جامعة حلب
كلية الهندسة الإلكترونية والكهربائية

PROGRAMMING A RASPBERRY PI WITH PYTHON

python™ + RaspberryPi

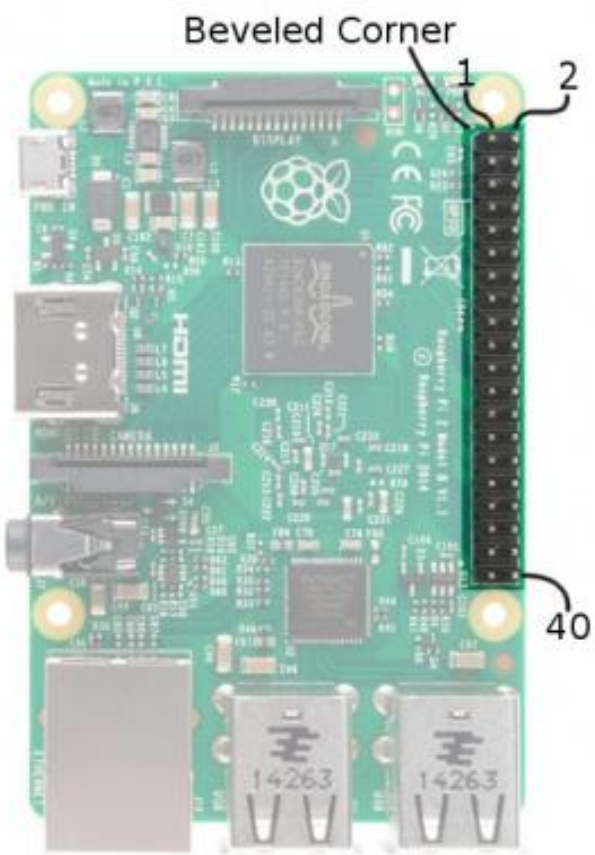
إعداد: م.علا جزماتي

GPIO (General Purpose Input/Output)

GPIO numbering

26 pins

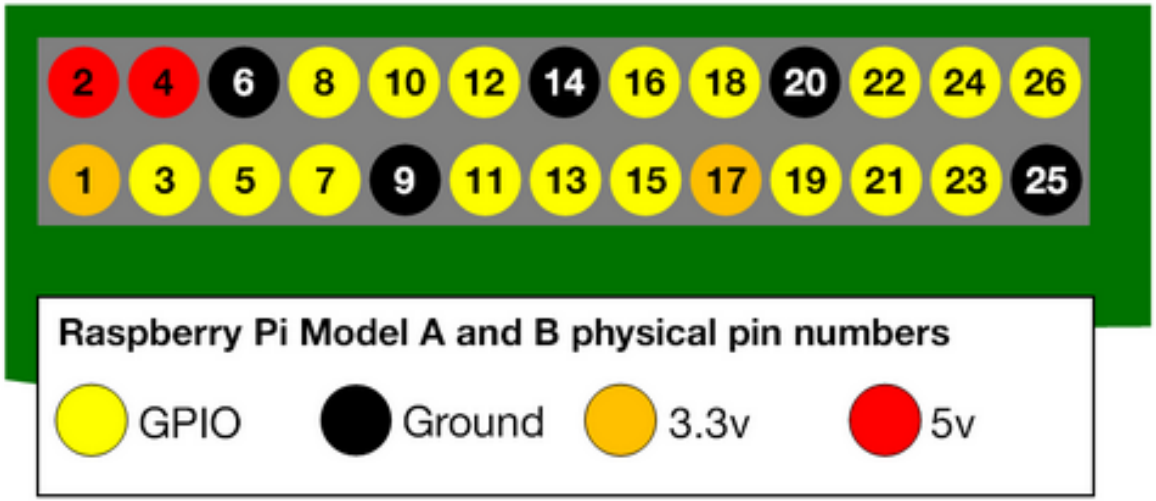
40 pins

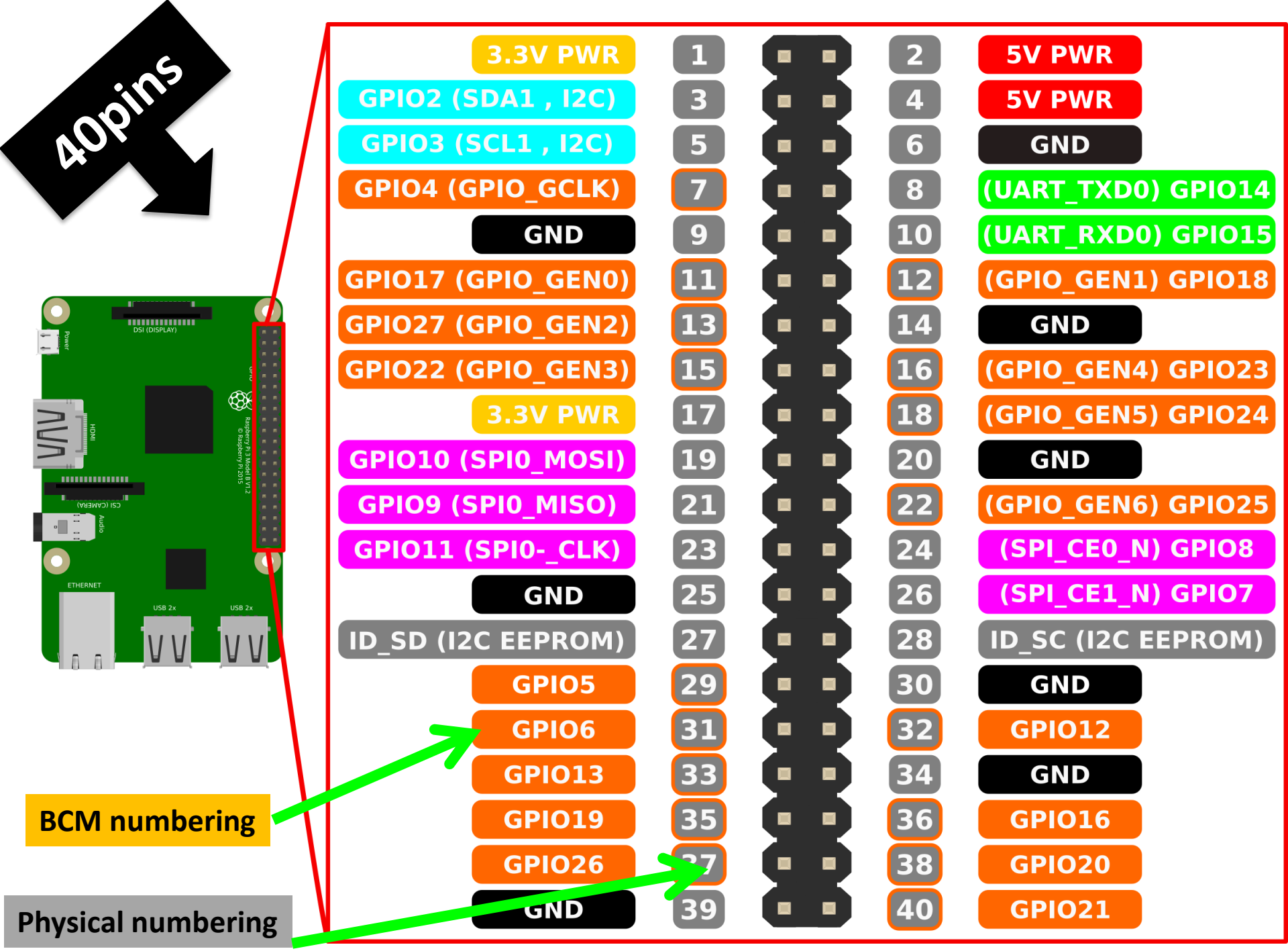


GPIO numbering

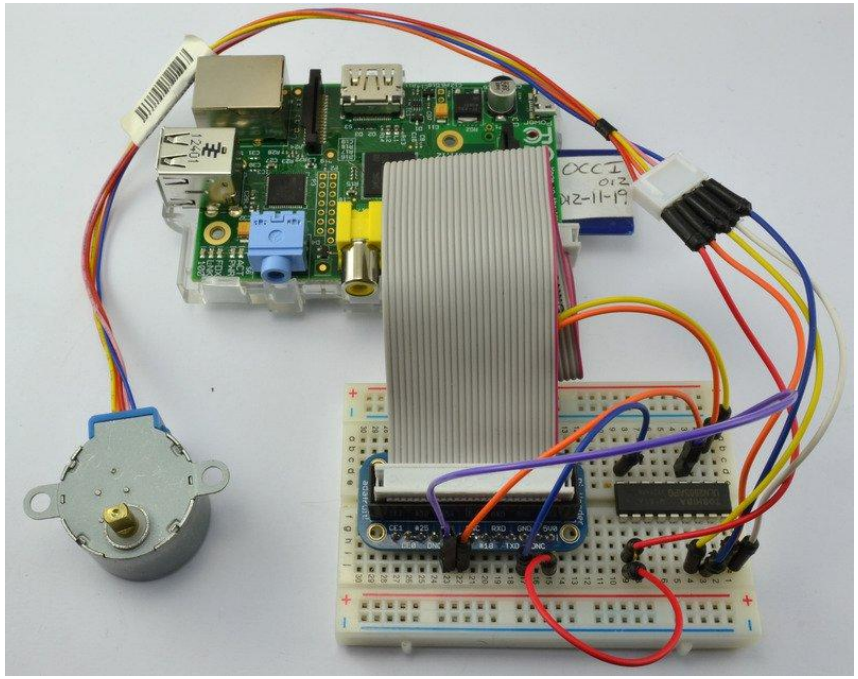
Physical numbering

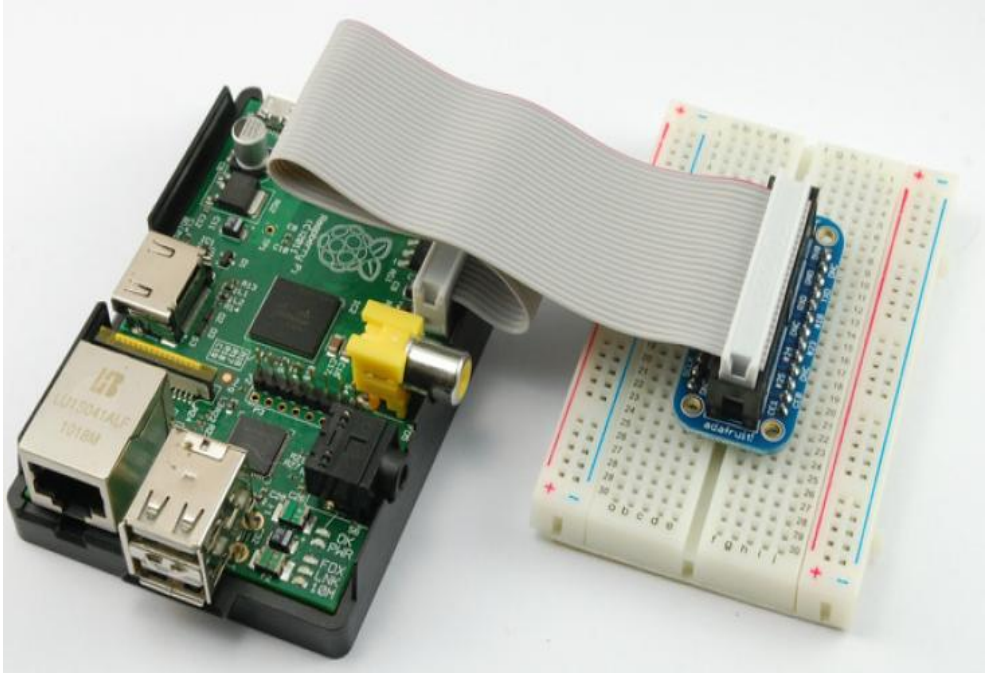
26 pins

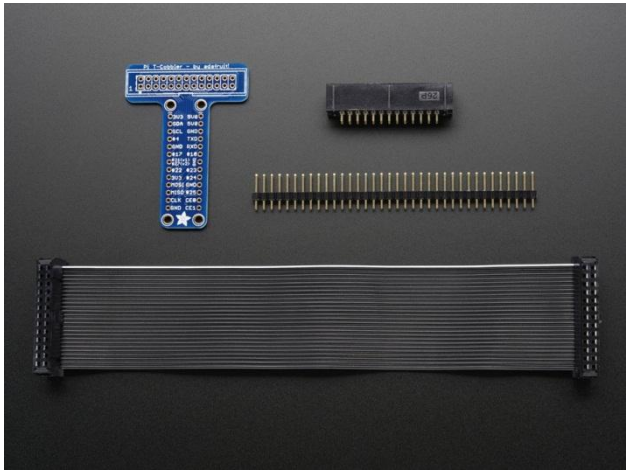


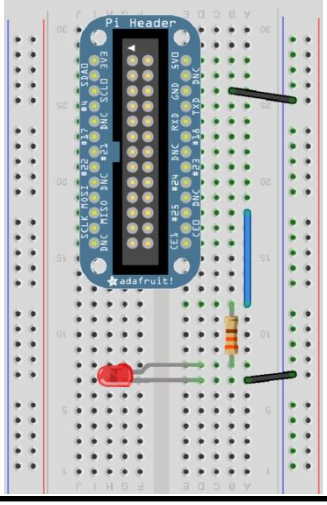

40pins
3.3V PWR 1
2 5V PWR
GPIO2 (SDA1 , I2C) 3
4 5V PWR
GPIO3 (SCL1 , I2C) 5
6 GND
GPIO4 (GPIO_GCLK) 7
8 (UART_TXD0) GPIO14
GND 9
10 (UART_RXD0) GPIO15
GPIO17 (GPIO_GEN0) 11
12 (GPIO_GEN1) GPIO18
GPIO27 (GPIO_GEN2) 13
14 GND
GPIO22 (GPIO_GEN3) 15
16 (GPIO_GEN4) GPIO23
3.3V PWR 17
18 (GPIO_GEN5) GPIO24
GPIO10 (SPI0_MOSI) 19
20 GND
GPIO9 (SPI0_MISO) 21
22 (GPIO_GEN6) GPIO25
GPIO11 (SPI0-_CLK) 23
24 (SPI_CE0_N) GPIO8
GND 25
26 (SPI_CE1_N) GPIO7
ID_SD (I2C EEPROM) 27
28 ID_SC (I2C EEPROM)
GPIO5 29
30 GND
GPIO6 31
32 GPIO12
GPIO13 33
34 GND
GPIO19 35
36 GPIO16
GPIO26 37
38 GPIO20
GND 39
40 GPIO21
BCM numbering
Physical numbering
DSI (DISPLAY)
Power
HDMI
CSI (CAMERA)
Audio
ETHERNET
USB 2x
USB 2x

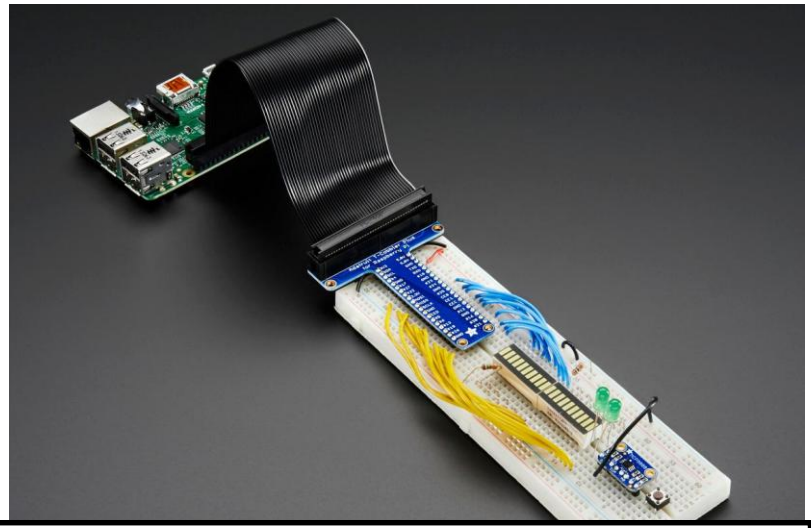
GPIO (General Purpose Input/Output)





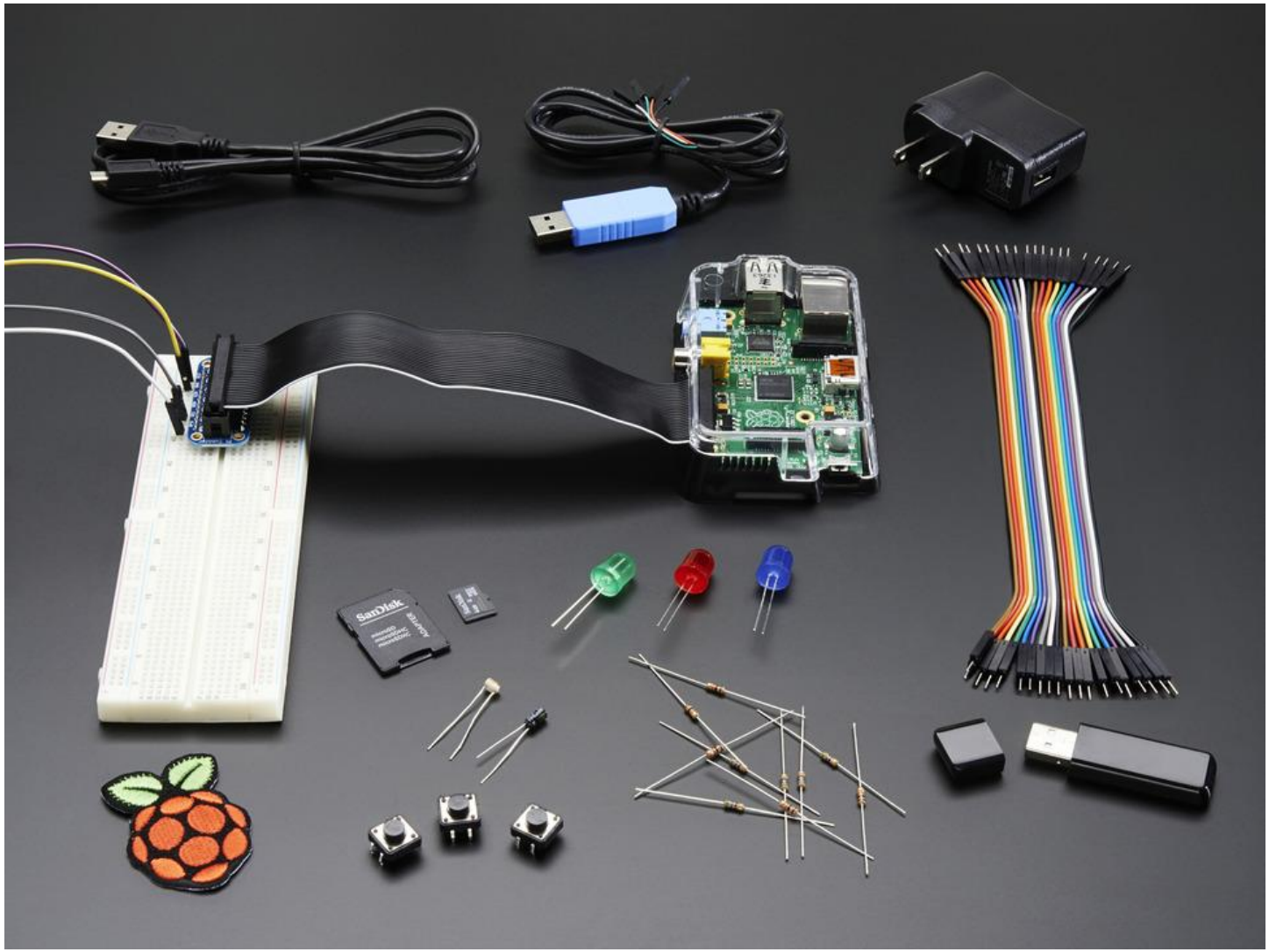




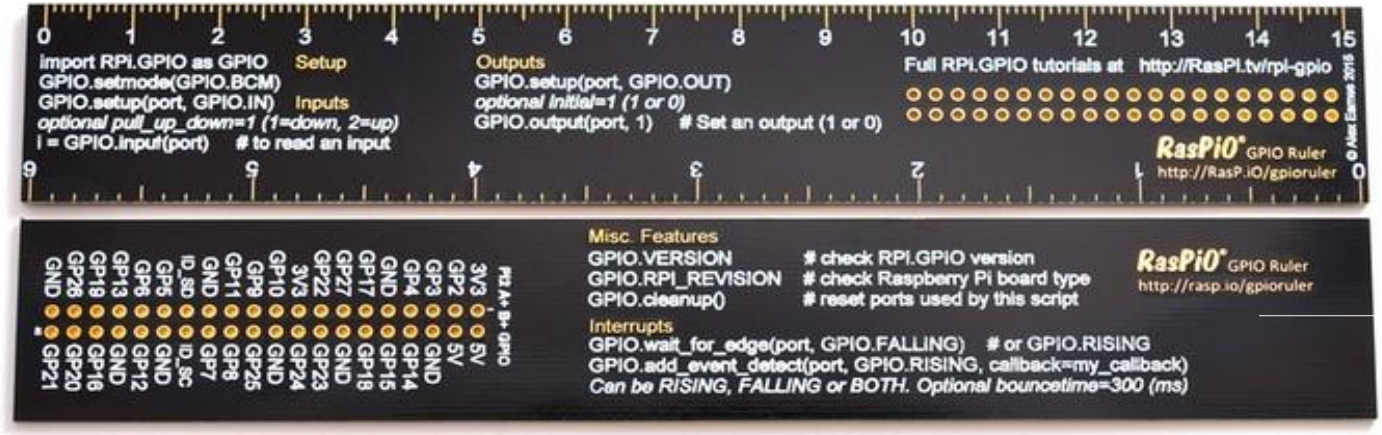


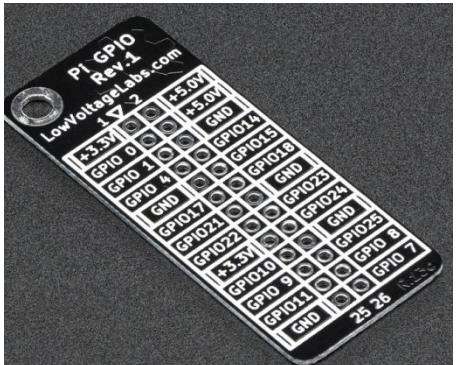
Adafruit Pi Unassembled T-Cobbler Breakout Kit for Raspberry Pi

GPIO (General Purpose Input/Output)

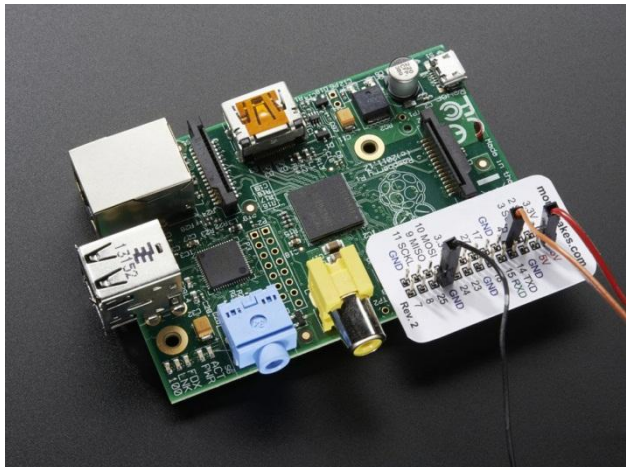


Pi Ruler


import RPI.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
Setup
GPIO.setup(port, GPIO.IN)
Inputs
optional pull_up_down=1 (1=down, 2=up)
i = GPIO.input(port) # to read an input
Outputs
GPIO.setup(port, GPIO.OUT)
optional initial=1 (1 or 0)
GPIO.output(port, 1) # Set an output (1 or 0)
Full RPI.GPIO tutorials at http://RasPi.tv/rpi-gpio
RasPiO GPIO Ruler
http://RasP.iO/gpioruler
© Alex Eames 2015
Misc. Features
GPIO.VERSION # check RPI.GPIO version
GPIO.RPI_REVISION # check Raspberry Pi board type
GPIO.cleanup() # reset ports used by this script
RasPiO GPIO Ruler
http://rasp.io/gpioruler
Interrupts
GPIO.wait_for_edge(port, GPIO.FALLING) # or GPIO.RISING
GPIO.add_event_detect(port, GPIO.RISING, callback=my_callback)
Can be RISING, FALLING or BOTH. Optional bouncetime=300 (ms)


Pi GPIO
Rev.1
LowVoltageLabs.com


monkmakes.com
3.3V 5V
2 SDA 5V
3 SCL GND
4 14 TXD
GND 15 RXD
17 18
27 GND
22 23
3.3V 24
10 MOSI GND
9 MISO 25
11 SCKL 8
GND 7
Rev. 2



لغة PYTHON

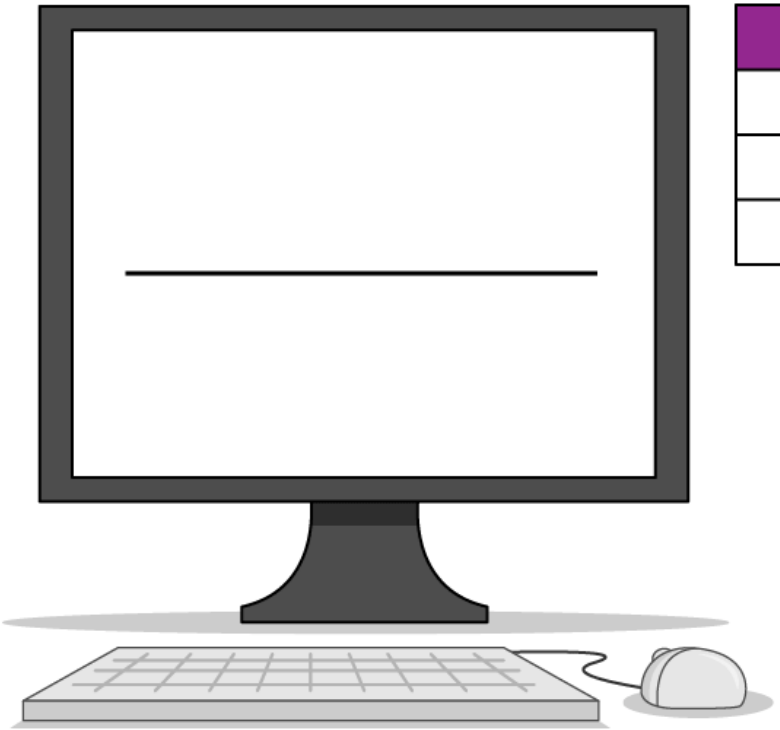
لغة بايثون Python:

- نشأت لغة بايثون في مركز العلوم والحاسب الآلي في امستردام عام 1991 على يد Guido van Rossumحيث تمت كتابة نواة اللغة بلغة C، وسميت بهذا الاسم الذي يعني (ثعبان ضخم يعيش في أنهار الأمازون وهو اسم فرقة مسرحية بريطانية كانت تعجب Guido).

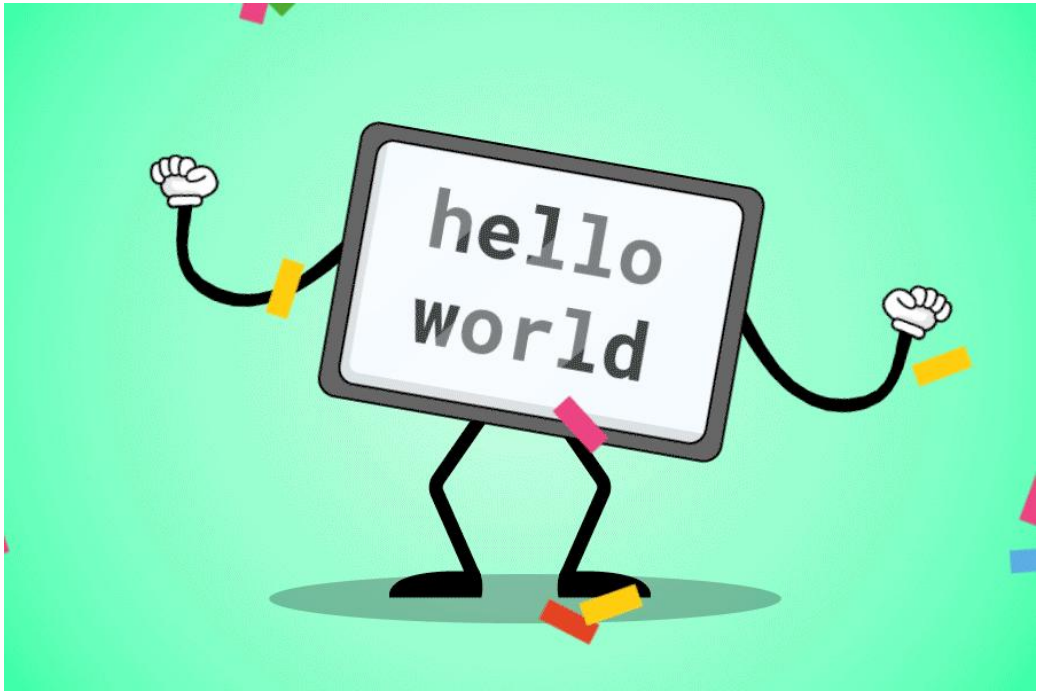
- تعمل على جميع أنظمة التشغيل وإصداراتها المختلفة وأنظمة الهواتف المحمولة.
- بسيطة فقراءة برنامج يكاد يشبه قراءة نص باللغة الإنكليزية، هذه الطبيعة الشبه رمزية (pseudo-code) لبايثون أحد أعظم أسرار قوتها. فتتيح لك التركيز على حل المشكلة لا اللغة نفسها.
- سهلة التعلم ومفتوحة المصدر.
- لغة برمجة عالية المستوى، فعندما تكتب البرامج لا تحتاج للاهتمام بالتفاصيل دقيقة المستوى مثل إدارة الذاكرة التي يستخدمها برنامجك، ....

- محمولة نظرا لطبيعتها كبرمجية مفتوحة المصدر، تم نقل بايثون إلى العديد من المنصات. كل ما تكتبه من برامج بايثون يمكن أن يعمل على أي من هذه المنصات دون أن يتطلب ذلك أي تغييرات على الإطلاق إذا كنت دقيقا بما فيه الكفاية لتجنب أي خصائص تعتمد على نظام محدد.
- مفسَّرة : البرنامج المكتوب بلغة مصرّفة (compiled) مثل c++ يتم تحويله من اللغة المصدر إلى اللغة التي يتكلمها حاسوبك (كود ثنائي بلغة الآلة) باستخدام المصرّف مع مختلف الخيارات والتعليمات. عند تشغيلك البرنامج، يقوم الرابط/المحمّل (linker/loader) بنسخ البرنامج من القرص الصلب إلى الذاكرة ويبدأ في تشغيله.
- بايثون -من ناحية أخرى- لا تحتاج التصريف إلى كود ثنائي. فقط شغل البرنامج مباشرة من الكود المصدر داخليا، فإن بايثون يحول كود المصدر إلى شكل وسيط يسمى bytecode ثم يترجم هذا إلى اللغة الأصلية لجهازك، ثم يشغله. كل هذا يجعل من الأسهل بكثير استخدام بايثون حيث لست بحاجة للاهتمام بتصريف البرنامج، أو التأكد من صحة مكتبات الربط وتحميلها، الخ، الخ. وهذا أيضا يجعل برامج بايثون الخاصة بك أكثر محمولية، بحيث يمكنك مجرد نسخ برنامج بايثون الخاص بك إلى حاسوب آخر، وبعدها يعمل!

- كائنية التوجه : تدعم بايثون البرمجة الإجرائية (procedure-oriented) وكذلك البرمجة الكائنية (object-oriented). في اللغات *إجرائية التوجه*، يتمحور البرنامج حول الإجراءات أو الدوال التي ليست سوى قطع من البرامج يمكن إعادة استخدامها. وفي اللغات *كائنية التوجه*، يتمحور البرنامج حول الكائنات (objects) التي تجمع بين البيانات والوظائف. ولبايثون طريقة قوية جدا ولكن تبسيطية لعمل البرمجة الكائنية خاصة عند مقارنتها باللغات الكبيرة مثل c++ أو جافا.
- قابلة للامتداد : إذا كنت في حاجة لجعل جزء حيوي من الكود يعمل سريعا جدا أو تريد إخفاء بعض الخوارزميات، فيمكنك كتابة هذا الجزء من برنامجك بلغة c++ وبعدها تستخدمه من برنامج بايثون الخاص بك.
- قابلة للتضمين : يمكنك تضمين بايثون في برامج c++ لإعطاء قدرات ال 'scripting'لمستخدمي برنامجك.
- مكتبات شاملة : مكتبة بايثون القياسية مكتبة ضخمة حقا. تساعدك على عمل مختلف الأشياء العادية



Name	Value



التعرف على الأساسيات

Basics of Python

Python has two operating modes.

- Interactive mode
- Standard mode

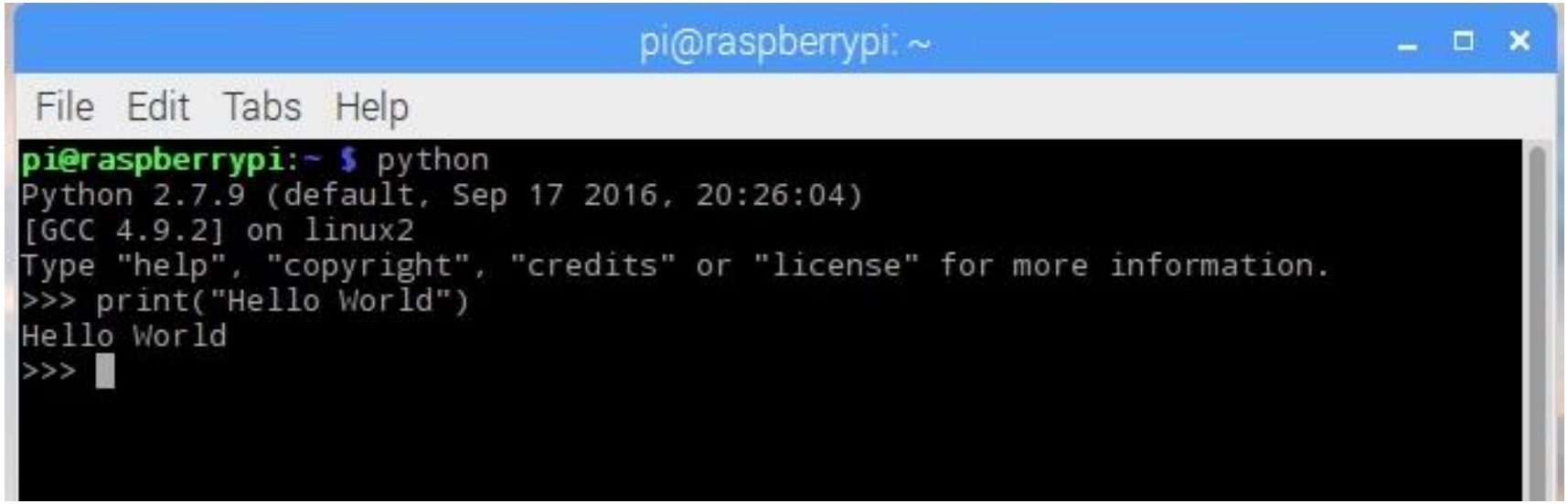
للتدريب على لغة python واستثمارها في أي مجال

يفضل تنصيب Anaconda

على أي حاسب الشخصي حيث تعد من أفضل البيئات البرمجية

https://www.anaconda.com/distribution/

1- Program Python with Terminal



Open a terminal in Raspbian and enter python.

It will display 'Python 2.7.9'.

Enter python3 and you'll see 'Python 3.4.2'.

We use Python 3 in our programming guides. You can open Python 3 in the terminal by just typing python3.

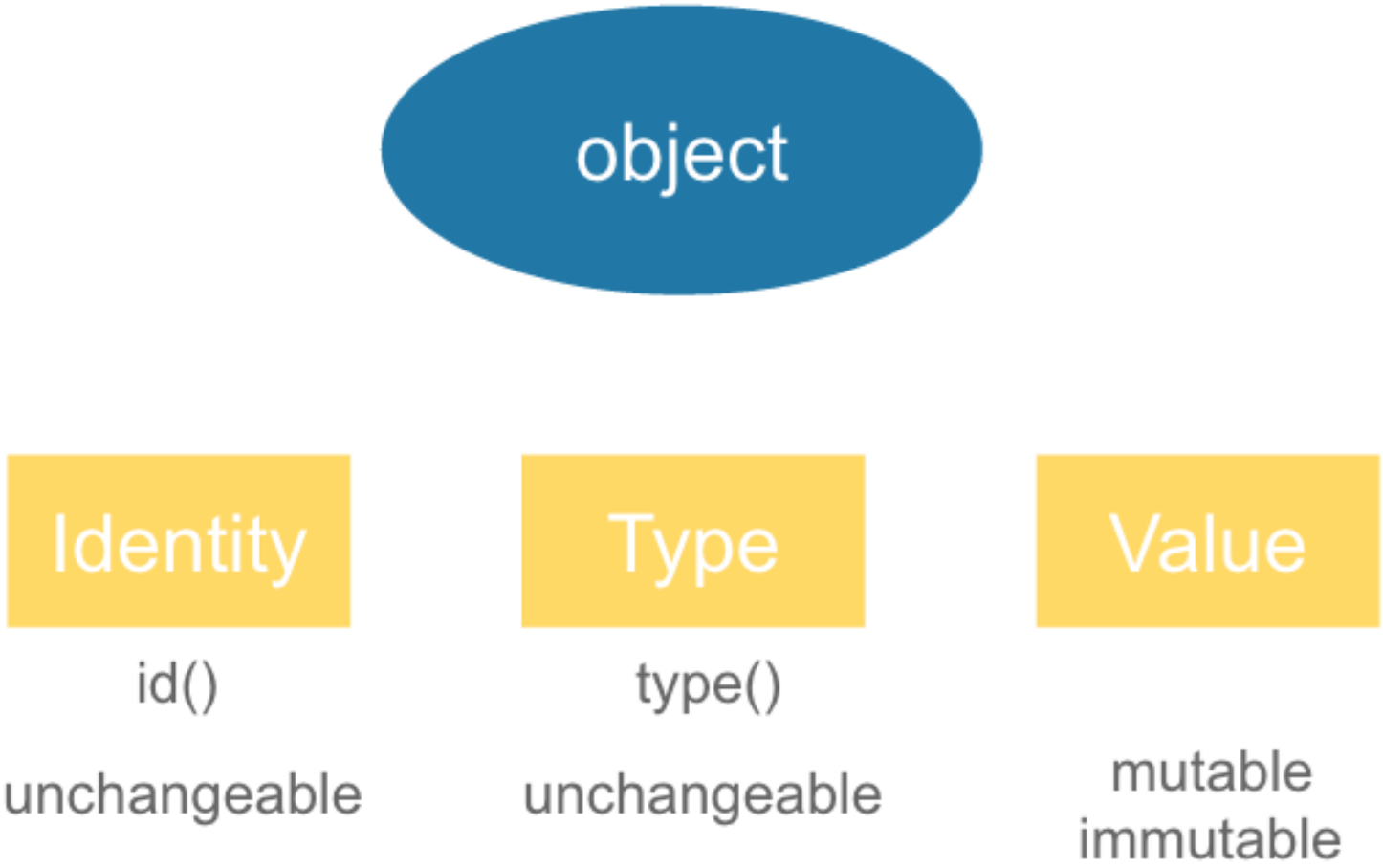
The '$' command-line prompt will be replaced with '>>>'. Here you can enter Python commands directly, just as you would terminal commands.

1- Program Python with Terminal

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

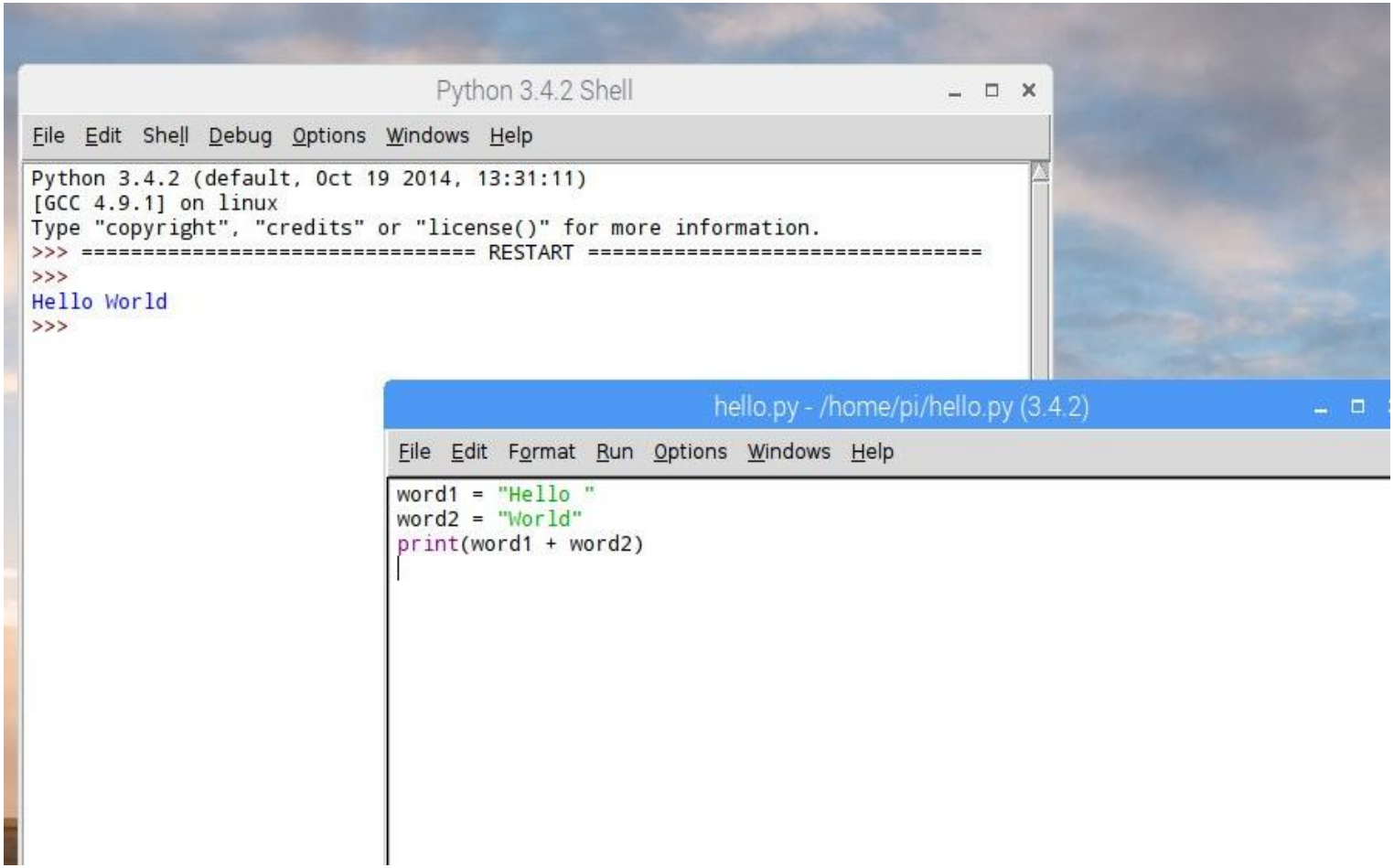
```
word1 = "Hello "
word2 = "World"
print(word1 + word2)
```

```
exit()
```


object
Identity
id()
unchangeable
Type
type()
unchangeable
Value
mutable
immutable

2- Program Python with the IDLE IDE

IDLE (**I**ntegrated **D**eve**L**opment **E**nvironment or **I**ntegrated **D**evelopment and **L**earning **E**nvironment)



For example, if you add together two strings they are combined:

```
name = "Harry"
job = "Wizard"
print("Yer a " + job + " " + name)
```

This prints the message “Yer a Wizard Harry”.

Let’s try a bit of maths:

```
number1 = 6
number2 = 9

print(number1 + number2)
```

Cast string into integers in Python

You can also cast strings into integers using the int() function.

This is particularly useful when you use input() to get a number from the user; the input is stored as a string. Let's create a program that asks for a number and exponent and raises the number to the power of the exponent (using the '**' symbol):

```
number = input("Enter a number: ")
exponent = input("Enter an exponent: ")
result = int(number) ** int(exponent)
```

We could just print out the result:

```
print(result)
```

But if we wanted to include a message, we need to type cast result to a string:

```
print(number + " raised to the power
" + exponent + " is " + str(result))
```

Branching with If in Python: learn the logic

```python
if True:
print("Hello World")
```

```python
if False:
print("Hello World")
```

```python
if True:
    print("The first branch ran")
else:
    print("The second branch ran")
```

```python
if False:
    print("The first branch ran")
else:
    print("The second branch ran")
```

Branching: use If, Else, and Elif for smarter coding

```
if False:
    print("The first block of code ran")
elif True:
    print("The second block of code ran")
else:
    print("The third block of code ran")
```

مثال:

Password.py

```
password = "qwerty"
attempt = input("Enter password: ")

if attempt == password:
    print("Welcome")
else:
    print("Incorrect password!")
```

Comparison operators in Python

These comparison operators are commonly used in conditions to determine if something is True or False:

== equal

!= not equal

< less than

<= less than or equal to

> greater than

>= greater than or equal to

<> less than or greater than

Modulo turns out to be handy in lots of ways. You can use % 2 to figure out if a number is odd or even:

```
10 % 2 == 0 # this is even
11 % 2 == 1 # this is odd
```

This program works out if a number is odd or even:

```
number = 10

if number % 2 == 0:
    print("The number is even")
else:
    print("The number is odd")
```

Learn to <u>comment</u> your code in Python

A mark of a good programmer is to use comments in your programs.

Comments are used to explain bits of your program to humans.

In Python, you start a comment line with <u>a hash symbol</u> (#).

```
# This is a comment. The whole line is ignored by the prog
# The print statement will run, as it has no comment

print("Hello World")

print("Goodbye World") # This is also a comment. But print
```

Comments help other users to read your program, but they will also help you understand what you're doing (long after you've forgotten).

It's a good habit to use comments in your programs.

IMPORT MODULES: LEARN TO CODE IN PYTHON

Using import in Python programs

```
import math
type(math)
```

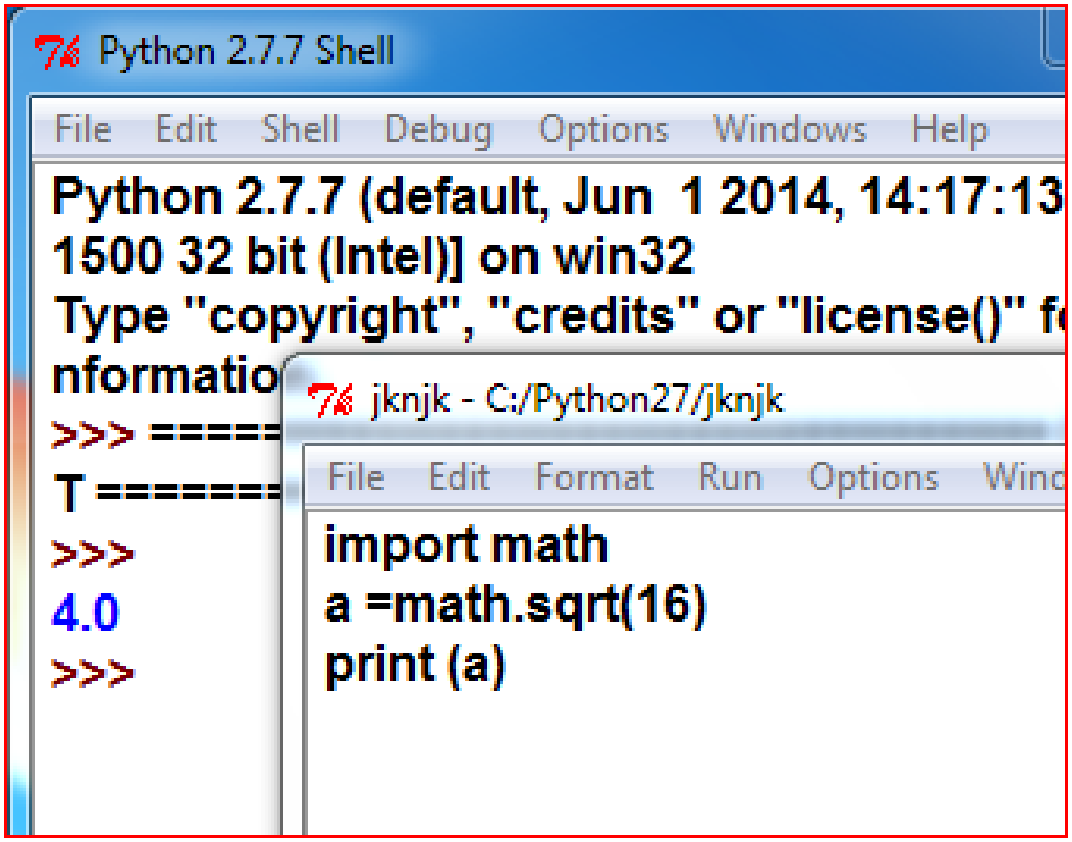
```
>>>
>>> type(math)

<type 'module'>
>>>
```

File Edit Format Run

```
import math
type(math)
```

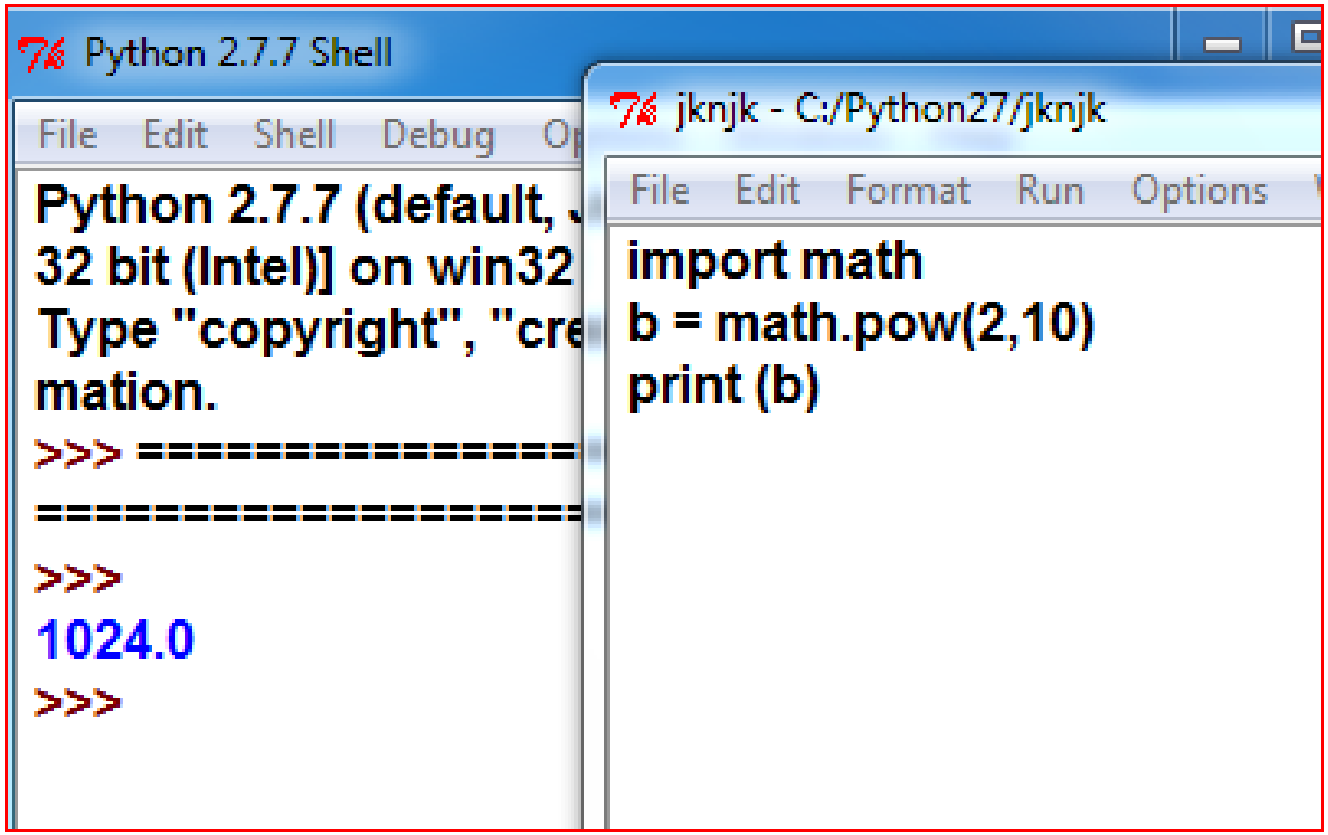
IMPORT MODULES: LEARN TO CODE IN PYTHON

Using import in Python programs



IMPORT MODULES: LEARN TO CODE IN PYTHON

Using import in Python programs



Import constant values in modules

You can also access constant values from a module, which are fixed variables contained in the module. These are like functions/methods, but without the parentheses.

```
math.pi
```

It's also possible to import methods and constants from modules using from. This enables you to use them inside your programs without dot notation (like regular functions). For example:

```
from math import pi
from math import e
from math import pow
```

Import constant values in modules

Now, whenever you type pi or e, you'll get pi and Euler's number. You can also use pow() just like a regular function. You can change the name of the function as you import it with as:

```
from math import pi as p
```

LOOPS: USING WHILE AND FOR IN PYTHON

While, condition and indent

```
while condition:
    indent
```

File Edit Format Run Options Windows Help

```
name = 'sara'
counter = 0
while counter < 3:
    print(name + " put the kettle on")
    counter = counter + 1

print("We will all have tea")
```

LOOPS: USING WHILE AND FOR IN PYTHON

jknjk - C:/Python27/jknjk

File Edit Format Run Options Windows Help

```
name = 'sara'
counter = 0
while counter < 3:
    print(name + " put the kettle on")
    counter = counter + 1

print("We will all have tea")
```

Python 2.7.7 Shell

File Edit Shell Debug Options Windows Help

```
Python 2.7.7 (default, Jun  1 2014, 14:1
32
Type "copyright", "credits" or "license(
>>> ================================ RES
>>>
Sara put the kettle on
Sara put the kettle on
Sara put the kettle on
We will all have tea
```

LOOPS: USING WHILE AND FOR IN PYTHON

```
for x in range (0, 10):
    print "hello"
```

```
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
```

LOOPS: USING WHILE AND FOR IN PYTHON

```
for x in "Camelot":
    print "Ni!"
```

```
Ni!
Ni!
Ni!
Ni!
Ni!
Ni!
Ni!
```

```
for x in "Camelot":
    print x
```

```
C
a
m
e
l
o
t
```

Type casting variables in Python

```
name = "Ben"
number = 10
print(name + number)
```

You'll get an error message: 'Type Error: Can't convert 'int' object to str implicitly'. This error is because Python can't add together a string and an integer, because they work differently. Ah, but not so fast! You can multiply strings and integers:

```
print(name * number)
```

- It'll print 'Ben' ten times: you'll get 'BenBenBenBenBenBenBenBenBenBen'.
- If you want to print out 'Ben10', you'll need to convert the <u>integer to a string</u>.
- You do this using a str() function and putting the integer inside the brackets. Here we do that, and store the result in a new variable called number_as_string:

```
number_as_string = str(number)
print(name + number_as_string)
```

This code will print out the name ‘Ben10’. This concept is known as ‘<u>type casting</u>’:
converting a variable from one type to another.

FUNCTIONS: LEARN CODE WITH PYTHON AND RASPBERRY PI

Learn to use Functions when programming Python on a Raspberry Pi

Using functions in Python programming

```
abs(2) # returns 2
abs(-2) # returns 2
```

```
positive_number = abs(-10)
```

Create a working function in Python

```
def function(parameter):
    return parameter
```

```
def absolute(number):
    if number < 0:
        return number * -1
    else:
        return number
```

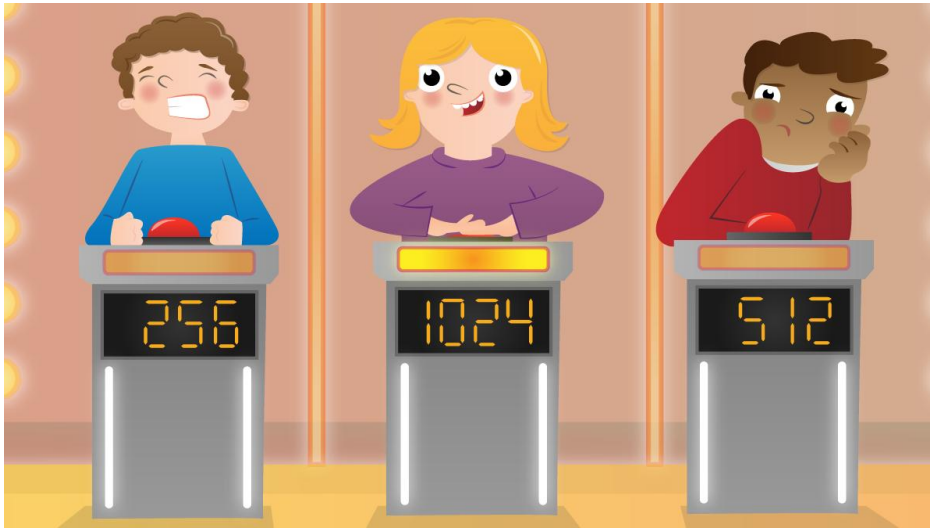
```
absolute(10)
absolute(-10)
```

Create a working function in Python

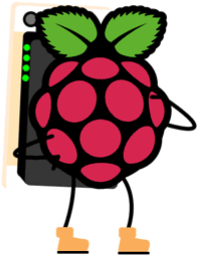
```
def AreaPerimeter (height, width):
    height = int(height)
    width = int(width)
    area = height * width
    perimeter = (2 * height) + (2 * width)
    print "The area is:" + area
    print (The perimeter is:" + perimeter
    return
```

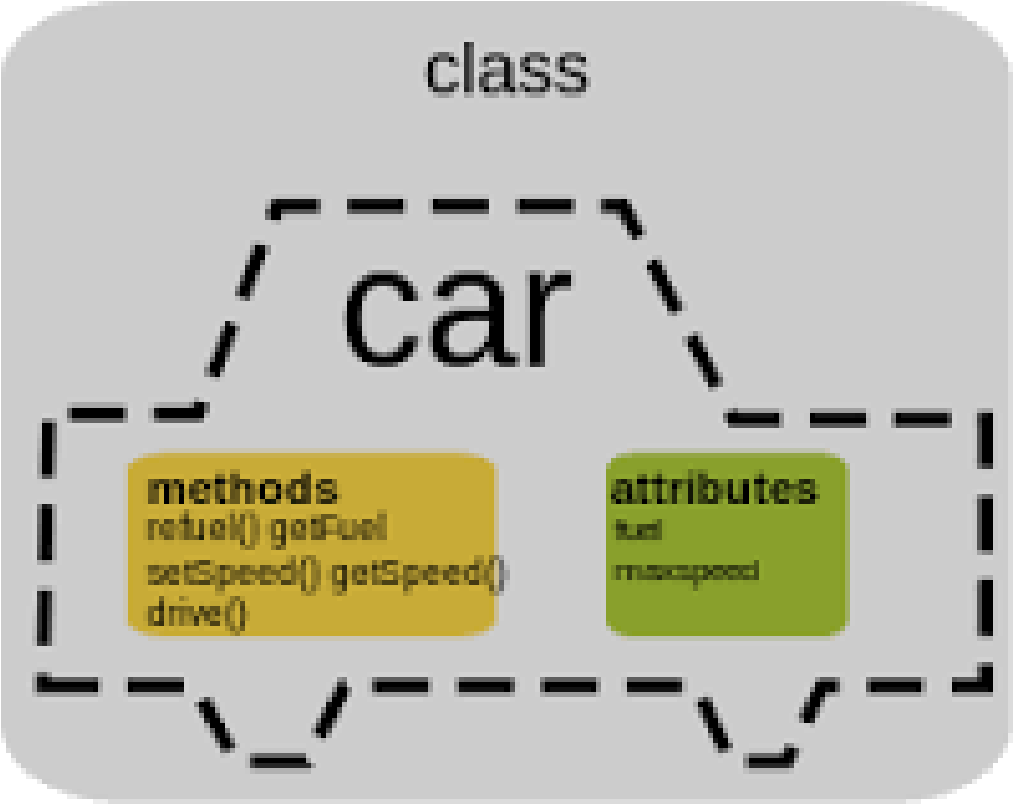
Simple reaction game using inputs, outputs and variables

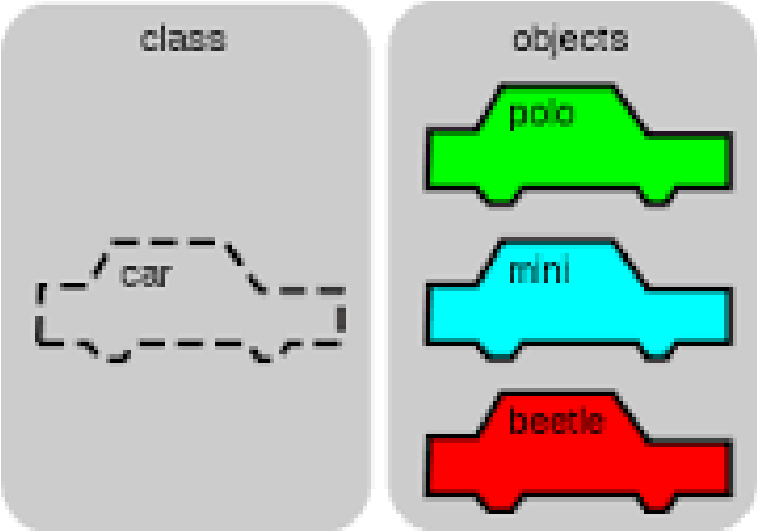
```
from time import sleep, time
sleep(3)
start = time()
print('Quick, hit the Enter key')
input()
stop = time()
reaction_time = stop - start
print('You took', reaction_time, 'seconds')
```



How To Construct Classes and Define Objects in Python 3




class
car
methods
refuel() getFuel
setSpeed() getSpeed()
drive()
attributes
fuel
maxspeed


class
car
objects
polo
mini
beetle

Example1:

fish-robot.py | Rocket.py | Shark.py

```python
class fish_robot():
    def __init__(self, name):
        self.name = name

    def swim(self):
        print(self.name + " is swimming")


myrobot = fish_robot("robot1")
myrobot.swim()
```

```
In [7]: runfile('C:/SPB_Data/
fish-robot.py', wdir='C:/
SPB_Data')
robot1 is swimming
```

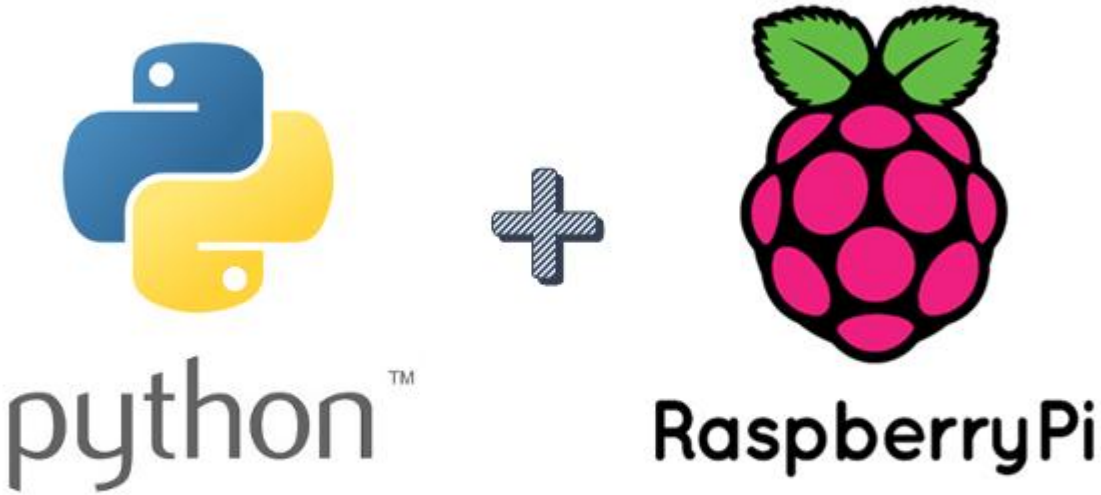
```python
class fish_robot():
    def __init__(self, name):
        self.name = name

    def swim(self):
        print(self.name + " is swimming")
    def localization(self, x=0, y=0):
        # Each rocket has an (x,y) position.
        self.x = x
        self.y = y
        print(self.name + "at" + str(x) + "," + str(y))
    def move_up(self):
        # Increment the y-position of the rocket.
        self.y += 1
        print("move to y :" + str(self.y))
myrobot = fish_robot("robot1")
myrobot.swim()
myrobot.localization(x=0, y=0)
myrobot.move_up()
```

```
In [22]: runfile('C:/SPB_Data/fish-robot.py'
robot1 is swimming
robot1at0,0
move to y :1
```

Example2:

ثانيا: برمجة أقطاب

GPIO (General Purpose Input/Output)

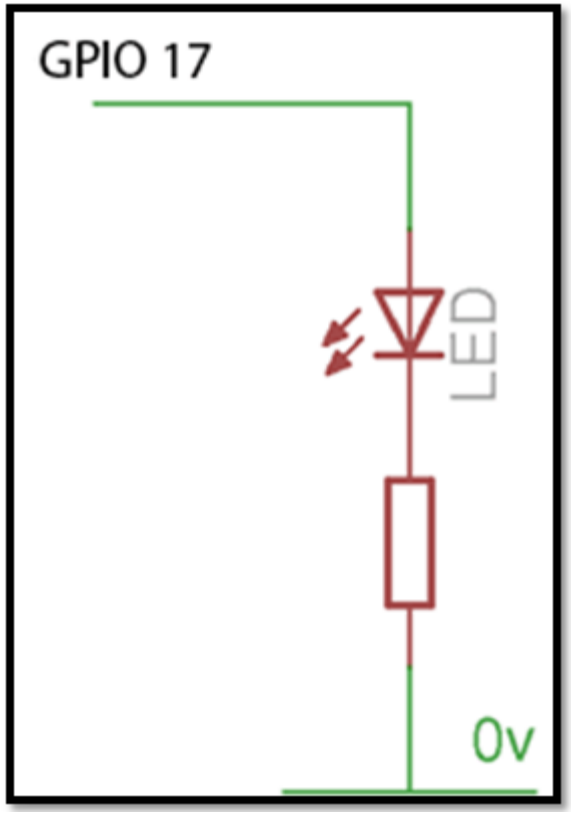


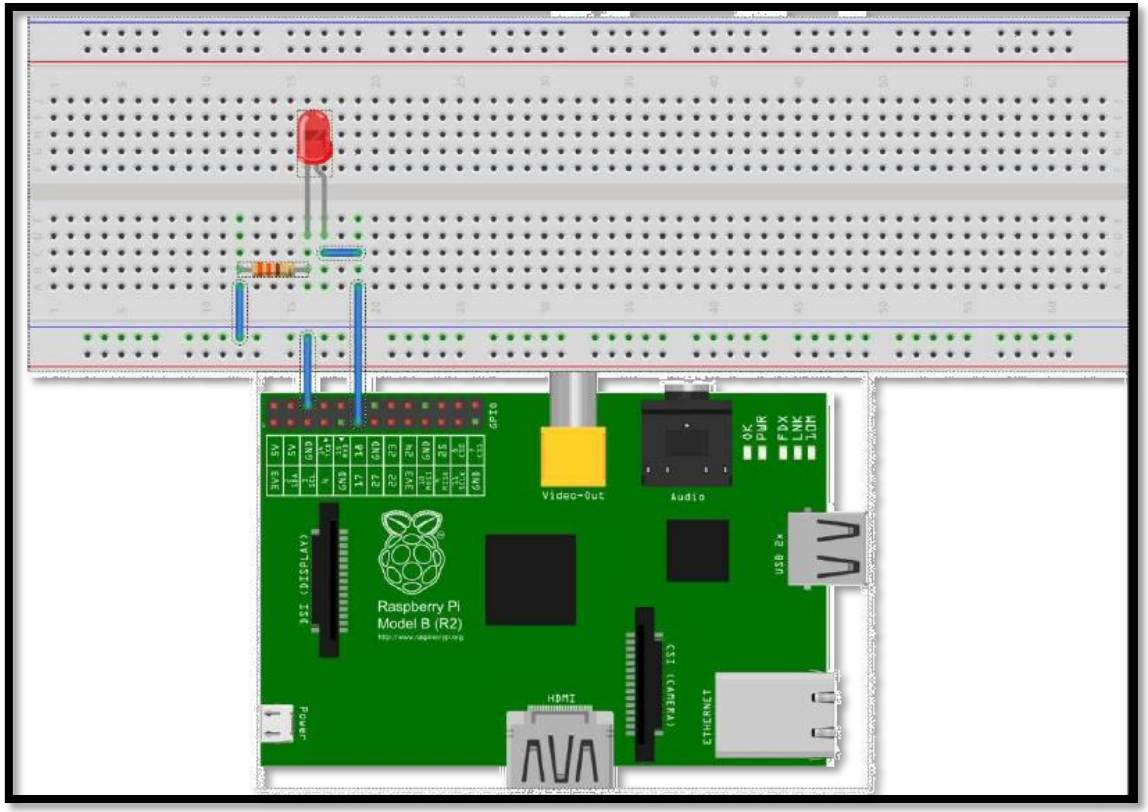
Import modules

module	• Methods
time	• Sleep() • Time()
RPi.GPIO	• Setmode() • Setup() • Setwarnings() • Output() • Input()

1

Connect an LED using a resistor between GPIO (P1-11) and GND





Connect an LED using a resistor between GPIO (P1-11) and GND

```
import time
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BOARD)
GPIO.setup(11, GPIO.OUT)
while True:
        GPIO.output(11,0)
        time.sleep(1)
        GPIO.output(11,1)
        time.sleep(1)
```

Thank You