https://www.halvorsen.blog

Raspberry Pi with Python

Hans-Petter Halvorsen

Free Textbook with lots of Practical Examples

Python for Software Development

Hans-Petter Halvorsen

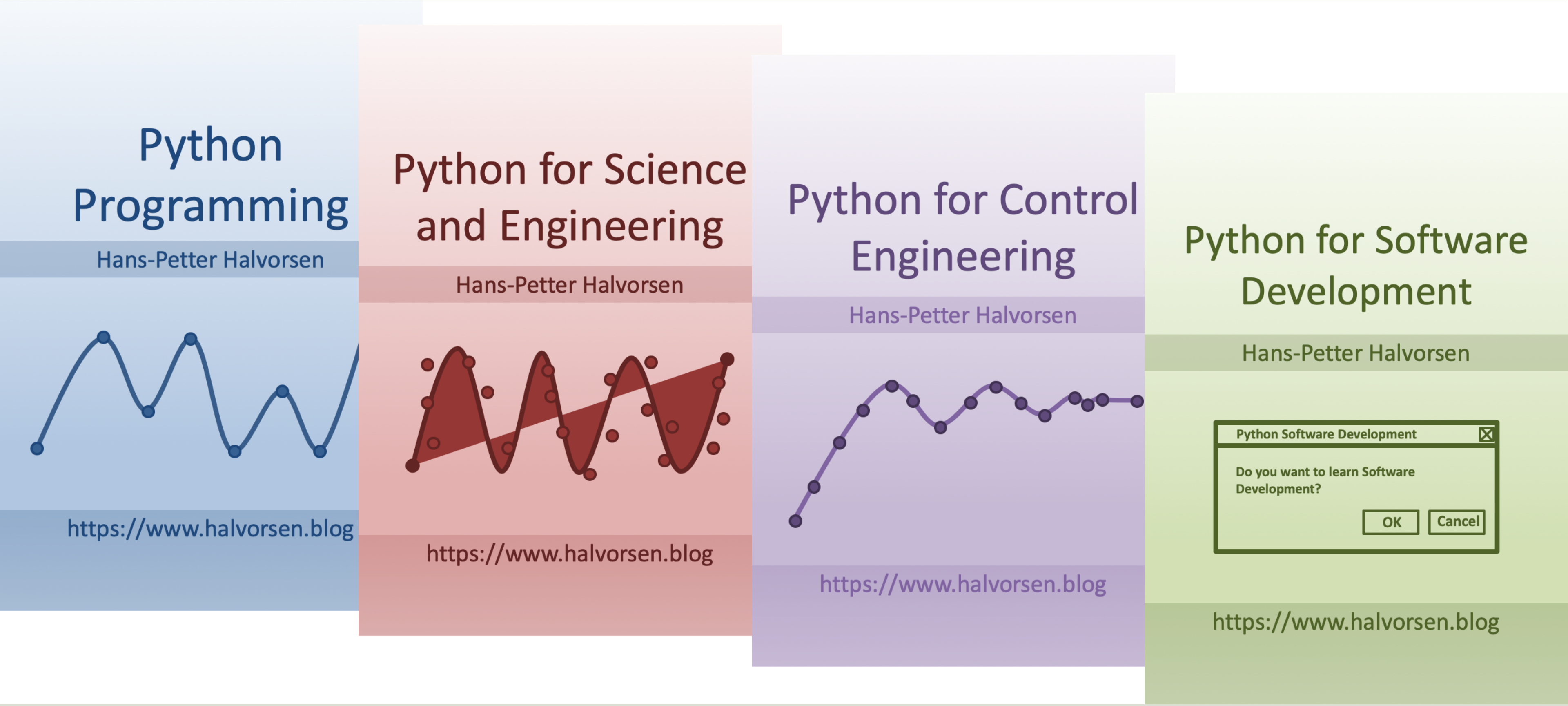
Python Software Development

Do you want to learn Software Development?

https://www.halvorsen.blog

https://www.halvorsen.blog/documents/programming/python/

Additional Python Resources



Python Programming

Hans-Petter Halvorsen

https://www.halvorsen.blog

Python for Science and Engineering

Hans-Petter Halvorsen

https://www.halvorsen.blog

Python for Control Engineering

Hans-Petter Halvorsen

https://www.halvorsen.blog

Python for Software Development

Hans-Petter Halvorsen

Python Software Development ☒

Do you want to learn Software Development?

OK Cancel

https://www.halvorsen.blog

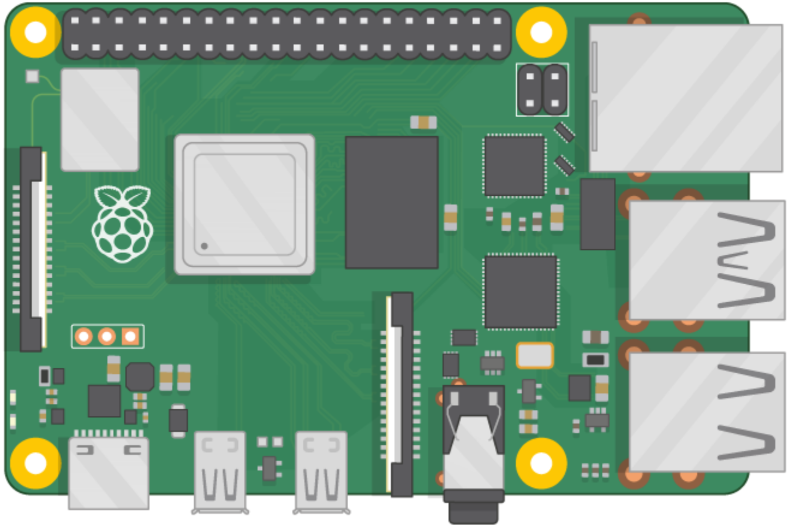
https://www.halvorsen.blog/documents/programming/python/

Contents

- Overview of Raspberry Pi
- Python on Raspberry Pi
  - Using the Thonny Python Editor
- Python
  - Basic Python Programming Examples
- Python Libraries/Packages
- GPIO with Examples

Raspberry Pi

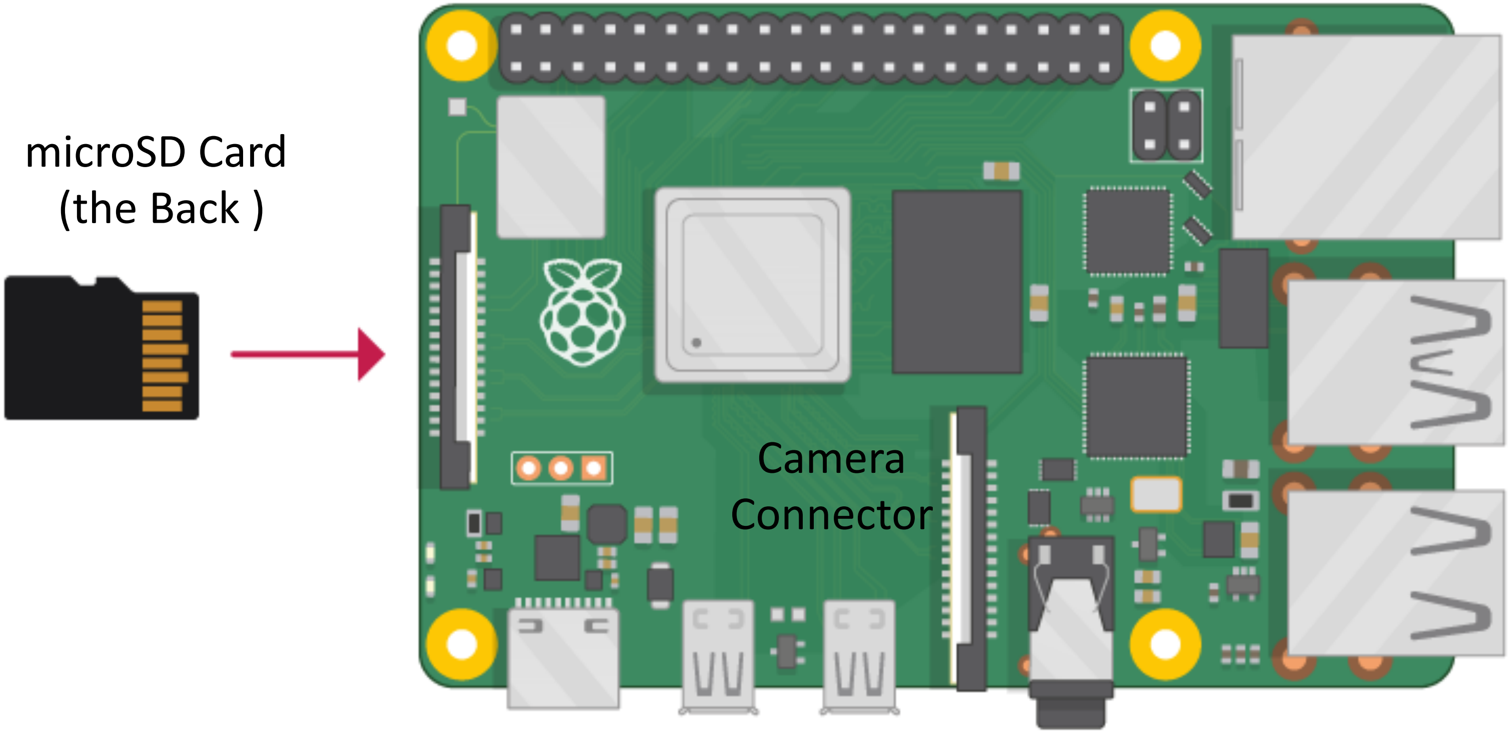
Raspberry Pi is a tiny (about 9x6cm), low-cost ($35+), single-board computer that supports embedded Linux operating systems



The recommended Operating System is called Raspberry Pi OS (Linux based)

https://www.raspberrypi.org

Raspberry Pi



GPIO Pins

microSD Card
(the Back)

Ethernet

Camera
Connector

USB A x 4

Power Supply (USB C)

microHDMI x 2

What Do you Need?

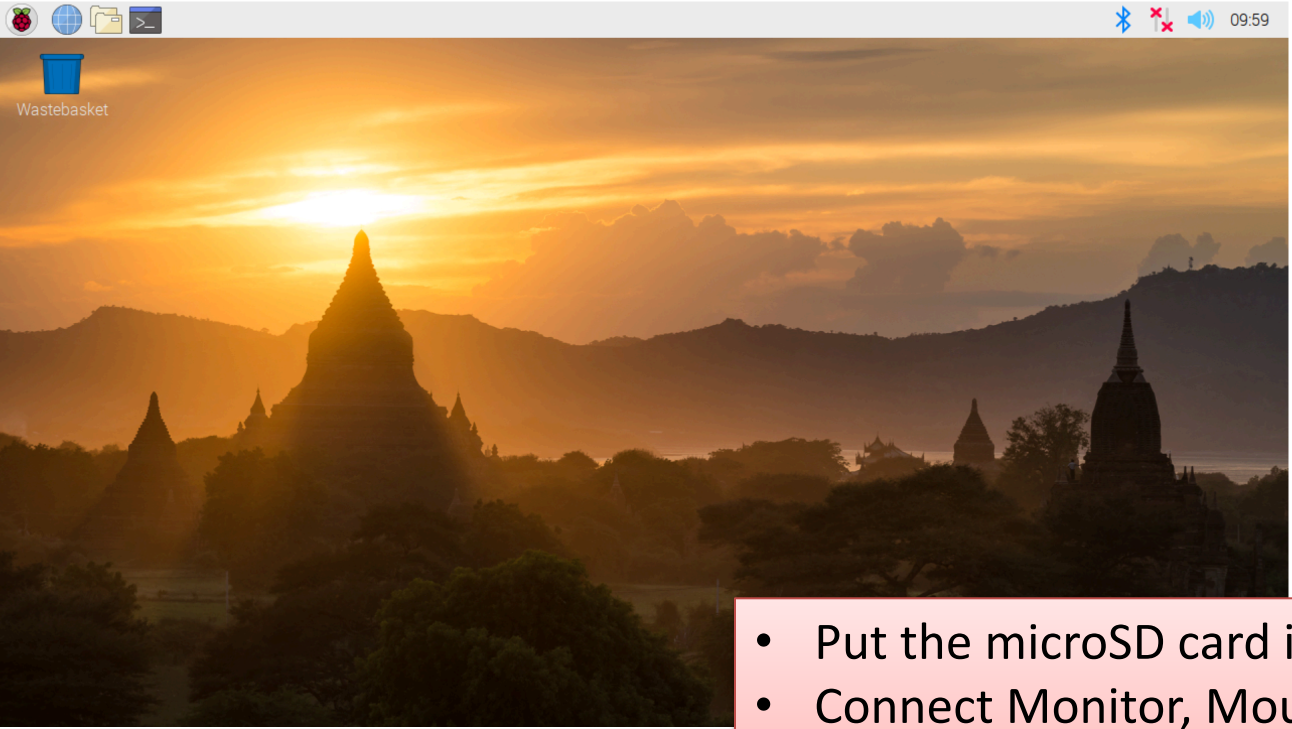
- Raspberry Pi
- microSD Card (+ Adapter)
- Power Supply
- microHDMI to HDMI Cable
- Monitor
- Mouse
- Keyboard

Raspberry Pi OS

- In order make your Raspberry Pi up and running you need to install an Operating System (OS)
- The OS for Raspberry Pi is called “Raspberry Pi OS“ (previously known as Raspbian)
- Raspberry Pi runs a version of an operating system called Linux (Windows and macOS are other operating systems).
- To install the necessary OS, you need a microSD card
- Then you use the “Raspberry Pi Imager“ in order to download the OS to the microSD card.

https://www.raspberrypi.org/software/

Start using Raspberry Pi



Raspberry Pi OS

- Put the microSD card into the Raspberry Pi
- Connect Monitor, Mouse and Keyboard
- Connect Power Supply
- Follow the Instructions on Screen to setup Wi-Fi

Remote Access

1. Install XRDP

https://en.wikipedia.org/wiki/Xrdp

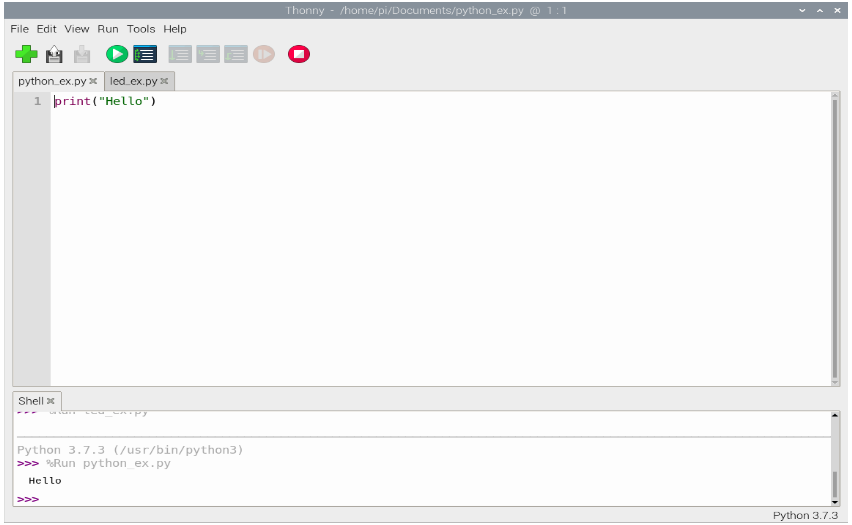
   - XRDP is a free and open-source implementation of Microsoft RDP (Remote Desktop Protocol) server. Install it by enter the following:
   - sudo apt-get install xrdp

2. Open Remote Desktop Connection (RDC) on your Windows Computer. RDS is also available for macOS

   - Enter Computer Name or IP Address
   - Default UserName is “pi“ and default Password is “raspberry“ (unless you have changed it)

Python on Raspberry Pi

- The Raspberry Pi OS comes with a basic Python Editor called ”Thonny“


Thonny - /home/pi/Documents/python_ex.py @ 1 : 1
File Edit View Run Tools Help
python_ex.py
led_ex.py
Shell
Python 3.7.3

```
print("Hello")
```

```
>>> %Run led_ex.py

Python 3.7.3 (/usr/bin/python3)
>>> %Run python_ex.py
  Hello
>>>
```

You can install and use others if you want

https://www.raspberrypi.org/documentation/usage/python/

Python Programming

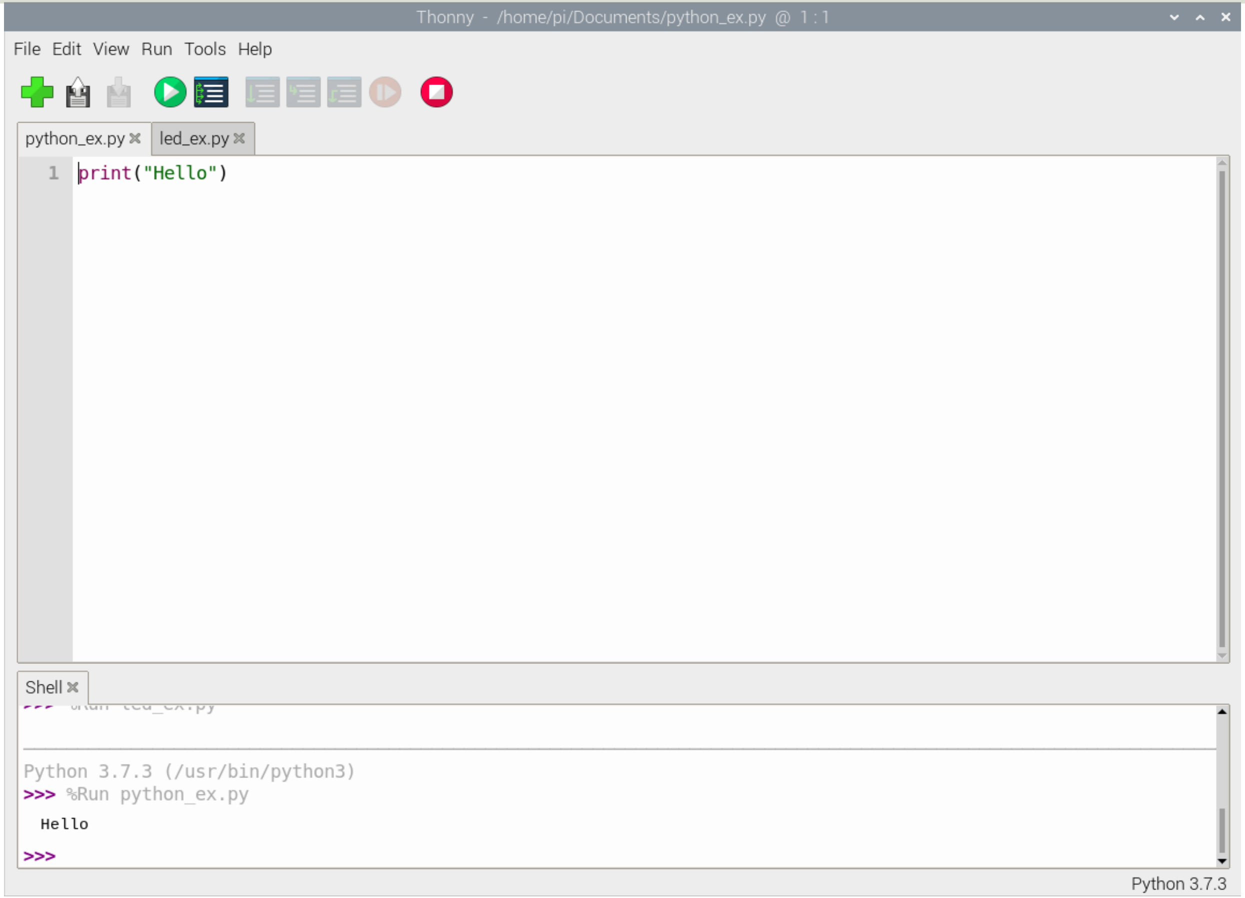
Hans-Petter Halvorsen

Python with Raspberry Pi

- Python is a fairly old Programming Language (1991) compared to many other Programming Languages like C# (2000), Swift (2014), Java (1995), PHP (1995).
- Python has during the last 10 years become more and more popular.
- Today, Python has become one of the most popular Programming Languages.
- The Raspberry Pi OS comes with a basic Python Editor called “Thonny“

https://www.raspberrypi.org/documentation/usage/python/

Hello World


Thonny - /home/pi/Documents/python_ex.py @ 1 : 1
File Edit View Run Tools Help
python_ex.py
led_ex.py
Shell
Python 3.7.3

```
print("Hello")
```

```
>>> %Run led_ex.py

Python 3.7.3 (/usr/bin/python3)
>>> %Run python_ex.py

  Hello

>>>
```

Here you also see the “Thonny“ Python Editor

Free Textbook with lots of Practical Examples

Python Programming

Hans-Petter Halvorsen

https://www.halvorsen.blog

https://www.halvorsen.blog/documents/programming/python/

Variables in Python

Creating variables:

```
> x = 3
> x
3
```

We can implement the formula $y(x) = ax + b$ like this:

$$y(x) = 2x + 4$$

We can use variables in a calculation like this:

```
> x = 3
> y = 3*x
> print(y)
```

```
> a = 2
> b = 4

> x = 3
> y = a*x + b
> print(y)
```

A variable can have a short name (like x and y) or a more descriptive name (sum, amount, etc). You don need to define the variables before you use them (like you need to to in, e.g., C/C++/C).

Calculations in Python

We can use variables in a calculation like this:

$$y(x) = 2x + 4$$

```
> a = 2
> b = 4
```

$y(3)$ = ?

```
> x = 3
> y = a*x + b
> print(y)
```

$y(5)$ = ?

```
> x = 5
> y = a*x + b
> print(y)
```

Math in Python

If we need only the sin() function, we can do like this:

```
from math import sin

x = 3.14
y = sin(x)
```

If we need many functions, we can do like this:

```
from math import *

x = pi
y = sin(x)
print(y)

y = cos(x)
print(y)

…
```

If we need a few functions, we can do like this:

```
from math import sin, cos

x = 3.14
y = sin(x)
print(y)

y = cos(x)
print(y)
```

We can also do like this:

```
import math
x = 3.14
y = math.sin(x)
print(y)
```

If-Else

If you have 2 conditions that you need to check, you can use If – Else:

```
a = 5
b = 8

if a > b:
    print("a is greater than b")
else:
    print("b is greater than a or a and b are equal")
```

Arrays

An array is a special variable, which can hold more than one value at a time

Example:

```
data = [1.6, 3.4, 5.5, 9.4]
```

Python does not have built-in support for Arrays, but Python Lists can be used instead.

Length of an Array (List):

```
N = len(data)
```

Get a specific element (Indexing):

```
x = data[2]
```

Change a specific element:

```
data[2] = 7.3
```

Add a new value to the end of the Array (List):

```
data.append(11.4)
```

For more advanced use of Arrays in Python you will have to import a library, like the **NumPy** library.

Using Arrays in Functions

Using Arrays in Functions

Note! statistics is a sub library in the Python Standard Library

Example:

```
from statistics import *

data = [1.6, 3.4, 5.5, 9.4]

m = mean(data)
sd = stdev(data)
datamin = min(data)
datamax = max(data)
```

For Loops

A For loop is used for iterating over a sequence. I guess all your programs will use one or more For loops. So if you have not used For loops before, make sure to learn it now.

Example:

```
cars = ["Ford", "Toyota", "Tesla"]

for car in cars:
  print(car)
```

Array (List) of Strings

Note! Python uses indentation (spaces)

Other Programming Languages uses curly brackets {} or Begin .. End

Example:

```
data = [1.6, 3.4, 5.5, 9.4]

for x in data:
    print (x)
```

Array (List) of Numbers

For Loops

The **range()** function is handy to use in For Loops:

```
N = 10

for x in range(N):
  print(x)
```

The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

You can also use the range() function like this:

```
start = 4
stop= 12 #but not including

for x in range(start, stop):
  print(x)
```

Or like this:

While Loops

```
i = 1
while i < 10:
  print(i)
  i = i + 1
```

```
1
2
3
4
5
6
7
8
9
```

```
data = [1.6, 3.4, 4.4, 5.5, 9.4]

max = 5

i = 0
while data[i] < max:
  print(data[i])
  i = i + 1
```

```
1.6
3.4
4.4
```

While Loops

```
data = [1.6, 3.4, 4.4, 5.5, 9.4]

N = len(data)

sum = 0

i = 0
while i < N:
  sum = sum + data[i]
  i = i + 1

print(sum)
```

```
24.3
```

Create Functions

Create the Function:

```
def add(x,y):
    z = x + y
    return z
```

Using the Function within the same script:

```
def add(x,y):
    z = x + y
    return z

# Using the Function:
x = 2
y = 5

z = add(x,y)

print(z)
```

Create Functions

- Although you can mix functions and code in one file, it is much better to create the functions in separate .py files
- In that way you can easily reuse the function in different Python scripts

1

We start by creating a separate Python File, e.g., "**myfunctions.py**" for the function:

myfunctions.py:

```
def average(x,y):

    return (x + y)/2
```

2 Next, we create a new Python File (e.g., "**testaverage.py**") where we use the function we created:

```
from myfunctions import average

a = 2
b = 3

c = average(a,b)

print(c)
```

Free Textbook with lots of Practical Examples

Python Programming

Hans-Petter Halvorsen

https://www.halvorsen.blog

https://www.halvorsen.blog/documents/programming/python/

Additional Resources

- Python Programming:
  https://www.halvorsen.blog/documents/programming/python/

- Python Programming Tutorial: Getting Started with the Raspberry Pi
  https://learn.sparkfun.com/tutorials/python-programming-tutorial-getting-started-with-the-raspberry-pi/

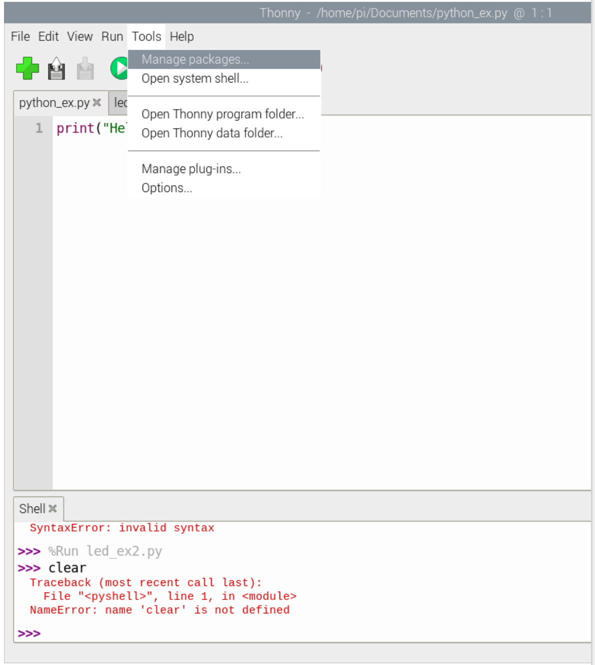
Python Libraries/ Packages

Python Packages/Libraries

- Rather than having all its functionality built into its core, Python was designed to be highly extensible.
- This approach has advantages and disadvantages.
- A disadvantage is that you need to install these packages separately and then later import these modules in your code.
- Some important packages are:
  - **NumPy** - NumPy is the fundamental package for scientific computing with Python
  - **Matplotlib** – With this library you can easily make plots in Python

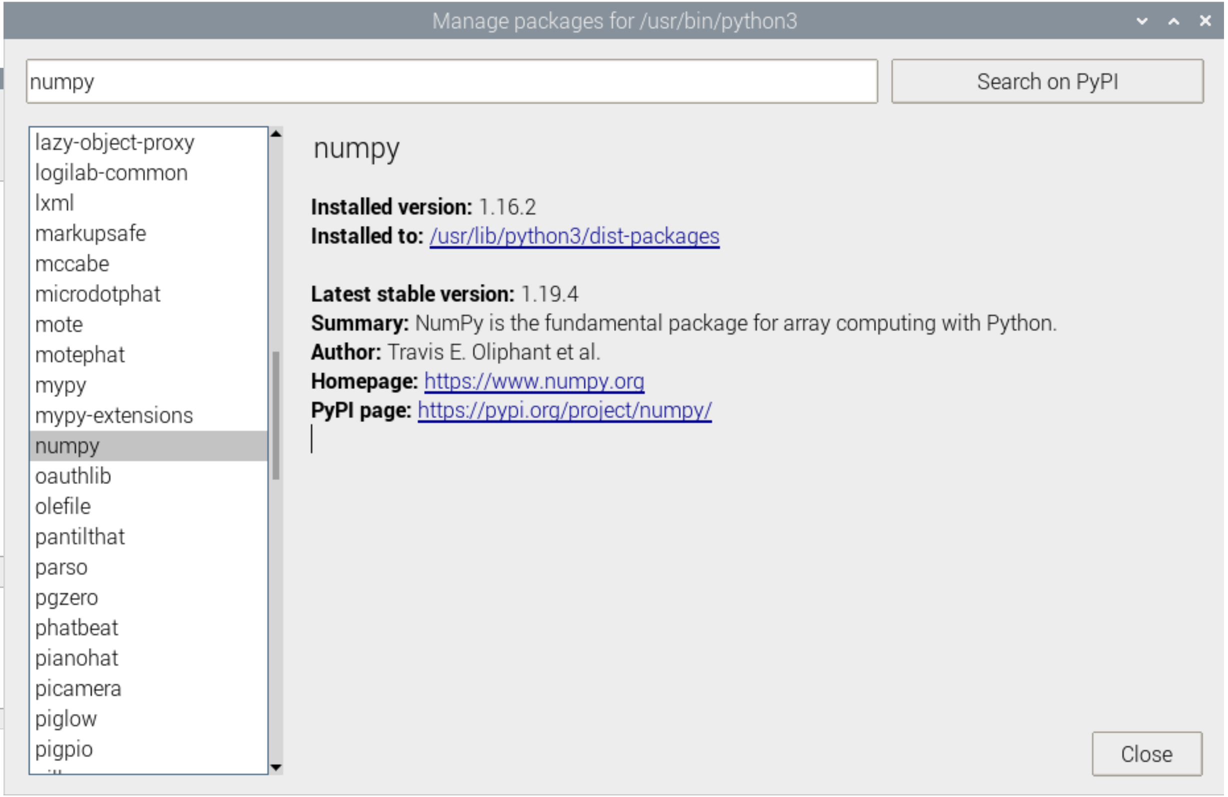
Python Packages with Thonny

Tools -> Manage packages…



```
print("He
```

```
SyntaxError: invalid syntax

>>> %Run led_ex2.py
>>> clear
Traceback (most recent call last):
  File "<pyshell>", line 1, in <module>
NameError: name 'clear' is not defined

>>>
```



lazy-object-proxy
logilab-common
lxml
markupsafe
mccabe
microdotphat
mote
motephat
mypy
mypy-extensions
numpy
oauthlib
olefile
pantilthat
parso
pgzero
phatbeat
pianohat
picamera
piglow
pigpio

numpy

Installed version: 1.16.2
Installed to: /usr/lib/python3/dist-packages

Latest stable version: 1.19.4
Summary: NumPy is the fundamental package for array computing with Python.
Author: Travis E. Oliphant et al.
Homepage: https://www.numpy.org
PyPI page: https://pypi.org/project/numpy/

Installing Python Packages

There are multiple ways to install Python Libraries/ Packages on Raspberry Pi

- apt: Some Python packages can be found in the Raspberry Pi OS archives and can be installed using apt. Example

```
sudo apt update
sudo apt install python3-picamera
```

- pip: Not all Python packages are available in the Raspberry Pi OS archives, and those that are can sometimes be out-of-date. If you can't find a suitable version in the Raspberry Pi OS archives, you can install packages from the Python Package Index (PyPI). To do so, use the pip tool. Example:

```
sudo pip3 install libraryname
```

- piwheels: piwheels is a Python package repository specifically for the Raspberry Pi

https://www.raspberrypi.org/documentation/linux/software/python.md

NumPy

- A Python Library for Numerical Operations, Arrays, etc.
- The NumPy Python Library is installed on the Raspberry Pi OS by default
- https://numpy.org

NumPy Example

Basic NumPy Example:

```
import numpy as np

x = 3

y = np.sin(x)

print(y)
```

In this example we use both the math module in the Python Standard Library and the NumPy library:

```
import math as mt
import numpy as np

x = 3

y = mt.sin(x)
print(y)

y = np.sin(x)
print(y)
```

As you see, NumPy also have also similar functions (e.g., sim(), cos(), etc.) as those who is part of the math library, but they are more powerful

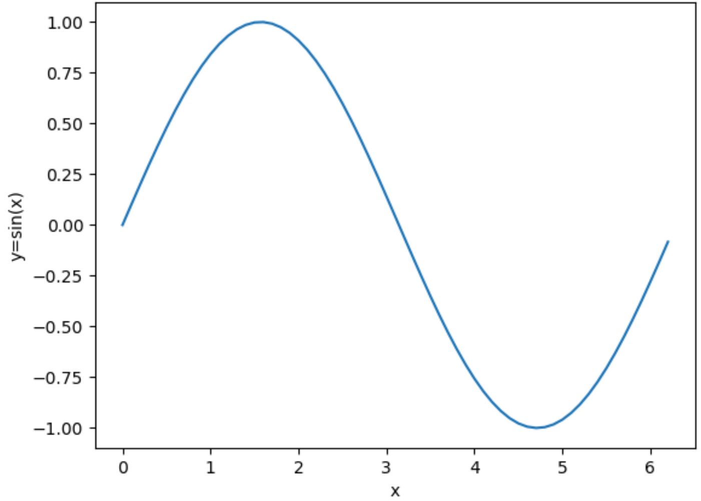
Matplotlib

- Typically you need to create some plots or charts. In order to make plots or charts in Python you will need an external library. The most used library is Matplotlib
- Matplotlib is a Python 2D plotting library
- Here you find an overview of the Matplotlib library: https://matplotlib.org
- The NumPy Python Library is NOT installed on the Raspberry Pi OS by default, so you must manually install it

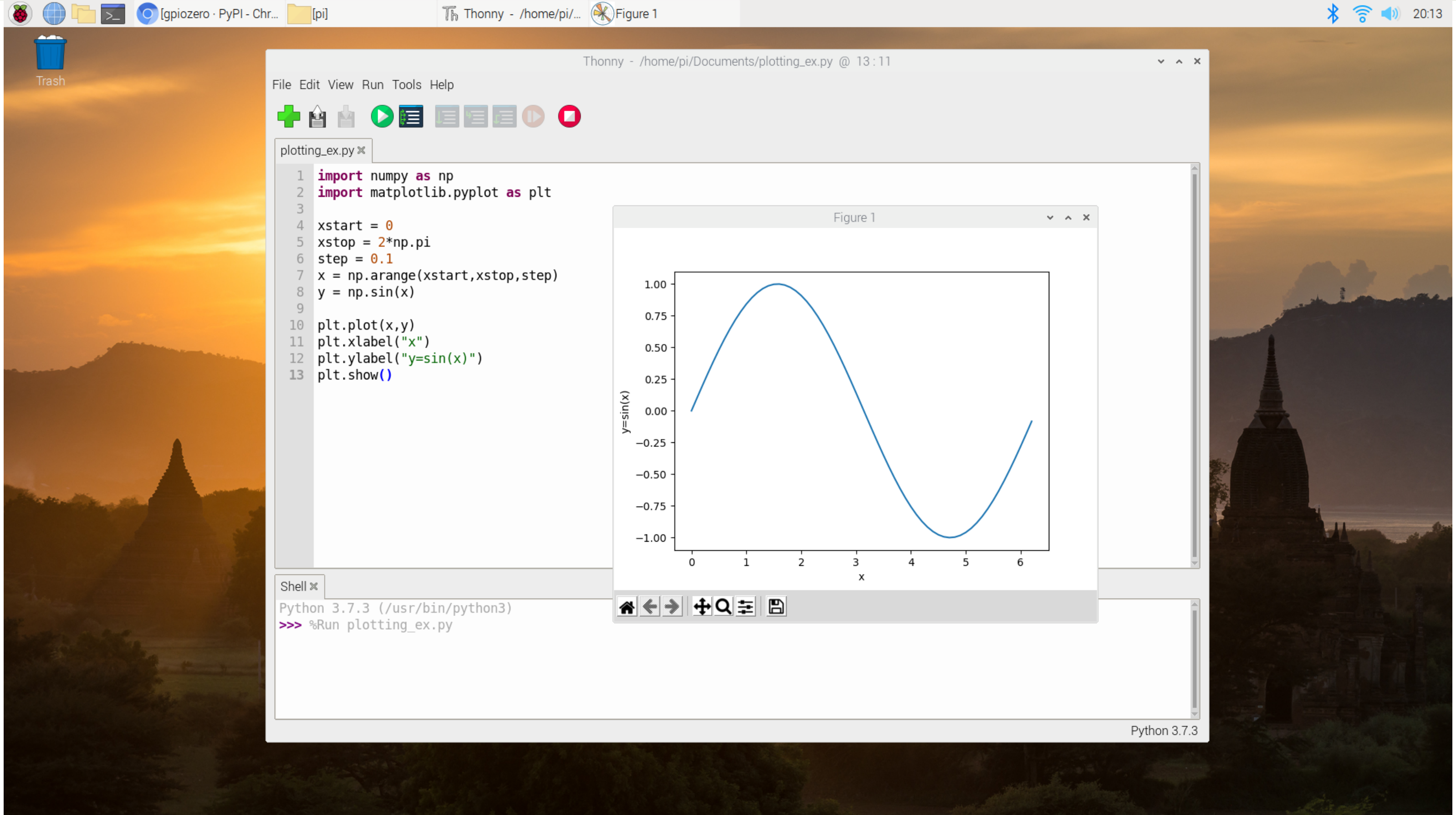
Matplotlib Example

Plotting a Sine Curve

```
import numpy as np
import matplotlib.pyplot as plt

xstart = 0
xstop = 2*np.pi
step = 0.1

x = np.arange(xstart, xstop, step)
y = np.sin(x)

plt.plot(x, y)
plt.xlabel('x')
plt.ylabel('y=sin(x)')
plt.show()
```



Matplotlib Example



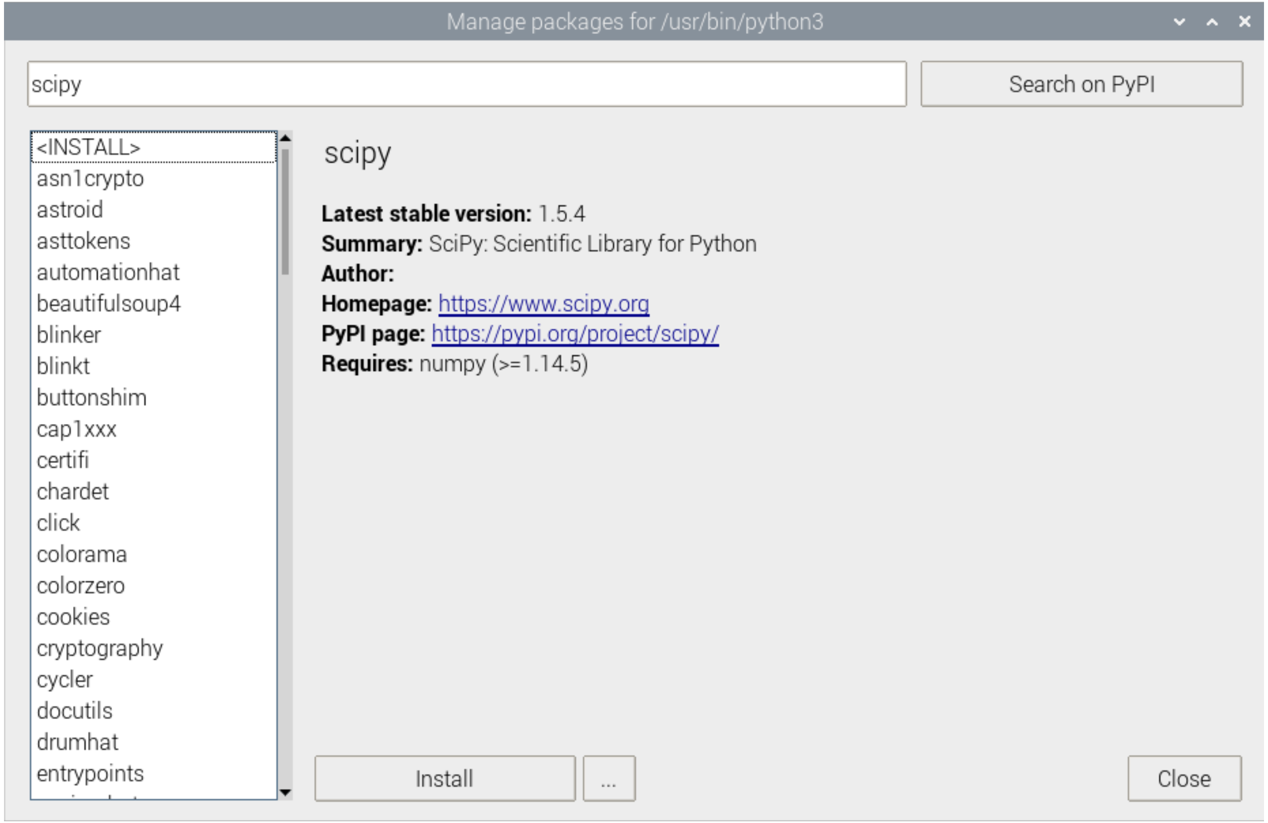
```python
import numpy as np
import matplotlib.pyplot as plt

xstart = 0
xstop = 2*np.pi
step = 0.1
x = np.arange(xstart,xstop,step)
y = np.sin(x)

plt.plot(x,y)
plt.xlabel("x")
plt.ylabel("y=sin(x)")
plt.show()
```

```
Python 3.7.3 (/usr/bin/python3)
>>> %Run plotting_ex.py
```

SciPy

- SciPy has many functions for Mathematics and Scientific Computing
- https://scipy.org
- https://docs.scipy.org/doc/scipy/reference/

Install SciPy with Thonny



scipy

Latest stable version: 1.5.4
Summary: SciPy: Scientific Library for Python
Author:
Homepage: https://www.scipy.org
PyPI page: https://pypi.org/project/scipy/
Requires: numpy (>=1.14.5)

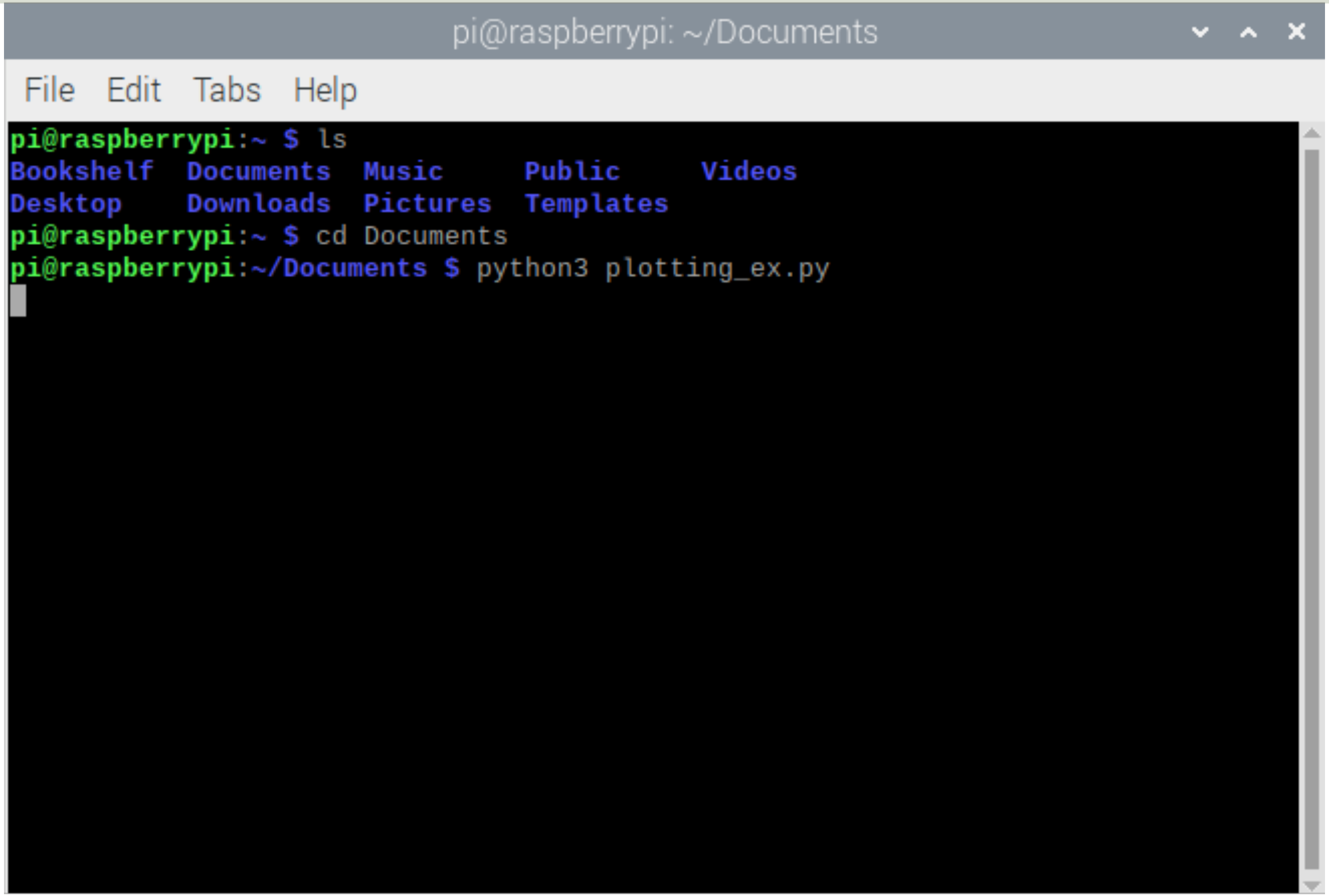
Python from Command Line

Python from Command Line

- You can write a Python file in a standard editor
- Then you run it as a Python script from the command line.
- Just navigate to the directory where the file is saved in (use commands cd and ls for navigation)

```
python3 hello.py
```

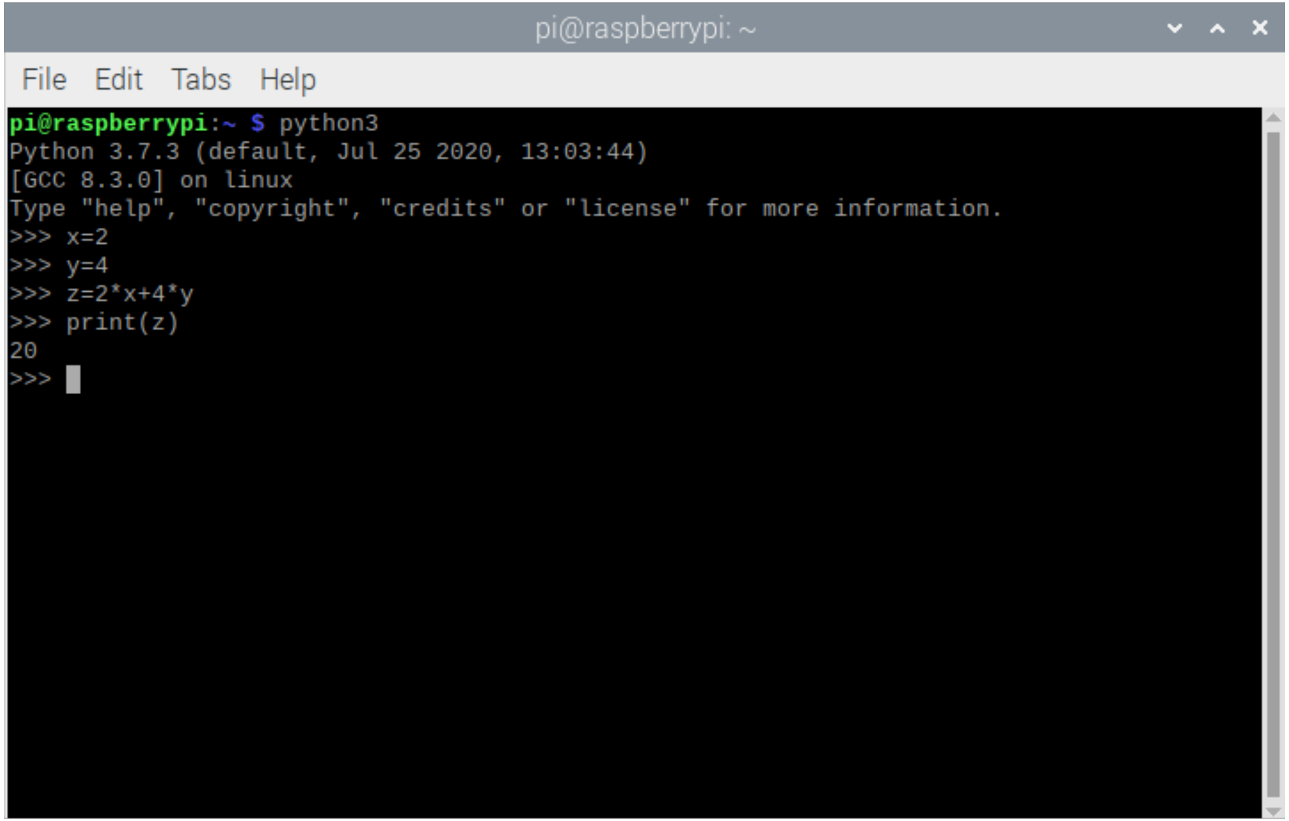
Python from Command Line



```
pi@raspberrypi:~ $ ls
Bookshelf  Documents  Music     Public     Videos
Desktop    Downloads  Pictures  Templates
pi@raspberrypi:~ $ cd Documents
pi@raspberrypi:~/Documents $ python3 plotting_ex.py
```

Python Shell from Terminal

Enter python3 in the Terminal



```
pi@raspberrypi:~ $ python3
Python 3.7.3 (default, Jul 25 2020, 13:03:44)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> x=2
>>> y=4
>>> z=2*x+4*y
>>> print(z)
20
>>> 
```

GPIO

GPIO Features

The GPIO pins are Digital Pins which are either True (+3.3V) or False (0V). These can be used to turn on/off LEDs, etc.

The Digital Pins can be either Output or Input.

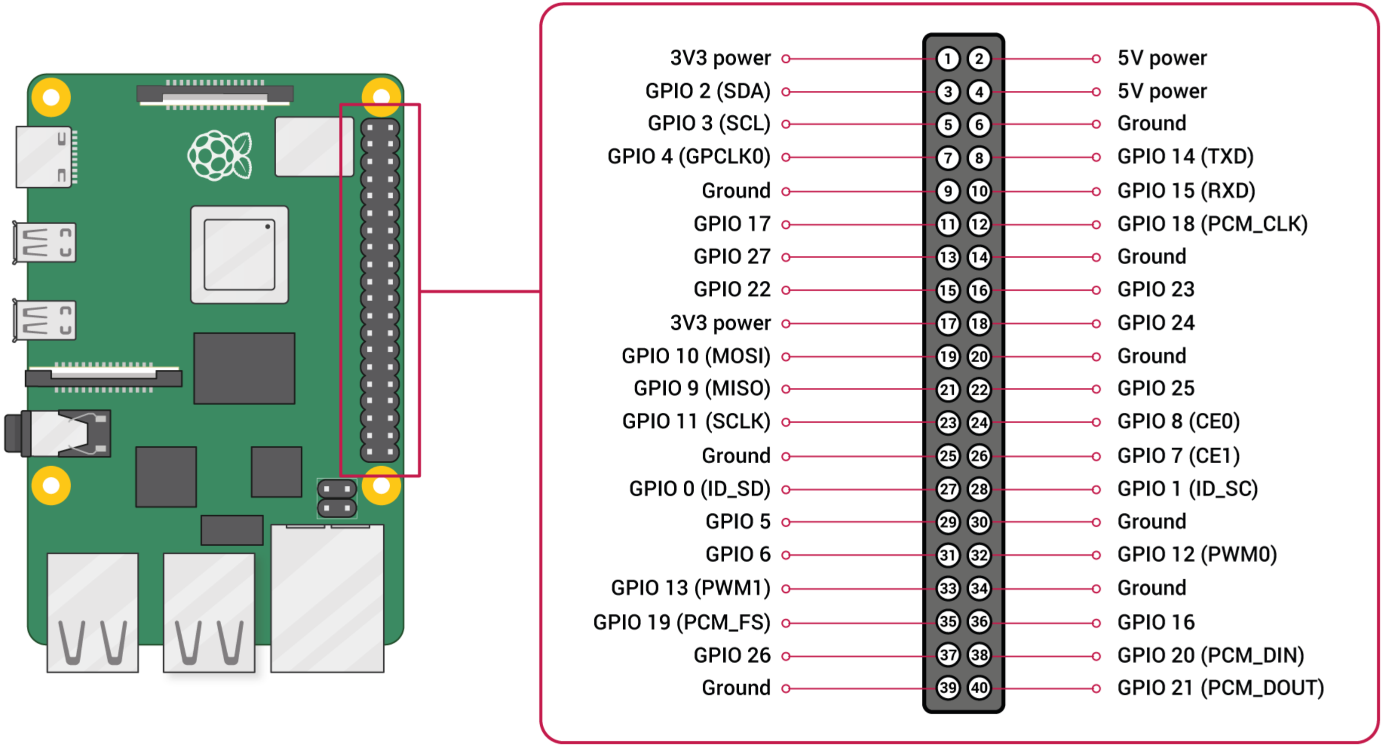
In addition, some of the pins also offer some other Features:

- PWM (Pulse Width Modulation)

Digital Buses (for reading data from Sensors, etc.):

- SPI
- I2C

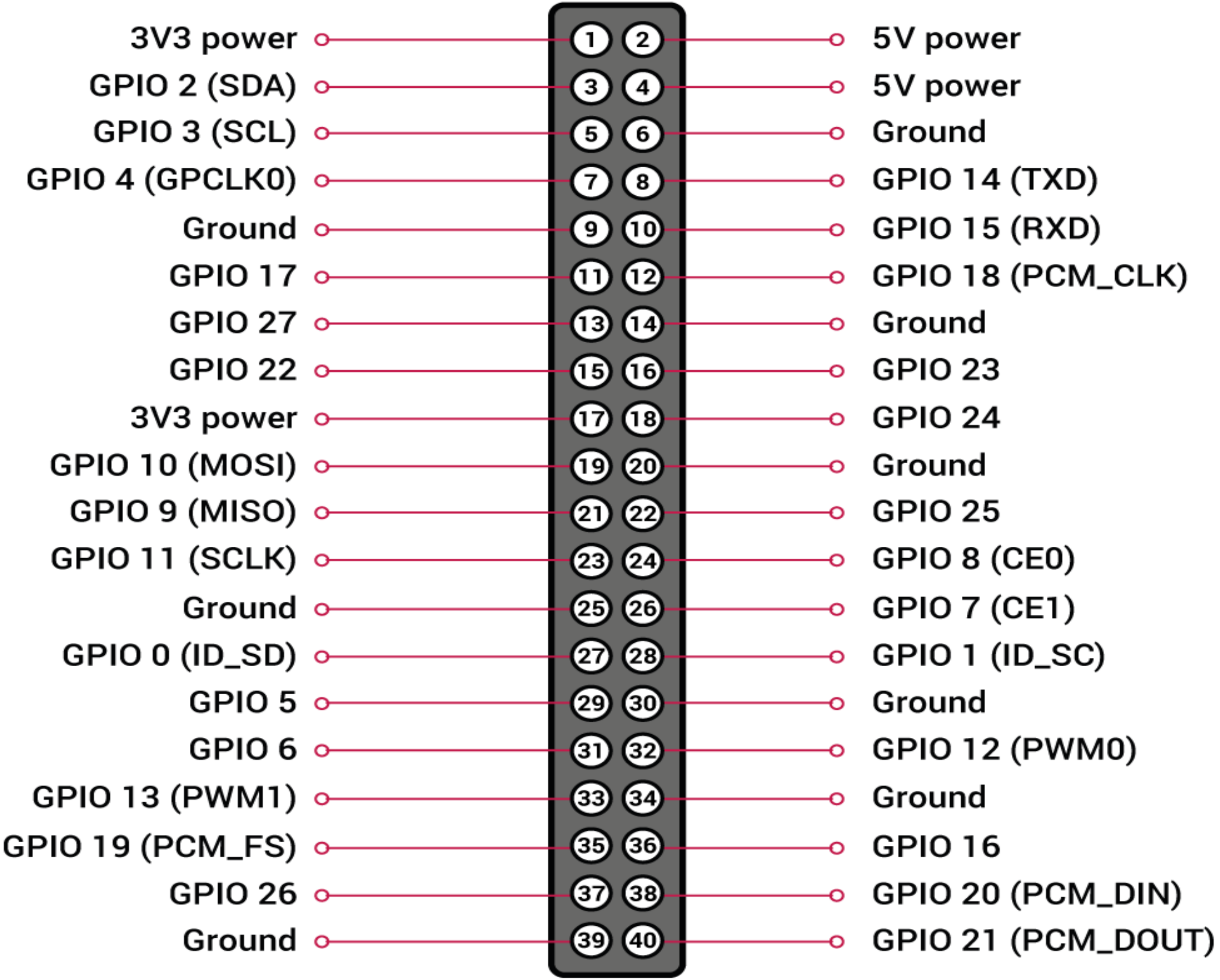
GPIO



3V3 power	1	2	5V power
GPIO 2 (SDA)	3	4	5V power
GPIO 3 (SCL)	5	6	Ground
GPIO 4 (GPCLK0)	7	8	GPIO 14 (TXD)
Ground	9	10	GPIO 15 (RXD)
GPIO 17	11	12	GPIO 18 (PCM_CLK)
GPIO 27	13	14	Ground
GPIO 22	15	16	GPIO 23
3V3 power	17	18	GPIO 24
GPIO 10 (MOSI)	19	20	Ground
GPIO 9 (MISO)	21	22	GPIO 25
GPIO 11 (SCLK)	23	24	GPIO 8 (CE0)
Ground	25	26	GPIO 7 (CE1)
GPIO 0 (ID_SD)	27	28	GPIO 1 (ID_SC)
GPIO 5	29	30	Ground
GPIO 6	31	32	GPIO 12 (PWM0)
GPIO 13 (PWM1)	33	34	Ground
GPIO 19 (PCM_FS)	35	36	GPIO 16
GPIO 26	37	38	GPIO 20 (PCM_DIN)
Ground	39	40	GPIO 21 (PCM_DOUT)

A powerful feature of the Raspberry Pi is the GPIO (general-purpose input/output) pins. The Raspberry Pi has a 40-pin GPIO header as seen in the image

GPIO



3V3 power	1	2	5V power
GPIO 2 (SDA)	3	4	5V power
GPIO 3 (SCL)	5	6	Ground
GPIO 4 (GPCLK0)	7	8	GPIO 14 (TXD)
Ground	9	10	GPIO 15 (RXD)
GPIO 17	11	12	GPIO 18 (PCM_CLK)
GPIO 27	13	14	Ground
GPIO 22	15	16	GPIO 23
3V3 power	17	18	GPIO 24
GPIO 10 (MOSI)	19	20	Ground
GPIO 9 (MISO)	21	22	GPIO 25
GPIO 11 (SCLK)	23	24	GPIO 8 (CE0)
Ground	25	26	GPIO 7 (CE1)
GPIO 0 (ID_SD)	27	28	GPIO 1 (ID_SC)
GPIO 5	29	30	Ground
GPIO 6	31	32	GPIO 12 (PWM0)
GPIO 13 (PWM1)	33	34	Ground
GPIO 19 (PCM_FS)	35	36	GPIO 16
GPIO 26	37	38	GPIO 20 (PCM_DIN)
Ground	39	40	GPIO 21 (PCM_DOUT)

GPIO with Python

Raspberry Pi GPIO and Python

- You can make all kinds of Python program on your Raspberry Pi
- But you could have used your ordinary desktop/laptop PC for that
- The UNIQUE thing with Raspberry Pi compared to an ordinary PC is the GPIO connector
- With GPIO you can connect LEDs, Sensors, control Motors, etc.
- You typically use Python in order communicate with GPIO connector
- That what's makes the combination Raspberry Pi + Python UNIQUE!

GPIO in Python

- In order to use and communicate with the GPIO Pins we typically use the Python Programming Language
- We can turn on LEDS, read data from different types of Sensors, etc.

https://www.raspberrypi.org/documentation/usage/gpio/python/

GPIO Zero

- The GPIO Zero Python Library can be used to communicate with GPIO Pins
- The GPIO Zero Python Library comes preinstalled with the Raspberry Pi OS (so no additional installation is necessary)

Resources:

- https://www.raspberrypi.org/documentation/usage/gpio/python/
- https://pypi.org/project/gpiozero/
- https://gpiozero.readthedocs.io/en/stable/
- https://gpiozero.readthedocs.io/en/stable/recipes.html

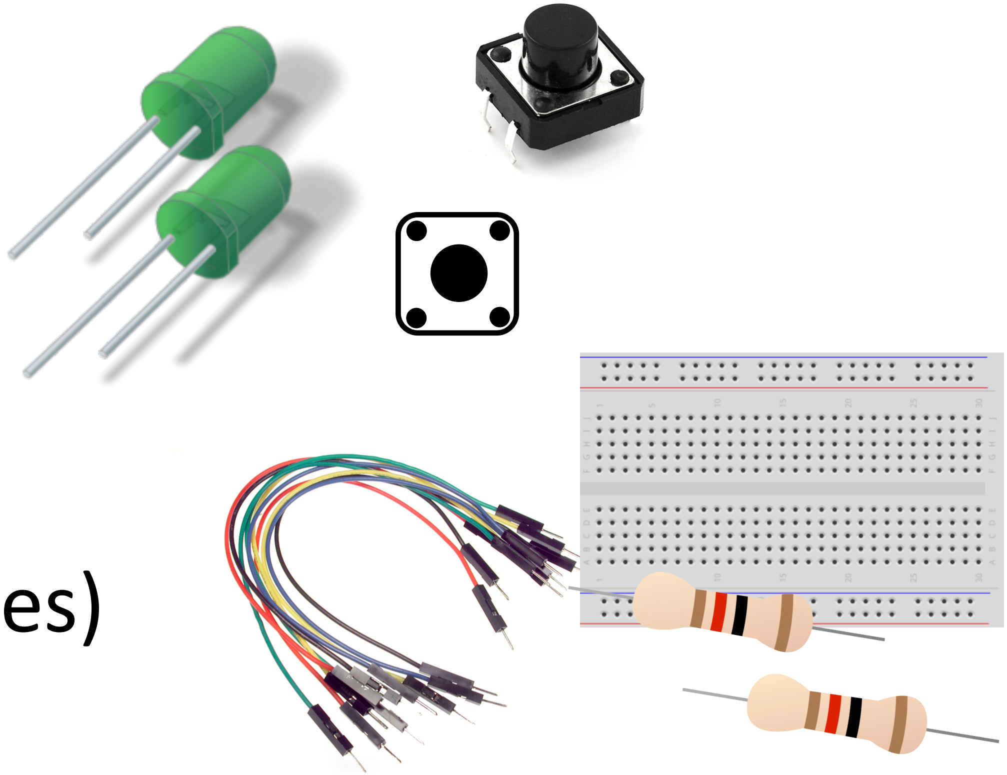
RPi.GPIO

- Rpi.GPIO is a module controlling the GPIO pins on the Raspberry Pi
- RPi.GPIO is a more “low-level“ Python Library than GPIO Zero. Actually, GPIO Zero is using RPi.GPIO
- The RPi.GPIO Python Library comes preinstalled with the Raspberry Pi OS (so no additional installation is necessary)

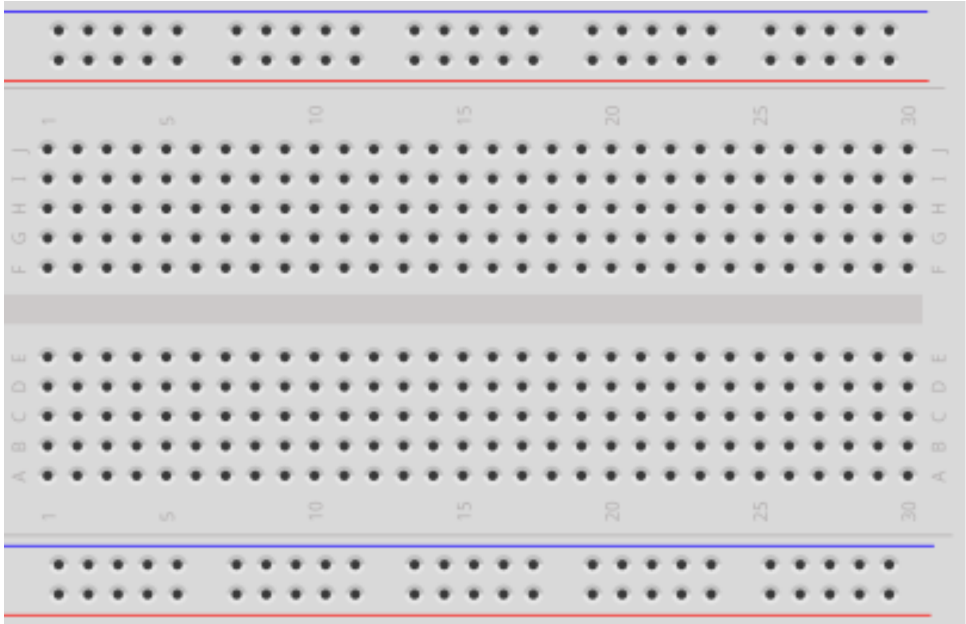
https://pypi.org/project/RPi.GPIO/

Necessary Equipment

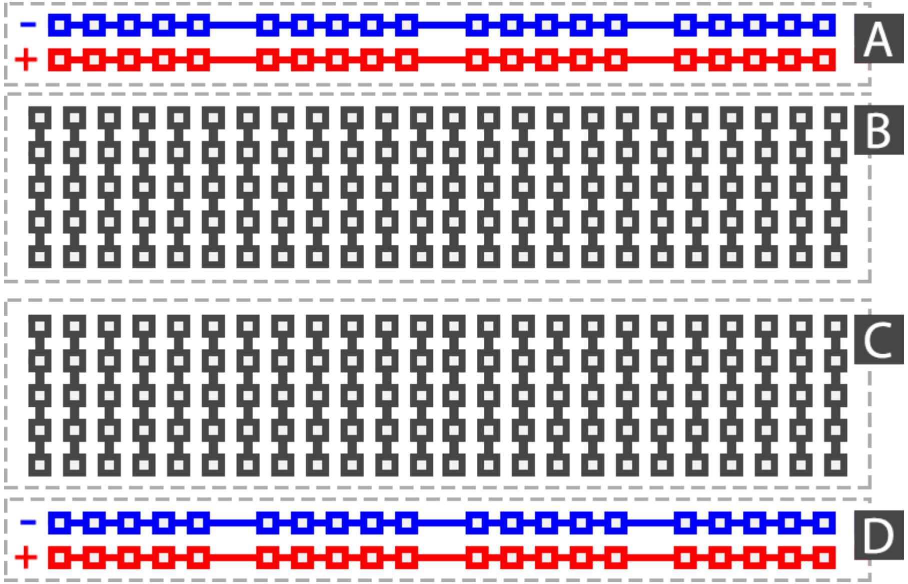
- Raspberry Pi
- Breadboard
- LEDs
- Push Buttons
- Resistors
- Wires (Jumper Wires)



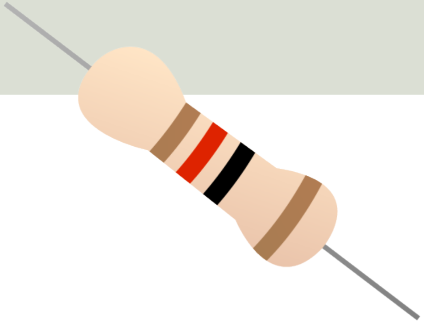
Breadboard



A breadboard is used to wire electric components together

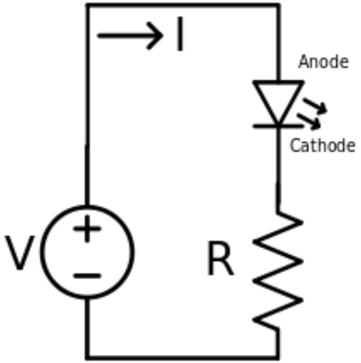


Resistors



Resistance is measured in Ohm (Ω)

Resistors comes in many sizes, e.g., 220Ω , 270Ω, 330Ω, 1kΩm 10kΩ, ...

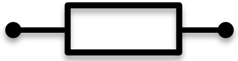


The resistance can be found using **Ohms Law**

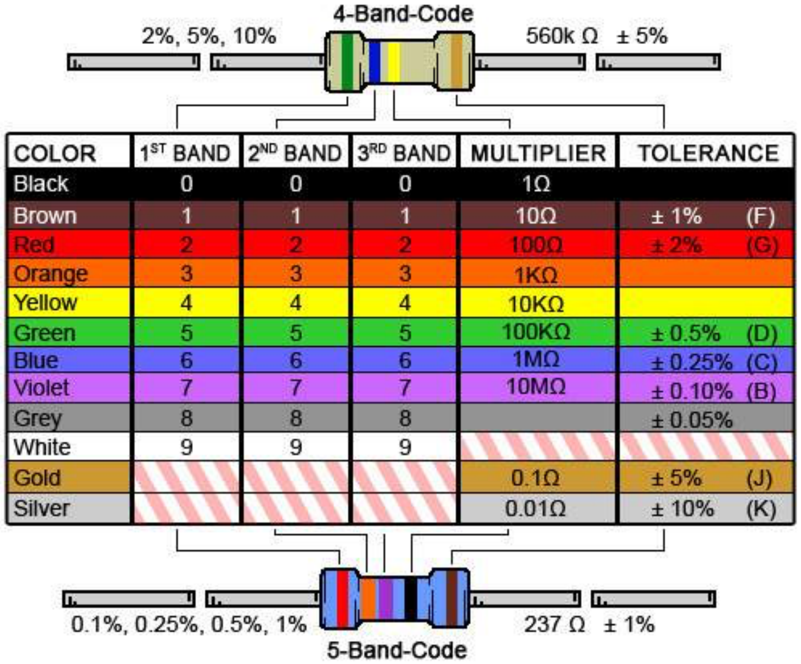
$$U = RI$$

https://en.wikipedia.org/wiki/Resistor

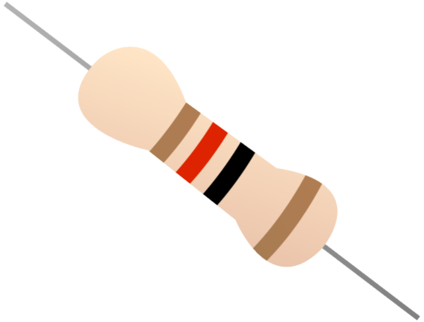
Electrical symbol:



Resistor Colors


4-Band-Code
2%, 5%, 10%
560k Ω ± 5%
0.1%, 0.25%, 0.5%, 1%
237 Ω ± 1%
5-Band-Code

COLOR	1ST BAND	2ND BAND	3RD BAND	MULTIPLIER	TOLERANCE
Black	0	0	0	1Ω	
Brown	1	1	1	10Ω	± 1% (F)
Red	2	2	2	100Ω	± 2% (G)
Orange	3	3	3	1KΩ	
Yellow	4	4	4	10KΩ	
Green	5	5	5	100KΩ	± 0.5% (D)
Blue	6	6	6	1MΩ	± 0.25% (C)
Violet	7	7	7	10MΩ	± 0.10% (B)
Grey	8	8	8		± 0.05%
White	9	9	9	▨	▨
Gold	▨	▨	▨	0.1Ω	± 5% (J)
Silver	▨	▨	▨	0.01Ω	± 10% (K)



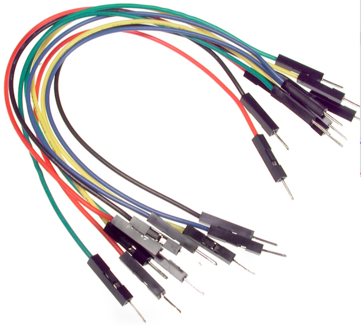
You can also use a **Multimeter**

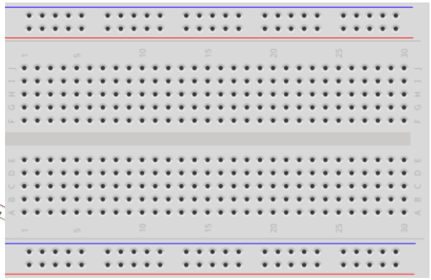
Resistor Calculator: http://www.allaboutcircuits.com/tools/resistor-color-code-calculator/

LED

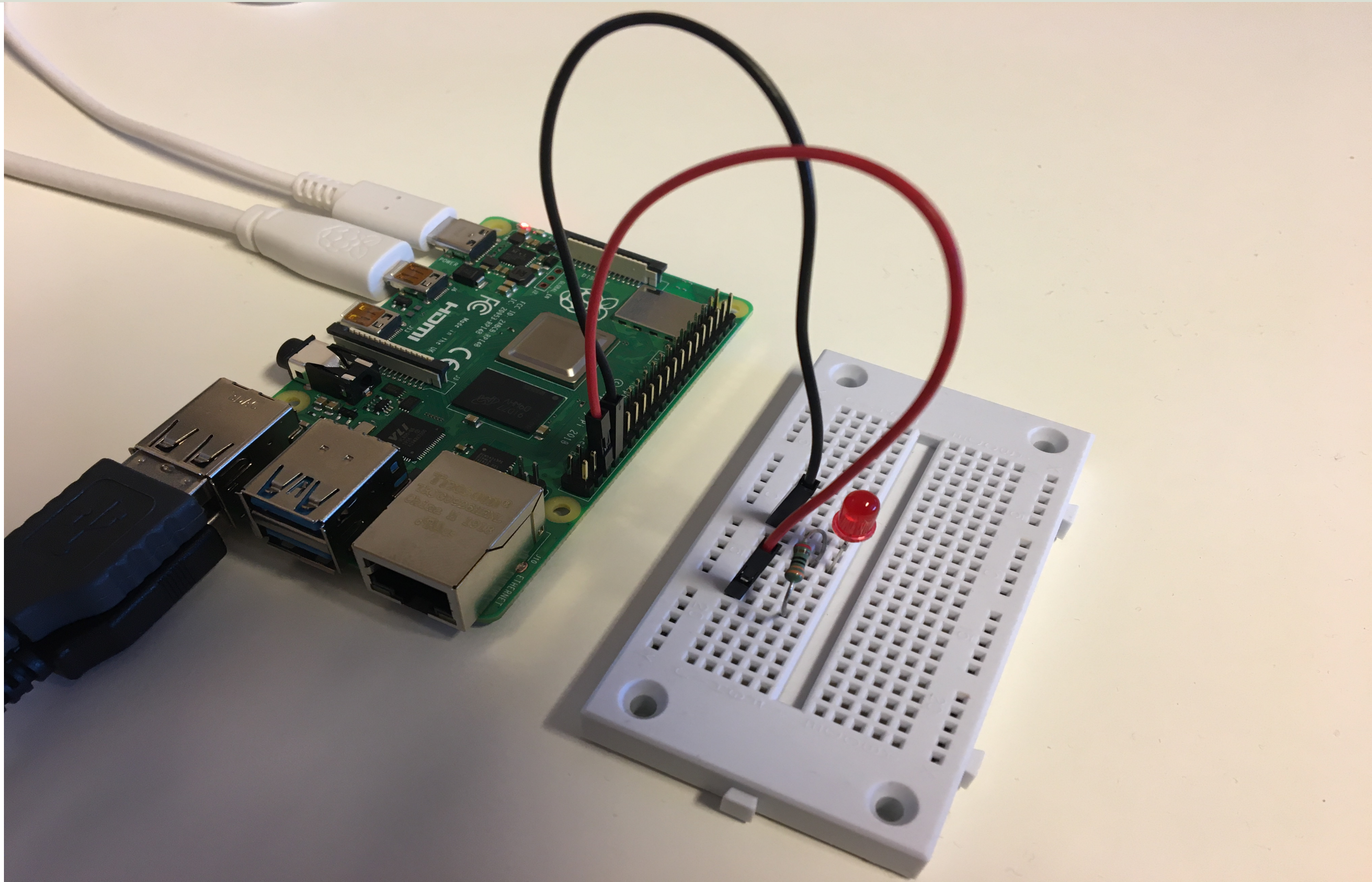
Necessary Equipment

- Raspberry Pi
- Breadboard
- LED
- Resistor, $R = 270\Omega$
- Wires (Jumper Wires)

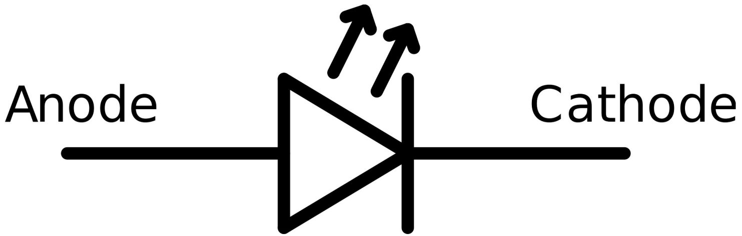


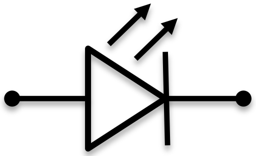


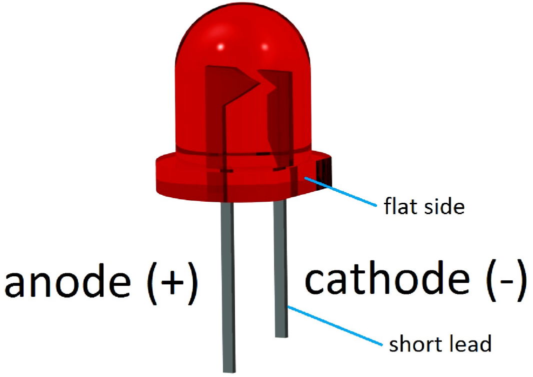
Setup and Wiring

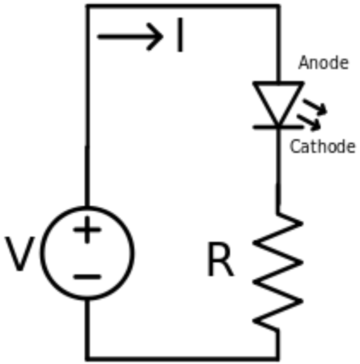


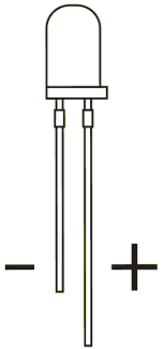
LED





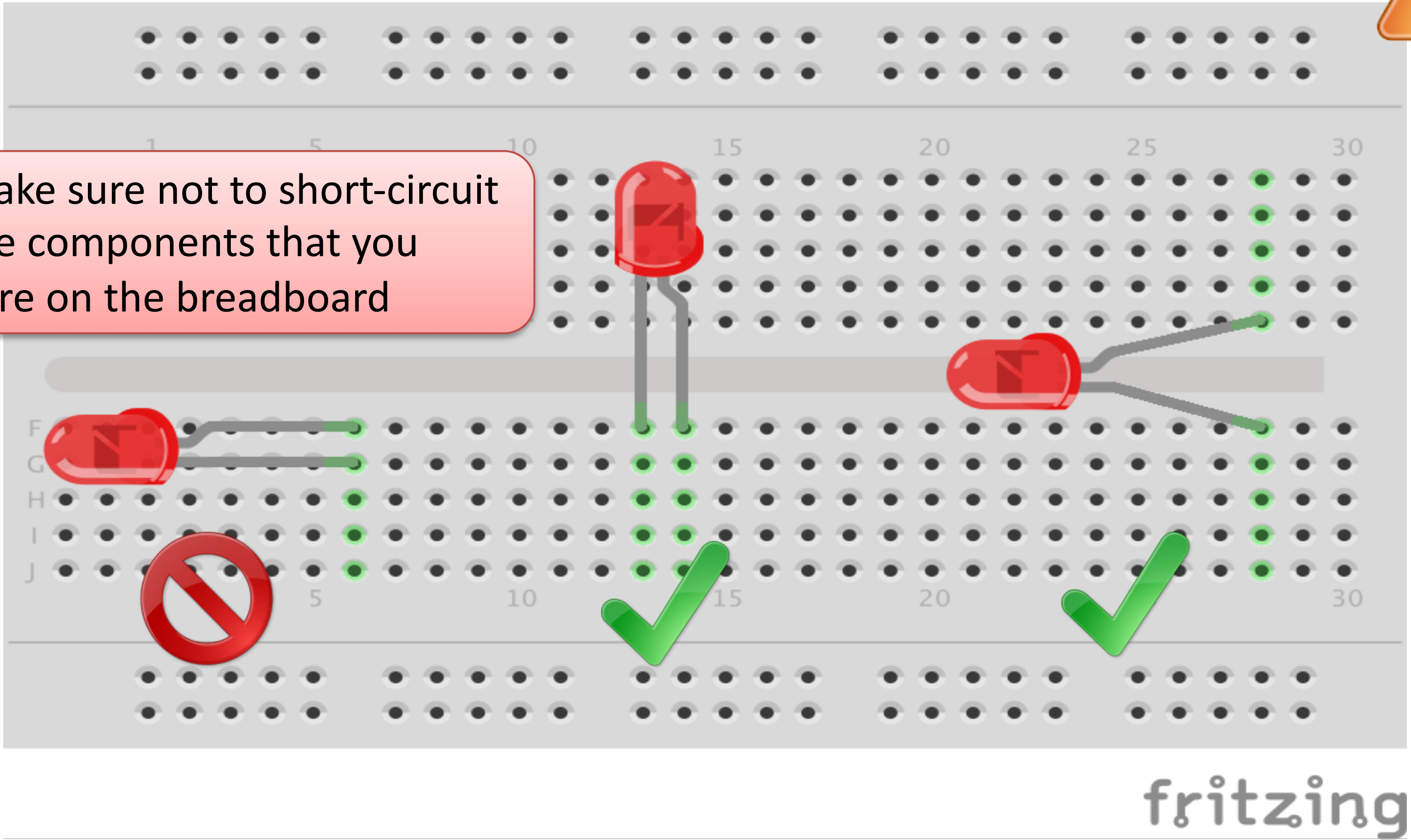






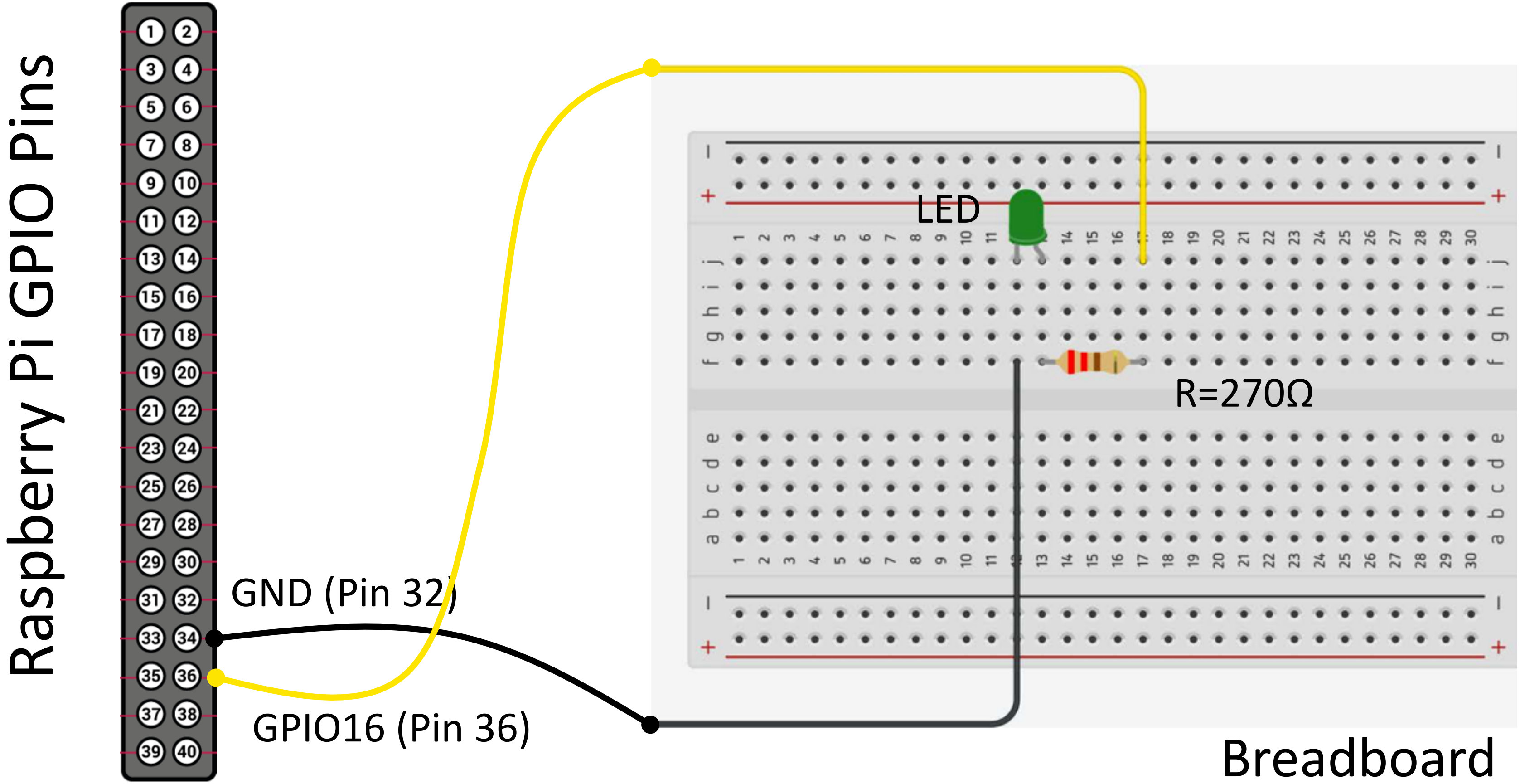
[Wikipedia]

Breadboard Wiring



Make sure not to short-circuit the components that you wire on the breadboard

LED Example



Raspberry Pi GPIO Pins

GND (Pin 32)

GPIO16 (Pin 36)

LED

R=270Ω

Breadboard

Why do you need a Resistor?

If the current becomes too large, the LED will be destroyed. To prevent this to happen, we will use a Resistor to limit the amount of current in the circuit.

What should be the size of the Resistor?

A LED typically need a current like 20mA (can be found in the LED Datasheet).
We use Ohm’s Law:

$$U = RI$$

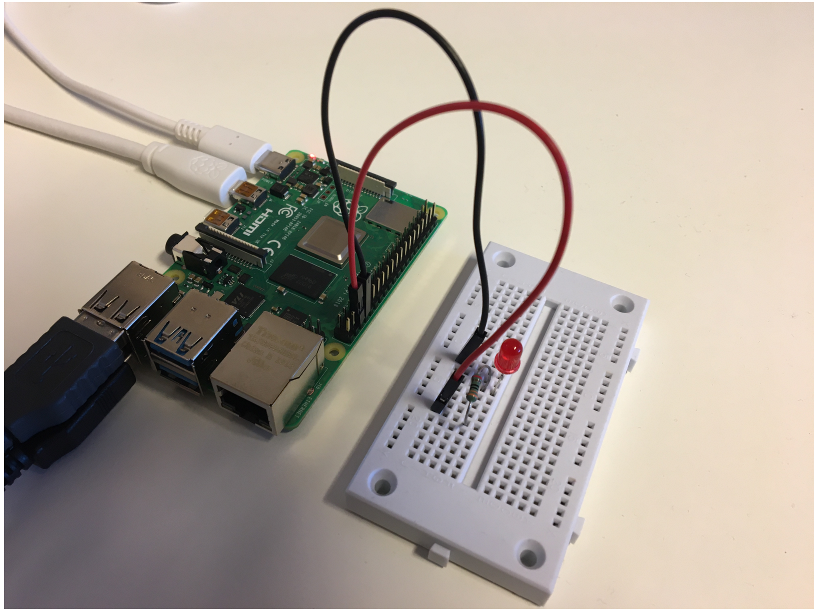
Arduino gives U=5V and I=20mA. We then get:

$$R = \frac{U}{I}$$

The Resistor needed will be $R = \frac{5V}{0.02A} = 250\Omega$. Resistors with R=250Ω is not so common, so we can use the closest Resistors we have, e.g., 270Ω

LED Example

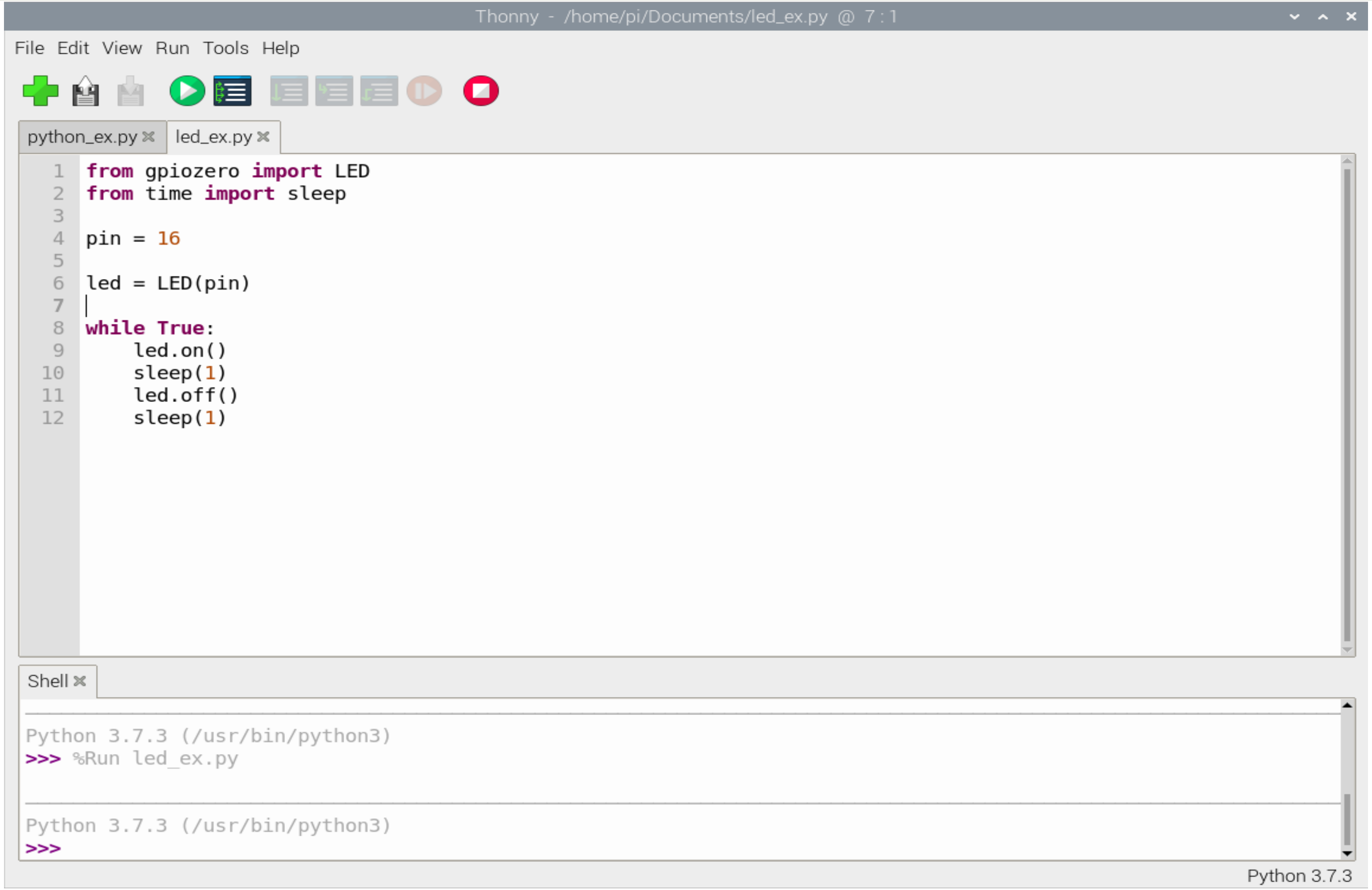
This Example “Runs for ever“



```
from gpiozero import LED
from time import sleep

pin = 16
led = LED(pin)

while True:
    led.on()
    sleep(1)
    led.off()
    sleep(1)
```

https://www.raspberrypi.org/documentation/usage/gpio/python/

LED Example



Thonny - /home/pi/Documents/led_ex.py @ 7 : 1
File Edit View Run Tools Help
python_ex.py
led_ex.py
Shell
Python 3.7.3

```
from gpiozero import LED
from time import sleep

pin = 16

led = LED(pin)

while True:
    led.on()
    sleep(1)
    led.off()
    sleep(1)
```

```
Python 3.7.3 (/usr/bin/python3)
>>> %Run led_ex.py

Python 3.7.3 (/usr/bin/python3)
>>>
```

LED Example

This example turns a LED on/off 10 times

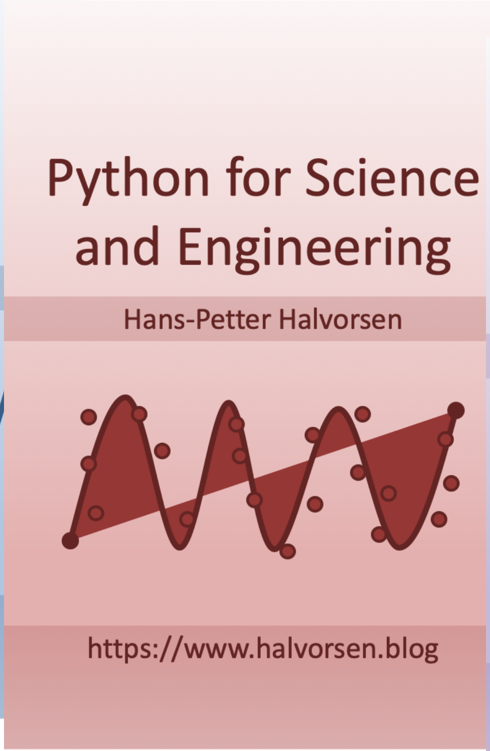
```
from gpiozero import LED
from time import sleep

pin = 16
led = LED(pin)

N = 10
for x in range(N):
    led.on()
    sleep(1)
    led.off()
    sleep(1)
```

Additional Python Resources

Python Programming

Hans-Petter Halvorsen

https://www.halvorsen.blog



Python for Science and Engineering

Hans-Petter Halvorsen

https://www.halvorsen.blog

Python for Control Engineering

Hans-Petter Halvorsen

https://www.halvorsen.blog

Python for Software Development

Hans-Petter Halvorsen

Python Software Development

Do you want to learn Software Development?

https://www.halvorsen.blog

https://www.halvorsen.blog/documents/programming/python/

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: https://www.halvorsen.blog