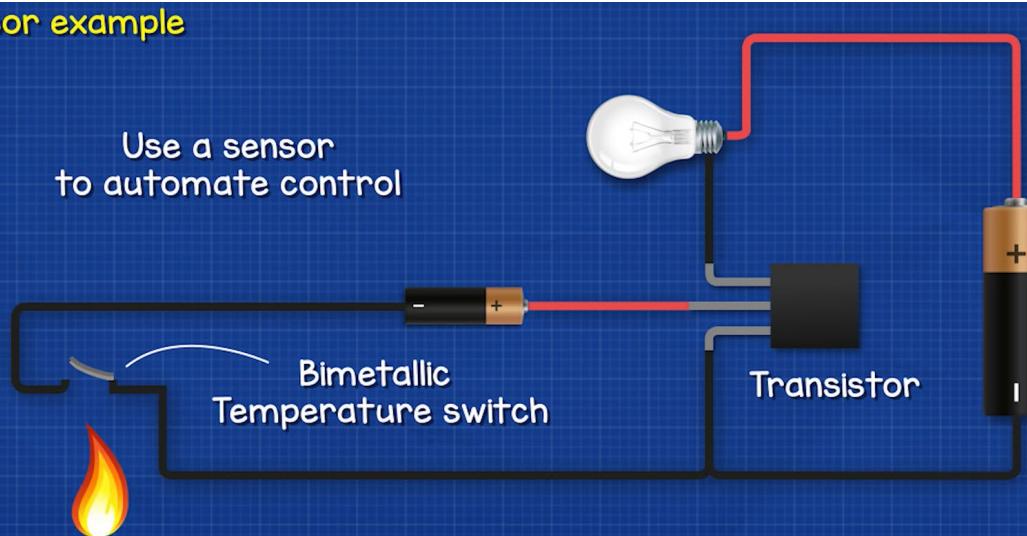


Transistors Application



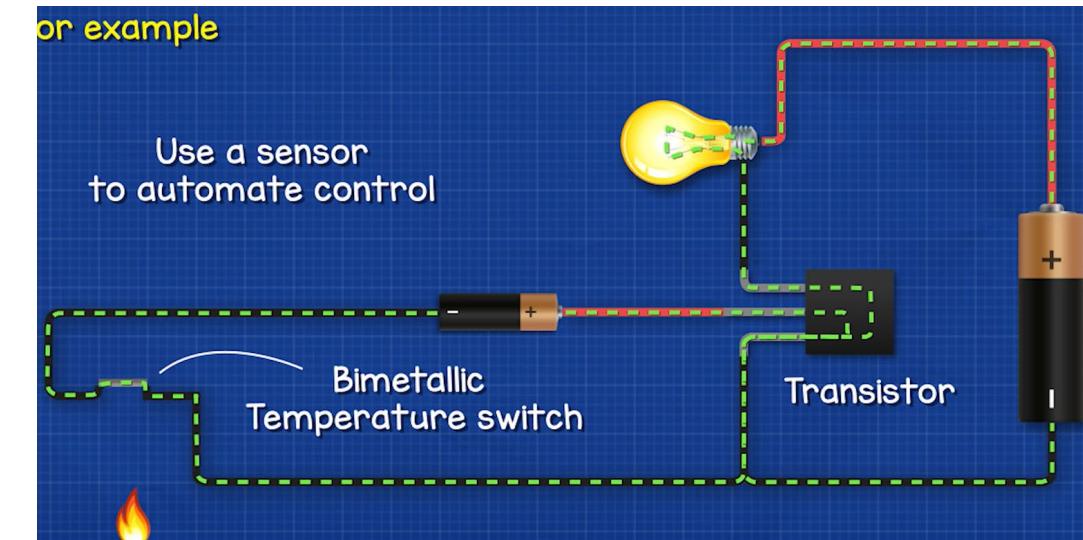
Sensor example

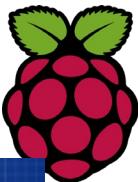
Use a sensor to automate control



or example

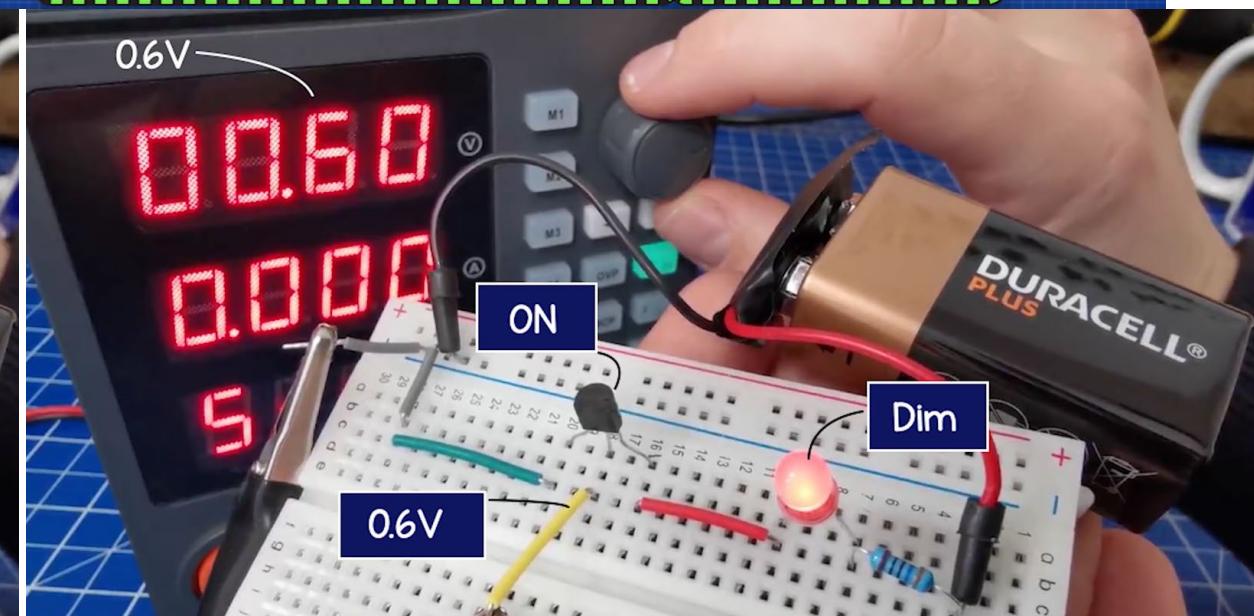
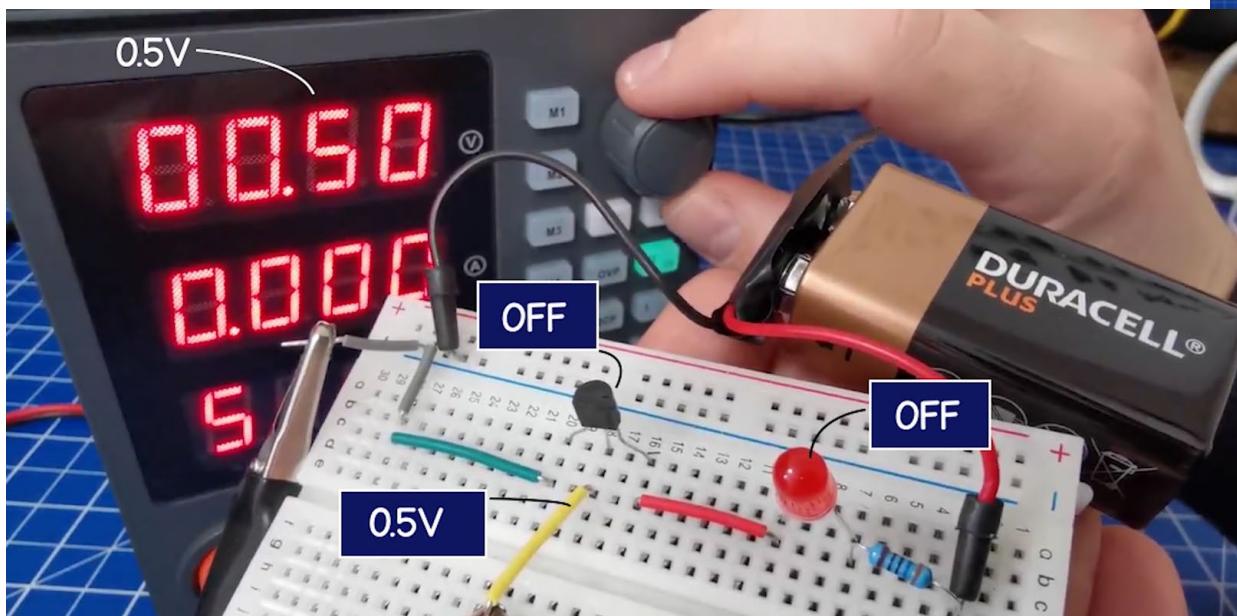
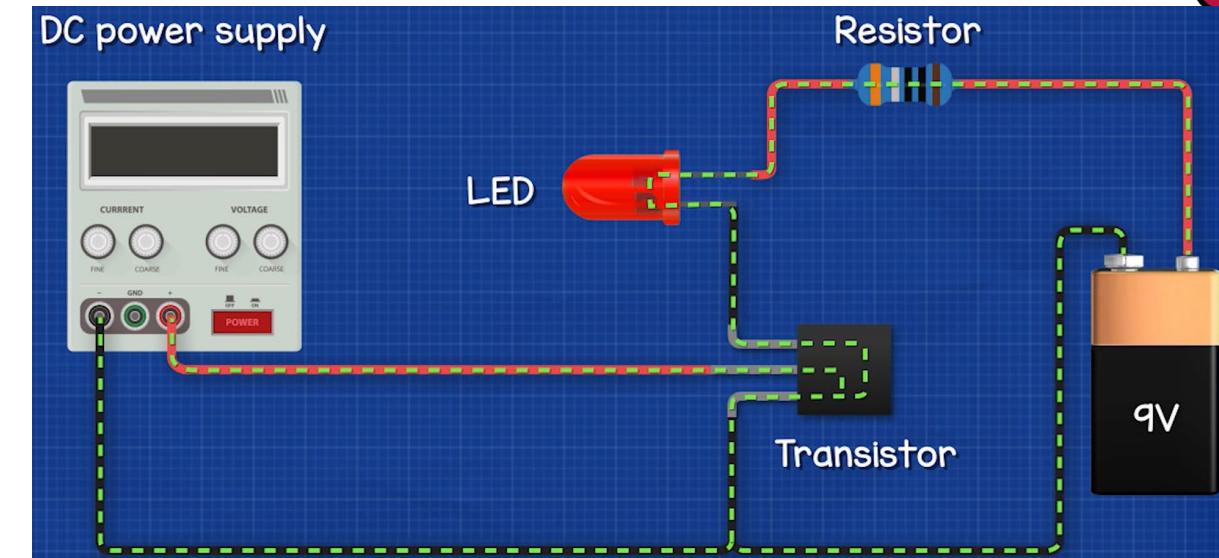
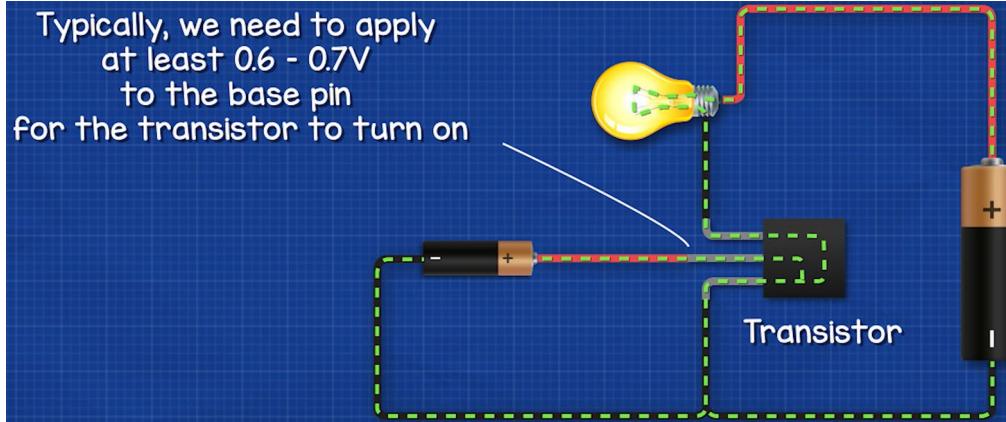
Use a sensor to automate control

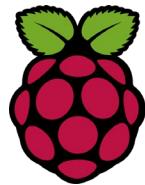




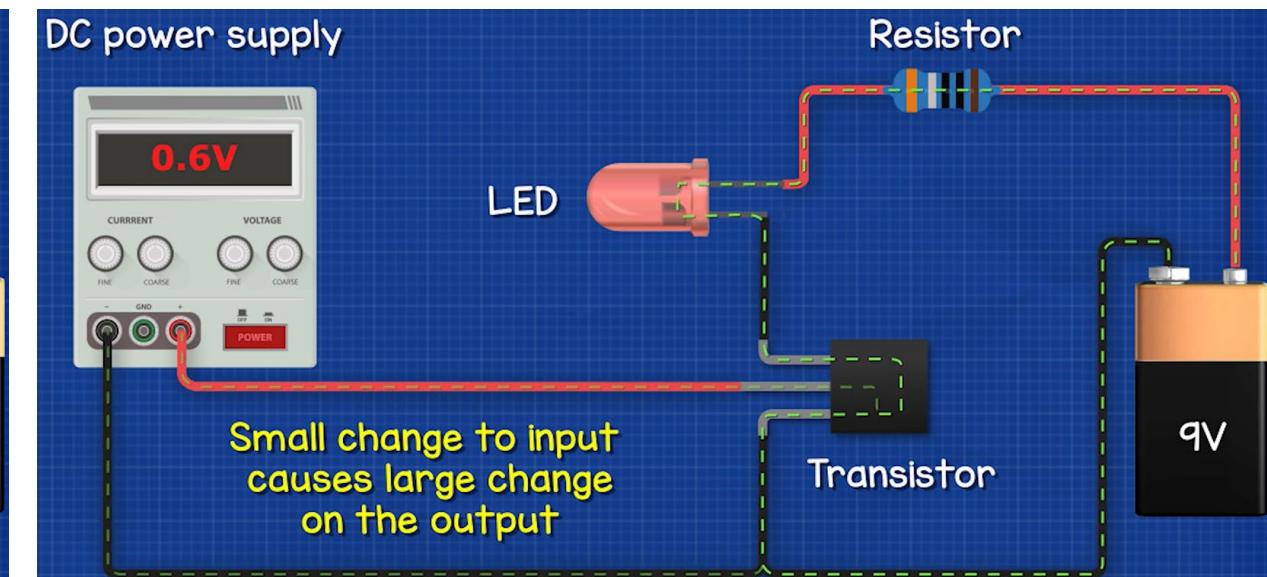
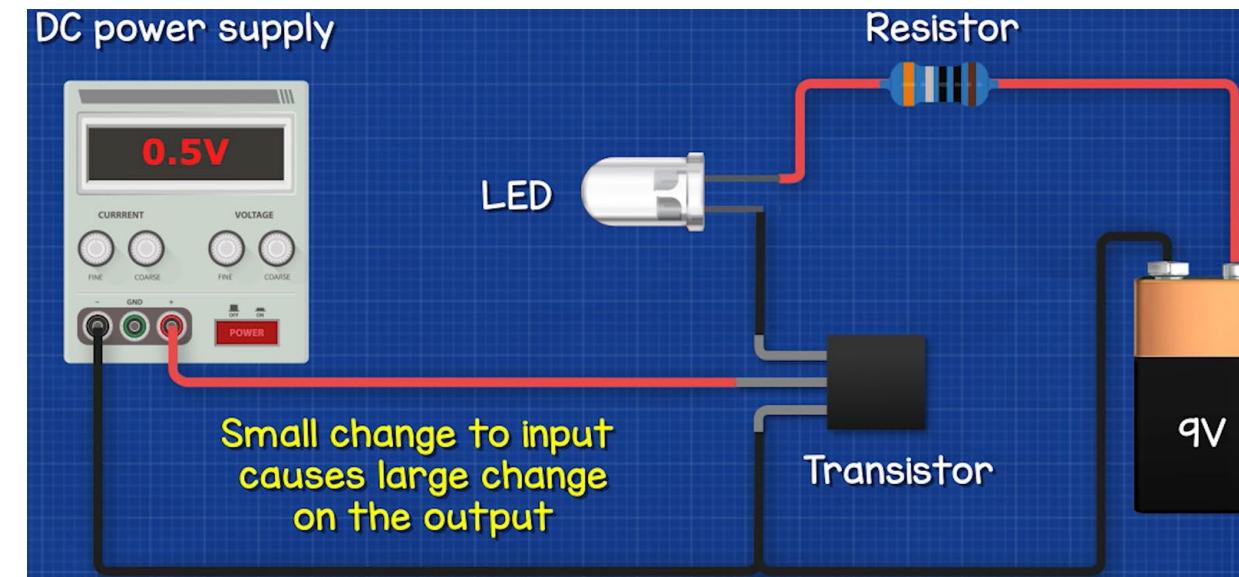
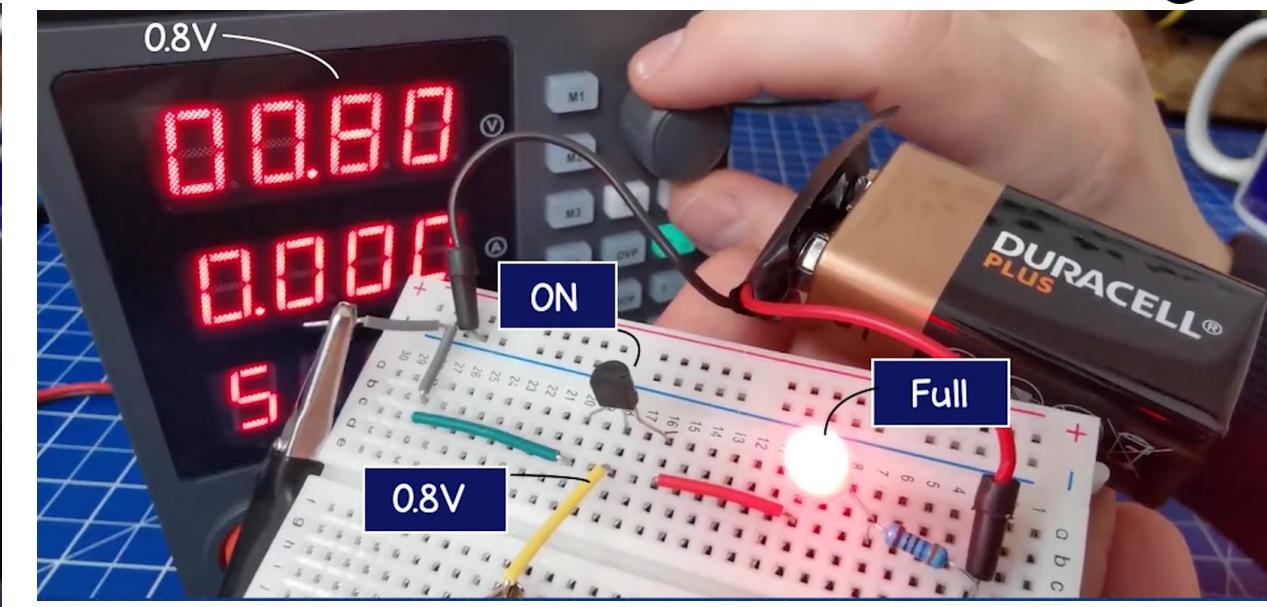
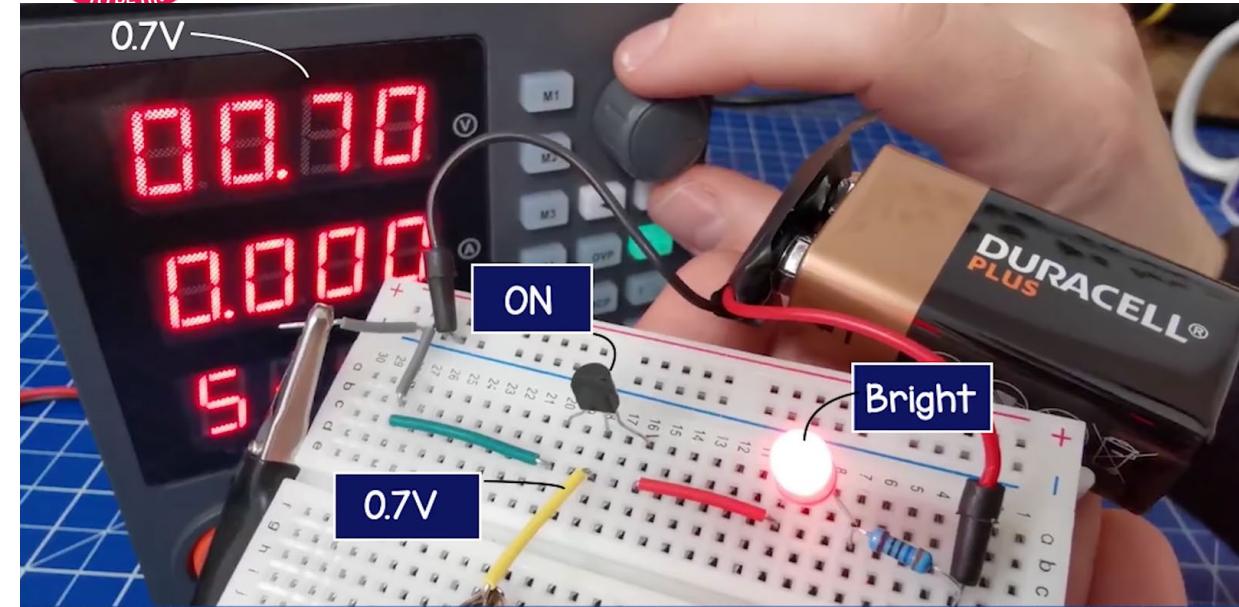
Transistors Application

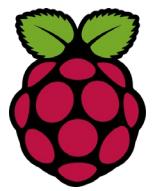
Typically, we need to apply at least 0.6 - 0.7V to the base pin for the transistor to turn on



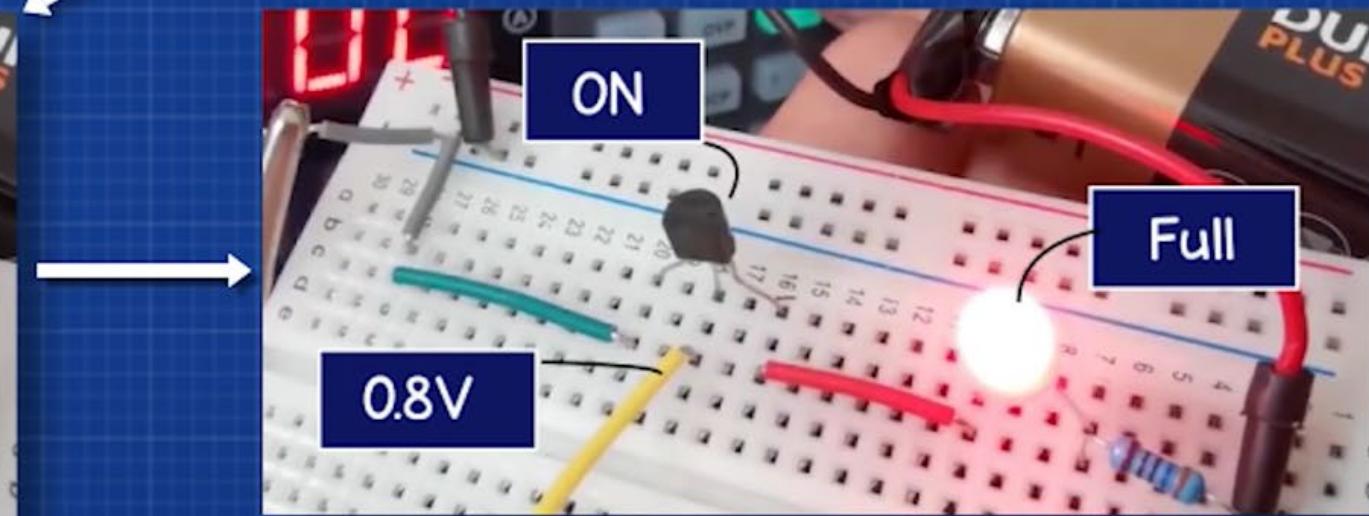
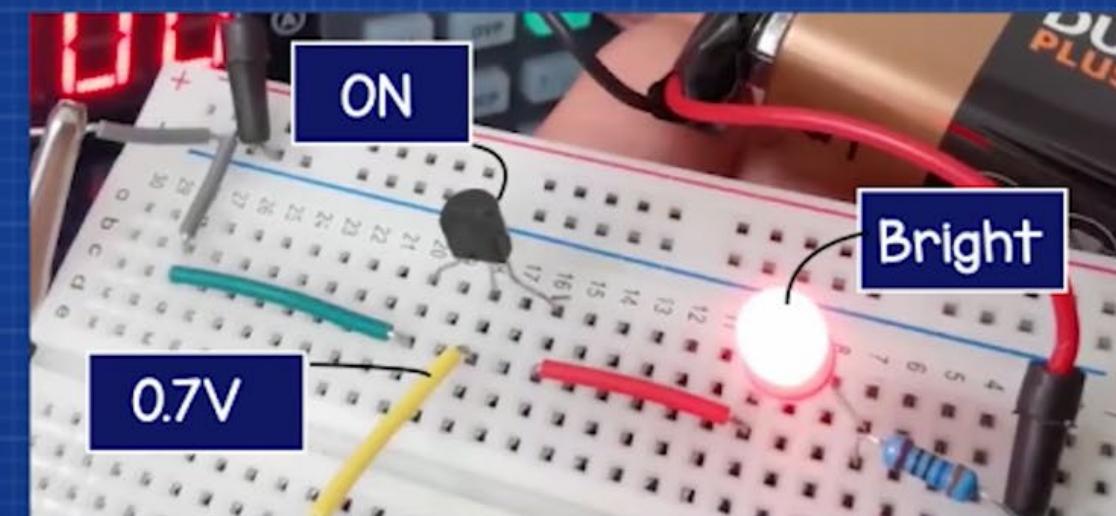
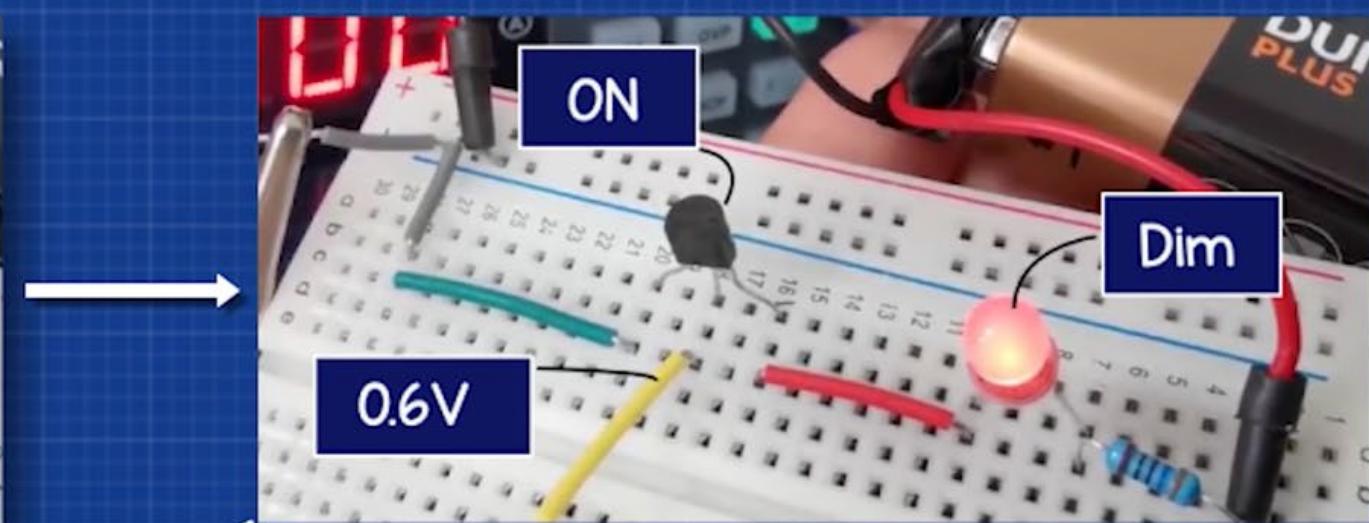
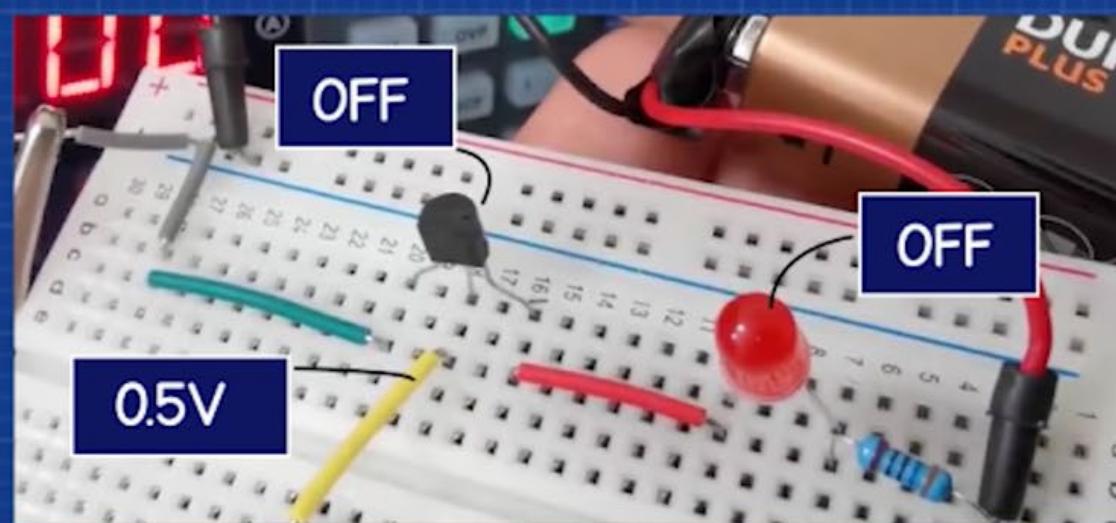


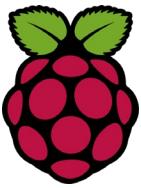
Transistors Application





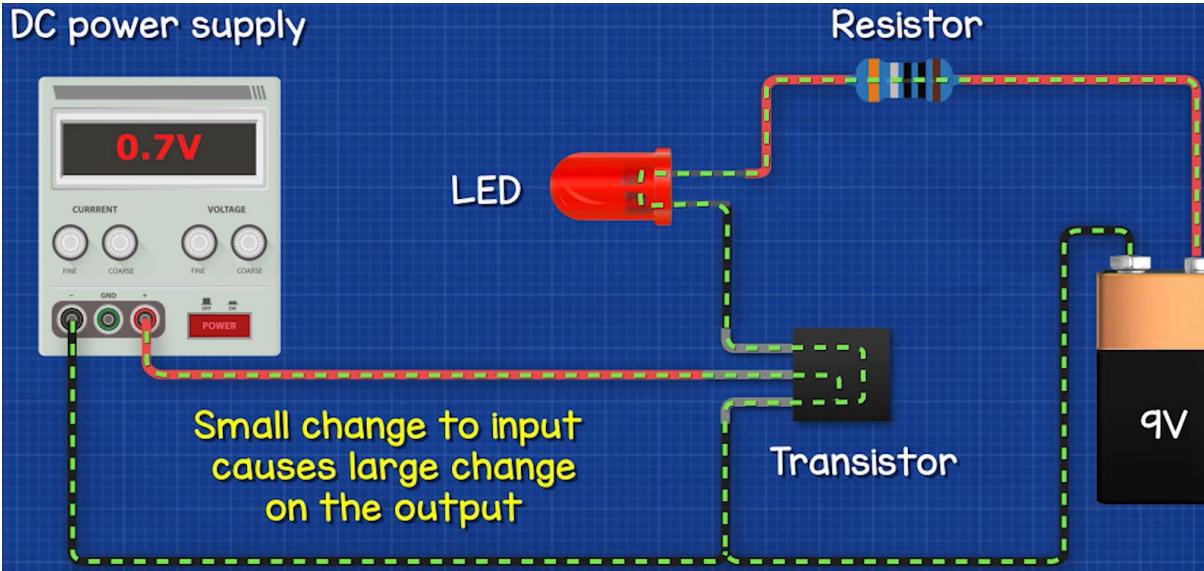
Transistors Application



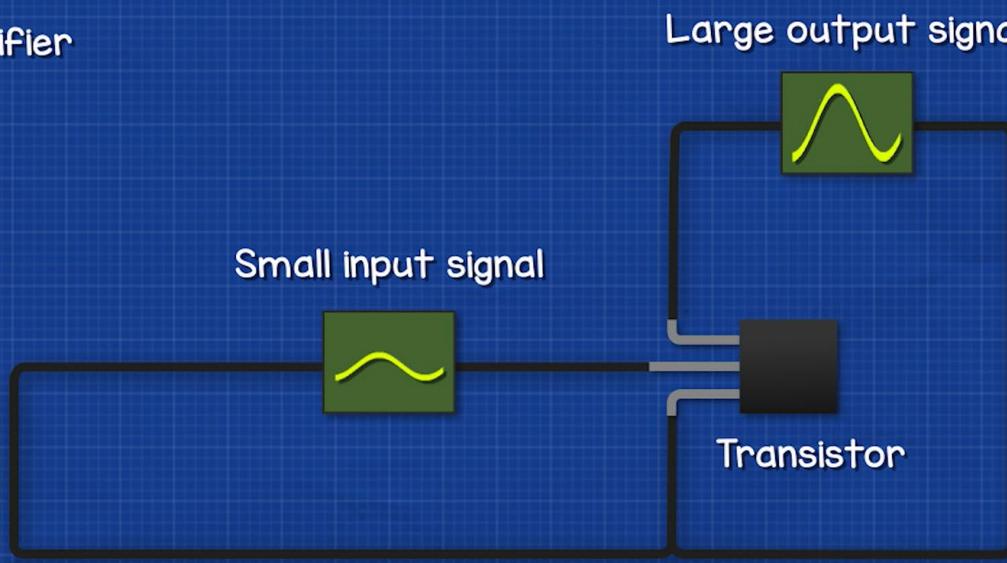


Transistors Application

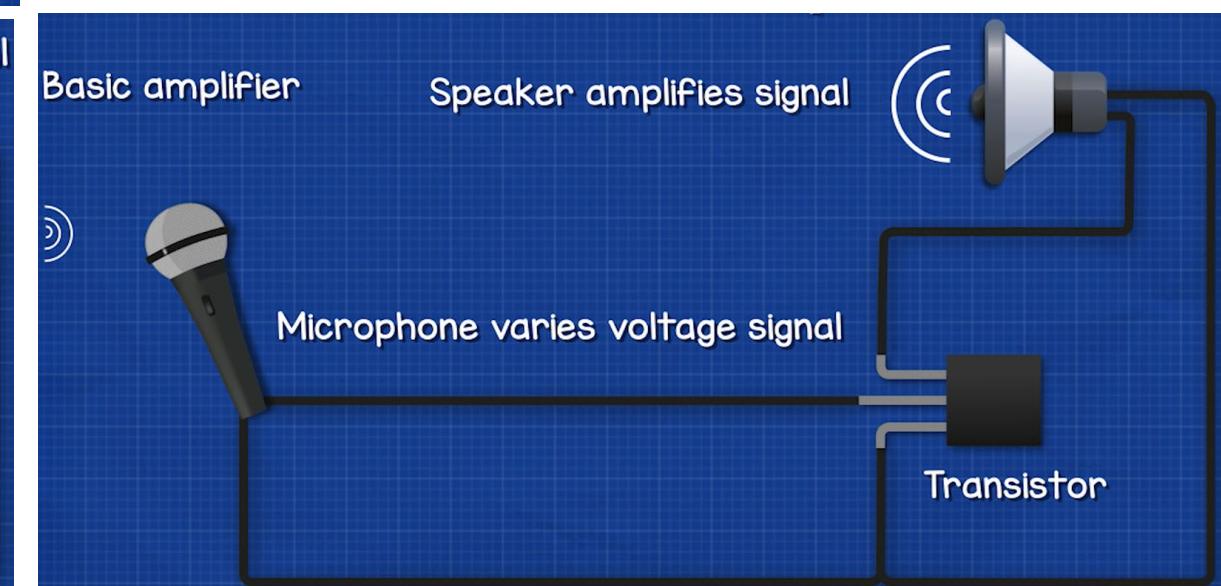
DC power supply

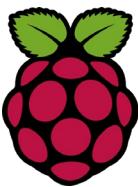


Basic amplifier



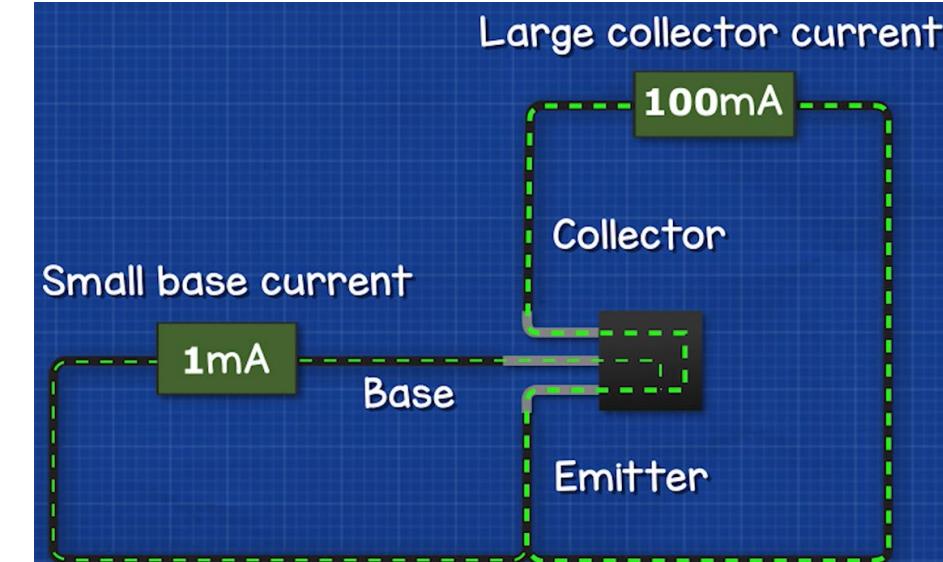
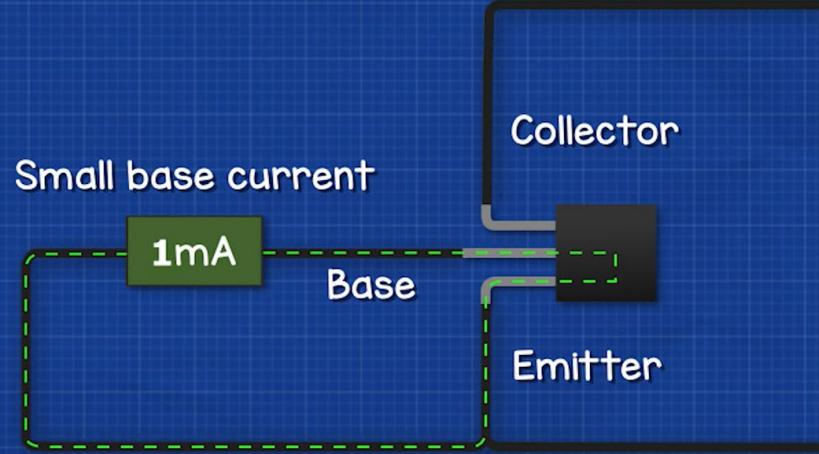
Basic amplifier





Transistors Current Gain

Current gain



$$\beta = \frac{\text{Collector current}}{\text{Base current}}$$

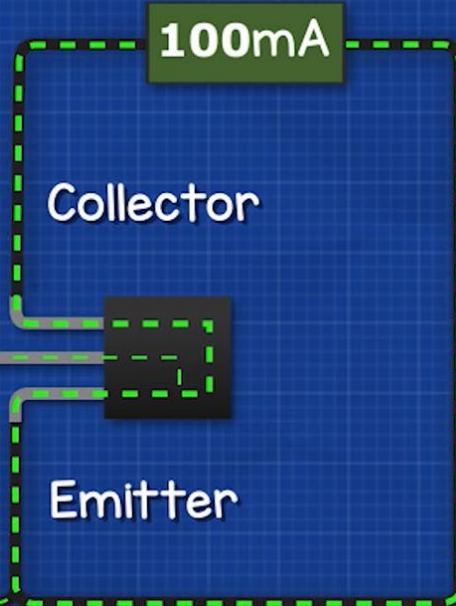
$$\beta = \frac{100\text{mA}}{1\text{mA}}$$

$$\beta = 100$$

Small base current

1mA

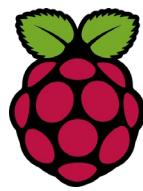
Base



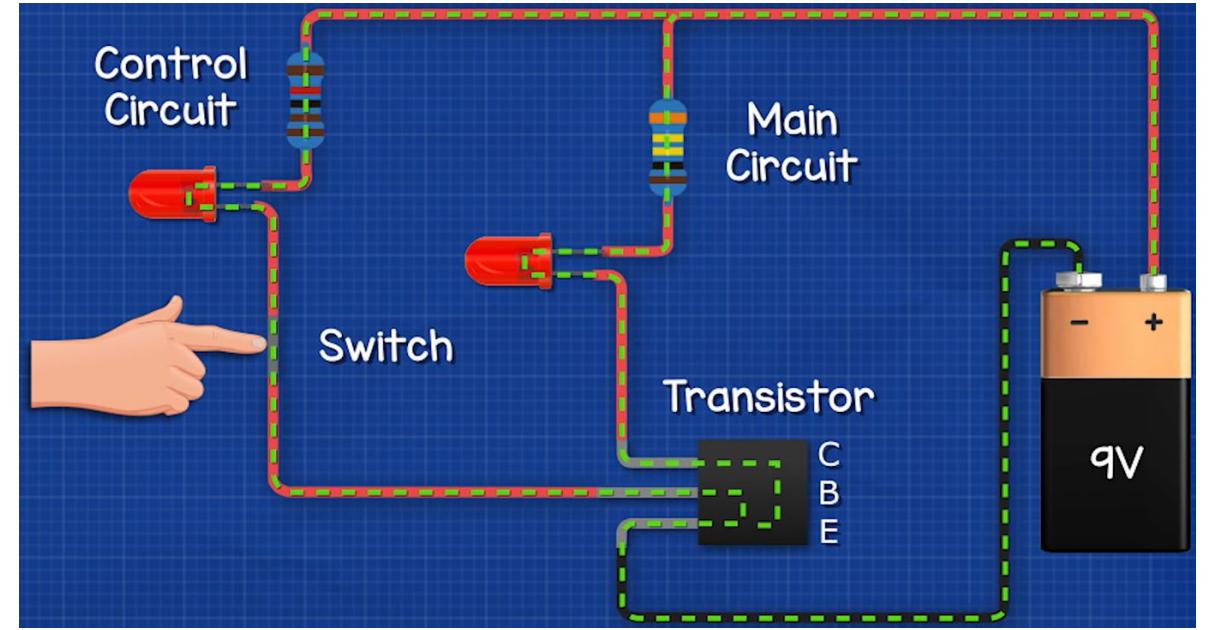
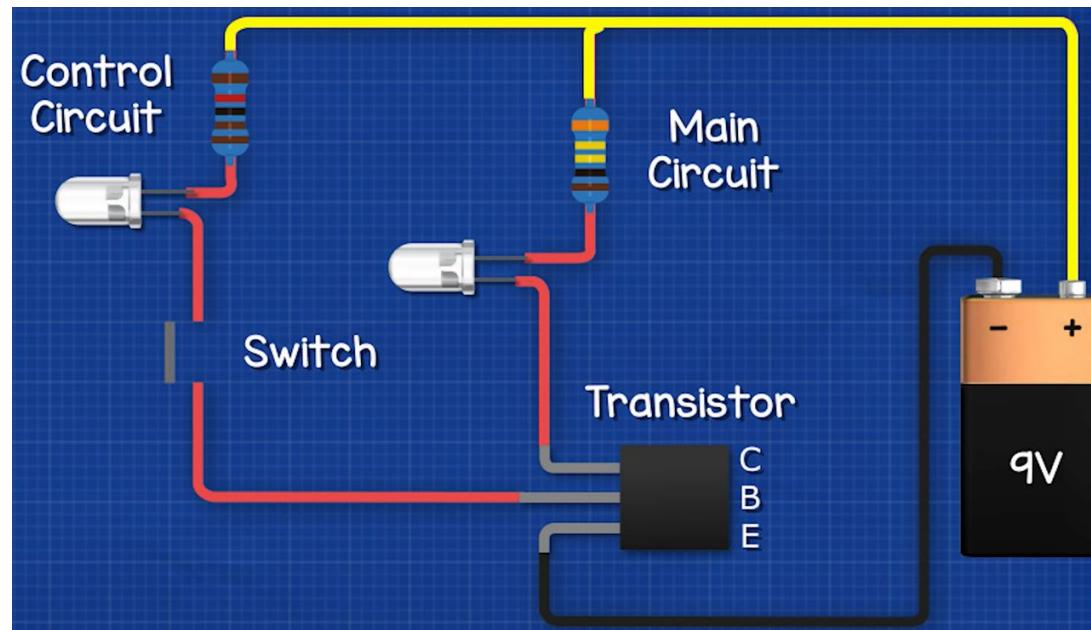
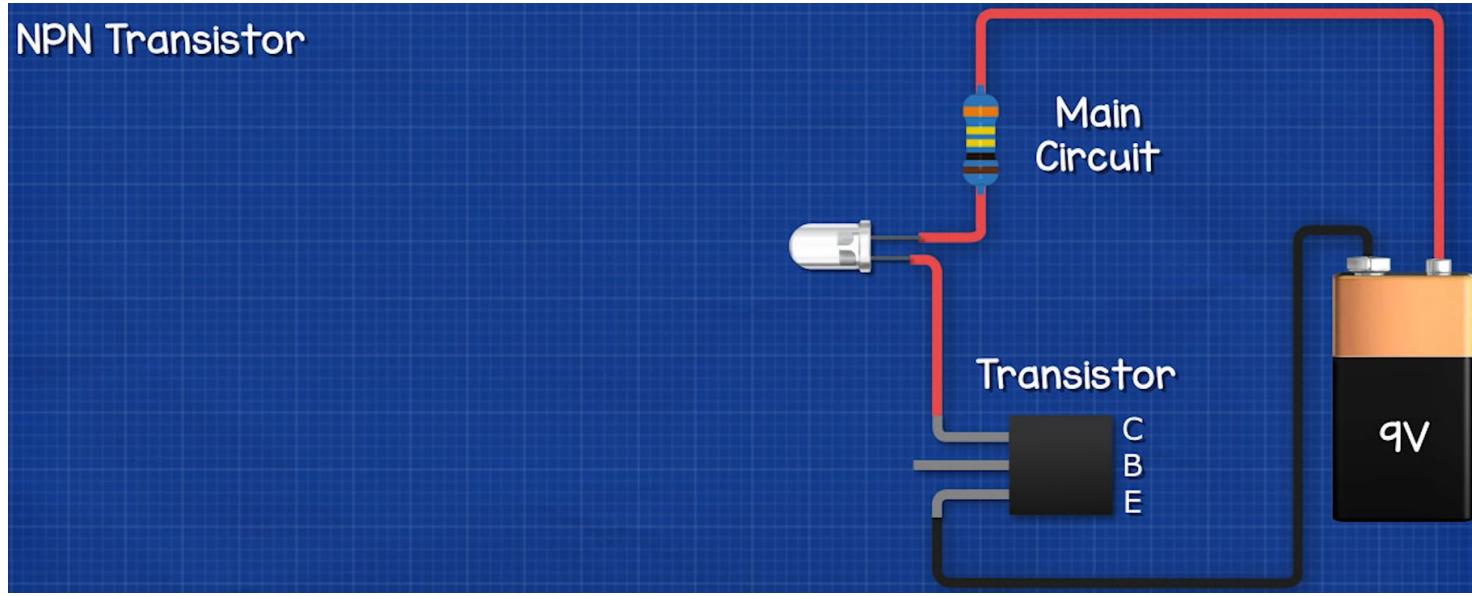
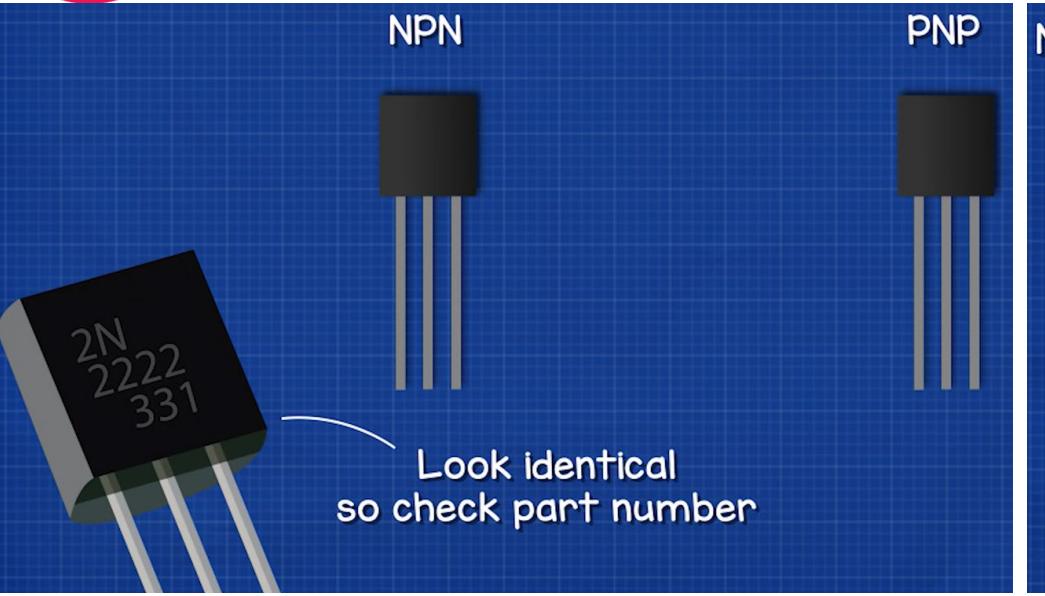
$$\beta = \frac{\text{Collector current}}{\text{Base current}}$$

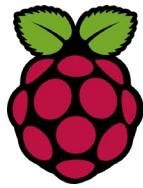
$$\text{Collector current} = \beta \times \text{Base current}$$

$$\text{Base current} = \frac{\text{Collector current}}{\beta}$$

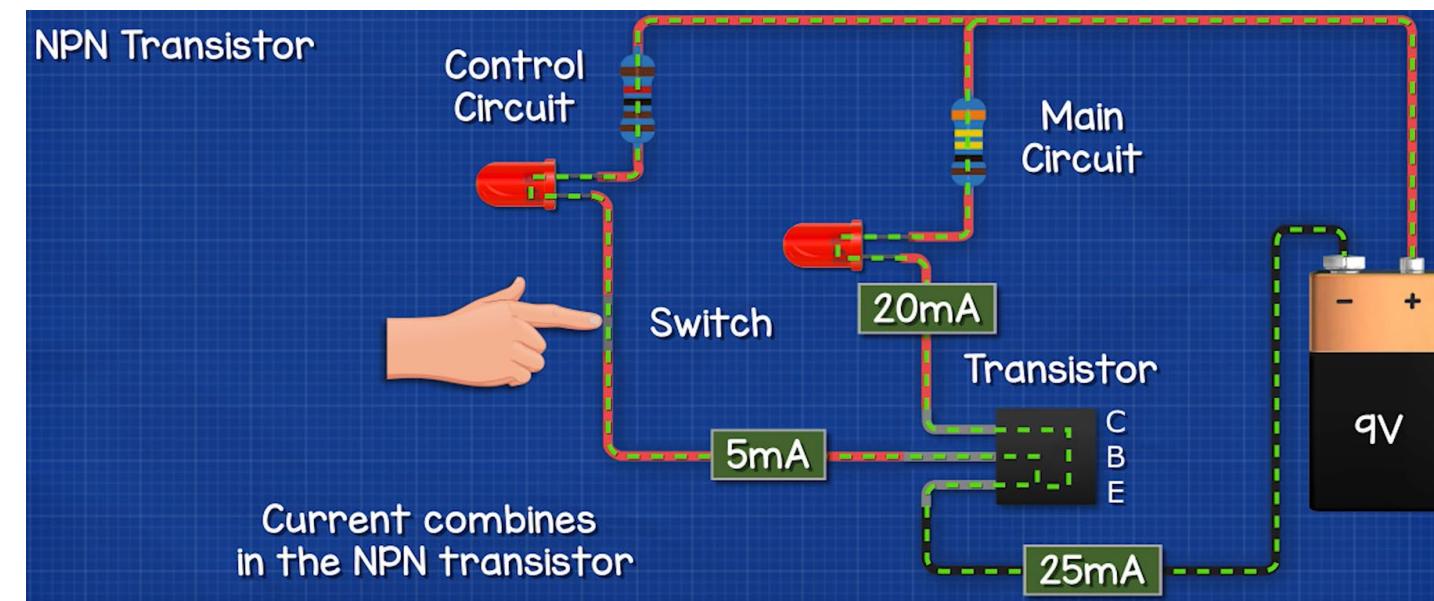
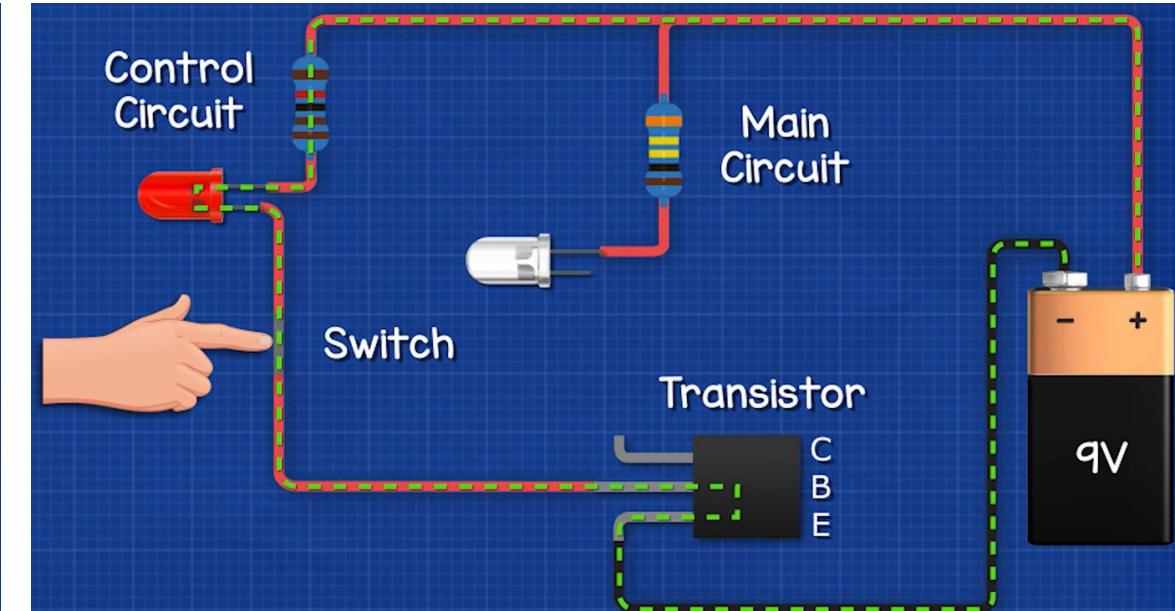
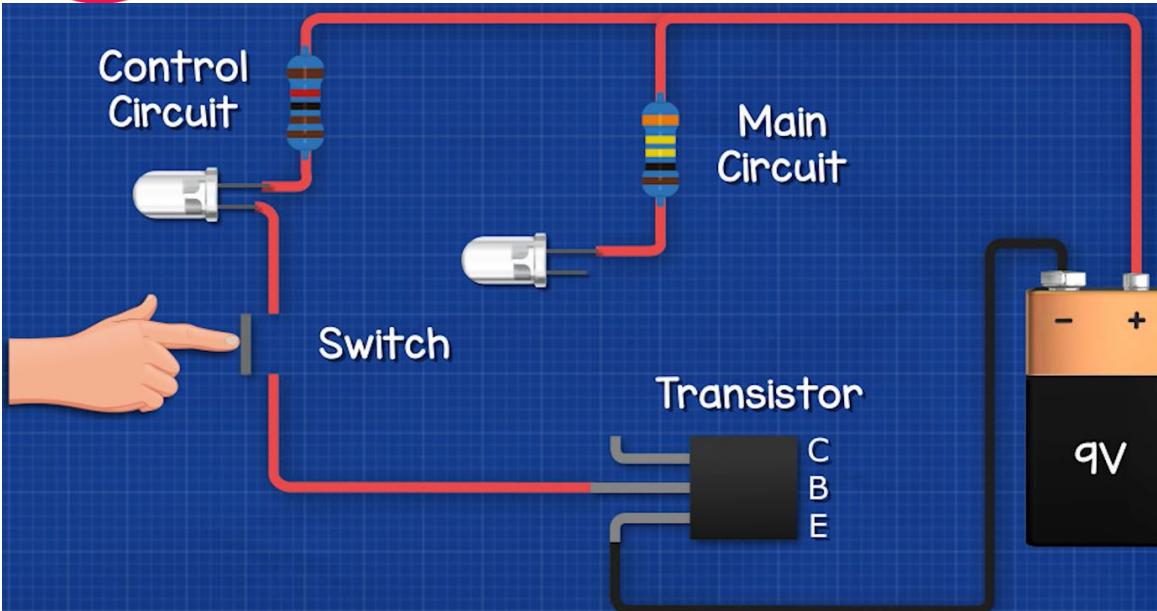


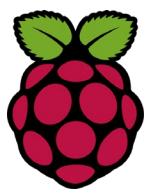
NPN and PNP Transistor





NPN and PNP Transistor





PNP Transistor

PNP Transistor

