

MEMO PYTHON 3

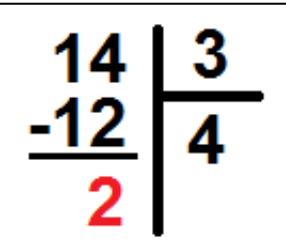
	Les types	Commentaire
Booléen	<pre>>>> mon_boolen=True >>> mon_boolen True >>> type(mon_boolen) <class 'bool'></pre>	Un booléen ne peut prendre que 2 valeurs True ou False.
Entier	<pre>>>> valeur=1235 >>> type(valeur) <class 'int'></pre>	Un entier est appelé aussi intéger
Float	<pre>nombre=12.58126 type(nombre) <class 'float'></pre>	<pre>>>> import sys >>> sys.float_info sys.float_info(max=1.7976931348623157e+308 etc....</pre>
String	<pre>>>> mot1="bonjour" >>> mot2='bye' >>> print(mot1,"/",mot2) bonjour / bye >>> type(mot1) <class 'str'></pre>	Une chaîne de caractère peut être délimitée par des apostrophes (simple quotes) ou des guillemets (double quotes)
Complexe	<pre>>>> z = 2+3j >>> z.real 2.0 >>> z.imag 3.0 >>> z.conjugate() (2-3j) >>> type(z) <class 'complex'></pre>	<pre>>>> from cmath import polar, rect >>> cmath.polar(z) (3.6055512754639896, 0.982793723247329) >>> cmath.rect(1.41,0.78) (1.0023880885973109+0.9916239810725782j)</pre>
Formatage	<pre>i=12 s = "%05d" % i print(s)</pre>	00012

	Le transtypage (cast)	Commentaire
Float -> integer	>>> a=3.45 >>> int(a) 3	La valeur a est affichée en valeur entière.
Integer-> float	>>> b=5 >>> c= float(b) print(c) 5.0	# ceci est un commentaire su une ligne """ ceci est un bloc de commentaire """
Int -> str	d=15 type(d) <class 'int'> e=str(d) print(e) 15 type(e) <class 'str'>	from random import * n = randint(1,6) print(n) 3 x = uniform(12, 18) print(x) 14.271572580135519 https://docs.python.org/3/library/random.html
Str-> int ou float	int("bonjour") Traceback (most recent call last): File "<input>", line 1, in <module> ValueError: invalid literal for int() with base 10: 'bonjour' int("10") 10	float("3.5") 3.5 float("hello") Traceback (most recent call last): File "<input>", line 1, in <module> ValueError: could not convert string to float: 'hello'

Liste des mots réservés :

And, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, import, in, is, lambda, not, or, pass, print, raise, return, try, while, yield, True, False

	Calculs	Calculs annexes
Addition	>>> a=1 >>> b=2 >>> c=a+b >>> print(c) 3	>>> c=a-b >>> print(c) -1
Multiplication	>>> a=5 >>> b=2 >>> c=a*b >>> print(c) 10	Où se trouve le nombre PI >>> from math import pi >>> print(pi) 3.141592653589793
Puissance	>>> a=2 >>> b=4 >>> c=a**b >>> print(c) 16	Fonctions trigonométriques >>> from math import cos,sin,pi >>> cos(pi/4) 0.7071067811865476
Racine carrée	>>> from math import sqrt >>> sqrt(36) 6.0	Charger la bibliothèque math complètement : >>> from math import * https://docs.python.org/fr/3.5/library/math.html
Division	>>> a=5 >>> b=3 >>> print(a/b) 1.6666666666666667	>>> a=7 >>> b=3 >>> print(a//b) 2 Deux // : division entière

	Calculs	Commentaires
Modulo	<pre>>>> a=14 >>> b=3 >>> print(a%b) 2</pre>	 <p>Le modulo est le reste de la division entière</p>
Décalage	<pre>>>> a=5 >>> a=a<<2 >>> print(a) 20</pre>	<pre>>>> a=a>>1 >>> print(a) 10</pre>
Incrémentation	<pre>>>> a=5 >>> a=a+1 >>> print(a) 6</pre>	<pre>>>> a=10 >>> a+=5 >>> print(a) 15</pre>
Test	<pre>>>> 1==2 False >>> 1==1 True >>> 3.5==3.5 True >>> 1>5 False</pre>	<pre>>>> (1==1) and (3>2) True >>> (1==1) and (3>5) False >>> (1==2) or (10>5) True not(1==2) True</pre>
Test suite	<pre>>>> 3!=4 True >>> 3==4 False</pre>	<p>Attention :</p> <p>Ne pas confondre un test deux égal == et une affectation un égal =</p>

	Instructions	Commentaires
Boucle for (range)	<pre>for a in range (0,3): print (a)</pre>	0 1 2 Process finished with exit code 0
Boucle for (step)	<pre>for a in range (0,10,2): print (a)</pre>	0 2 4 6 8 Process finished with exit code 0
Boucle for (in)	<pre>mot="bonjour" for lettre in mot: print (lettre)</pre>	b o n j o u r Process finished with exit code 0
Condition IF (1)	<pre>a = 25 if a > 20: print("chiffre supérieur à 20"); else: print("chiffre inférieur ou égal à 20");</pre>	chiffre supérieur à 20
Condition IF (2)	<pre>a = 1 b = 3 if a == 2 and b == 3: print("ok") else: print("non")</pre>	non

	Instructions	Résultat affiché sur la console
Condition IF (3)	<pre>a = "hello" if a == "hello": print("bonjour"); else: print("au revoir");</pre>	bonjour
Switch	<pre>a = 3 if a==1: print("Bonjour") elif a==2: print("ça va") elif a==3: print("Au revoir") else: print("c'est pas le bon chiffre")</pre>	Au revoir
While	<pre>i = 0 while (i <= 6): print("Voici la ligne ", i) i+=1</pre>	Voici la ligne 0 Voici la ligne 1 Voici la ligne 2 Voici la ligne 3 Voici la ligne 4 Voici la ligne 5 Voici la ligne 6
Binaire hexa	<pre>print(bin(12)) print(hex(0b11001)) print(0x80)</pre>	0b1100 0x19 128

	Instructions	Résultat affiché sur la console														
Extraction dans une chaîne	<pre>phrase = "bonjour" print (phrase[1:4])</pre>	<p>onj</p> <table border="1"> <tr> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> <tr> <td>b</td><td>o</td><td>n</td><td>j</td><td>o</td><td>u</td><td>r</td></tr> </table> <p>Le chiffre 4 est exclu</p>	0	1	2	3	4	5	6	b	o	n	j	o	u	r
0	1	2	3	4	5	6										
b	o	n	j	o	u	r										
Majuscule/minuscules	<pre>chaine = "NE PARLEZ PAS SI FORT !" print(chaine.lower()) chaine="nsi" chaine=chaine.upper() print(chaine)</pre>	<p>ne parlez pas si fort !</p> <p>NSI</p>														
Concaténation	<pre>w = "Washington" t= "Touchard" lycee= w+" "+t print (lycee) print (len(lycee))</pre>	<p>Washington Touchard</p> <p>19</p>														
Sortir d'un menu	<pre>chaine = "" while chaine.lower() != "q": print("Tapez 'Q/q' pour quitter...") chaine = input() print("Merci !")</pre>	<p>Tapez 'Q/q' pour quitter...</p> <p>s</p> <p>Tapez 'Q/q' pour quitter...</p> <p>q</p> <p>Merci !</p>														

	Instructions	Résultat affiché sur la console	
Input	<pre>a = float(input("saisir un nombre decimal: ")) print (a**3) b = int(input("saisir un nombre entier: ")) print (b**2) c=str(input("saisir une chaine de caractères : ")) print ("il y a",len(c),"caractere(s) dans votre chaine")</pre>	saisir un nombre decimal: 2.3 6.899999999999995 saisir un nombre entier: 5 25 saisir une chaine de caractères : hello il y a 5 caractere(s) dans votre chaine	
format	<pre>prenom ="Clint" nom="Eastwood" chaine = "Je m'appelle {} {}".format(prenom, nom) print(chaine)</pre>	Je m'appelle Clint Eastwood	
Chaine divers	<pre>chaine="bonjour" print(chaine.capitalize()) # La première lettre en majuscule chaine = " une chaine avec des espaces " print(chaine.strip()) # On retire les espaces au début et à la fin de la chaîne titre = "introduction" print(titre.upper().center(20))</pre>	Bonjour une chaine avec des espaces INTRODUCTION	
Aide str	<pre>help(str) #voir aussi en annexe 1 capitalize(self, /) casefold(self, /) center(self, width, fillchar=' ', /) count(...) encode(self, /, encoding='utf-8', errors='strict') endswith(...) expandtabs(self, /, tabsize=8) find(...) format(...) format_map(...) index(...) isalnum(self, /) isalpha(self, /)</pre>	<pre> isascii(self, /) isdecimal(self, /) isdigit(self, /) isidentifier(self, /) islower(self, /) isnumeric(self, /) isprintable(self, /) isspace(self, /) istitle(self, /) isupper(self, /) join(self, iterable, /) ljust(self, width, fillchar=' ', /) lower(self, /)</pre>	<pre> lstrip(self, chars=None, /) partition(self, sep, /) replace(self, old, new, count=-1, /) rfind(...) rindex(...) rjust(self, width, fillchar=' ', /) rpartition(self, sep, /) rsplit(self, /, sep=None, maxsplit=-1) rstrip(self, chars=None, /) Etc....</pre>

	Instructions	Résultat affiché sur la console												
append	<pre>liste=[1, 9, 8] liste.append(77) print(liste)</pre>	[1, 9, 8, 77]												
affichage	<pre>liste=[1, 9, 8, 45, 12] print(liste[3]) print(liste[3:]) print(liste[:3])</pre>	45 [45, 12] [1, 9, 8]												
extend	<pre>liste_un=[1, 3, 'toto'] liste_deux=[89, 2.5, 'titi'] liste_un.extend(liste_deux) print(liste_un)</pre>	[1, 3, 'toto', 89, 2.5, 'titi']												
remove	<pre>liste=[1, 9, 8, 45, 12] liste.remove(8) print(liste)</pre>	[1, 9, 45, 12] <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td> </tr> <tr> <td>1</td><td>9</td><td>8</td><td>45</td><td>12</td> </tr> </table>	0	1	2	3	4	1	9	8	45	12		
0	1	2	3	4										
1	9	8	45	12										
del	<pre>liste=[1, 9, 8, 45, 12, 78] del liste[2:4] print(liste)</pre>	[1, 9, 12, 78] <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td> </tr> <tr> <td>1</td><td>9</td><td>8</td><td>45</td><td>12</td><td>78</td> </tr> </table> <p>Indice 4 exclu</p>	0	1	2	3	4	5	1	9	8	45	12	78
0	1	2	3	4	5									
1	9	8	45	12	78									
count	<pre>liste=[1, 9, 8, 9, 12, 9] print(liste.count(9))</pre>	3												
Sort	<pre>liste=[1, 45, 2, 10, 17, 5] liste.sort() print(liste)</pre>	[1, 2, 5, 10, 17, 45]												
Sorted	<pre>liste=[1, 45, 2, 10, 17, 5] liste_triee=sorted(liste, reverse=True) print(liste) print(liste_triee)</pre>	[1, 45, 2, 10, 17, 5] [45, 17, 10, 5, 2, 1]												

	Instructions	Résultat affiché sur la console
Initialisation(1)	<pre>liste=[] for i in range(0,10): liste.append(0) print(liste) #ou tab=[0] *10 print(tab)</pre>	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Initialisation(2)	<pre>liste=[0 for i in range(0,10)] print(liste) liste=[i for i in range(0,10)] print(liste) liste=[i*2 for i in range(0,10)] print(liste)</pre>	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0] [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] [0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
Initialisation(3)	<pre>liste=[i*2 if i==5 else i for i in range(0,10)] print(liste)</pre>	[0, 1, 2, 3, 4, 10, 6, 7, 8, 9]
Deux dimensions	<pre>n = 3 m = 4 tab = [[0] * m for _ in range(n)] tab[1][1]=2 print (tab)</pre>	[[0, 0, 0, 0], [0, 2, 0, 0], [0, 0, 0, 0]]
affichage	<pre>a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]] for i in range(len(a)): for j in range(len(a[i])): print(a[i][j], end=' ') print()</pre>	1 2 3 4 5 6 7 8 9

Visualiser un code pas à pas graphiquement

<http://pythontutor.com/visualize.html#mode=edit>

```
liste=[5, 6, 7, 8, 9, 4]
somme=0
for n in range(len(liste)):
    somme=somme+liste[n]
print(somme)
```

Résultat :

39

Write code in Python 3.6

```
1 liste=[5,6,7,8,9,4]
2 somme=0
3 for n in range(len(liste)):
4     somme=somme+liste[n]
5 print(somme)
6 |
```

Help improve this tool by completing a [short user survey](#)

Visualize Execution
Live Programming Mode

Déroulement de l'exécution pas à pas

Python 3.6

```
1 liste=[5, 6, 7, 8, 9, 4]
2 somme=0
3 for n in range(len(liste)):
4     somme=somme+liste[n]
5 print(somme)
```

[Edit this code](#)

→ line that has just executed
→ next line to execute

Click a line of code to set a breakpoint; use the Back and Forward buttons to jump there.

[<< First](#) [< Back](#) Step 9 of 16 Forward > [Last >>](#)

Print output (drag lower right corner to resize)

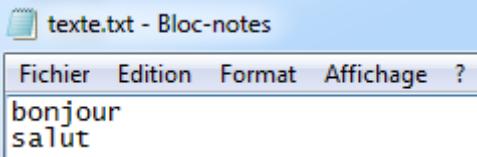
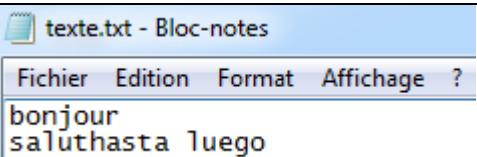
Frames	Objects
Global frame	list
liste	0 1 2 3 4 5
somme	18
n	2

formatage	<pre> print("pi vaut plus ou moins {0:7}.".format(3.14)) print("pi vaut plus ou moins {0:7.4}.".format(3.14159265359)) print("pi vaut plus ou moins {0:07.4}.".format(3.14159265359)) for x in range(1,11): print('{0:2d} {1:3d} {2:4d}'.format(x, x*x, x*x*x)) </pre>	pi vaut plus ou moins 3.14 pi vaut plus ou moins 3.142 pi vaut plus ou moins 003.142 1 1 1 2 4 8 3 9 27 4 16 64 5 25 125 6 36 216 7 49 343 8 64 512 9 81 729 10 100 1000
random	<pre> import random secret = random.randint(0, 5) n = int(input('valeur : ')) if n == secret: print("gagné!") else : print("La valeur était", secret) </pre>	valeur : 1 La valeur était 3
Tuple(1)	<pre> tuple=('a',1,5.6,[1,2,45]) print("longueur du tuple",len(tuple)) for i in tuple: print(i) </pre>	longueur du tuple 4 a 1 5.6 [1, 2, 45]
Tuple(2)	<pre> tuple_1 = (5, 2, 25, 56) tuple_2 = ("jack", "tony") tuple_3 = tuple_1 + tuple_2 print(tuple_3) </pre>	(5, 2, 25, 56, 'jack', 'tony')
Tuple(3)	<pre> tuple_1 = ("jack", "tony") tuple_2 = tuple_1 * 2 print(tuple_2) </pre>	('jack', 'tony', 'jack', 'tony') Rem : un tuple est non mutable (on ne peut pas changer le contenu)

	Instructions	Résultat affiché sur la console
Tuple(3)	<pre>def recuperer_resultats(): a=1 b=0.56 c=["toto", 45] return (a,b,c) print(recuperer_resultats())</pre>	(1, 0.56, ['toto', 45])
Tuple(4)	<pre>def recuperer_resultats(): a=1 b=0.56 c=["toto", 45] return (a,b,c) res=recuperer_resultats() print(len(res)) for items in res: print(items)</pre>	3 1 0.56 ['toto', 45]
Tuple(5)	<pre>def recuperer_resultats(): a=1 b=0.56 c=["toto", 45] return (a,b,c) res=recuperer_resultats() print(len(res)) for indice in range(len(res)): print(res[indice])</pre>	3 1 0.56 ['toto', 45]
Dictionnaire(1)	<pre>stock={"poires":5, "pommes":10, "fraises":35} print(len(stock)) print(stock["fraises"])</pre>	3 35
Dictionnaire(1)	<pre>stock={"poires":5, "pommes":10, "fraises":35} for i in stock: print(i,stock[i])</pre>	poires 5 pommes 10 fraises 35

	Instructions	Résultat affiché sur la console
Dictionnaire(3)	<pre>stock = {'poires': 5, 'pommes': 10, 'fraises': 35} print('fraises' in stock) print('fraises' in stock.keys()) print(7 in stock.values()) print(5 in stock.values()) print(3 in stock.values())</pre>	True True False True False
Dict.(4)	<pre>stock = {'poires': 5, 'pommes': 10, 'fraises': 35} for nom, num in stock.items(): print(nom, '->', num)</pre>	poires -> 5 pommes -> 10 fraises -> 35
Dict.(5)	<pre>inventaire="" for nom in stock.keys(): inventaire += nom + ', ' print(inventaire)</pre>	poires, pommes, fraises,
Dict.(6)	<pre>for num in stock.values(): print (num)</pre>	5 10 35
Ensemble(1)	<pre>H = set([1,2,3,6,3,2,1,8,5]) H.add(4) # un nouvel élémént est ajouté H.add(1) # si l'élément est déjà présent, il ne se passe rien print('H =',H) H.remove(8) print('H =',H)</pre>	$H = \{1, 2, 3, 4, 5, 6, 8\}$ $H = \{1, 2, 3, 4, 5, 6\}$
Ensemble(2)	<pre>H = set([1,2,3,6,3,2,1,8,5]) G = set([1,10,3]) print('H',chr(8745), 'G =',H & G) #& intersection. print('H',chr(8746), 'G =',H G) # union. print('H',chr(8726), 'G =', H - G) # - difference print('H',chr(8854), 'G =', H ^ G) #^ symmetric difference</pre>	$H \cap G = \{1, 3\}$ $H \cup G = \{1, 2, 3, 5, 6, 8, 10\}$ $H \setminus G = \{8, 2, 5, 6\}$ $H \ominus G = \{2, 5, 6, 8, 10\}$

	Instructions	Résultat affiché sur la console																																							
split	<pre>chaine="pensez au c++ après le python" nouvelle_chaine=chaine.split() print(nouvelle_chaine) chaine="alors,quoi de neuf" nouvelle_chaine=chaine.split(',') print(nouvelle_chaine)</pre>	[pensez', 'au', 'c++', 'après', 'le', 'python'] ['alors', 'quoi de neuf']																																							
	<pre>chaine="hello" print(chaine*3)</pre>	hellohellohello Impossible à faire en C++																																							
join	<pre>liste=['scotty','say','hello', 'computer'] nouvelle_chaine=" ".join(liste) print(nouvelle_chaine)</pre>	scotty says hello computer																																							
count	<pre>chaine="hello lycée touchard" print(chaine.count('o'))</pre>	2 En effet il y a 2 'o' dans la chaîne																																							
find	<pre>chaine="hello lycée touchard" print(chaine.find('tou'))</pre>	12																																							
	<table border="1"> <tr> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td></tr> <tr> <td>h</td><td>e</td><td>l</td><td>l</td><td>o</td><td></td><td>l</td><td>y</td><td>c</td><td>é</td><td>e</td><td></td><td>t</td><td>o</td><td>u</td><td>c</td><td>h</td><td>a</td><td>r</td><td>d</td></tr> </table>	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	h	e	l	l	o		l	y	c	é	e		t	o	u	c	h	a	r	d
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19																						
h	e	l	l	o		l	y	c	é	e		t	o	u	c	h	a	r	d																						
print	<pre>print("Hello World"); print("Aujourd'hui"); print("C'est \"Dommage!\""); print("Hum \\o/"); """ \' La simple quote (') \" La double quote (") \n Le passage à la ligne ASCII \t La tabulation horizontale \v La tabulation verticale \\ L'anti slash (Backslash \) """ </pre>	Hello World Aujourd'hui C'est "Dommage!" Hum \o/ \' La simple quote (') \" La double quote (") \n Le passage à la ligne ASCII \t La tabulation horizontale \v La tabulation verticale \\ L'anti slash (Backslash \) """																																							

	Instructions	Résultat affiché sur la console
Ecriture dans un fichier texte	<pre>import os os.chdir('E:\python mooc') fichier=open("texte.txt","w") fichier.write("bonjour\n") fichier.write("salut") fichier.close()</pre>	 <p>W : crée le fichier, si il existe, le fichier est écrasé</p>
Ajouter du texte dans un fichier	<pre>import os os.chdir('E:\python mooc') fichier=open("texte.txt","a") fichier.write("hasta luego") fichier.close()</pre>	 <p>a : ajoute des données en fin de fichier</p>
Lecture d'un fichier texte	<pre>import os os.chdir('E:\python mooc') fichier=open("texte.txt","r") chaine=fichier.read() print(chaine) fichier.close()</pre>	bonjour saluthasta luego

La liste des méthodes standard sur les fichiers est la suivante:

Instruction	Effet
f=open("fichier") :	ouvre "fichier" en lecture
f=open("fichier","w") :	ouvre "fichier" en écriture
f=open("fichier","a") :	ouvre "fichier" en écriture en rajoutant après les données déjà présentes
f.read() :	retourne le contenu du fichier f
f.readline() :	lit une ligne
f.readlines() :	renvoie la liste des lignes de f
f.write(s) :	écrit la chaîne de caractères s dans le fichier f
f.close() :	ferme f

	Instructions	Résultat affiché sur la console
Ecriture dans un fichier binaire	<pre>import os, pickle os.chdir('E:\python_mooc') a=10 b=2.3 c="bonjour" fichier=open ("donnees.bin", "wb") pickle.dump(a,fichier) pickle.dump(b,fichier) pickle.dump(c,fichier) fichier.close()</pre>	<p>Contenu du fichier avec un éditeur hexadécimal :</p>  <p>wb : crée le fichier en binaire</p>
Lecture d'un fichier binaire	<pre>import os, pickle os.chdir('E:\python_mooc') fichier=open ("donnees.bin", "rb") a=pickle.load(fichier) b=pickle.load(fichier) c=pickle.load(fichier) fichier.close() print(a,b,c)</pre>	10 2.3 bonjour

Contenu du Fichier.txt :

bonjour,
il fait toujours beau
en bretagne.

	Instructions	Résultat affiché sur la console
With open(1)	<pre>with open('fichier.txt') as f: print(f.readline(),end='') print(f.readline(),end='')</pre>	bonjour, il fait toujours beau
With open(2)	<pre>with open('fichier.txt') as f: for ligne in f: print(ligne.split())</pre>	['bonjour,'] ['il', 'fait', 'toujours', 'beau'] ['en', 'bretagne.']}

	Instructions	Résultat affiché sur la console
Fichier csv (écriture)	<pre>import csv csvData = [['Personne', 'Age'], ['Peter', '22'], ['Tony', '45'], ['Bruce', '50']] with open('person.csv', 'w') as csvFile: writer = csv.writer(csvFile) writer.writerows(csvData) csvFile.close()</pre>	
Fichier csv (lecture)	<pre>import csv f = open("personne.csv", "r") c = csv.reader(f, delimiter=',') tableau = [] for ligne in c: tableau.append(ligne) f.close() print(tableau)</pre>	[['Personne', 'Age'], ['Peter', '22'], ['Tony', '45'], ['Bruce', '50']]
Fichier json (écriture)	<pre>import json dico = {"agrumes": {"oranges": 4, "citrons": 2, "pamplemousse": 54}, "salades": {"batavia": 2, "laitue": 9}} f = open('fruits_legumes.json', 'w') json.dump(dico, f, indent=4) f.close()</pre>	{ "agrumes": { "oranges": 4, "citrons": 2, "pamplemousse": 54 }, "salades": { "batavia": 2, "laitue": 9 } }
Fichier json (lecture)	<pre>import json g = open('fruits_legumes.json', 'r') fruitslegumes = json.load(g) g.close() print(fruitslegumes)</pre>	{'agrumes': {'oranges': 4, 'citrons': 2, 'pamplemousse': 54}, 'salades': {'batavia': 2, 'laitue': 9}}

<https://jsonformatter.curiousconcept.com/>

	Instructions	Résultat affiché sur la console
Lambda(1)	<pre>g = lambda x: x**2 print(g(3))</pre>	6
Lambda(2)	<pre>x = lambda a, b, c : a + b + c print(x(5, 6, 2))</pre>	13
Lambda(3)	<pre>L = [1, 2, 3, 4] print(list(map(lambda x: x**2, L)))</pre>	[1, 4, 9, 16]
filtré	<pre>def even_fn(x): if x % 2 == 0: return True return False print(list(filter(even_fn, [1, 3, 2, 5, 20, 21])))</pre>	[2, 20]

	Instructions	Résultat affiché sur la console
Les fonctions	<pre>def table_de(table): for chiffre in range(1,10+1): print(chiffre,"x",table,"=",chiffre*table) table_de(5)</pre>	1 x 5 = 5 2 x 5 = 10 3 x 5 = 15 4 x 5 = 20 5 x 5 = 25 6 x 5 = 30 7 x 5 = 35 8 x 5 = 40 9 x 5 = 45 10 x 5 = 50
Fonction avec retour	<pre>def calcul(chiffre,puissance=2): return chiffre**puissance print(calcul(3)) print(calcul(3,4))</pre>	9 81

Tracé de courbes avec matplotlib

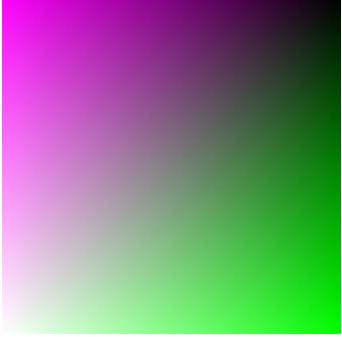
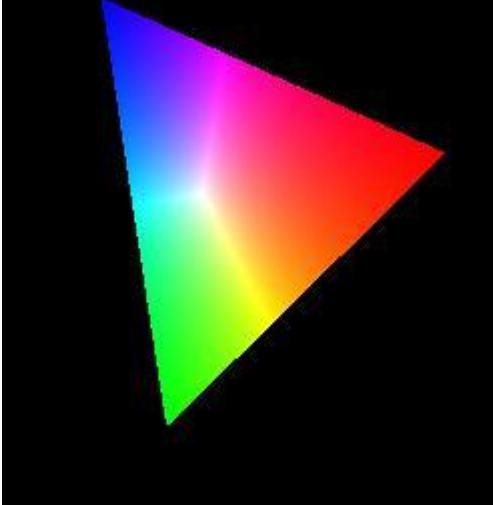
<https://matplotlib.org/2.0.2/Matplotlib.pdf>

	Instructions	Résultat affiché sur la console
Tracé à partir de points	<pre>from pylab import * x = array([1, 3, 4, 6]) y = array([2, 3, 5, 1]) plot(x, y) show() # affiche la figure a l'ecran</pre>	
cosinus	<pre>from pylab import * x = linspace(0, 2*pi, 30) y = cos(x) plot(x, y) show() # affiche la figure a l'ecran</pre>	
parabole	<pre>from pylab import * x = linspace(-5, 7, 50) y = +2*x**2-4*x-40 plot(x, y) show() # affiche la figure</pre>	

<https://pillow.readthedocs.io/en/stable/>

Affichage d'une image	Instructions <pre>from PIL import Image img=Image.open("lighthouse.jpg") largeur,hauteur=img.size print (largeur,hauteur) img.show()</pre> 960 640	
Résultat		

	Instructions
Conversion en noir et blanc	<pre>from PIL import Image img=Image.open("lighthouse.jpg") largeur,hauteur=img.size for x in range(largeur): for y in range(hauteur): r,v,b=img.getpixel((x,y)) moyenne=int((r+v+b)/3) img.putpixel((x,y),(moyenne,moyenne,moyenne)) img.show()</pre>
Résultat	

	Instructions	Résultat affiché
Degradé	<pre>from PIL import Image img=Image.new('RGB', (256,256), (0,0,0)) largeur,hauteur=img.size for y in range(hauteur): for x in range(largeur): img.putpixel((x,y), (255-x, y, 255-x)) img.show() img.save("degrade.jpg")</pre>	
Palette	<pre>from PIL import Image img=Image.new('RGB', (256,256), (0,0,0)) largeur,hauteur=img.size for r in range(1,256): for v in range(256): for b in range(256): X = 0.49 * r + 0.31 * v + 0.20 * b; Y = 0.17697 * r + 0.81240 * v + 0.01063 * b; Z = 0.01 * v + 0.99 * b; x = int(X * 300 / (X + Y + Z)); y = int(Y * 300 / (X + Y + Z)); img.putpixel((x,y), (r,v,b)) img.show() img.save("palette.jpg")</pre>	 (le temps de traitement est long)

Instructions

```
#https://python-visualization.github.io/folium/quickstart.html
import folium

m=folium.Map(
    location=[ 47.995332,  0.204976],
    tiles='OpenStreetMap',
    zoom_start=20
)
folium.Marker(
    location=[ 47.995653,  0.20267],
    popup='Washington place',
    icon=folium.Icon(icon='cloud')
).add_to(m)

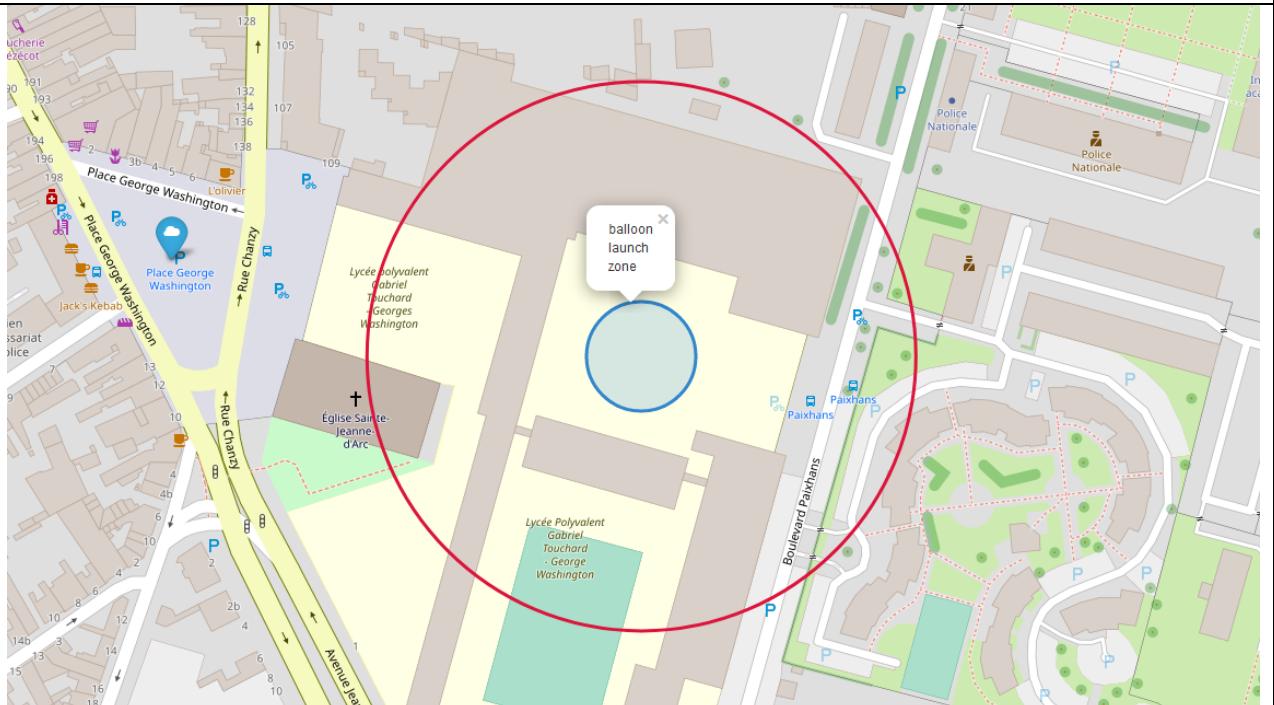
folium.Circle(
    radius=100,
    location=[ 47.995332,  0.204976],
    popup='The high school',
    color='crimson',
    fill=False,
).add_to(m)

folium.CircleMarker(
    location=[ 47.995332,  0.204976],
    radius=50,
    popup='balloon launch zone',
    color='#3186cc',
    fill=True,
    fill_color='#3186cc'
).add_to(m)

m.save('index.html')
```

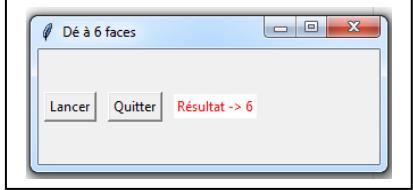
Génération d'une page web

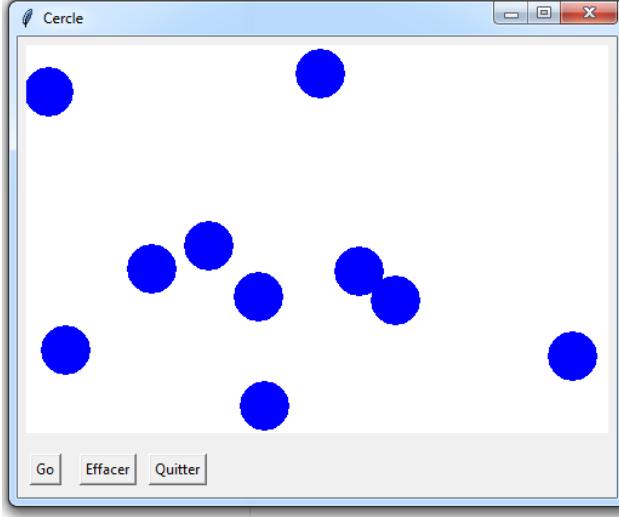
Résultat



<https://python-graph-gallery.com/288-map-background-with-folium/>

https://www.python-course.eu/tkinter_buttons.php

	Instructions <pre>import tkinter as tk def write_slogan(): print("Tkinter is easy to use!") root = tk.Tk() frame = tk.Frame(root) frame.pack() button = </pre>
Boutons (1)	<pre>tk.Button(frame, text="QUIT", fg="red", command=quit) button.pack(side=tk.LEFT) slogan = tk.Button(frame, text="Hello", command=write_slogan) slogan.pack(side=tk.LEFT) root.mainloop()</pre> 
Boutons (2)	Instructions <pre>#http://fsincere.free.fr/isn/python/cours_python_tkinter.php from tkinter import * import random def NouveauLance(): nb = random.randint(1, 6) Texte.set('Résultat -> ' + str(nb)) # Création de la fenêtre principale (main window) Mafenetre = Tk() Mafenetre.title('Dé à 6 faces') Mafenetre.geometry('300x100') # Création d'un widget Button (bouton Lancer) BoutonLancer = Button(Mafenetre, text ='Lancer', command = NouveauLance) # Positionnement du widget avec la méthode pack() BoutonLancer.pack(side = LEFT, padx = 5, pady = 5) # Création d'un widget Button (bouton Quitter) BoutonQuitter = Button(Mafenetre, text ='Quitter', command = Mafenetre.destroy) BoutonQuitter.pack(side = LEFT, padx = 5, pady = 5) Texte = StringVar() NouveauLance() # Création d'un widget Label (texte 'Résultat -> x') LabelResultat = Label(Mafenetre, textvariable = Texte, fg ='red', bg ='white') LabelResultat.pack(side = LEFT, padx = 5, pady = 5) Mafenetre.mainloop()</pre> 

	Instructions
Canevas	<pre># script cercle.py # (C) Fabrice Sincère from tkinter import * import random def Cercle(): """ Dessine un cercle de centre (x,y) et de rayon r """ x = random.randint(0,Largeur) y = random.randint(0,Hauteur) r = 20 Canevas.create_oval(x-r, y-r, x+r, y+r, outline='blue', fill='blue') def Effacer(): """ Efface la zone graphique """ Canevas.delete(ALL) # Création de la fenêtre principale (main window) Mafenetre = Tk() Mafenetre.title('Cercle') # Création d'un widget Canvas (zone graphique) Largeur = 480 Hauteur = 320 Canevas = Canvas(Mafenetre, width = Largeur, height = Hauteur, bg ='white') Canevas.pack(padx =5, pady =5) # Création d'un widget Button (bouton Go) BoutonGo = Button(Mafenetre, text ='Go', command = Cercle) BoutonGo.pack(side = LEFT, padx = 10, pady = 10) # Création d'un widget Button (bouton Effacer) BoutonEffacer = Button(Mafenetre, text ='Effacer', command = Effacer) BoutonEffacer.pack(side = LEFT, padx = 5, pady = 5) # Création d'un widget Button (bouton Quitter) BoutonQuitter = Button(Mafenetre, text ='Quitter', command = Mafenetre.destroy) BoutonQuitter.pack(side = LEFT, padx = 5, pady = 5) Mafenetre.mainloop()</pre>
Résultat	

Guizero plus simple que tkinter (surcouche logicielle)

	Instructions
Guizero	<pre>#https://lawsie.github.io/guizero/ from guizero import App, PushButton, CheckBox, info, Drawing import playsound as ps def play(): info("play", "ecoutons de la musique ?") ps.playsound("test.mp3", False) #non bloquant # ps.playsound("test.mp3") #bloquant app=App(width=256,height=256,layout='grid') bouton=PushButton(app,command=play,grid=[0,0]) checkbox=CheckBox(app,text="check here",grid=[2,0]) drawing=Drawing(app,grid=[4,2]) drawing.rectangle(0,0,20,20,color='red') #trace un rectangle dans la case 4,2 app.display()</pre>
Résultat	

APIXU

HOME API WEATHER DOCS PRICING

SIGNUP LOGIN

Popular: Gran Canaria | Tenerife | Majorca | Lanzarote | Fuerteventura | Crete | New York | Rome | Dubai | London | Sydney | Moscow

FREE, FAST, SIMPLE AND FULLY MANAGED

JSON and XML Weather API and Geo Developer API

API Key

Key: cf965220-1f5c-41e7-806 [Get Started]

Current Plan: Free [Change Plan]

Status: LIVE

Fields: Change Response Fields

Regenerate key

Has your key been compromised? You can generate a new key.

Before you proceed?

- You will need to change your apps to use the new key.
- Your statistics will be reset.
- This action cannot be undone.

I would like to regenerate api key.

Regenerate Key

[Interactive](#) | [Analytics](#) | [Documentation](#)

Current Weather

HTTP: <http://api.apixu.com/v1/current.json?key=cf965220-1f5c-41e7-806&q=Paris>

HTTPS: <https://api.apixu.com/v1/current.json?key=cf965220-1f5c-41e7-806&q=Paris>

Forecast Weather

HTTP: <http://api.apixu.com/v1/forecast.json?key=cf965220-1f5c-41e7-806&q=Paris>

HTTPS: <https://api.apixu.com/v1/forecast.json?key=cf965220-1f5c-41e7-806&q=Paris>

	Instructions
api xu	<pre> import requests, json urlData = "https://api.apixu.com/v1/current.json?key=cf9652aabf6e4ea6868105327190806&q=le -mans" response = requests.get(urlData) data = response.json() print(type(data)) print(len(data)) print(data) print(data.keys()) location = data["location"] print(type(location)) print(location["name"]) print(location["region"]) print(location["country"]) print(location["lat"]) print(location["lon"]) #ou plus vite for nom, num in location.items(): print(nom, '->', num) current = data["current"] for nom, num in current.items(): print(nom, '->', num) </pre>
Résultat	<pre> <class 'dict'> 2 {'location': {'name': 'Le Mans', 'region': 'Pays de la Loire', 'country': 'France', 'lat': 48.0, 'lon': 0.2, 'tz_id': 'Europe/Paris', 'localtime_epoch': 1560001514, 'localtime': '2019-06-08 15:45'}, 'current': {'last_updated_epoch': 1560000611, 'last_updated': '2019-06-08 15:30', 'temp_c': 18.0, 'temp_f': 64.4, 'is_day': 1, 'condition': {'text': 'Moderate rain', 'icon': '//cdn.apixu.com/weather/64x64/day/302.png', 'code': 1189}, 'wind_mph': 11.9, 'wind_kph': 19.1, 'wind_degree': 240, 'wind_dir': 'WSW', 'pressure_mb': 1022.0, 'pressure_in': 30.7, 'precip_mm': 0.0, 'precip_in': 0.0, 'humidity': 56, 'cloud': 100, 'feelslike_c': 18.0, 'feelslike_f': 64.4, 'vis_km': 10.0, 'vis_miles': 6.0, 'uv': 4.0, 'gust_mph': 21.5, 'gust_kph': 34.6}} dict_keys(['location', 'current']) <class 'dict'> name -> Le Mans region -> Pays de la Loire country -> France lat -> 48.0 lon -> 0.2 tz_id -> Europe/Paris localtime_epoch -> 1560001514 localtime -> 2019-06-08 15:45 last_updated_epoch -> 1560000611 last_updated -> 2019-06-08 15:30 temp_c -> 18.0 temp_f -> 64.4 is_day -> 1 condition -> {'text': 'Moderate rain', 'icon': '//cdn.apixu.com/weather/64x64/day/302.png', 'code': 1189} wind_mph -> 11.9 wind_kph -> 19.1 wind_degree -> 240 wind_dir -> WSW pressure_mb -> 1022.0 pressure_in -> 30.7 precip_mm -> 0.0 precip_in -> 0.0 humidity -> 56 cloud -> 100 feelslike_c -> 18.0 feelslike_f -> 64.4 vis_km -> 10.0 vis_miles -> 6.0 uv -> 4.0 gust_mph -> 21.5 gust_kph -> 34.6 </pre>

The screenshot shows the Code Beautify JSON Viewer. A modal window titled "Enter Url" is displayed, containing the URL "https://api.apixu.com/v1/current.json?key=cf9652aabf6e". Below the URL input is a "Sample URL" field, which is currently empty. At the bottom of the modal are two buttons: "Load" and "Cancel". To the left of the modal, there is a dark-themed code editor window showing a single line of JSON code:

```
1 [{"location": {"name": "Le Mans", "region": "Pays de la Loire", "country": "France", "lat": 48.0, "lon": 0.2, "tz_id": "Europe/Paris", "localtime_epoch": 1560002731, "localtime": "2019-06-08 16:05"}, "current": {"last_updated_epoch": 1560002414, "last_updated": "2019-06-08 16:00", "temp_c": 18.0, "temp_f": 64.4, "is_day": 1, "condition": {"text": "Moderate rain", "icon": "/cdn.apixu.com/weather/64x64/day/302.png", "code": 1189}, "wind_mph": 8.1, "wind_kph": 13.0, "wind_degree": 240, "wind_dir": "WSW", "pressure_mb": 1022.0, "pressure_in": 30.7, "precip_mm": 0.0, "precip_in": 0.0, "humidity": 56, "cloud": 75, "feelslike_c": 18.0, "feelslike_f": 64.4, "vis_km": 10.0, "vis_miles": 6.0, "uv": 4.0, "gust_mph": 20.6, "gust_kph": 33.1}}]
```

The screenshot shows the Code Beautify JSON Viewer. On the left is a dark-themed code editor showing the JSON code from the previous screenshot. In the center is a panel with various tools:

- "Result mode:" dropdown (set to "tree")
- "Load Url" button
- "Browse" button
- "Tree Viewer" button (highlighted)
- "2 Tab Space" dropdown
- "Beautify" button

On the right is a light-themed code editor showing the JSON code with line numbers and syntax highlighting. The "Beautify" button is also present here.

```
1 [{"location": {"name": "Le Mans", "region": "Pays de la Loire", "country": "France", "lat": 48.0, "lon": 0.2, "tz_id": "Europe/Paris", "localtime_epoch": 1560002731, "localtime": "2019-06-08 16:05"}, "current": {"last_updated_epoch": 1560002414, "last_updated": "2019-06-08 16:00", "temp_c": 18, "temp_f": 64.4, "is_day": 1, "condition": {"text": "Moderate rain", "icon": "/cdn.apixu.com/weather/64x64/day/302.png", "code": 1189}, "wind_mph": 8.1, "wind_kph": 13, "wind_degree": 240, "wind_dir": "WSW", "pressure_mb": 1022, "pressure_in": 30.7, "precip_mm": 0, "precip_in": 0, "humidity": 56, "cloud": 75, "feelslike_c": 18, "feelslike_f": 64.4, "vis_km": 10, "vis_miles": 6, "uv": 4, "gust_mph": 20.6, "gust_kph": 33.1}}]
```

The screenshot shows the 'API keys' section of the OpenWeatherMap website. At the top, there's a navigation bar with links for Support Center, Weather in your area, Weather, Maps, API, Price, and Partners. Below the navigation is a sub-navigation bar with links for New Products, Setup, API keys (which is underlined), Services, Payments, Billing plans, and Block logs. A message box states: 'You can generate as many API keys as needed for your subscription. We accumulate the total load from all keys.' Below this, a table lists one API key:

Key	Name
3d7[REDACTED]a3c	Default

Next to the key value are two small icons: a gear and a trash can.

Programme page suivante

Résultat :

Enter city name : teloche

```
dict_keys(['coord', 'weather', 'base', 'main', 'visibility', 'wind', 'clouds', 'dt', 'sys',  
'timezone', 'id', 'name', 'cod'])  
{'coord': {'lon': 0.27, 'lat': 47.89}, 'weather': [{"id": 803, 'main': 'Clouds', 'description':  
'broken clouds', 'icon': '04d'}], 'base': 'stations', 'main': {'temp': 291.84, 'pressure':  
1022, 'humidity': 55, 'temp_min': 290.93, 'temp_max': 292.59}, 'visibility': 10000,  
'wind': {'speed': 3.6, 'deg': 240}, 'clouds': {'all': 75}, 'dt': 1560002090, 'sys': {'type':  
1, 'id': 6570, 'message': 0.0139, 'country': 'FR', 'sunrise': 1559966457, 'sunset':  
1560023707}, 'timezone': 7200, 'id': 2973192, 'name': 'Teloche', 'cod': 200}
```

Temperature (in kelvin unit) = 291.84

atmospheric pressure (in hPa unit) = 1022

humidity (in percentage) = 55

description = broken clouds

NSI : Python 3

```
import requests, json

# Enter your API key here
api_key = "3d71c286cd789d06043a2d7bc65afa3c"

# base_url variable to store url
base_url = "http://api.openweathermap.org/data/2.5/weather?"

# Give city name
city_name = input("Enter city name : ")

# complete_url variable to store
# complete url address
complete_url = base_url + "appid=" + api_key + "&q=" + city_name

# get method of requests module
# return response object
response = requests.get(complete_url)

# json method of response object
# convert json format data into
# python format data
data = response.json()
print(data.keys())
print(data)

# Now x contains list of nested dictionaries
# Check the value of "cod" key is equal to
# "404", means city is found otherwise,
# city is not found
if data["cod"] != "404":

    # store the value of "main"
    # key in variable y
    y = data["main"]

    # store the value corresponding
    # to the "temp" key of y
    current_temperature = y["temp"]

    # store the value corresponding
    # to the "pressure" key of y
    current_pressure = y["pressure"]

    # store the value corresponding
    # to the "humidity" key of y
    current_humidiy = y["humidity"]

    # store the value of "weather"
    # key in variable z
    wx = data["weather"]

    # store the value corresponding
    # to the "description" key at
    # the 0th index of z
    weather_description = wx[0]["description"]

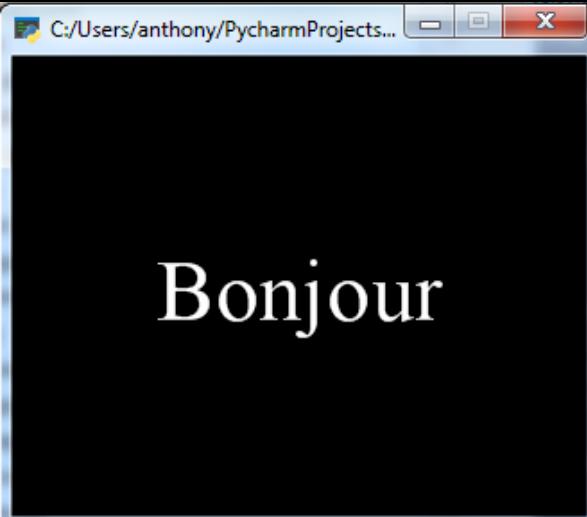
    # print following values
    print(" Temperature (in kelvin unit) = " +
          str(current_temperature) +
          "\n atmospheric pressure (in hPa unit) = " +
          str(current_pressure) +
          "\n humidity (in percentage) = " +
          str(current_humidiy) +
          "\n description = " +
          str(weather_description))

else:
    print(" City Not Found ")
```

Pygame

	Instructions
Click	<pre> import pygame pygame.init() clock = pygame.time.Clock() CIEL = 0, 200, 255 VERT = 0, 255, 0 ROUGE = 255, 0, 0 loop = True fenetre = pygame.display.set_mode((640, 480)) background = pygame.Surface(fenetre.get_size()) pygame.display.set_caption('Exemple') background.fill(CIEL) fenetre.blit(background, (0, 0)) while loop: for event in pygame.event.get(): if event.type == pygame.QUIT: loop = False #fermeture de la fenêtre (croix rouge) elif event.type == pygame.MOUSEBUTTONDOWN: x, y = pygame.mouse.get_pos() #recupération des coord x,y de la souris btn=pygame.mouse.get_pressed() #recupération des états des boutons de la souris print(btn) if btn[0]==1: rect_green = pygame.draw.rect(fenetre, VERT, [x, y, 100, 50]) elif btn[2]==1: rect_green = pygame.draw.rect(fenetre, ROUGE, [x, y, 100, 50]) elif btn[1]==1: loop=False elif event.type == pygame.KEYDOWN: #lecture du clavier if event.key == pygame.K_ESCAPE or event.unicode == 'q': loop = False # Actualisation de l'affichage pygame.display.flip() # 10 fps clock.tick(10) </pre>
Résultat	

Pyglet

	Instructions
Test	<pre> import pyglet from pyglet.window import key from pyglet.window import mouse window = pyglet.window.Window(width=320, height=256) label = pyglet.text.Label('Bonjour', font_name='Times New Roman', font_size=36, x=window.width//2, y=window.height//2, anchor_x='center', anchor_y='center') @window.event def on_draw(): window.clear() label.draw() @window.event def on_key_press(symbol, modifiers): if symbol == key.A: print('The "A" key was pressed.') elif symbol == key.P: music = pyglet.resource.media('test.mp3') #http://avbin.github.io/AVbin/Download.html #avbin (win32) dans c:/windows/system32 music.play() @window.event def on_mouse_press(x, y, button, modifiers): if button == mouse.LEFT: print('The left mouse button was pressed.') pyglet.app.run() </pre>
Résultat	 <p>Touche a Affichage d'un message</p> <p>Touche p Lecture d'un fichier mp3</p>

A suivre...

Pygal : Création de fichiers svg

https://fr.wikipedia.org/wiki/Scalable_Vector_Graphics

	Instructions																												
Test	<pre>import pygal bar_chart = pygal.Bar() # First import pygal bar_chart.add('Fibonacci', [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]) # Then create a bar graph object bar_chart.add('Fibonacci', [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]) # Add some values bar_chart.render_to_file('bar_chart.svg') # Save the svg to a file</pre>																												
Résultat	<table border="1"> <thead> <tr> <th>Index</th> <th>Value</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td></tr> <tr><td>2</td><td>1</td></tr> <tr><td>3</td><td>2</td></tr> <tr><td>4</td><td>3</td></tr> <tr><td>5</td><td>5</td></tr> <tr><td>6</td><td>8</td></tr> <tr><td>7</td><td>13</td></tr> <tr><td>8</td><td>21</td></tr> <tr><td>9</td><td>34</td></tr> <tr><td>10</td><td>55</td></tr> <tr><td>11</td><td>89</td></tr> <tr><td>12</td><td>144</td></tr> </tbody> </table>	Index	Value	0	0	1	1	2	1	3	2	4	3	5	5	6	8	7	13	8	21	9	34	10	55	11	89	12	144
Index	Value																												
0	0																												
1	1																												
2	1																												
3	2																												
4	3																												
5	5																												
6	8																												
7	13																												
8	21																												
9	34																												
10	55																												
11	89																												
12	144																												

Fonctionne sous windows, problème sous linux ?

	Instructions
Création de base	<pre> import sqlite3 conn = sqlite3.connect('base.db') c = conn.cursor() # Create table c.execute("CREATE TABLE IF NOT EXISTS base(id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE, nom TEXT, prenom TEXT, adresse TEXT, cp INT, ville TEXT);") # Insert a row of data c.execute("INSERT INTO base(nom, prenom, adresse,cp,ville) VALUES ('Wayne','Bruce','32 chem de la chauve souris',72000,'LE MANS')"); c.execute("INSERT INTO base(nom, prenom, adresse,cp,ville) VALUES ('Parker','Peter','16 bd de la tarantule',44000,'NANTES')"); c.execute("INSERT INTO base(nom, prenom, adresse,cp,ville) VALUES ('Banner','Bruce','3 rue du geant vert',75000,'PARIS')"); c.execute("INSERT INTO base(nom, prenom, adresse,cp,ville) VALUES ('Rogers','Steeve','5 rue du bouclier',49000,'ANGERS')"); c.execute("INSERT INTO base(nom, prenom, adresse,cp,ville) VALUES ('Stark','Tony','2 bd de la tour',72330,'PARIGNE LE POLIN')"); # Save (commit) the changes conn.commit() c.execute("SELECT * FROM base") rows = c.fetchall() for row in rows: print(row) conn.close() #close the connection. </pre>
Résultat	<pre> (1, 'Wayne', 'Bruce', '32 chem de la chauve souris', 72000, 'LE MANS') (2, 'Parker', 'Peter', '16 bd de la tarantule', 44000, 'NANTES') (3, 'Banner', 'Bruce', '3 rue du geant vert', 75000, 'PARIS') (4, 'Rogers', 'Steeve', '5 rue du bouclier', 49000, 'ANGERS') (5, 'Stark', 'Tony', '2 bd de la tour', 72330, 'PARIGNE LE POLIN') </pre>

	Instructions
Lecture d'une base	<pre> import sqlite3 conn = sqlite3.connect('base.db') def requete(req): print("-----") c = conn.cursor() c.execute(req) rows = c.fetchall() for row in rows: print(row) requete("SELECT * FROM base ORDER BY cp") requete("SELECT * FROM base WHERE prenom='Bruce'") id = 3; requete("UPDATE base SET prenom='hulk' where id="+str(id)) requete("SELECT * FROM base WHERE id=" + str(id)) conn.close() </pre>
Résultat	<pre> ----- (2, 'Parker', 'Peter', '16 bd de la tarantule', 44000, 'NANTES') (4, 'Rogers', 'Steeve', '5 rue du bouclier', 49000, 'ANGERS') (1, 'Wayne', 'Bruce', '32 chem de la chauve souris', 72000, 'LE MANS') (5, 'Stark', 'Tony', '2 bd de la tour', 72330, 'PARIGNE LE POLIN') (3, 'Banner', 'Bruce', '3 rue du geant vert', 75000, 'PARIS') ----- (1, 'Wayne', 'Bruce', '32 chem de la chauve souris', 72000, 'LE MANS') (3, 'Banner', 'Bruce', '3 rue du geant vert', 75000, 'PARIS') ----- (3, 'Banner', 'hulk', '3 rue du geant vert', 75000, 'PARIS') </pre>

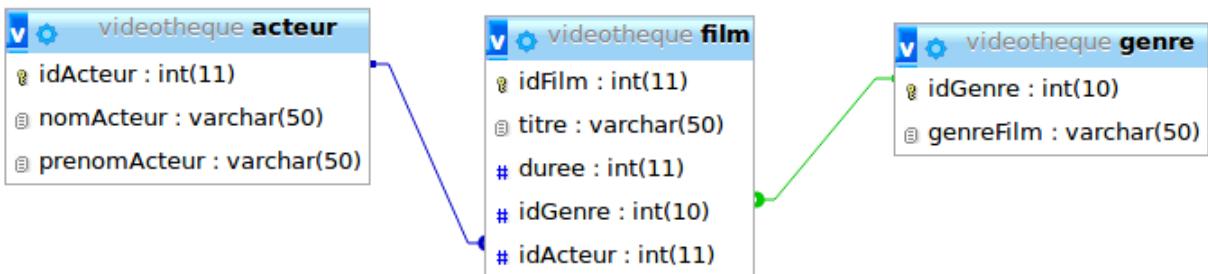
Afficher le contenu d'une base en ligne :

<https://sqliteonline.com/>

ou

<https://sqlitebrowser.org/dl/>

Clés étrangères



Les tables ont la structure suivante :

genre				
Champs	Type	Clef primaire	auto_increment	clef étrangère
<u>idGenre</u>	int(10)	X	X	
genreFilm	varchar(50)			

acteur				
Champs	Type	Clef primaire	auto_increment	clef étrangère
<u>idActeur</u>	int(11)	X	X	
nomActeur	varchar(50)			
prenomActeur	varchar(50)			

film				
Champs	Type	Clef primaire	auto_increment	clef étrangère
<u>idFilm</u>	int(11)	X	X	
titre	varchar(50)			
duree	int(11)			
idGenre	int(10)			X
idActeur	int(11)			X

Contenu des tables

genre	
<u>idGenre</u>	genreFilm
1	Comedie
2	Drame
3	Policier
4	Western
5	Science-fiction
6	Dessin animé
7	Aventure

acteur		
<u>idActeur</u>	nomActeur	prenomActeur
1	Ford	Harisson
2	Weather	Sigourney
3	Travolta	John
4	Smith	Will
5	Marceau	Sophie
6	Snipes	Wesley
7	Lhermitte	Thierry
8	Roberts	Julia
9	Di Caprio	Leonardo
10	Gable	Clark
11	Russel	Kurt
12	Houston	Angelica

film				
<u>idFilm</u>	titre	duree	idGenre	idActeur
1	Star Wars 4	117	5	1
2	Star Wars 5	122	5	1
3	Alien 3	110	5	2
4	Allo maman ici bebe	92	1	3
5	Bad boys	113	3	4
6	La famille Addams	95	1	12
7	La fille de d artagnan	124	7	5
8	Blade	116	5	6
9	Le pere noel est une ordure	90	1	7
10	New York 1997	99	5	11
11	Pretty Woman	114	1	8
12	Titanic	187	7	9
13	Just married	132	1	8
14	Autant en emporte le vent	175	7	10
15	Indiana Jones	183	7	1

Création de la base videotheque et des trois tables (genre, acteur, film)

```

import sqlite3
conn = sqlite3.connect('videotheque.db')
c = conn.cursor()

# Create tables
c.execute("create table IF NOT EXISTS genre("
          "idGenre INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE not null, "
          "genreFilm varchar(50) not null; ;")
c.execute("create table IF NOT EXISTS acteur("
          "idActeur INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE not null, "
          "nomActeur varchar(50) not null, "
          "prenomActeur varchar(50) not null); ;")
c.execute("create table IF NOT EXISTS film("
          "idFilm INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE not null, "
          "titre varchar(50) not null, "
          "duree int(11) not null, "
          "idGenre INTEGER, "
          "idActeur INTEGER, "
          "FOREIGN KEY(idGenre) REFERENCES genre(idGenre), "
          "FOREIGN KEY(idActeur) REFERENCES acteur(idActeur)); ;")

# Insert a row of data

c.execute( "insert into genre(genreFilm)values('Comedie');")
c.execute( "insert into genre(genreFilm)values('Drame');")
c.execute( "insert into genre(genreFilm)values('Policier');")
c.execute( "insert into genre(genreFilm)values('Western');")
c.execute( "insert into genre(genreFilm)values('Science-fiction');")
c.execute( "insert into genre(genreFilm)values('Dessin animé');")
c.execute( "insert into genre(genreFilm)values('Aventure');")

c.execute( "insert into acteur(nomActeur,prenomActeur)values('Ford','Harrison');")
c.execute( "insert into acteur(nomActeur,prenomActeur)values('Weaver','Sigourney');")
c.execute( "insert into acteur(nomActeur,prenomActeur)values('Travolta','John');")
c.execute( "insert into acteur(nomActeur,prenomActeur)values('Smith','Will');")
c.execute( "insert into acteur(nomActeur,prenomActeur)values('Marceau','Sophie');")
c.execute( "insert into acteur(nomActeur,prenomActeur)values('Snipes','Wesley');")
c.execute( "insert into acteur(nomActeur,prenomActeur)values('Lhermitte','Thierry');")
c.execute( "insert into acteur(nomActeur,prenomActeur)values('Roberts','Julia');")
c.execute( "insert into acteur(nomActeur,prenomActeur)values('Di Caprio','Leonardo');")
c.execute( "insert into acteur(nomActeur,prenomActeur)values('Gable','Clark');")
c.execute( "insert into acteur(nomActeur,prenomActeur)values('Russel','Kurt');")
c.execute( "insert into acteur(nomActeur,prenomActeur)values('Houston','Angelica');")

c.execute( "insert into film(titre,duree,idGenre,idActeur)values('Star wars 4','117','5','1');")
c.execute( "insert into film(titre,duree,idGenre,idActeur)values('Star wars 5','122','5','1');")
c.execute( "insert into film(titre,duree,idGenre,idActeur)values('Alien 3','110','5','2');")
c.execute( "insert into film(titre,duree,idGenre,idActeur)values('Allo Maman, ici Bebe','92','1','3');")
c.execute( "insert into film(titre,duree,idGenre,idActeur)values('Bad Boys','113','3','4');")
c.execute( "insert into film(titre,duree,idGenre,idActeur)values('La Famille Addams','95','1','12');")
c.execute( "insert into film(titre,duree,idGenre,idActeur)values('La Fille de d'artagnan','124','7','5');")
c.execute( "insert into film(titre,duree,idGenre,idActeur)values('Blade','116','5','6');")
c.execute( "insert into film(titre,duree,idGenre,idActeur)values('Le Pere Noel est une Ordure','90','1','7');")
c.execute( "insert into film(titre,duree,idGenre,idActeur)values('New York 1997','99','5','11');")
c.execute( "insert into film(titre,duree,idGenre,idActeur)values('Pretty Woman','114','1','8');")
c.execute( "insert into film(titre,duree,idGenre,idActeur)values('Titanic','187','7','9');")
c.execute( "insert into film(titre,duree,idGenre,idActeur)values('Just Married','132','1','8');")
c.execute( "insert into film(titre,duree,idGenre,idActeur)values('Autant en emporte le vent','175','7','10');")
c.execute( "insert into film(titre,duree,idGenre,idActeur)values('Indiana Jones','183','7','1');")

# Save (commit) the changes
conn.commit()
conn.close() #close the connection.

```

Utilisation de la base videotheque

```

import sqlite3
conn = sqlite3.connect('videotheque.db')

def requete(req):
    print("-----")
    c = conn.cursor()
    c.execute(req)
    rows = c.fetchall()
    for row in rows:
        print(row)

#Affichage tous les films d'Harisson Ford
requete("select titre from acteur,film where acteur.idActeur=film.idActeur and nomActeur='Ford';")

#Affichage de tous les films qui ont une durée supérieur à 120 minutes ?
requete("select titre from film where duree>120;")

#Affichage de tous les films de science-fiction
requete("select titre from genre,film where genre.idGenre=film.idGenre and genreFilm='Science-fiction';")

#Affichage de tous les films d'aventures d'Harisson Ford ?
requete("select titre from genre,film,acteur where genre.idGenre=film.idGenre and acteur.idActeur=film.idActeur and genreFilm='Aventure' and nomActeur='Ford';")

#Affichage de tous les films dans l'ordre décroissant de durée
requete("select titre,duree from film order by duree desc;")

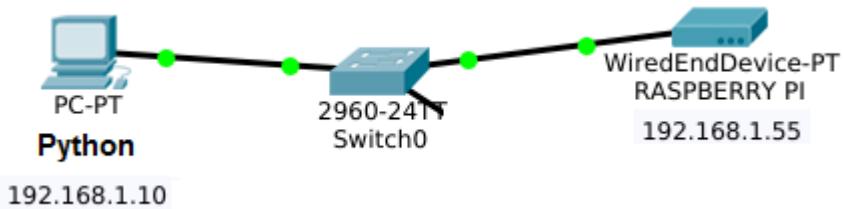
conn.close()

```

Résultats dans la console :

----- ('Star wars 4',) ('Star wars 5',) ('Indiana Jones',) ----- ('Star wars 5',) ('La Fille de d artagnan',) ('Titanic',) ('Just Married',) ('Autant en emporte le vent',) ('Indiana Jones',) ----- ('Star wars 4',) ('Star wars 5',) ('Alien 3',) ('Blade',) ('New York 1997',)	----- ('Indiana Jones',) ----- ('Titanic', 187) ('Indiana Jones', 183) ('Autant en emporte le vent', 175) ('Just Married', 132) ('La Fille de d artagnan', 124) ('Star wars 5', 122) ('Star wars 4', 117) ('Blade', 116) ('Pretty Woman', 114) ('Bad Boys', 113) ('Alien 3', 110) ('New York 1997', 99) ('La Famille Addams', 95) ('Allo Maman, ici Bebe', 92) ('Le Pere Noel est une Ordure', 90)
---	---

Comment interroger une base de donnée dans un Raspberry PI depuis un PC distant avec mysql.connector ?



Un Raspberry pi installé avec la suite logicielle LAMP : Linux Apache MySql Php

```
sudo apt-get install apache2 php5 mysql-server phpmyadmin
```

https://philippe.ddns.net/documentation/Raspberry_pi/

Dans phpmyadmin, un compte **user1**, mot de passe **toto** avec les droits administrateurs

The screenshot shows the phpMyAdmin interface. The left sidebar shows a database structure with "localhost:3306" selected. The main area displays the "user" table from the "mysql" database. The table has columns: Host, User, Password, and Select_priv. The data shows:

Host	User	Password	Select_priv
%	user1	*1844F2B11CCAEF3B31F573A1384F608BB6DE3DF9	Y
localhost	phpmyadmin	*1844F2B11CCAEF3B31F573A1384F608BB6DE3DF9	N
localhost	root	*1844F2B11CCAEF3B31F573A1384F608BB6DE3DF9	Y
%	user2	*63D85DCA15EAFFC58C908FD2FAE50CCBC60C4EA2	N

Le serveur doit pouvoir être accessible depuis un client

```
pi@raspberrypi:~ $ nano /etc/mysql/mariadb.conf.d/50-server.cnf
#bind-address      = 127.0.0.1 (mettre cette ligne en commentaire)
```

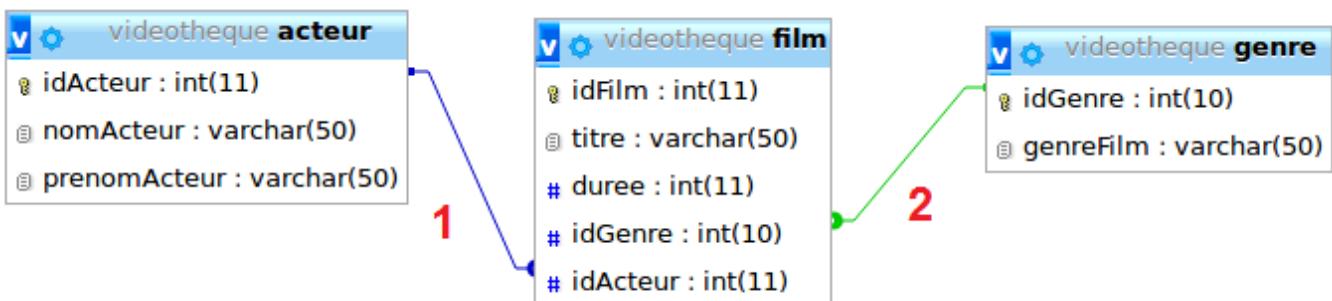
Requete sql avec mysql.connector

	Instructions
Réquête SQL	<pre>import mysql.connector db = mysql.connector.connect(user='user1', password='toto', host='192.168.1.55', database='videotheque') # Create a Cursor object to execute queries. cur = db.cursor() # Select data from table using SQL query. cur.execute("select titre from genre,film where genre.idGenre=film.idGenre and genreFilm='Science-fiction';") for row in cur.fetchall(): print (row[0])</pre>
Résultat	Star wars 1 Star wars 2 Alien 3 blade new york 1997

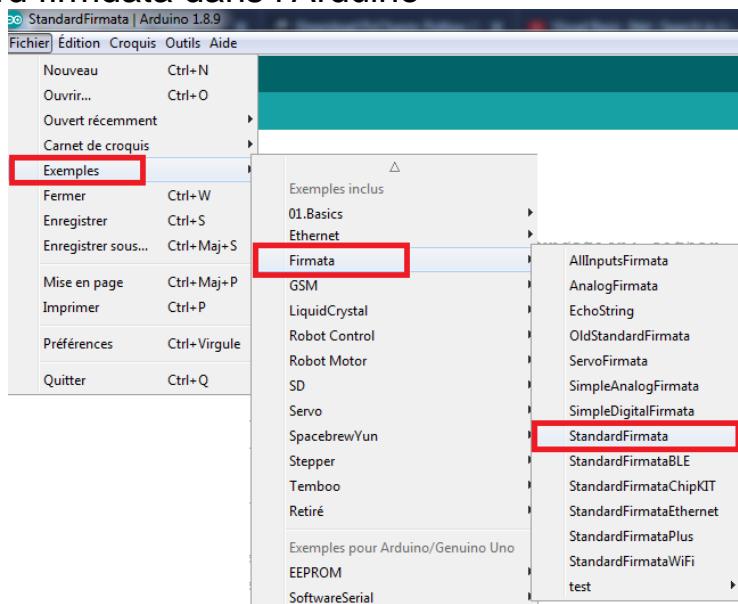
Un convertisseur sqlite to mysql est disponible ici

<https://www.rebasedata.com/convert-sqlite-to-mysql-online>

Il faudra refaire les liens dans phpmyadmin à l'aide du concepteur de vues.



Charger le standard firmata dans l'Arduino



Instructions

```
# script pour communication "arduino"
# (C) Gaillard Cédric
# nécessite firmadata standard sur arduino

from tkinter import *
import random
import pyfirmata2

broche = 13 # Pin 13 is used
port = pyfirmata2.Arduino.AUTO_DETECT
# Creer une carte
uno = pyfirmata2.Arduino(port)

def mettreUN():
    uno.digital[broche].write(1)

def mettreZERO():
    uno.digital[broche].write(0)
# fenêtre principale
fenetre = Tk()

fenetre.title('Arduino')
fenetre.geometry('300x300')

# Bouton mise à 0
bouton0 = Button(fenetre, text='0', height=2, width=3, fg='BLACK',
command=mettreZERO)
bouton0.pack(side=LEFT, padx=50, pady=10)

# Bouton mise à 1
bouton1 = Button(fenetre, text='1', height=2, width=3, fg='BLACK', command=mettreUN)
bouton1.pack(side=LEFT, padx=10, pady=10)

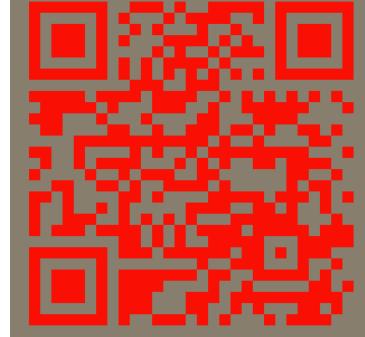
# Label
label_titre = Label(fenetre, text='LED13 arduino', bg='#F0F0F0', fg='RED')
label_titre.pack(side=LEFT, padx=10, pady=50)

fenetre.mainloop()
```



Génération d'un QRcode

<https://pypi.org/project/qrcode/>

	Instructions	Résultat affiché
QRcode(1)	<pre>import qrcode img = qrcode.make('https://touchard- washington.paysdelaloire.e-lyco.fr/') img.show() img.save("qrcode.png")</pre>	
QRcode(2)	<pre>import qrcode qr = qrcode.QRCode(version=1, error_correction=qrcode.constants.ERROR_CORRECT_L, box_size=10, border=5,) qr.add_data('https://touchard- washington.paysdelaloire.e-lyco.fr/') qr.make(fit=True) img = qr.make_image(fill_color="red", back_color="gray") img.show()</pre>	

Décodage d'un QRcode

<https://pypi.org/project/xqrcode/>

	Instructions	Résultat affiché
QRcode(3)	<pre>import xqrcode results = xqrcode.decode_from_file('qrcode.png') print(results) print(results[0]['data']) [{'type': 'QRCODE', 'data': 'https://touchard-washington.paysdelaloire.e-lyco.fr/'} https://touchard-washington.paysdelaloire.e-lyco.fr/]</pre>	

<https://opencv-python-tutorials.readthedocs.io/en/latest/index.html>

	Instructions	Résultat affiché
OpenCV(1)	<pre>import numpy as np import cv2 # Load an color image in grayscale img = cv2.imread('lighthouse.jpg',0) # Load an color image in color #img = cv2.imread('lighthouse.jpg',1) cv2.imshow('image',img) k = cv2.waitKey(0) if k == 27: # wait for ESC key to exit cv2.destroyAllWindows()</pre>	

	Instructions	Résultat affiché
OpenCV(2)	<pre>#https://matplotlib.org/2.0.2/Matplotlib.pdf import numpy as np import cv2 from matplotlib import pyplot as plt img = cv2.imread('lighthouse.jpg',0) plt.imshow(img, cmap = 'magma', interpolation = 'bicubic') #plt.imshow(img, cmap = 'gray', interpolation = 'bicubic') plt.xticks([]), plt.yticks([]) # to hide tick values on X and Y axis plt.show()</pre>	

https://docs.opencv.org/3.1.0/d7/d1b/group_imgproc_misct.html

	Instructions	Résultat affiché
OpenCV(3)	<pre>import numpy as np import cv2 cap = cv2.VideoCapture(0) while(True): # Capture frame-by-frame ret, frame = cap.read() # Our operations on the frame come here #gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) color = cv2.cvtColor(frame, 0) # Display the resulting frame #cv2.imshow('frame',gray) cv2.imshow('frame', color) if cv2.waitKey(1) & 0xFF == ord('q'): break # When everything done, release the capture cap.release() cv2.destroyAllWindows()</pre>	

https://docs.opencv.org/3.1.0/d1/db7/tutorial_py_histogram_begins.html

	Instructions	Résultat affiché
OpenCV(4)	<pre> import cv2 import numpy as np from matplotlib import pyplot as plt img = cv2.imread('lighthouse.jpg') color = ('b','g','r') for i,col in enumerate(color): histr = cv2.calcHist([img],[i],None,[256],[0,256]) plt.plot(histr,color = col) plt.xlim([0,256]) plt.show() </pre>	
OpenCV(5)	<pre> # https://docs.opencv.org/3.1.0/d7/d4d/tutorial_py_thresholding.html import cv2 import numpy as np from matplotlib import pyplot as plt img = cv2.imread('lighthouse.jpg',0) img = cv2.medianBlur(img,5) ret,th1 = cv2.threshold(img,127,255, cv2.THRESH_BINARY) th2 = cv2.adaptiveThreshold(img,255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY,11,2) th3 = cv2.adaptiveThreshold(img,255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY,11,2) titles = ['Original Image', 'Global Thresholding (v = 127)', 'Adaptive Mean Thresholding', 'Adaptive Gaussian Thresholding'] images = [img, th1, th2, th3] for i in range(4): plt.subplot(2,2,i+1),plt.imshow(images[i],'gray') plt.title(titles[i]) plt.xticks([]),plt.yticks([]) plt.show() </pre>	<p>Original Image</p> <p>Global Thresholding ($v = 127$)</p> <p>Adaptive Mean Thresholding</p> <p>Adaptive Gaussian Thresholding</p>

Ethernet : Client pour serveur industriel type Sollae

https://python.developpez.com/cours/TutoSwinnen/?page=page_20#L18.3

<https://pymotw.com/2/socket/tcp.html>

<https://www.eztcp.com/en/home/>

Client (provisoire)

```
# Définition d'un client réseau rudimentaire
# Ce client dialogue avec un serveur ad hoc

import socket,sys

HOST="192.168.1.77"
PORT=23

mySocket=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
try:
    print("connexion avec un module sollae (telnet)")
    mySocket.connect((HOST,PORT))
except socket.error:
    print("la connexion a échoué")
    sys.exit()
print("connexion établie avec le serveur transfert des données")
mySocket.send("hello serveur".encode("Utf8"))

while 1:
    msgServeur = mySocket.recv(1024) #attente des données du serveur
    if msgServeur.upper() == "FIN\r".encode("Utf8"):
        break
    print("S>", msgServeur)
    msgClient = input("C> ")
    mySocket.send(msgClient.encode("Utf8"))
print("Connexion interrompue.")
mySocket.close()
```

S : serveur , C : client

connexion avec un module sollae (telnet)

connexion établie avec le serveur transfert des données

S> b'le sollae envoie des donnees\r'

C> ok j'ai bien reçu les données

S> b'fin\r'

Connexion interrompue.

Serveur (provisoire)

```

# Définition d'un serveur réseau rudimentaire
# Ce serveur attend la connexion d'un client, pour entamer un dialogue avec lui
import socket, sys

HOST = '192.168.1.13'
PORT = 23

# 1) création du socket :
mySocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# 2) liaison du socket à une adresse précise :
try:
    mySocket.bind((HOST, PORT))
except socket.error:
    print ("La liaison du socket à l'adresse choisie a échoué.")
    sys.exit()

while 1:
    # 3) Attente de la requête de connexion d'un client :
    print ("Serveur prêt, en attente de requêtes ...")
    mySocket.listen(5)

    # 4) Etablissement de la connexion :
    connexion, adresse = mySocket.accept()
    print ("Client connecté, adresse IP %s, port %s" % (adresse[0], adresse[1]))

    # 5) Dialogue avec le client :
    connexion.send("Vous êtes connecté au serveur Marcel. Envoyez vos messages.".encode("Utf8"))
    msgClient = connexion.recv(1024)
    while 1:
        print ("C>", msgClient)
        if msgClient.upper() == "FIN\r\n".encode("Utf8") or msgClient == "":
            break
        msgServeur = input("S> ")
        connexion.send(msgServeur.encode("Utf8"))
        msgClient = connexion.recv(1024)

    # 6) Fermeture de la connexion :
    connexion.send("Au revoir !".encode("Utf8"))
    print ("Connexion interrompue.")
    connexion.close()

    ch = input("<R>ecommencer <T>erminer ? ")
    if ch.upper() == 'T':
        break

```

Test avec putty comme client telnet
 Serveur prêt, en attente de requêtes ...
 Client connecté, adresse IP 192.168.1.21, port 47491
 S> bonjour
 C> b'ok\r\n'
 S> a bientot
 C> b'fin\r\n'
 Connexion interrompue.
 <R>ecommencer <T>erminer ?

Serveur web en python

Serveur

```
import http.server

PORT = 8888
server_address = ("", PORT)

server = http.server.HTTPServer
handler = http.server.CGIHTTPRequestHandler
handler.cgi_directories = ["/"]
print("Serveur actif sur le port :", PORT)

httpd = server(server_address, handler)
httpd.serve_forever()
```

Page web

```
# coding: utf-8
# A la racine du projet

import cgi

form = cgi.FieldStorage()
print("Content-type: text/html; charset=utf-8\n")

print(form.getvalue("name"))

html = """<!DOCTYPE html>
<head>
    <title>Mon programme</title>
</head>
<body>
    <form action="/index.py" method="post">
        <input type="text" name="name" value="Votre nom" />
        <input type="submit" name="send" value="Envoyer information au serveur">
    </form>
</body>
</html>
"""

print(html)
```

Résultat

The screenshot shows a web browser window with the following elements:

- Header bar with back, forward, refresh, and home icons.
- Address bar showing the URL: localhost:8888/index.py.
- Form fields:
 - A text input field labeled "Votre nom" containing the text "bidochon".
 - A button labeled "Envoyer information au serveur" (Send information to the server).

<https://www.eclipse.org/paho/clients/python/>

<http://www.steves-internet-guide.com/publishing-messages-mqtt-client/>

<https://shiftr.io/>

<https://github.com/f4goh/MQTT-Tutoriel>

Exemple : Publish et subscribe

```
import paho.mqtt.client as mqtt

# The callback for when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
    print(msg.topic+" "+str(msg.payload))

client = mqtt.Client("python") #id must be unique
client.username_pw_set("weatherSensors", "bme280Sensors") #login, password
client.on_message = on_message

client.connect("broker.shiftr.io", 1883, 60)

client.subscribe("/sensors/temperature")

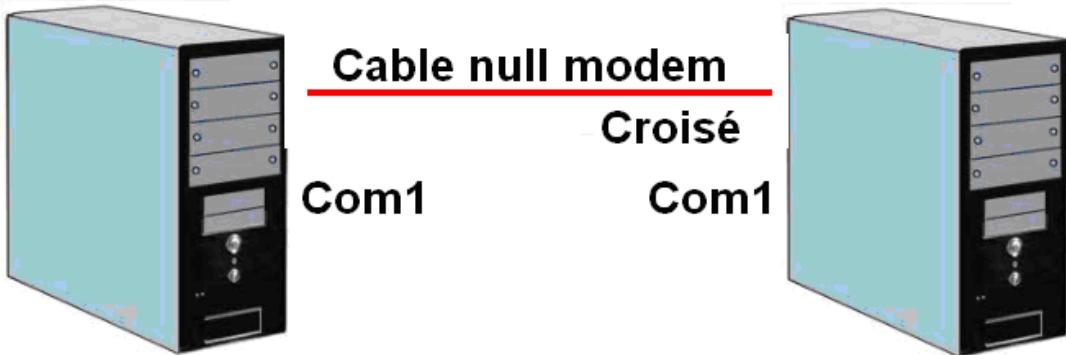
temp=0
while temp !=-1:
    temp=int(input("valeur de la temperature positive "))
    if temp>=0:
        client.publish("/sensors/temperature", temp)
print("fin de connection")
client.connected_flag=False
client.disconnect_flag=True
```

valeur de la temperature positive 30
valeur de la temperature positive 15
valeur de la temperature positive -1
fin de connection

sensors/temperature
20
python - 15:03:58 - 00 NR



Liaison série (envoi et réception de caractères)



```
import serial
ser = serial.Serial('com1', 9600, timeout=0) # open serial port
#ser = serial.Serial('/dev/ttyUSB0') # open serial port
print("le port est ouvert ?", ser.is_open)
print(ser.name) # check which port was really used
ser.write(b'hello') # write a hello string
packet = bytearray()
packet.append(0x41)
packet.append(0x42)
packet.append(0x43)
ser.write(packet) #write ABC
ser.close() # close port
```

```
import serial
ser = serial.Serial('com1', 9600, timeout=None) # open serial port
#ser = serial.Serial('/dev/ttyUSB0') # open serial port
print("le port est ouvert ?", ser.is_open)
print(ser.name) # check which port was really used
x = ser.read() # read one byte
s = ser.read(10) # read up to ten bytes (timeout)
print(x,s)
ser.close()
```

le port est ouvert ? True

com1

b'f' b'0123456789'

<https://pythonhosted.org/pyserial/>

	Instructions
Trie par sélection	<pre> import random N = 7 BIG_N = N*N*N x=[] def init_liste(): for i in range(0, N): x.append(random.randint(0,BIG_N)) print("Tri par selection : ") init_liste(); print("Avant tri : ",x) compteur = 0 for i in range(0,N): i_min= i for j in range(i+1,N): if(x[j]<x[i_min]): i_min=j if (i != i_min): k=x[i] x[i]=x[i_min] x[i_min]=k print (x[i_min], '<-->', x[i], ' : ', x) compteur+=1 print("Apres tri croissant : ", x) print("Nombre d'echanges : " , compteur) </pre>
Résultat	<p>Tri par selection :</p> <p>Avant tri : [128, 291, 9, 57, 107, 330, 5]</p> <p>128 <--> 5 : [5, 291, 9, 57, 107, 330, 128]</p> <p>291 <--> 9 : [5, 9, 291, 57, 107, 330, 128]</p> <p>291 <--> 57 : [5, 9, 57, 291, 107, 330, 128]</p> <p>291 <--> 107 : [5, 9, 57, 107, 291, 330, 128]</p> <p>291 <--> 128 : [5, 9, 57, 107, 128, 330, 291]</p> <p>330 <--> 291 : [5, 9, 57, 107, 128, 291, 330]</p> <p>Apres tri croissant : [5, 9, 57, 107, 128, 291, 330]</p> <p>Nombre d'echanges : 6</p>

	Instructions
Trie par insertion	<pre> import random N = 7 BIG_N = N*N*N x=[] def init_liste(): for i in range(0, N): x.append(random.randint(0,BIG_N)) print("Tri par insertion : ") init_liste(); print("Avant tri : ",x) compteur = 0 for i in range(0,N): k=x[i] j=i while j>0 and k<x[j-1]: x[j]=x[j-1] j-=1 x[j]=k print(i+1,"premieres valeurs triees: ",x) print("Apres tri croissant : ", x) </pre>
Résultat	<p>Tri par insertion :</p> <p>Avant tri : [235, 130, 86, 199, 90, 179, 303]</p> <p>1 premières valeurs triées: [235, 130, 86, 199, 90, 179, 303]</p> <p>2 premières valeurs triées: [130, 235, 86, 199, 90, 179, 303]</p> <p>3 premières valeurs triées: [86, 130, 235, 199, 90, 179, 303]</p> <p>4 premières valeurs triées: [86, 130, 199, 235, 90, 179, 303]</p> <p>5 premières valeurs triées: [86, 90, 130, 199, 235, 179, 303]</p> <p>6 premières valeurs triées: [86, 90, 130, 179, 199, 235, 303]</p> <p>7 premières valeurs triées: [86, 90, 130, 179, 199, 235, 303]</p> <p>Apres tri croissant : [86, 90, 130, 179, 199, 235, 303]</p>
Version récursive	<pre> print("Tri par insertion version recursive : ") init_liste(); print("Avant tri : ",x) def insere(t,j): if j>0 and t[j]<t[j-1]: t[j-1],t[j]=t[j],t[j-1] insere(t,j-1) def tri_insertion(t,j=1): if j<len(t): insere(t,j) tri_insertion(t,j+1) tri_insertion(x) </pre>

	Instructions
Trie à bulles	<pre> import random N = 7 BIG_N = N*N*N x= [] def init_liste(): for i in range(0, N): x.append(random.randint(0,BIG_N)) print("Tri à bulles :") init_liste(); print("Avant tri : ",x) compteur = 0 permutation=True while permutation==True: permutation=False compteur+=1 for i in range(0,N-compteur): if x[i]>x[i+1]: permutation=True x[i],x[i+1]=x[i+1],x[i] print("tri en cours : ",x) print("Apres tri croissant : ", x) print("Nombre passages : " , compteur) </pre>
Résultat	<p>Tri à bulles :</p> <p>Avant tri : [180, 285, 129, 249, 99, 130, 100]</p> <p>tri en cours : [180, 129, 249, 99, 130, 100, 285]</p> <p>tri en cours : [129, 180, 99, 130, 100, 249, 285]</p> <p>tri en cours : [129, 99, 130, 100, 180, 249, 285]</p> <p>tri en cours : [99, 129, 100, 130, 180, 249, 285]</p> <p>tri en cours : [99, 100, 129, 130, 180, 249, 285]</p> <p>tri en cours : [99, 100, 129, 130, 180, 249, 285]</p> <p>Apres tri croissant : [99, 100, 129, 130, 180, 249, 285]</p> <p>Nombre passages : 6</p>

Recherche dichotomique

Recherche dichotomique	<p>Instructions</p> <pre>#On suppose que l'élément recherché se trouve dans la liste def recherche_dichotomique_recursive(element, liste_triee): if len(liste_triee)==1 : return 0 m = len(liste_triee)//2 if liste_triee[m] == element : return m elif liste_triee[m] > element : return recherche_dichotomique_recursive(element, liste_triee[:m]) else : return m + recherche_dichotomique_recursive(element, liste_triee[m:]) def recherche_dichotomique(element, liste_triee) : a = 0 b = len(liste_triee)-1 m = (a+b)//2 while a < b : if liste_triee[m] == element : return m elif liste_triee[m] > element : b = m-1 else : a = m+1 m = (a+b)//2 return a list=[1,5,12,45,201,452,987] # 0 1 2 3 4 5 6 print("valeur à l'indice ", recherche_dichotomique(45,list)) print("valeur à l'indice ",recherche_dichotomique_recursive(45,list))</pre>
Résultat	valeur à l'indice 3 valeur à l'indice 3

POO

Manque le diagramme de classe ici

```
from math import sqrt

class Point:
    def __init__(self, _x, _y):
        self.x = _x
        self.y = _y

    def get_x(self):
        return self.x

    def get_y(self):
        return self.y

    def module(self):
        return sqrt(self.x**2+self.y**2)

    def __str__(self):
        return "({},{})".format(self.x, self.y)

p=Point(3,7)
print(p)
print("le point ({}, {}) a pour module {:.3g}".format(p.get_x(), p.get_y(), p.module()))
```

(3,7)

le point (3,7) a pour module 7.62

POO et héritage

Manque les diagrammes de classe ici

Classe mère	Classe fille
<pre>class Compte_bancaire: def __init__(self): self.solde=0 def Deposer(self,somme): self.solde+=somme def Retirer(self,retrait): retrait_accept=False if (self.solde>retrait): self.solde-=retrait retrait_accept=True return retrait_accept def Obtenir_solde(self): return self.solde def __str__(self): return "{}".format(self.solde)</pre>	<pre>class Compte_courant(Compte_bancaire): def __init__(self): super().__init__() self.decouvert_autorise=0 def Changer_decouvert(self,nouveau_decouvert): self.decouvert_autorise=nouveau_decouvert def Retirer(self,retrait): #surcharge retrait_accept=False if (self.solde+self.decouvert_autorise>=retrait): self.solde-=retrait retrait_accept=True return retrait_accept</pre>
<pre>compte=Compte_bancaire() compte.Deposer(100) print(compte.Obtenir_solde()) print(compte) print(compte.Retirer(50)) print(compte) print(compte.Retirer(150)) print(compte)</pre>	<pre>compte=Compte_courant() compte.Deposer(100) print(compte) compte.Chaenger_decouvert(500) print(compte.Retirer(150)) print(compte)</pre>
100 100 True 50 False 50	100 True -50

Récursivité

	Instructions	Résultat affiché sur la console
factoriel	<pre>def fact(n): f = 1 for i in range(n): f = f * (i+1) return(f) print(fact(5))</pre>	120
PGCD	<pre>def pgcd(a, b): if (b == 0): return (a) else: return (pgcd(b, a % b)) print(pgcd(27,18))</pre>	9
Decimal en binaire	<pre>def dec2bin(n): if n == 0: return '0' else: return dec2bin(n//2) + str(n%2) print(dec2bin(15))</pre>	01111
Binaire en, decimal	<pre>def bin2dec(n,poids): if n == 0: return 0 else: return (n%2)*2**poids+bin2dec(n//10,poids+1) print(bin2dec(1010,0))</pre>	10
Fibonacci	<pre>#1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, etc... def Fibonacci(n): if n<=1: return 1 else: return Fibonacci(n-1) + Fibonacci(n-2); print(Fibonacci(10))</pre>	89

	Instructions	Résultat
romain	<pre> def valeur(c): nombre={ 'M' : 1000, 'D' : 500, 'C' : 100, 'L' : 50, 'X' : 10, 'V' : : 5, 'I' : 1} return nombre[c] def calcul(chaine): if len(chaine)==1: return valeur(chaine[0]) else: if valeur(chaine[0])>=valeur(chaine[1]): return (valeur(chaine[0]))+calcul(chaine[1:]) else: return calcul(chaine[1])-valeur(chaine[0]) print(calcul('MDLIC')) </pre>	1649
suite	<pre> #f(1) = 3, #f(n+1) = f(n) + 3 def mult(n): if n == 1: return 3 else: return mult(n-1) + 3 for i in range(1,10): print(i,mult(i)) </pre>	1 3 2 6 3 9 4 12 5 15 6 18 7 21 8 24 9 27
Triangle de Pascal	<pre> def pascal(col, row): if (col == 0) or (col == row): return 1 else: return pascal(col - 1, row - 1) + pascal(col, row - 1) def Triangle_de_Pascal(num): for r in range(num): for c in range(r + 1): print(str(pascal(c, r)) ,end=' ') print() Triangle_de_Pascal(10) </pre>	
reverse	<pre> def reverse(a): if len(a) == 0: return a else: return reverse(a[1:]) + a[0] print(reverse("Bonjour,comment allez-vous ?")) </pre>	? suov-zella tnemmoc,ru ojnoB
somme	<pre> def somme(lst): if len(lst) == 1: return lst[0] else: return lst[0] + somme(lst[1:]) a = [2, 7, 5, 8, 9] print(somme(a)) </pre>	31

Jeux d'essai

	Instructions	Résultat
Calcul simple	<pre>def calcul(x): y=2*x return y print("pass test1",calcul(3)==6) print("pass test2",calcul(2)!=5)</pre>	pass test1 True pass test2 True
Recherche de caractère	<pre>def rechercher_caractere(chaine,car): trouve=False for c in chaine: if (c==car): trouve=True return trouve print("pass test1",rechercher_caractere('bonjour','o')==True) print("pass test1",rechercher_caractere('hello','a')==False)</pre>	pass test1 True pass test2 True
Somme	<pre>def somme(lst): if len(lst) == 1: return lst[0] else: return lst[0] + somme(lst[1:]) print("pass test1",somme([2, 7, 5, 8, 9])==31)</pre>	pass test1 True
Calcul formel	<pre>from sympy import * x = Symbol('x') f=(x + 3)**2 g=(f.expand()) print(g) v=limit(sin(x)/x, x, 0) print(v) derivee=diff((x + 3)**2, x) print(derivee) integrale=integrate(derivee, x) print(integrale)</pre>	<p>https://docs.sympy.org/latest/tutorial/calculus.html</p> <p> $x^{**2} + 6*x + 9$ 1 $2*x + 6$ $x^{**2} + 6*x$ </p>

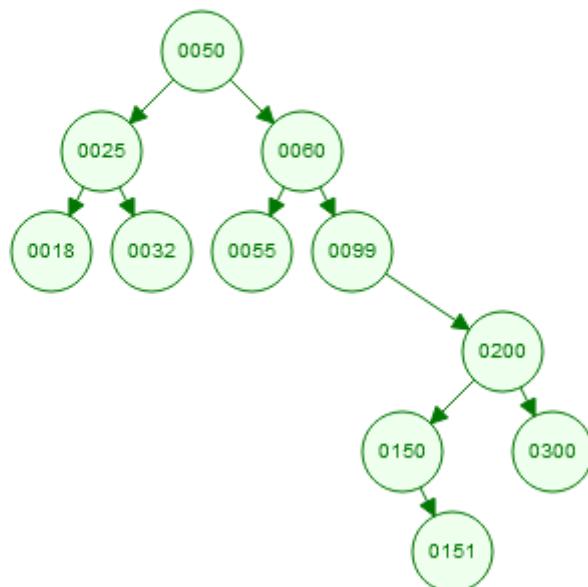
Les arbres binaires

<https://github.com/joowani/binarytree>

Mode automatique	Mode manuel
<pre>from binarytree import tree, bst, heap my_bst = bst(height=3, is_perfect=True) print(my_bst) print(my_bst.properties)</pre>	<pre>from binarytree import Node root = Node(8) root.left = Node(3) root.right = Node(10) root.right.right = Node(14) root.left.left = Node(1) root.left.right = Node(6) root.right.right.left = Node(13) root.left.right.left = Node(4) root.left.right.right = Node(7) print(root)</pre>
<pre> graph TD 7 --- 3 7 --- 11 3 --- 1 3 --- 5 11 --- 9 11 --- 13 1 --- 0 1 --- 2 5 --- 4 5 --- 6 9 --- 8 9 --- 10 13 --- 12 13 --- 14 </pre>	<pre> graph TD 8 --- 3 8 --- 10 3 --- 1 3 --- 6 10 --- 14 1 --- 4 1 --- 7 6 --- 13 </pre>
<pre>{'height': 3, 'size': 15, 'is_max_heap': False, 'is_min_heap': False, 'is_perfect': True, 'is_strict': True, 'is_complete': True, 'leaf_count': 8, 'min_node_value': 0, 'max_node_value': 14, 'min_leaf_depth': 3, 'max_leaf_depth': 3, 'is_bst': True, 'is_balanced': True}</pre>	

Binary Search Tree

Insert
Delete
Find
Print



<https://www.cs.usfca.edu/~galles/visualization/BST.html>

Parcours préfixe, suffixe, infixe

Instructions	suite	
<pre> class Noeud: def __init__(self, _valeur): self.valeur = _valeur self.filsGauche = None self.filsDroit = None def creerNoeud(_valeur): n = Noeud(_valeur) n.filsGauche = None; n.filsDroit = None; return n def inserer(a, n): if a.valeur > n: if a.filsGauche == None: a.filsGauche = creerNoeud(n) else: inserer(a.filsGauche, n) else: if a.filsDroit == None: a.filsDroit = creerNoeud(n) else: inserer(a.filsDroit, n); def afficherInfixe(a): if a.filsGauche != None: afficherInfixe(a.filsGauche) print(a.valeur) if a.filsDroit != None: afficherInfixe(a.filsDroit) def afficherPrefixe(a): print(a.valeur) if a.filsGauche != None: afficherPrefixe(a.filsGauche) if a.filsDroit != None: afficherPrefixe(a.filsDroit) def afficherSuffixe(a): if a.filsGauche != None: afficherPrefixe(a.filsGauche) if a.filsDroit != None: afficherPrefixe(a.filsDroit) print(a.valeur) </pre>	<pre> a = creerNoeud(50) inserer(a, 25) inserer(a, 60) inserer(a, 18) inserer(a, 32) inserer(a, 55) inserer(a, 99) inserer(a, 200) inserer(a, 150) inserer(a, 300) inserer(a, 151) </pre>	
	<pre>print("Infixe :")</pre>	
	<pre>afficherInfixe(a)</pre>	
	<pre>print("Prefixe :")</pre>	
	<pre>afficherPrefixe(a)</pre>	
	<pre>print("Suffixe :")</pre>	
	<pre>afficherSuffixe(a)</pre>	
Infixe :	Prefixe :	Suffixe :
18	50	25
25	25	18
32	18	32
50	32	60
55	60	55
60	55	99
99	99	200
150	200	150
151	150	151
200	151	300
300	300	50

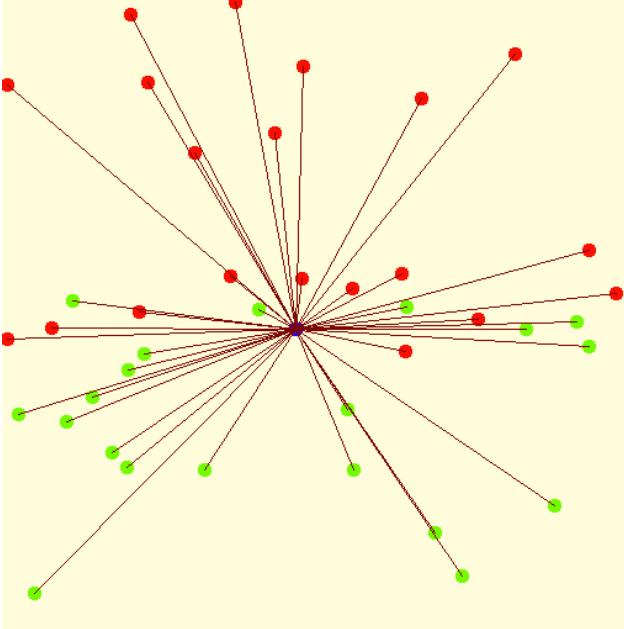
Minimum, maximum, hauteur, nb nœuds

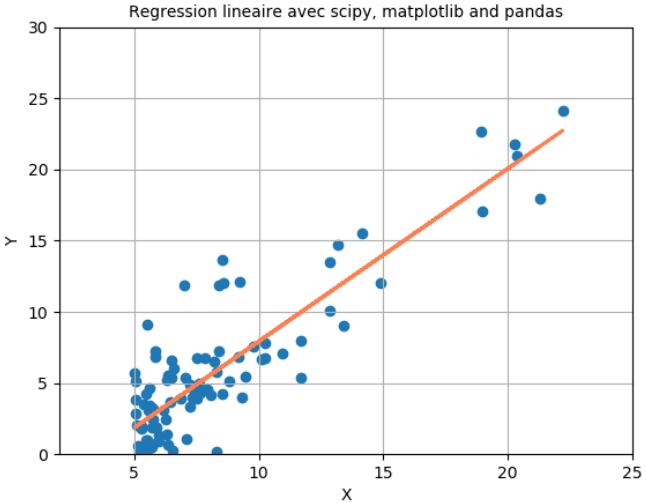
Instructions		
Minimum, maximum, hauteur, nb nœuds	<pre> def minimum(a): if a.filsGauche != None: return minimum(a.filsGauche) else: return a.valeur def maximum(a): if a.filsDroit != None: return maximum(a.filsDroit) else: return a.valeur def hauteur(a): if a == None: return -1 else: hg = hauteur(a.filsGauche) hd = hauteur(a.filsDroit) if (hg > hd): return hg + 1 else: return hd + 1 def nbnoeuds(a): if (a != None): return 1 + nbnoeuds(a.filsGauche) + nbnoeuds(a.filsDroit) else: return 0 def etqPresente(a, etq): if a == None: return False else: return a.valeur == etq or etqPresente(a.filsGauche, etq) or etqPresente(a.filsDroit, etq) def nboc(a, etq): if a == None: return 0 else: if a.valeur == etq: res = 1 else: res = 0 res = res + nboc(a.filsGauche, etq) + nboc(a.filsDroit, etq) return res </pre>	
	<pre> print("minimum :", hauteur(a)) print("maximum :", maximum(a)) print("nb noeuds :", nbnoeuds(a)) print("hauteur :", hauteur(a)) print(etqPresente(a, 160)) print("nb occurrences :", nboc(a, 200)) </pre>	minimum : 5 maximum : 300 nb noeuds : 11 hauteur : 5 False nb occurrences : 1

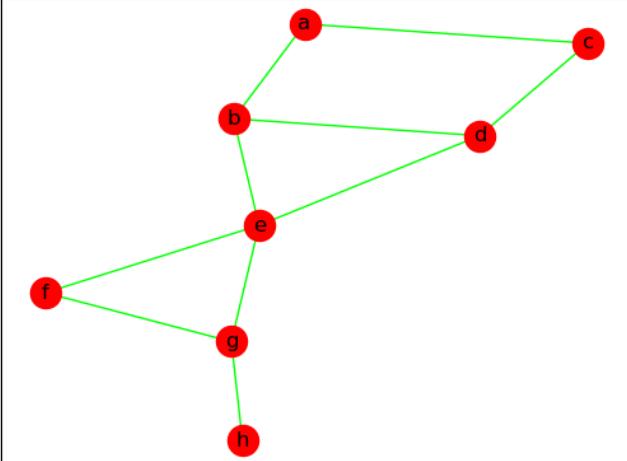
Trier une table à l'aide d'un arbre binaire

	Instructions
Trie	<pre> def remplirTableau(t, a): if a.filsGauche!=None: remplirTableau(t,a.filsGauche); t.append(a.valeur) if a.filsDroit!=None: remplirTableau(t,a.filsDroit); def trier(t): a = creerNoeud(t[0]) for n in range(1,len(t)): inserer(a,t[n]) t[:]=[] n=remplirTableau(t,a) tableau=[] for i in range(0,10): tableau.append(random.randint(0,100)) print(tableau) trier(tableau); print(tableau) </pre>
Résultat	<pre>[18, 38, 96, 90, 61, 47, 9, 76, 44, 20] [9, 18, 20, 38, 44, 47, 61, 76, 90, 96]</pre>

	Instructions
k-NN	<pre> from PIL import Image, ImageDraw, ImageFont import random from math import sqrt MaxVoisins=20 #nb de voisins max par couleur K=5 #nb de voisins plus proches à vérifier < MaxVoisins #definition du code couleur en lien avec RGB et le texte couleurVoisinRouge=(0,(255,0,0),'rouge') couleurVoisinVert=(1,(0,255,0),'vert') couleurintrus=(0,0,255) #tracé d'un cercle def cercle(centreX,centreY,rayon,couleur): draw = ImageDraw.Draw(img) draw.ellipse((centreX - rayon, centreY - rayon, centreX + rayon, centreY + rayon), fill=couleur) #tracé d'une ligne def ligne(x1,y1,x2,y2): draw = ImageDraw.Draw(img) draw.line((x1, y1, x2, y2), fill=128) #génère un voisin de dimension MaxVoisins def creationVoisin(tableau,debut,fin): for nb in range(MaxVoisins): x = random.randint(0, largeur) y = random.randint(int(debut * hauteur), int(fin * hauteur)) tableau.append((x,y)) return tableau #dessine le voisin def afficheVoisin(tableau,couleur): for nb in range(MaxVoisins): cercle(tableau[nb][0],tableau[nb][1],5,couleur) #génère un intrus def creationintrus(debut,fin): x = random.randint(int(debut * largeur), int(fin * largeur)) y = random.randint(int(debut * hauteur), int(fin * hauteur)) return x,y #affiche un intrus def afficheintrus(i,couleur): cercle(i[0], i[1], 5, couleur) def getKey(item): return item[1] #calcule la distance entre 2 pts def calculDistance(tableauDistance,tableauVoisin,intrus,codeCouleur): for n in range(MaxVoisins): tableauDistance.append((codeCouleur,int(sqrt(pow(tableauVoisin[n][0]-intrus[0],2)+pow(tableauVoisin[n][1]-intrus[1],2))))) ligne(tableauVoisin[n][0],tableauVoisin[n][1],intrus[0],intrus[1]) </pre>

	Instructions
k-NN(suite)	<pre> img=Image.new('RGB', (512,512), (255,255,255)) largeur,hauteur=img.size voisinRouge=[] creationVoisin(voisinRouge,0,0.6) #entre 0 et 60% de la hauteur de l'image print('coord voisin 1',voisinRouge) afficheVoisin(voisinRouge,couleurVoisinRouge[1]) voisinVert=[] creationVoisin(voisinVert,0.3,1) #entre 30% et 100% de la hauteur de l'image print('coord voisin 2',voisinVert) afficheVoisin(voisinVert,couleurVoisinVert[1]) intrus=creationintrus(0.4,0.6) #entre 40% et 60% de la hauteur de l'image print('coord intrus',intrus) afficheintrus(intrus,couleurintrus) tableauDistance=[] calculDistance(tableauDistance,voisinRouge,intrus,couleurVoisinRouge[0]) calculDistance(tableauDistance,voisinVert,intrus,couleurVoisinVert[0]) tableauDistance=sorted(tableauDistance,key=getKey) print(tableauDistance) sommeCodeCouleur1=0 sommeCodeCouleur2=0 for indice in range(K): if tableauDistance[indice][0]==0: sommeCodeCouleur1+=1 if tableauDistance[indice][0]==1: sommeCodeCouleur2+=1 print('K=',K,',',couleurVoisinRouge[2],':',sommeCodeCouleur1,',',couleurVoisinVert[2],':',sommeCodeCouleur2) print("la couleur de l'intrus sera : ",end=' ') if(sommeCodeCouleur1>sommeCodeCouleur2): print(couleurVoisinRouge[2]) else: print(couleurVoisinVert[2]) img.show() </pre>
Résultat	 <pre> coord voisin 1 [(324, 222), (221, 108), (185, 224), (498, 238), (476, 203), (4, 69), (327, 285), (156, 124), (340, 80), (40, 266), (243, 226), (118, 67), (4, 275), (284, 234), (416, 44), (111, 253), (244, 54), (189, 2), (386, 259), (104, 12)] coord voisin 2 [(52, 342), (328, 249), (73, 322), (208, 251), (280, 332), (89, 367), (26, 481), (373, 467), (102, 300), (285, 381), (476, 281), (101, 379), (425, 267), (164, 381), (448, 410), (351, 432), (57, 244), (466, 261), (115, 287), (13, 336)] coord intrus (238, 267) exemple : (1, 34) : couleur vert à une distance de 34 (0, 41) : couleur rouge à une distance de 41 [(1, 34), (0, 41), (0, 56), (0, 68), (1, 77), (0, 90), (1, 91), (0, 97), (1, 123), (1, 124), (0, 127), (1, 135), (1, 139), (0, 148), (0, 159), (0, 164), (1, 173), (1, 176), (1, 179), (1, 182), (1, 187), (0, 198), (1, 199), (1, 200), (0, 213), (0, 213), (1, 228), (0, 233), (0, 234), (1, 235), (1, 238), (1, 241), (0, 246), (1, 254), (0, 261), (0, 269), (0, 285), (0, 288), (1, 301), (0, 306)] K= 5 , rouge : 3 , vert : 2 la couleur de l'intrus sera: rouge </pre>

Régression linéaire	Instructions <pre>#https://mrmint.fr/regression-lineaire-python-pratique import pandas as pd import matplotlib.pyplot as plt from scipy import stats #slope : coefficient directeur #intercept : ordonnée à l'origine def predict(x): return slope * x + intercept df = pd.read_csv("univariate_linear_regression_dataset.csv") print ('nombre de valeurs dans le fichier csv',len(df)) X = df.iloc[0:len(df),0] #selection de la première colonne de notre dataset Y = df.iloc[0:len(df),1] axes = plt.axes() axes.set_xlim([2, 25]) #redimensionnement des axes axes.set_ylim([0, 30]) axes.grid() #affichage d'une grille plt.scatter(X,Y) #tracé des points #https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.linregress.html #calcul des coefficients slope, intercept, r_value, p_value, std_err = stats.linregress(X, Y) #calcul du tableau de valeurs à l'aide des coefs slope et intercept droite = predict(X) plt.plot(X, droite, color='coral', linewidth=2) plt.title("Regression lineaire avec scipy, matplotlib and pandas", fontsize=10) plt.xlabel('X') plt.ylabel('Y') plt.savefig("regression_lineaire.png") print('-----') print('Y=',slope,'* X +',intercept) valeur=20.27 print('Y =',predict(valeur),' pour X =',valeur) plt.show()</pre>
Résultat	 <pre>nombre de valeurs dans le fichier csv 96 ----- Y= 1.213547253908358 * X + - 4.211504005424086 Y = 20.38709883129833 pour X = 20.27</pre>

Instructions	
Graph non orienté (1) <pre>#https://networkx.github.io/documentation/stable/tutorial.html import networkx as nx import matplotlib.pyplot as plt g=nx.Graph()#non orienté for n in range(8): g.add_node(chr(ord('a')+n)) #ou bien """ g.add_node('a') g.add_node('b') g.add_node('c') g.add_node('d') g.add_node('e') g.add_node('f') g.add_node('g') g.add_node('h') """ g.add_edge('a', 'b') g.add_edge('a', 'c') g.add_edge('b', 'd') g.add_edge('d', 'c') g.add_edge('b', 'e') g.add_edge('d', 'e') g.add_edge('e', 'f') g.add_edge('f', 'g') g.add_edge('g', 'e') g.add_edge('g', 'h') matrice=nx.adjacency_matrix(g) print(matrice.todense()) print(nx.info(g)) nx.draw_networkx(g,node_color='#FF0000',edge_color='#00FF00') # displays the node and edge plt.show() # displays the networkx graph on matplotlib canvas</pre>	
Résultat	 <pre>[[0 1 1 0 0 0 0 0] [1 0 0 1 1 0 0 0] [1 0 0 1 0 0 0 0] [0 1 1 0 1 0 0 0] [0 1 0 1 0 1 1 0] [0 0 0 0 1 0 1 0] [0 0 0 0 1 1 0 1] [0 0 0 0 0 0 1 0]] Name: Type: Graph Number of nodes: 8 Number of edges: 10 Average degree: 2.5000</pre>

Instructions

Graph orienté (2)

```
#https://networkx.github.io/documentation/stable/tutorial.html
import networkx as nx
import matplotlib.pyplot as plt
g=nx.DiGraph() #orienté

g.add_node('a', pos=(1, 8))
g.add_node('b', pos=(0, 6))
g.add_node('c', pos=(0.5, 6))
g.add_node('d', pos=(2, 6))
g.add_node('e', pos=(0, 3))
g.add_node('f', pos=(1, 4))
g.add_node('g', pos=(1, 1))

g.add_edge('a', 'b', w=9)
g.add_edge('a', 'c', w=20)
g.add_edge('a', 'd', w=17)
g.add_edge('c', 'f', w=40)
g.add_edge('f', 'g', w=8)
g.add_edge('d', 'g', w=1)
g.add_edge('b', 'e', w=30)
g.add_edge('e', 'f', w=2)

matrice=nx.adjacency_matrix(g, weight='w')
print(matrice.todense())

for edge in g.edges(data=True):
    print(edge)

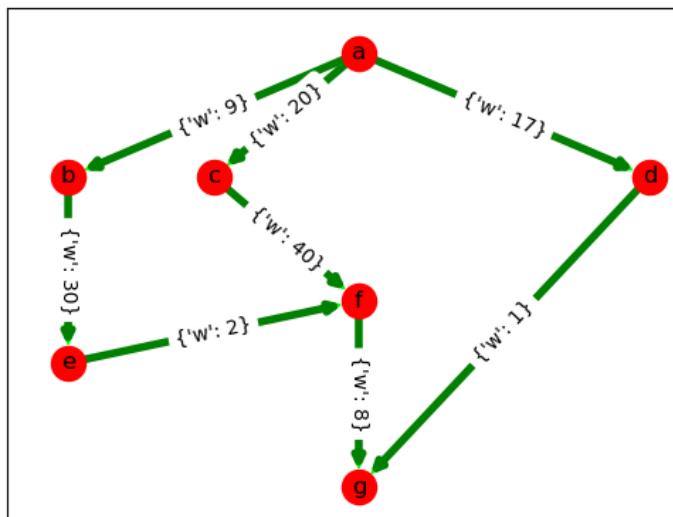
print("plus court chemin",nx.dijkstra_path(g, 'a', 'g', weight='w'))

print(nx.info(g))
pos = nx.spring_layout(g)
pos=nx.get_node_attributes(g, 'pos')

nx.draw_networkx(g, pos=pos, node_color="#FF0000", edge_color="#00FF00")
nx.draw_networkx_edge_labels(g, pos=pos)
nx.draw_networkx_edges(g, pos, width=4, edge_color='g', arrows=True)

plt.show() # displays graph on matplotlib canvas
```

Résultat



```
[[ 0  9 20 17  0  0  0]
 [ 0  0  0  0 30  0  0]
 [ 0  0  0  0  0 40  0]
 [ 0  0  0  0  0  0  1]
 [ 0  0  0  0  0  0  2]
 [ 0  0  0  0  0  0  8]
 [ 0  0  0  0  0  0  0]]
(['a', 'b', {'w': 9}], ['a', 'c', {'w': 20}], ['a', 'd', {'w': 17}], ['b', 'e', {'w': 30}], ['c', 'f', {'w': 40}], ['d', 'g', {'w': 1}], ['e', 'f', {'w': 2}], ['f', 'g', {'w': 8}])
plus court chemin ['a', 'd', 'g']
Name:
Type: DiGraph
Number of nodes: 7
Number of edges: 8
Average in degree: 1.1429
Average out degree: 1.1429
```

	Instructions
Graphé (3)	<pre>#génération à partir d'une matrice import networkx as nx import numpy as np import matplotlib.pyplot as plt A = [[0, 100, 0, 0, 40, 0], [100, 0, 20, 0, 0, 70], [0, 20, 0, 80, 50, 0], [0, 0, 80, 0, 0, 30], [40, 0, 50, 0, 0, 60], [0, 70, 0, 30, 60, 0]] a = np.array(A) g = nx.from_numpy_matrix(a) for edge in g.edges(data=True): print(edge) print("plus court chemin",nx.dijkstra_path(g, 0, 3,weight='weight')) print(nx.info(g)) pos = nx.spring_layout(g) nx.draw_networkx(g, pos=pos, node_color='#FF0000', edge_color='#00FF00') nx.draw_networkx_edge_labels(g, pos=pos) nx.draw_networkx_edges(g, pos, width=4, edge_color='g', arrows=True) plt.show() # displays the networkx graph on matplotlib canvas</pre>
Résultat	<pre>(0, 1, {'weight': 100}) (0, 4, {'weight': 40}) (1, 2, {'weight': 20}) (1, 5, {'weight': 70}) (2, 3, {'weight': 80}) (2, 4, {'weight': 50}) (3, 5, {'weight': 30}) (4, 5, {'weight': 60}) plus court chemin [0, 4, 5, 3] Name: Type: Graph Number of nodes: 6 Number of edges: 8 Average degree: 2.6667</pre>

Algorithme de Boyer-Moorehttps://fr.wikipedia.org/wiki/Algorithme_de_Boyer-Moore

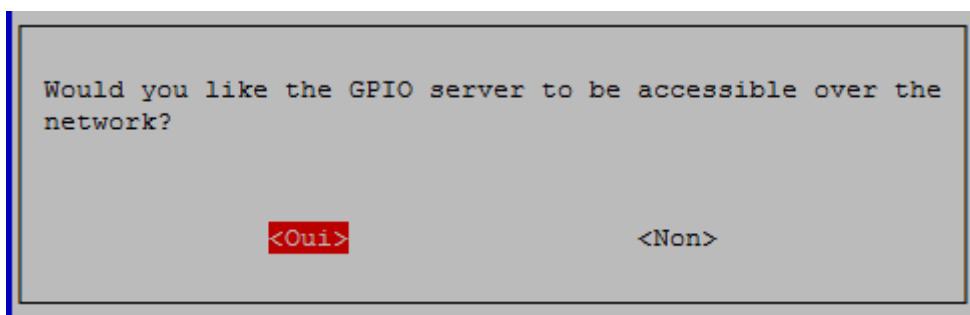
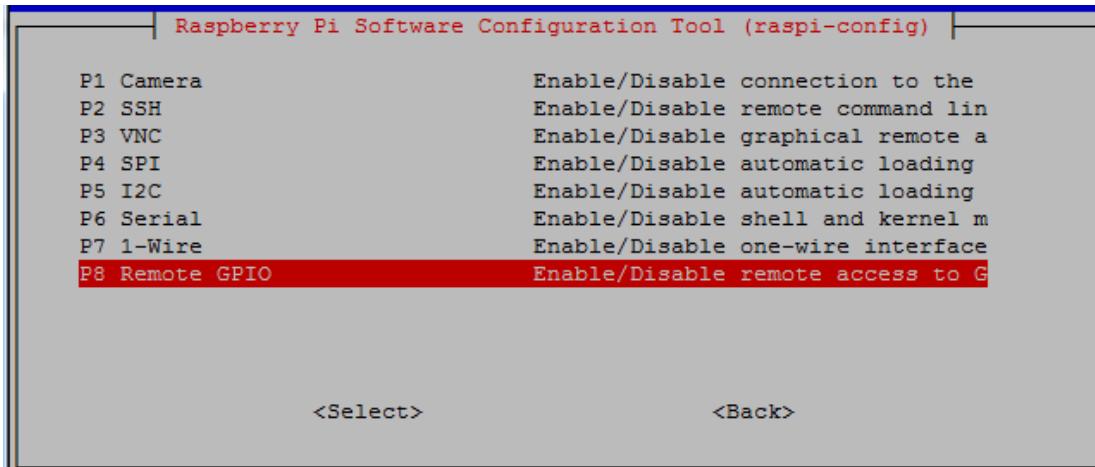
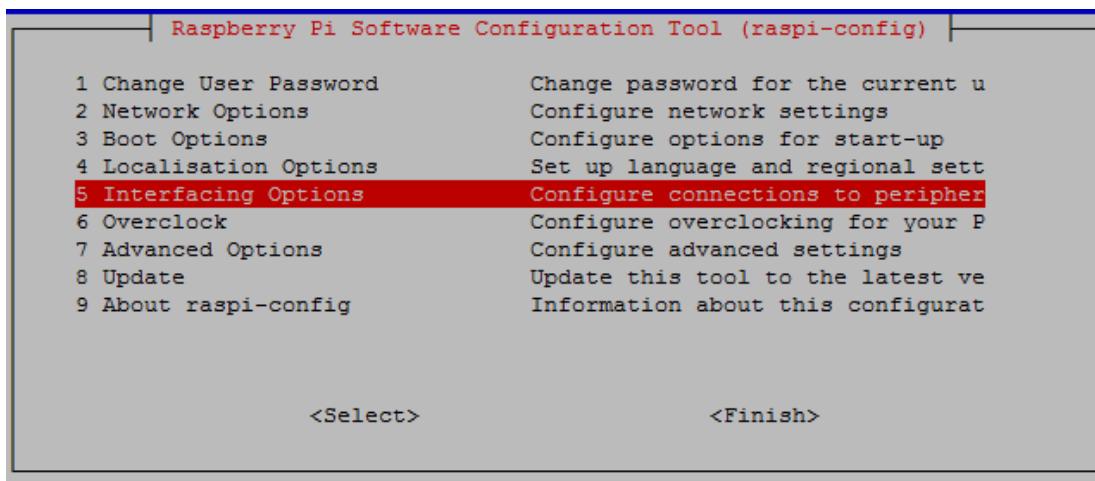
Boyer-Moore	<p>Instructions</p> <pre> occurrences = dict() def table(pattern, alphabet): for letter in alphabet: occurrences[letter] = pattern.rfind(letter) def last(letter): return occurrences[letter] def boyer_moore_match(text, pattern): """Find occurrence of pattern in text.""" alphabet = set(text) table(pattern, alphabet) m = len(pattern) n = len(text) i = m - 1 # text index j = m - 1 # pattern index while i < n: if text[i] == pattern[j]: if j == 0: return i else: i -= 1 j -= 1 else: #l = last(text[i]) l=occurrences[text[i]] i = i + m - min(j, 1 + l) j = m - 1 return -1 def show_match(text, pattern): print('Text: %s' % text) p = boyer_moore_match(text, pattern) print('Match: %s%s' % ('.' * p, pattern)) text = 'abacaababadcabacabaabb' pattern = 'abacab' show_match(text, pattern) text = 'Lorem ipsum dolor sit amet, empor invidunt ut labore et dolore magna aliquyam.' pattern = 'dolor' show_match(text, pattern) show_match(text, pattern + 'e') </pre>
Résultat	Text: abacaababadcabacabaabb Match:abacab Text: Lorem ipsum dolor sit amet, empor invidunt ut labore et dolore magna aliquyam. Match:dolor Text: Lorem ipsum dolor sit amet, empor invidunt ut labore et dolore magna aliquyam. Match:dolore

Avantages	Inconvénients
<ul style="list-style-type: none"> - Facile à mettre en service - Permet de commander les E/S très facilement 	<ul style="list-style-type: none"> - Lent (normal c'est du python remote) - Pas de support I²C (A ce jour) - Le code est exécuté sur le PC client

Installation coté Raspberry PI : https://gpiozero.readthedocs.io/en/stable/remote_gpio.html

En ligne de commande : (installé sur un Raspbian Stretch Lite, sans interface graphique)

pi@raspberrypi:~ \$ sudo raspi-config



Finish, puis Reboot

Installation du logiciel côté RPI :

```
$ sudo apt install pigpio
```

Lancer pigpiod au démarrage du Raspberry :

```
sudo systemctl enable pigpiod
```

ou manuellement

```
$ sudo pigpiod
```

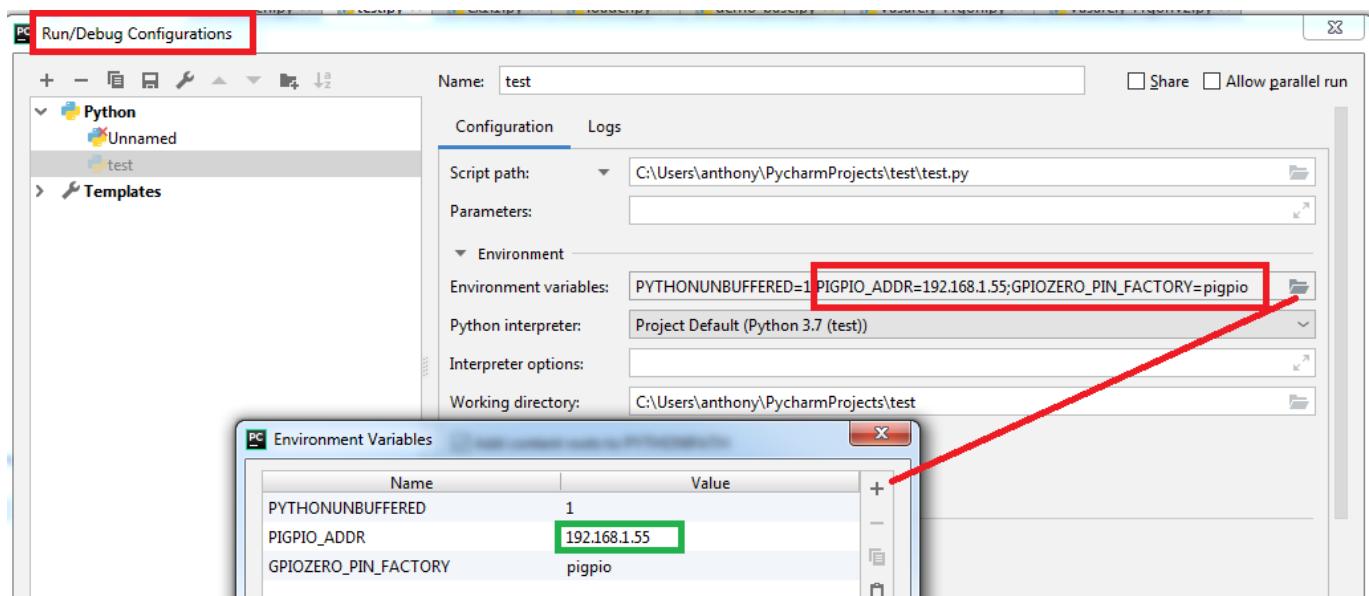
Prendre connaissance de l'adresse IP du RPI

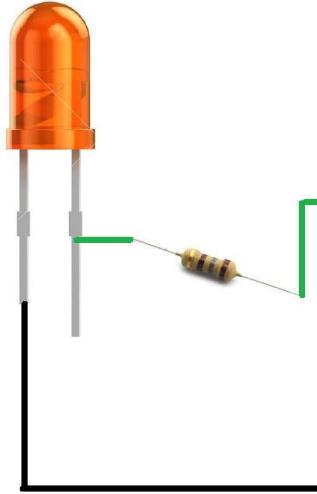
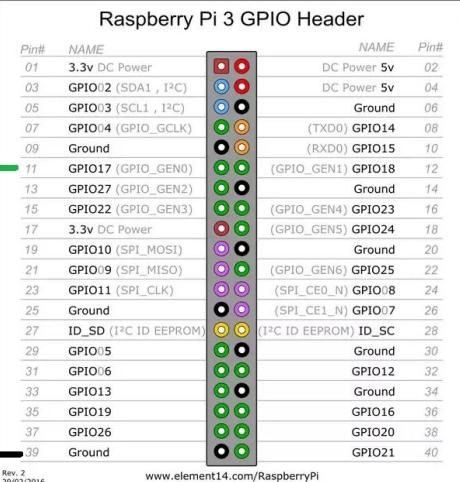
```
pi@raspberrypi:~ $ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.1.55 netmask 255.255.255.0 broadcast
      192.168.1.255
```

Configuration côté Client : (ici Pycharm)

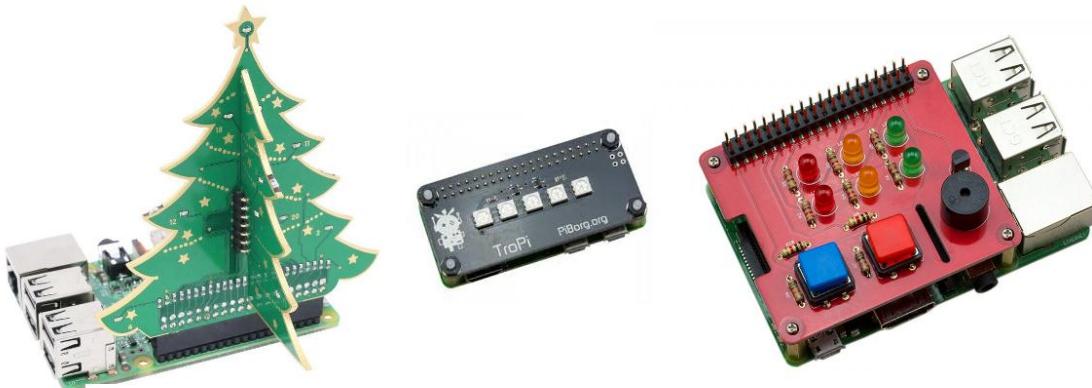
Installer gpiozero et pigpio (voir en dernière page)

Configurer les variables d'environnement du logiciel avec l'adresse IP de votre RPI



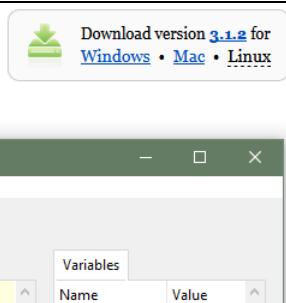
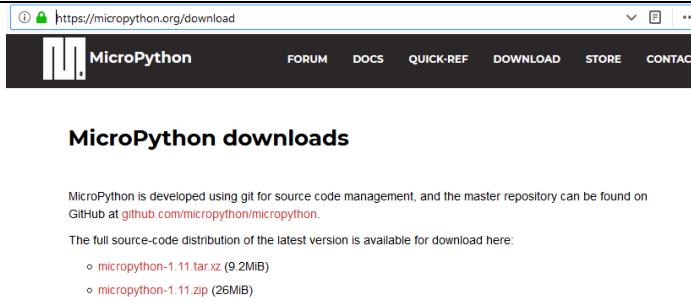
Instructions	GPIO header
<pre>#Cabler une led + résistance de 180 ohms en série sur la broche 11 (gpio17) from gpiozero import * from gpiozero import LED from time import sleep red = LED(17) print('{0}'.format(pi_info())) while True: red.on() sleep(1) red.off() sleep(1)</pre>	 
<pre>J8: 3V3 (1) (2) 5V GPIO2 (3) (4) 5V GPIO3 (5) (6) GND GPIO4 (7) (8) GPIO14 GND (9) (10) GPIO15 GPIO17 (11) (12) GPIO18 GPIO27 (13) (14) GND GPIO22 (15) (16) GPIO23 3V3 (17) (18) GPIO24 GPIO10 (19) (20) GND GPIO9 (21) (22) GPIO25 GPIO11 (23) (24) GPIO8 GND (25) (26) GPIO7 GPIO0 (27) (28) GPIO1 GPIO5 (29) (30) GND GPIO6 (31) (32) GPIO12 GPIO13 (33) (34) GND GPIO19 (35) (36) GPIO16 GPIO26 (37) (38) GPIO20 GND (39) (40) GPIO21</pre>	<pre>-----. oooooooooooooooooooooo J8 +=== 1ooooooooooooooooooooo USB +=== Pi Model 3B V1.2 +---+ +=== D SoC USB S +=== I +---+ C +=== S Net pwr HDMI I A +=== `- ----- ---- V -----' Revision : a02082 SoC : BCM2837 RAM : 1024Mb Storage : MicroSD USB ports : 4 (excluding power) Ethernet ports : 1 Wi-fi : True Bluetooth : True Camera ports (CSI) : 1 Display ports (DSI): 1</pre>

Les extensions RPI pour gpiozero sont les suivantes :



Le bus SPI est supporté, il est possible de câbler des interfaces de style MCPxxxx en attendant un update pour le bus I²C.

ESP8266 et Micropython

Editeur	Firmware
<p>https://thonny.org/ (Windows, linux installation via pip) ou https://codewith.mu/ (Windows, esp8266 non supporté sous linux)</p> <p>Thonny Python IDE for beginners</p> 	<p>https://micropython.org/download</p> 

Flasher le firmware dans le 8266 (<https://micropython.org/download#esp8266>)
Avec nodemcu firmware disponible ici : <https://github.com/f4goh/NSI>

<https://micropython.org/download#esp8266>

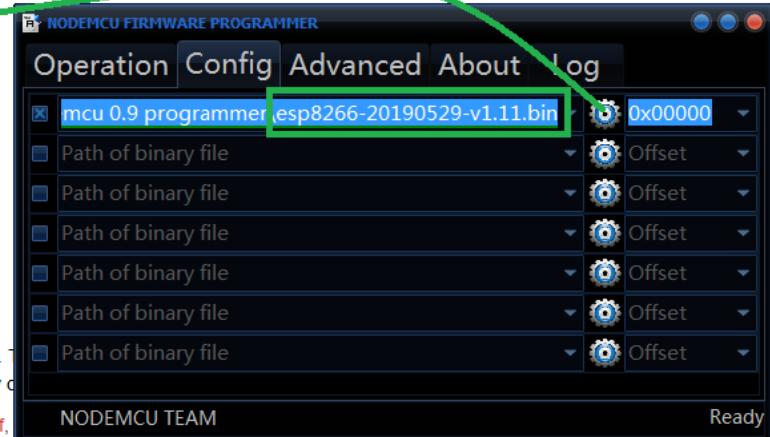
Firmware for ESP8266 boards

The following files are stable firmware for the ESP8266. Program your board using the esptool.py program as described in the tutorial.

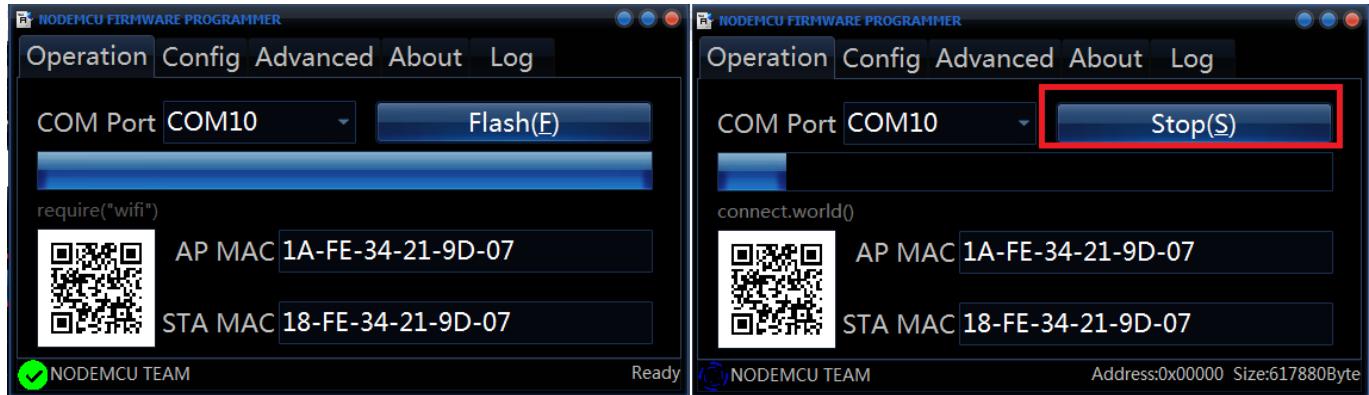
- o [esp8266-20190529-v1.11.bin](#) (elf, map) (latest)
- o [esp8266-20190125-v1.10.bin](#) (elf, map)
- o [esp8266-20180511-v1.9.4.bin](#) (elf, map)
- o [esp8266-20171101-v1.9.3.bin](#) (elf, map)
- o [esp8266-20170823-v1.9.2.bin](#) (elf, map)
- o [esp8266-20170612-v1.9.1.bin](#) (elf, map)
- o [esp8266-20170526-v1.9.bin](#) (elf, map)
- o [esp8266-20170108-v1.8.7.bin](#) (elf, map)

The following are daily builds of the ESP8266 firmware. They are not automatically started, and debugging is enabled by default.

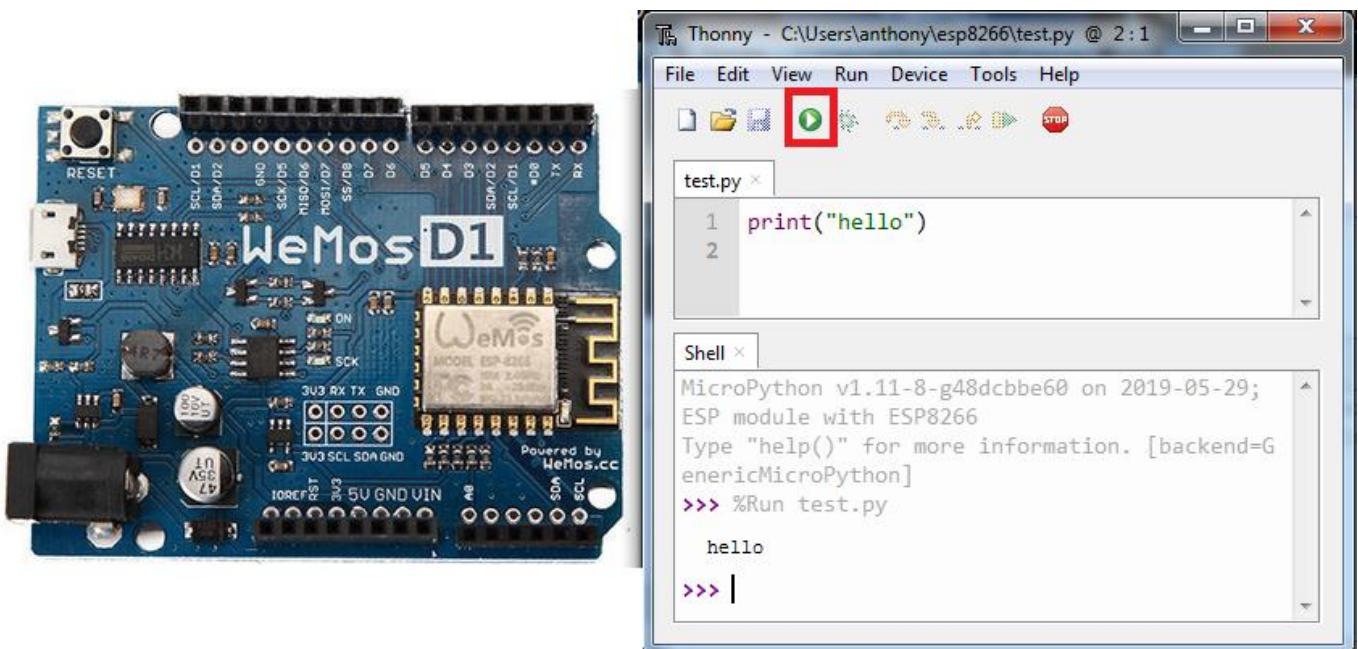
- o [esp8266-20190610-v1.11-37-g62f004ba4.bin](#) (elf, map)



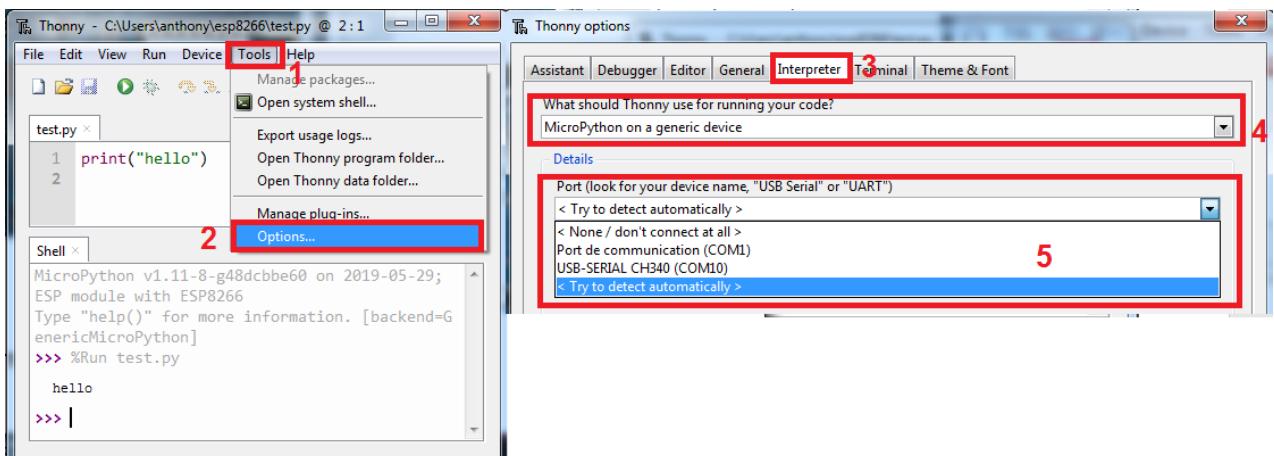
Puis bouton Flash.



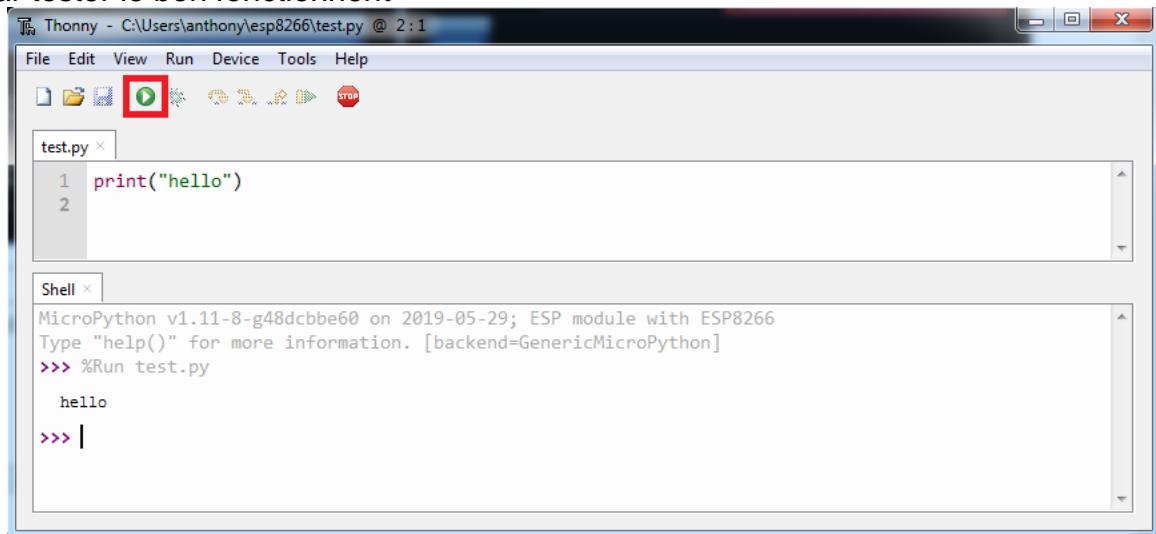
Faire un reset de la carte esp8266 **avant** de lancer Thonny



Configuration en mode esp8266



Finir par tester le bon fonctionnement



ESP32 et Micropython

Pour programmer le firmware de l'esp32, il faudra installer esptool sous linux en ligne de commande

<https://github.com/espressif/esptool>

```
sudo apt install python-pip
sudo pip install --upgrade pip
sudo pip install esptool
pip install pyserial
sudo pip install pyserial
```

Repérer le port série de la carte esp32

```
nsi@nsi-LIFEBOOK-A555:~$ ls /dev/tty*
/dev/tty17  /dev/tty32  /dev/tty48  /dev/tty63      /dev/ttyS2    /dev/ttyS7
/dev/tty18  /dev/tty33  /dev/tty49  /dev/tty7      /dev/ttyS20   /dev/ttys8
/dev/tty19  /dev/tty34  /dev/tty5   /dev/tty8      /dev/ttyS21   /dev/ttys9
/dev/tty2   /dev/tty35  /dev/tty50  /dev/tty9      /dev/ttyS22   /dev/ttys22
/dev/tty20  /dev/tty36  /dev/tty51  /dev/ttysprintk /dev/ttyS23
/dev/tty21  /dev/tty37  /dev/tty52  /dev/ttyS0     /dev/ttys24
/dev/tty22  /dev/tty38
/dev/tty53  /dev/ttyS1           /dev/ttyS25
```

Télécharger le firmware pour l'esp32 (<https://micropython.org/download#esp32>)

<p>Standard firmware:</p> <ul style="list-style-type: none"> o esp32-20190610-v1.11-37-g62f004ba4.bin (latest) o esp32-20190529-v1.11.bin o esp32-20190125-v1.10.bin o esp32-20180511-v1.9.4.bin o esp32--bluetooth.bin 	
--	---

Renommer le fichier par esp32.bin

Exécuter les 2 commandes suivantes :

```
python esptool.py --port /dev/ttyUSB0 erase_flash
python esptool.py --chip esp32 --port /dev/ttyUSB0 write_flash -z 0x1000 esp32.bin
```

Commandes afin de changer l'adresse MAC de l'ESP32 si nécessaire

```
python espefuse.py --port /dev/ttyUSB0 summary
python espefuse.py --port /dev/ttyUSB0 dump
python espefuse.py --port /dev/ttyUSB0 mac
python espefuse.py --port /dev/ttyUSB0 get_custom_mac
python espefuse.py --port /dev/ttyUSB0 burn_custom_mac de:ad:be:ef:fe:00
python espefuse.py --port /dev/ttyUSB0 get_custom_mac
```

Suite après la configuration du point d'accès

Installer un point d'accès Wifi RaspAP

<https://raspbian-france.fr/creer-un-hotspot-wi-fi-en-moins-de-10-minutes-avec-la-raspberry-pi/>

```
sudo cp /etc/wpa_supplicant/wpa_supplicant.conf /etc/wpa_supplicant/wpa_supplicant.conf.sav
sudo cp /dev/null /etc/wpa_supplicant/wpa_supplicant.conf
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

ajouter dans le fichier

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
```

Ctrl+o, ctrl+x : sauver, quitter

```
wget -q https://git.io/v0EUQ -O /tmp/raspap && bash /tmp/raspap
sudo reboot
```

IP address: 10.3.141.1

Username: **admin**

Password: **secret**

DHCP range: 10.3.141.50 to 10.3.141.255

SSID: raspi-webgui

Password: a definir dans le portail ci dessous

The screenshot shows the RaspAP WiFi Portal v1.5 interface. The left sidebar contains navigation links: Tableau de bord, Configurer client WiFi, Configurer hotspot, Configurer réseau, Configurer server DHCP, Configurer Auth, Change Thème, Data usage, Système, and About RaspAP. The main dashboard has a red header with the RaspAP logo. A green notification bar at the top says "Interface is up." Below it are four cards: "Informations d'interface" (Interface info), "Informations sans fil" (Wireless info), "Statistiques d'interface" (Interface stats), and "Appareils connectés" (Connected devices). The "Informations d'interface" card shows details like Nom de l'interface (wlan0), IPv4 Address (10.3.141.1), Masque de sous-réseau (255.255.255.0), and Adresse Mac (b8:27:eb:07:2a:81). The "Informations sans fil" card shows AP Mac Adresse (Not connected), Bitrate, Niveau du signal, Puissance de transmission (31.00 dBm), and Fréquence MHz. The "Statistiques d'interface" card shows Paquets reçus (1597), Octets reçus (222792 (217,57 KB)), Paquets transférés (2140), and Octets transférés (1903992 (1,82 MB)). The "Appareils connectés" card lists one device: Honor_6X (Adresse IP: 10.3.141.237, Adresse Mac: 44:c3:46:b8:d2).

Comment configurer le proxy (utile dans un lycée avec Raspbian)

<https://www.raspberrypi.org/documentation/configuration/use-a-proxy.md>

Configuration de raspbian via le terminal

sudo nano /etc/environment

```
export http_proxy="http://username:password@proxyipaddress:proxyport"
export https_proxy="http://username:password@proxyipaddress:proxyport"
export no_proxy="localhost, 127.0.0.1"
```

Pour que vos commandes via sudo gardent ces paramètres,

sudo visudo

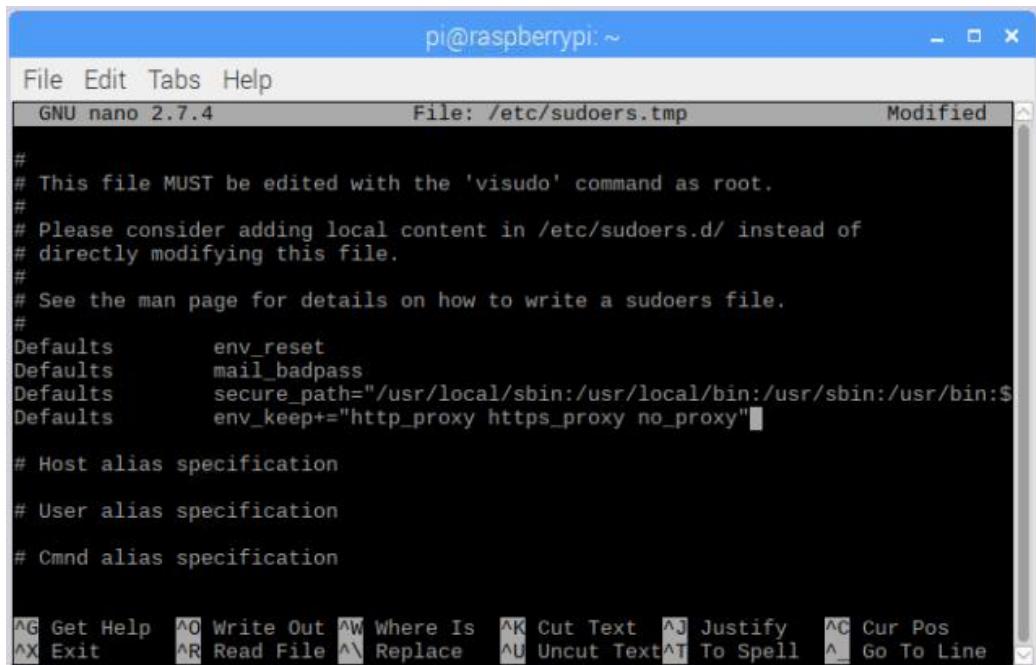
```
Defaults    env_keep+="http_proxy https_proxy no_proxy"
```

Pour que le système de package APT utilise le proxy, créer un fichier 10proxy dans /etc/apt/apt.conf.d/ :

```
cd /etc/apt/apt.conf.d
sudo nano 10proxy
```

```
Acquire::http::proxy "http://username:password@proxyipaddress:proxyport";
```

```
Acquire::https::proxy "http://username:password@proxyipaddress:proxyport";
```



```
pi@raspberrypi: ~
File Edit Tabs Help
GNU nano 2.7.4          File: /etc/sudoers.tmp          Modified
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults      env_reset
Defaults      mail_badpass
Defaults      secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:$
Defaults      env_keep+="http_proxy https_proxy no_proxy"

# Host alias specification

# User alias specification

# Cmnd alias specification

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos
^X Exit      ^R Read File  ^Y Replace   ^U Uncut Text  ^T To Spell  ^_ Go To Line
```

MicroPython ESP8266

<http://docs.micropython.org/en/v1.9.3/esp8266/esp8266/tutorial/index.html>

Quels sont les modules déjà intégrés ?

```

Shell ×
MicroPython v1.11-8-g48dcbe60 on 2019-05-29; ESP module with ESP8266
Type "help()" for more information. [backend=GenericMicroPython]
>>> help("modules")

__main__          http_client      socket           upip
_boot             http_client_ssl  ssd1306 ■       upip_utarfile
_onewire          http_server     ssl              upysh
_webrepl          http_server_ssl struct           urandom
apa102 ■         initsetup      sys              ure
array             io               time             urequests
binascii          json            uasyncio/_init_ urllib/urequest
btree             lwip            uasyncio/core   uselect
builtins          machine ■      ubinascii      usocket
collections       math            ucollections  ussl
dht ■            micropython    ucryptolib    ustruct
ds18x20 ■        neopixel ■    uctypes        utime
errno             network        uerrno        utimeq
esp               ntptime        uhashlib      uwebsocket
example_pub_button onewire       uio             webrepl
example_sub_led   os             ujson          webrepl_setup
flashbdev         port_diag     umqtt/robust ■ websocket_helper
framebuf          random        umqtt/simple  zlib
gc                re             uos
hashlib           select        Plus any modules on the filesystem
>>> |
```

En **rouge** le hardware géré en natif. On remarque le protocole mqtt intéressant pour la gestion des IOT.
En **vert** (machine). C'est le module qui gère les périphériques.

<http://docs.micropython.org/en/v1.9.3/esp8266/library/machine.html>

```
class Pin – control I/O pins
class Signal – control and sense external I/O devices
class UART – duplex serial communication bus
class SPI – a Serial Peripheral Interface bus protocol (master side)
class I2C – a two-wire serial protocol
class RTC – real time clock
class Timer – control hardware timers
class WDT – watchdog timer
```

Le reste à découvrir au fur et à mesure. Il est possible d'ajouter des modules en faisant :

<div style="border: 1px solid #ccc; padding: 5px; font-family: monospace;"> Device Tools Help <hr/> Soft reboot Ctrl+D Upload current script as main script Ctrl+U Unload current script as boot script Upload current script with current name Show device's main script Show device's boot script Close serial connection </div>	<p>A noter :</p> <p>L'esp8266 en mode micropython contient 2 fichiers importants.</p> <p>Boot.py : est l'équivalent du setup sur Arduino. (Programme exécuté au démarrage) Ensuite si l'esp contient un fichier main.py, alors celui-ci sera exécuté juste après le boot.py</p>
--	--

Configuration réseau avec le point d'accès wifi

http://docs.micropython.org/en/v1.9.3/esp8266/esp8266/tutorial/network_basics.html?highlight=sta_if%20connect

Dans la console :

```
>>> import network
>>> sta_if=network.WLAN(network.STA_IF)
>>> sta_if.active()
False

>>> sta_if.active(True)

#5 ets_task(4020f4d8, 28, 3fff9e30, 10) -> normal pas de connexion définie

>>> sta_if.connect('raspi-webgui','touchardNSI')
>>> sta_if.isconnected()
True

>>> sta_if.ifconfig()
('10.3.141.104', '255.255.255.0', '10.3.141.1', '10.3.141.1')
```

Après un Reset ou une remise sous tension :

```
>>> import network
>>> sta_if=network.WLAN(network.STA_IF)
>>> sta_if.isconnected()
True
```

Afficher l'adresse MAC :

```
>>> import network
>>> import ubinascii
>>> mac = ubinascii.hexlify(network.WLAN().config('mac'),':').decode()
>>> print(mac)
a0:20:a6:21:9d:fa
```

Remarque :

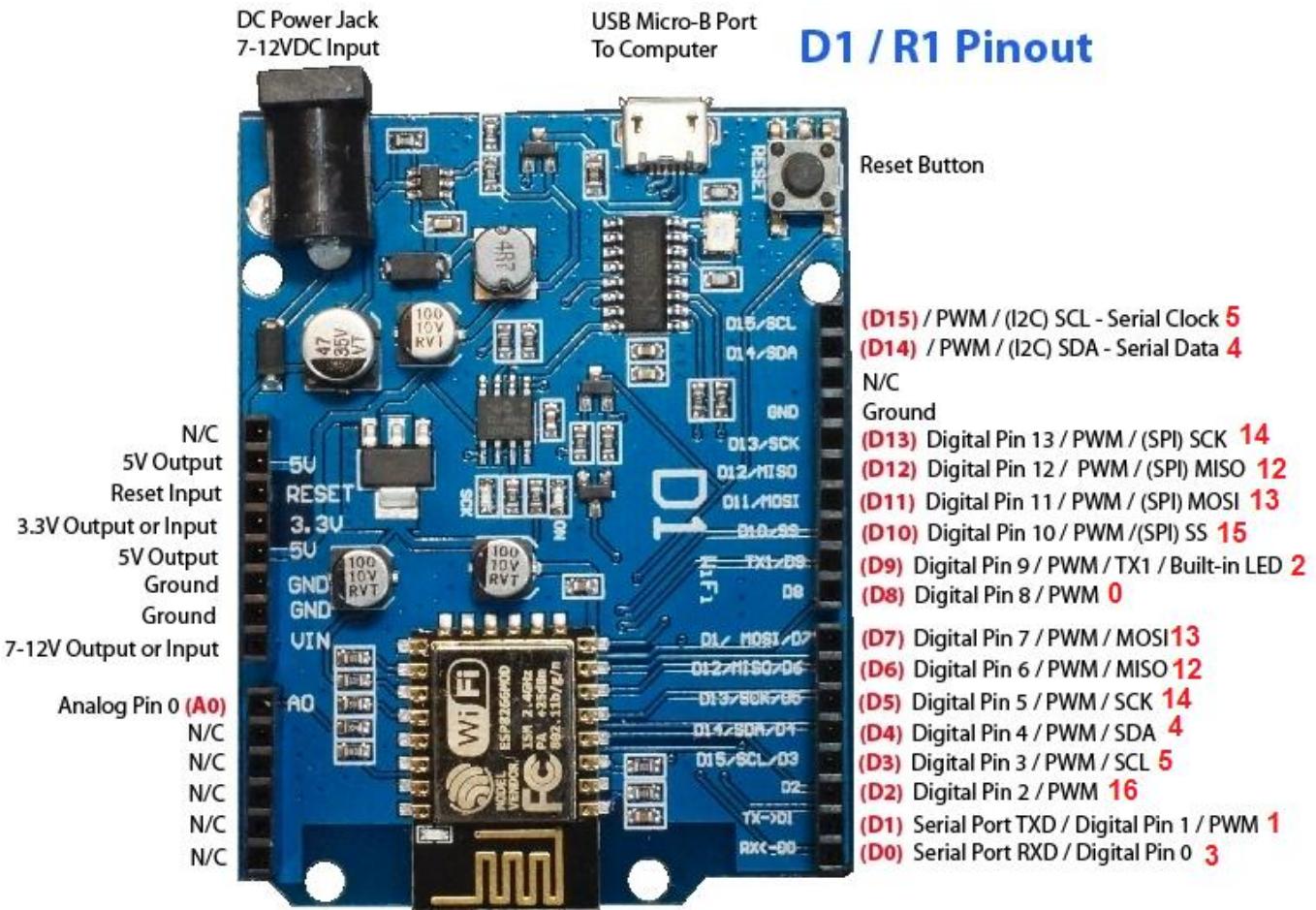
Par défaut REPL est activé, c'est-à-dire que l'on peut commander l'esp8266 avec le port Série via les drivers USB ch340/ch341.

Il existe aussi WebREPL, c'est aussi un moyen de commander l'esp8266 mais à travers le réseau wifi. Celui-ci est désactivé par défaut. Bien sûr, il faut au préalable configurer l'esp8266 sur son point d'accès via un port série.
Pour activer WebREPL :

```
>>> import webrepl_setup
WebREPL daemon auto-start status: disabled

Would you like to (E)nable or (D)isable it running on boot?
(Empty line to quit)
> E
To enable WebREPL, you must set password for it
New password (4-9 chars): toto
Confirm password: toto
Changes will be activated after reboot
Would you like to reboot now? (y/n) y #reste à utiliser un client WebREPL
```

Clignoter une Led



only pins 0, 2, 4, 5, 12, 13, 14, 15, and 16 can be used.

<http://docs.micropython.org/en/v1.9.3/esp8266/esp8266/tutorial/pins.html>

```
import machine
import time
led = machine.Pin(2, machine.Pin.OUT)
while(True):
    led.on()
    time.sleep_ms(500)
    led.off()
    time.sleep_ms(500)

#ctrl+c pour stopper
```

>>> %Run test.py
Traceback (most recent call last):

File "C:\Users\anthony\esp8266\test.py", line 7, in <module>

KeyboardInterrupt:

L'auto compléition fonctionne CTRL+ESPACE, mais pas à chaque fois

Clignoter deux Leds

En utilisant la programmation asynchrone avec librairie uasyncio

<https://github.com/peterhinch/micropython-async/blob/master/TUTORIAL.md>

Attention la dernière version d'uasyncio n'est pas dans le firmware de micropython. Surveillez les mises à jour.

```

import machine
import uasyncio as asyncio

led1 = machine.Pin(2, machine.Pin.OUT)
led2 = machine.Pin(15, machine.Pin.OUT)

async def blink(led, delay):  # coroutine
    while True:
        print(led, "on")
        led.on()
        await asyncio.sleep_ms(delay)
        print(led, "off")
        led.off()
        await asyncio.sleep_ms(delay)

# boucle d'événements
loop = asyncio.get_event_loop()
loop.create_task(blink(led1, 500))  # Schedule ASAP
loop.create_task(blink(led2, 1000))  # Schedule ASAP
loop.run_forever()

```

#ctrl+c pour stopper

Traceback (most recent call last):
File "C:\Users\anthony\esp8266\test.py", line 21, in <module>
File "uasyncio/core.py", line 173, in run_forever
File "uasyncio/__init__.py", line 69, in wait
KeyboardInterrupt:

>>>

```

>>> import uasyncio
>>> dir(uasyncio)
['__class__', '__name__', '__path__', 'DEBUG', 'log', 'select', 'sleep', 'sleep_ms', 'time', 'ucollections',
 'uerrno', 'utimeq', 'type_gen', 'set_debug', 'CancelledError', 'TimeoutError', 'EventLoop', 'SysCall',
 'SysCallI', 'StopLoop', 'IORead', 'IOWrite', 'IOReadDone', 'IOWriteDone', 'get_event_loop', 'SleepMs',
 'cancel', 'TimeoutObj', 'wait_for_ms', 'wait_for', 'coroutine', 'ensure_future', 'Task', '_socket',
 'PollEventLoop', 'StreamReader', 'StreamWriter', 'open_connection', 'start_server', 'uasyncio', 'core']

```

Remarque :

Par défaut il n'y a pas le module uasyncio dans l'esp32, il faut l'installer manuellement après être connecté sur votre point d'accès wifi.

```

>>> import upip
>>> upip.install('micropython-uasyncio')

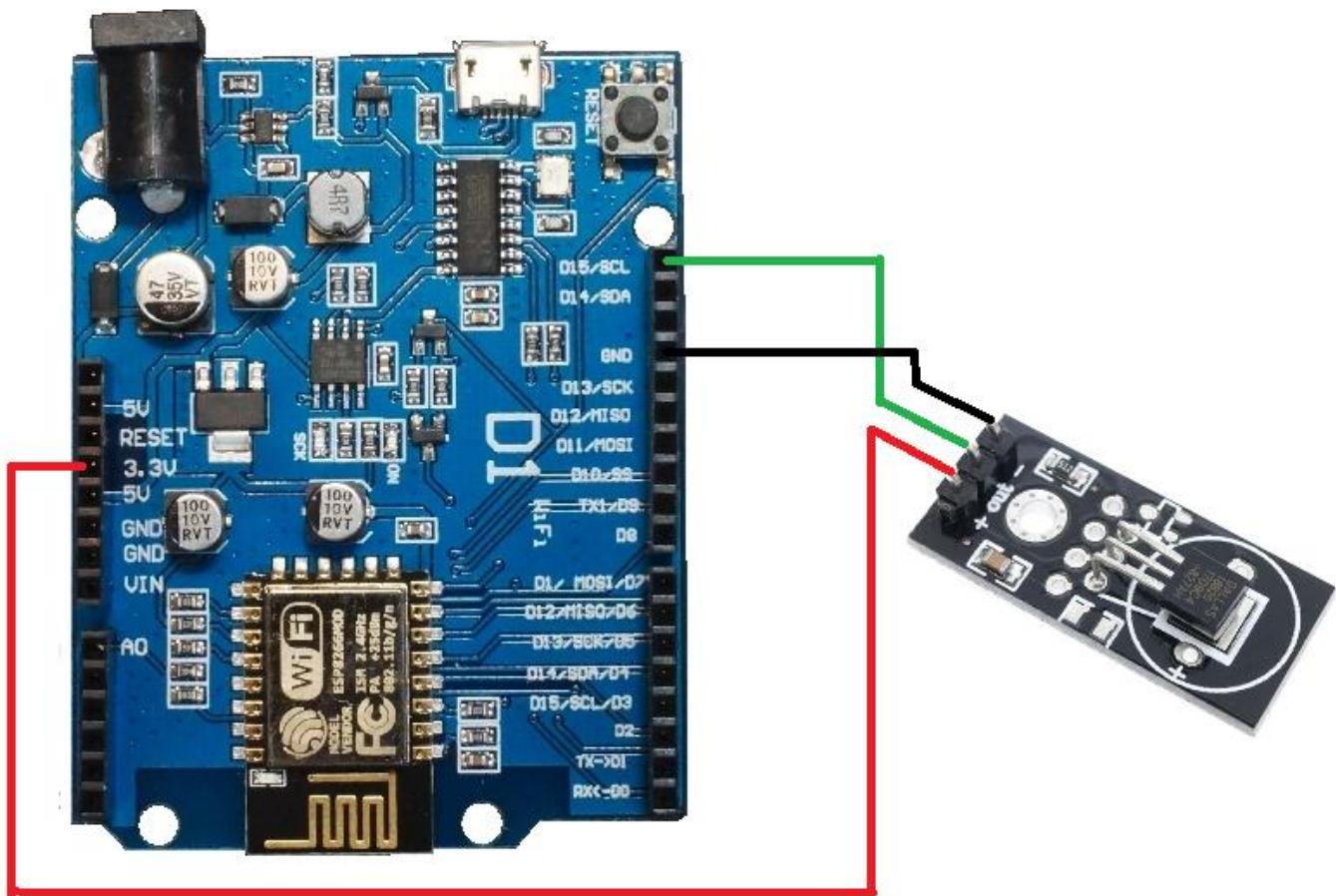
Installing to: /lib/
Warning: micropython.org SSL certificate is not validated
Installing micropython-uasyncio 2.0 from https://micropython.org/pi/uasyncio/uasyncio-2.0.tar.gz
Installing micropython-uasyncio.core 2.0 from https://micropython.org/pi/uasyncio.core/uasyncio.core-2.0.tar.gz

upip.install('micropython-ssd1306') # pour l'afficheur oled ssd1306

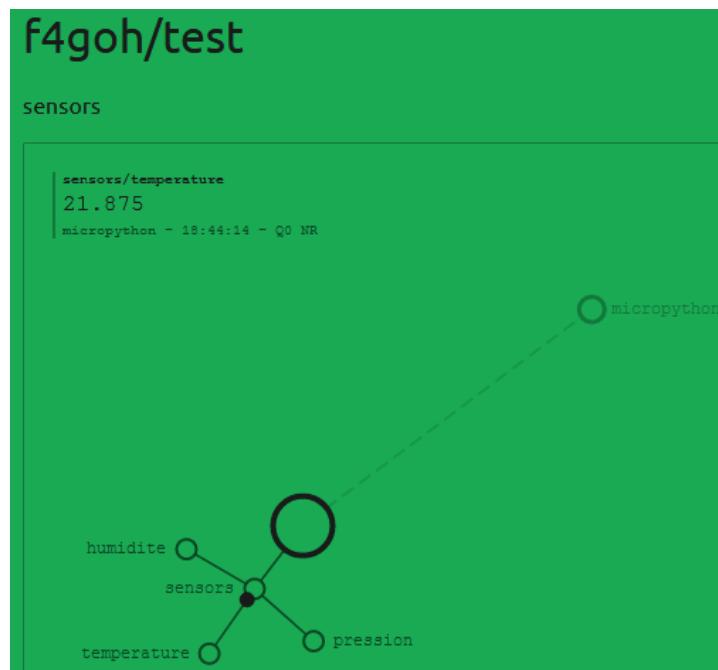
```

Envoyer une valeur de température vers un broker MQTT

Schéma de câblage avec un capteur de température ds18b20 onewire



Résultats sur le broker <https://shiftr.io/f4goh/test>



La température est bien reçue sur le broker.

<https://github.com/f4goh/MQTT-Tutoriel>

Le programme est compatible avec le broker <https://thingspeak.com/>

```

from umqtt.robust import MQTTClient
import network
import sys
import time
from time import sleep_ms
from machine import Pin
from onewire import OneWire
from ds18x20 import DS18X20

sta_if = network.WLAN(network.STA_IF)
print(sta_if.active())
print(sta_if.ifconfig())

myMqttClient = b"micropython"

THINGSPEAK_URL = b"broker.shiftr.io"
THINGSPEAK_USER_ID = b'weatherSensors'
THINGSPEAK_MQTT_API_KEY = b'bme280Sensors'
client = MQTTClient(client_id=myMqttClient,
                     server=THINGSPEAK_URL,
                     user=THINGSPEAK_USER_ID,
                     password=THINGSPEAK_MQTT_API_KEY,
                     ssl=False)

try:
    client.connect()
    print("connection ok");
except Exception as e:
    print('could not connect to MQTT server {}{}'.format(type(e).__name__, e))
    sys.exit()

bus = OneWire( Pin(5) )
ds = DS18X20( bus )

# Scanner tous les périphériques sur le bus
# Chaque périphérique à une adresse spécifique
roms = ds.scan()
for rom in roms:
    print( rom )

PUBLISH_PERIOD_IN_SEC = 30

while True:
    try:
        ds.convert_temp()
        # attendre OBLIGATOIREMENT 750ms
        sleep_ms( 750 )
        for rom in roms:
            temp_celsius = ds.read_temp(rom)
            print( "Temp: %s" % temp_celsius )
            client.publish("/sensors/temperature", str(temp_celsius))
            print("publish ok");
            time.sleep(PUBLISH_PERIOD_IN_SEC)
    except KeyboardInterrupt:
        print('Ctrl-C pressed...exiting')
        client.disconnect()
        sys.exit()

print("exit")

```

True
('10.3.141.104', '255.255.255.0', '10.3.141.1', '10.3.141.1')
connection ok
bytearray(b'\x10\xd0\x13U\x03\x08\x00\xc6')
Temp: 22.25
publish ok
Ctrl-C pressed...exiting

Timers en micropython

La précision du signal généré sur la broche 13 n'a rien de comparable avec le même programme en langage C.

```
"""
timer irq
the ESP32 has 4 hardware timers. For this example, we will use timer 0.
"""

import machine

led = machine.Pin(16, machine.Pin.OUT)
timer = machine.Timer(0)

def handleInterrupt(timer):
    global ledState
    ledState ^= 1
    led.value(ledState)

ledState = 0

timer.init(freq=1000, mode=machine.Timer.PERIODIC, callback=handleInterrupt)
# timer.init(period=1000, mode=machine.Timer.PERIODIC, callback=handleInterrupt)

while True:
    pass
```

Scanner I2C

```
"""
scan i2c esp32
Scan i2c bus...
i2c devices found: 1
Decimal address: 60 | Hexa address: 0x3c
"""

import machine
i2c = machine.I2C(scl=machine.Pin(4), sda=machine.Pin(5))

print('Scan i2c bus...')
devices = i2c.scan()

if len(devices) == 0:
    print("No i2c device !")
else:
    print('i2c devices found:',len(devices))

for device in devices:
    print("Decimal address: ",device," | Hexa address: ",hex(device))
```

Afficheur OLED SSD1306

```
"""
oled test
"""

import machine, ssd1306

i2c = machine.I2C(scl=machine.Pin(4), sda=machine.Pin(5))
oled = ssd1306.SSD1306_I2C(128, 64, i2c, 0x3c)
oled.fill(0)
oled.text("Hello f4goh", 0, 0)
oled.show()
```

Emission et réception sur l'UART 2 avec asyncio

```
"""
esp32 pinout
gpio16 rx
gpio17 tx
"""

import uasyncio as asyncio
from machine import UART
uart = UART(2, 9600)

async def sender():
    swriter = asyncio.StreamWriter(uart, {})
    while True:
        await swriter.awrite('Hello uart. \n')
        print('Wrote')
        await asyncio.sleep(2)

async def receiver():
    sreader = asyncio.StreamReader(uart)
    while True:
        res = await sreader.readline()
        print('Received', res)

loop = asyncio.get_event_loop()
loop.create_task(sender())
loop.create_task(receiver())
loop.run_forever()
```

Réception GPS sur l'UART 2 avec asyncio

A tester : https://github.com/alexmrqt/micropython-gps/blob/master/adafruit_gps.py

```
"""
esp32 pinout
gpio16 rx relié à la sortie GPS
gpio17 tx
"""

import uasyncio as asyncio
from machine import UART
uart = UART(2, 9600)

async def receiver():
    sreader = asyncio.StreamReader(uart)
    while True:
        res = await sreader.readline()
        nmea=res.split(b',')
        if nmea[0]==b'$GPGGA':
            #print('Received', nmea)
            print(int(float(nmea[1].decode())),end=',')
            print('latitude :',float(nmea[2].decode()),end=',')
            print(' longitude :',float(nmea[4].decode())))

loop = asyncio.get_event_loop()
loop.create_task(receiver())
loop.run_forever()
```

133405,latitude : 4753.415, longitude : 16.6092
133406,latitude : 4753.415, longitude : 16.6092
133407,latitude : 4753.415, longitude : 16.6092

**Fonctions prédefinies**

- ▷ `abs(x)` : valeur absolue de `x`
- ▷ `int(x)` : valeur `x` convertie en entier
- ▷ `float(x)` : valeur `x` convertie en réel
- ▷ `str(x)` : valeur `x` (int ou float), convertie en str
- ▷ `list(x)` : valeur `x` convertie en liste
- ▷ `tuple(x)` : valeur `x` convertie en tuple
- ▷ `dict(x)` : séquence de couples `x` convertie en dictionnaire
- ▷ `set(x)` : `x` converti en ensemble
- ▷ `help(x)` : aide sur `x`
- ▷ `dir(x)` : liste des attributs de `x`
- ▷ `type(x)` : type de `x`
- ▷ `print(...)` : imprime
- ▷ `input(x)` : imprime le string `x` et lit le string qui est introduit
- ▷ `round(x [,ndigits])` : valeur arrondie du float `x` à `ndigits` chiffres (par défaut 0)
- ▷ `range([start], stop, [step])` : retourne une suite d'entiers
- ▷ `sorted(s)` : retourne une liste avec les éléments de `s` triés

Gather, scatter et keyword arguments

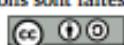
- ▷ `def fun(*args)` : gather des arguments en un tuple args
- ▷ `fun(*s)` : scatter de la séquence `s` lors de l'appel

Opérations et méthodes sur les séquences (str, list, tuples)

- ▷ `len(s)` : longueur de la séquence `s`
- ▷ `min(s)`, `max(s)` : élément minimum, maximum
- ▷ `sum(s)` : (ne fonctionne pas pour les string) : somme de tous les éléments (valeur numérique)
- ▷ `s.index(value, [start, [stop]])` : premier indice de value dans `s[start:stop]`
- ▷ `s.count(sub [,start [,end]])` : nombre d'occurrences sans chevauchement de sub dans `s[start:end]`
- ▷ `enumerate(s)` : construit une séquence de couples dont le ième élément (à partir de 0) vaut le couple (`i, s[i]`)
- ▷ `zip(a,b), zip(a,b,c), ...` : construit une séquence de couples, resp. triples, ..., dont le ième élément reprend le ième élément de chaque séquence `a, b, c`

Méthodes sur les chaînes de caractères (str)

- ▷ `s.lower()`, `s.upper()` : string avec caractères en minuscules respectivement en majuscules
- ▷ `s.islower()`, `s.isdigit()`, `s.isalnum()`, `s.isalpha()`, `s.isupper()` : vrai si dans `s` on n'a (respectivement) que des minuscules, des chiffres, des caractères alphanumériques, alphabétiques, majuscules
- ▷ `s.find(sub [,start [,end]])` : premier indice de `s` où le sous string `sub` est trouvé dans `s[start:end]`
- ▷ `s.replace(old, new[, co])` : copie de `s` en remplaçant toutes les (ou les `co` premières) occurrences de `old` par `new`.
- ▷ `s.format(...)` : copie de `s` après formatage
- ▷ `s.capitalize()` : copie de `s` avec première lettre en majuscule
- ▷ `s.strip()` : copie de `s` en retirant les blancs en début et fin
- ▷ `s.join(t)` : créer un str qui est le résultat de la concaténation des éléments de la séquence de str `t` chacun séparé par le str `s`
- ▷ `s.split([sep [,maxsplit]])` : renvoie une liste d'éléments séparés dans `s` par le caractère `sep` (par défaut blanc); au maximum `maxsplit` séparations sont faites (par défaut l'infini)

**Opérateurs et méthodes sur les listes**

- ▷ `s.append(v)` : ajoute un élément valant `v` à la fin de la liste
- ▷ `s.extend(s2)` : rajoute à `s` tous les éléments de la liste `s2`
- ▷ `s.insert(i,v)` : insère l'objet `v` à l'indice `i`
- ▷ `s.pop([i])` : supprime l'élément d'indice `i` de la liste (par défaut le dernier) et retourne la valeur de l'élément supprimé
- ▷ `s.remove(v)` : supprime la première valeur `v` dans `s`
- ▷ `s.reverse()` : renverse l'ordre des éléments de la liste, le premier et le dernier élément échangent leurs places, ...
- ▷ `s.sort(key=None, reverse=False)` : trie `s` en place
- ▷ `s.copy()` : shallow copie superficielle de `s`
- ▷ `del s[i]`, `del s[i:j]` : supprime un ou des éléments de `s`

Méthodes sur les dictionnaires (dict)

- ▷ `d.clear()` : supprime tous les éléments de `d`
- ▷ `d.copy()` : shallow copie de `d`
- ▷ `{}.fromkeys(s, v)` : crée un dict avec les clés de `s` et la valeur `v`
- ▷ `d.get(k [,v])` : renvoie la valeur `d[k]` si elle existe `v` sinon
- ▷ `d.items()` : liste des items (k,v) de `d`
- ▷ `d.keys()` : liste des clés
- ▷ `d.pop(k [,v])` : enlève `d[k]` s'il existe et renvoie sa valeur ou `v` sinon
- ▷ `d.popitem()` : supprime un item arbitraire (k,v) et retourne l'item sous forme de tuple
- ▷ `d.setdefault(k [,v])` : `d[k]` si elle existe sinon `v` et rajoute `d[k]=v`
- ▷ `d.update(s)` : `s` est une liste de tuples que l'on rajoute à `d`
- ▷ `d.values()` : liste des valeurs de `d`
- ▷ `del d[k]` : supprime l'élément de clé `k` de `d`

Méthodes sur les ensembles (set)

- ▷ `s = set(v)` : initialise `s` : un set contenant les valeurs de `v`
- ▷ `s.add(v)` : ajoute l'élément `v` au set `s` (ne fait rien s'il y est déjà)
- ▷ `s.clear()` et `s.copy()` : idem dictionnaires
- ▷ `s.remove(v)` : supprime l'élément `v` du set (erreur si `v` n'est pas présent dans `s`)
- ▷ `s.discard(v)` : si `v` existe dans `s`, le supprime
- ▷ `s.pop()` : supprime et renvoie un élément arbitraire de `s`

Modules

- ▷ `math` : accès aux constantes et fonctions mathématiques (pi, sin(), sqrt(x), exp(x), floor(x) (valeur plancher), ceil(x) (valeur plafond), ...) : exemple : `math.ceil(x)`
- ▷ `copy`: `copy(s)`, `deepcopy(s)` : shallow et deepcopy de `s`

Méthodes sur les fichiers

- ▷ `f = open('fichier')` : ouvre 'fichier' en lecture (autre paramètres possibles : 'w'(en écriture), 'a'(en écriture avec ajout), encoding ='utf-8': encodage UTF-8)
- ▷ `with open('fichier') as f` : ouvre 'fichier' pour traitement à l'intérieur du with
- ▷ `for ligne in open('fichier')` : ouvre et traite chaque ligne de 'fichier' et le ferme à la fin du for
- ▷ `f.read()` : retourne le contenu du fichier texte `f`
- ▷ `f.readline()` : lit une ligne
- ▷ `f.readlines()` : renvoie la liste des lignes de `f`
- ▷ `f.write(s)` : écrit la chaîne de caractères `s` dans le fichier `f`
- ▷ `f.close()` : ferme `f`

16.08.2018



Annexe 2

Ensemble et dictionnaire

Un **ensemble** `s` :

- est créé en donnant ses éléments entre accolades `{e1, e2, ...}` ou avec la fonction `set(s)` où `s` est une séquence d'éléments ;
- `set()` crée un ensemble vide ;
- les propriétés mathématiques des ensembles s'appliquent : les doublons sont supprimés ;
- `len(s)` donne le nombre d'éléments de l'ensemble `s` ;
- Ayant deux ensembles `a` et `b`: les opérateurs suivants existent
 - `a | b` (union),
 - `a & b` (intersection),
 - `a - b` (différence),
 - `a ^ b` (différence symétrique)

La différence symétrique prend les éléments dans un des deux ensembles mais pas dans l'autre ;

- la préposition `in` peut également être utilisée comme test d'appartenance ou dans une instruction `for` pour itérer sur tous les éléments de l'ensemble
- les opérateurs relationnels (`==`, `!=`, `<`, `<=`, `>`, `>=`) correspondent aux opérateurs d'égalité ou non et d'inclusion stricte ou non dans un sens ou l'autre.

un dictionnaire `d` :

- est créé en donnant ses éléments `clé : valeur` entre accolade `{k1:v1, k2:v2, ...}` ;
- un dictionnaire vide est créé en mettant deux accolades sans rien à l'intérieur `{}` ;
- chaque clé d'un dictionnaire identifie un élément et peut être de n'importe quel type *non modifiable* (une clé ne peut être une liste ou un ensemble ou un autre dictionnaire). les valeurs correspondantes peuvent être de n'importe quel type ;
- nous pouvons accéder à une composante de `d` en écrivant `d[key]` où `key` est une valeur de clé ;
- `d[cle] = valeur` ajoute une composante `cle : valeur` à `d` si elle n'existe pas ou modifie la composante `cle` si elle existe déjà dans `d` ;
- nous pouvons utiliser la préposition `in` pour tester si une clé fait partie de `d` ou dans un `for` pour itérer sur toutes les clés de `d` ;
- ayant deux dictionnaires `d1` et `d2`, les seuls opérateurs relationnels qui fonctionnent sont l'égalité `d1 == d2` et la différence `d1 != d2` ;
- `del(d[k])` supprime l'élément de `d` de clé `k`.

Manipulation des dictionnaires :

- `d.clear()` : supprime tous les éléments de `d`
- `d.copy()` : shallow copie de `d`
- `{}.fromkeys(s, v)` : crée un dict avec les clés de `s` et la valeur `v`
- `d.get(k [,v])` : renvoie la valeur `d[k]` si elle existe `v` sinon
- `d.items()` : liste des items `(k,v)` de `d`
- `d.keys()` : liste des clés
- `d.pop(k [,v])` : enlève `d[k]` s'il existe et renvoie sa valeur ou `v` sinon
- `d.popitem()` : supprime un item `(k,v)` et retourne l'item sous forme de tuple
- `d.setdefault(k [,v])` : `d[k]` si elle existe sinon `v` et rajoute `d[k] = v`
- `d.update(s)` : `s` est une liste de tuples que l'on rajoute à `d`
- `d.values()` : liste des valeurs de `d`

Memento PYTHON

Types de variables - fonctions de conversions

- Type integer → Nombre entier**
`int()` → convertit si possible un décimal ou texte en entier
- Type float → Nombre décimal**
`float()` → convertit si possible un entier ou texte en décimal
- Type string → Chaîne de caractères (texte)**
suite de signes définie en la délimitant par des guillemets
`str()` → convertit un nombre en chaîne
- Type boolean → Logique**
ne prend que deux valeurs : True et False
- Affectation =**
`x = ...` → lire « x prend la valeur ... »

astuces

```
x="12" # x est du type sting
y=3 # y est du type integer
z=x+y # erreur
z=int(x)+y # donne 15
z=x+str(y) # donne "123"
```

```
a=(1+1>2) # a booleen qui vaut False
b=(2*3==6) # b booleen qui vaut True
```

```
a,b=12,15 # équivaut à a=12 et b =15
x+=2 # équivaut à x=x+2
x-=1 # équivaut à x=x-1
```

Entrées, sorties console, opérations numériques

- Entrée → `input(" message")`** : lit un texte saisi au clavier .
 - ☞ Renvoie donc toujours une chaîne de caractères.
 - ☞ conversion possible en nombre par `int()` ou `float()`
- Sortie en console → `print(, , ...)`** : affiche en console les valeurs de tout type en les séparant par une tabulation.
- Opérations sur les nombres**
 - `/` → division décimale
 - `//` → quotient de la division entière
 - `%` → reste de la division entière
 - `**` → puissance (remarque : $a^{**0.5} = \sqrt{a}$)
 - `abs()` → valeur absolue
 - `round(x,d)` → arrondi le nombre x à d décimales

```
a=input("Texte?") # a type string
b=float(input("Nombre?")) # b type float
```

```
a=45*2
print("Coût=",a,"€") # affiche "Coût= 90 €"
```

```
d=11/4 # d vaut 2.75
q=11//4 # q vaut 2
r=11%4 # r vaut 3 car 11= 2*4+3
p=4**3 # p vaut 64
r=16**0.5 # r vaut racine(16) soit 4
x=abs(-7-10) # x vaut 3
y=round(4/3,4) # y vaut 1,3333
```

Chaînes de caractères

- Concaténation +** → attache les textes pour n'en former qu'un
- Caractères d'échappement**
le signe `\` permet de transformer le caractère qui suit
`\n` → saut de ligne (new). `\t` → tabulation
`\\" ou \\'` → guillemet qui ne ferme pas la chaîne
- longueur d'une chaîne :**
`len()` → renvoie le nombre de caractères d'une chaîne, espaces compris.
- Indexation** Chaque caractère de la chaîne est indexé (numéroté) en commençant par 0
`Chaine[i]` → renvoie le caractère de rang i
 - ☞ astuces :
 - `MaChaine[-1]` → dernier caractère
 - `MaChaine[-2]` → avant dernier caractère, etc...
 - `MaChaine[i:j]` → caractères indéxés de i à j-1
 - ☞ Attention : on ne peut pas modifier un caractère d'une chaîne par son index, seulement le lire !
- Code ASCII**
`chr(x)` → renvoie le caractère de code ASCII x
`ord(char)` → renvoie le code ASCII du caractère char
 - ☞ `chr(10)` ou `chr(13)` → saut de ligne. `chr(9)` → tabulation

```
T1="Ceci est" # variable type string
T2=" un test." # variable type string
T=T1+T2 # T vaut "Ceci est un test"
```

```
T="ceci est \nun test"# Imprime: ceci est
print(T) # un test
T="Il m\'a dit:\\"Attention\\"
print(T) # imprime : Il m'a dit:"Attention"
```

```
T="Ceci est un test"
L=len(T) # entier valant 16
print(T[0]) # écrit 'C'
x=T[2] # x vaut 'c'
y=T[L] # erreur de dépassement
z=T[-1] # z vaut 't' (dernière lettre)
z=T[-1] # z vaut 't' (dernier index)
z=T[-2] # z vaut 't' (avant dernier index)
z=T[1:6] # z vaut 'eci e' (indexe 1 à 5)
```

```
T="Python"
T[1]="y" # erreur: affectation non autorisée
```

```
x=chr(65) # x vaut 'A'
y=ord('B') # y vaut 66
```

Listes et tuples

- Liste** → suite indexée et modifiable d'éléments de tout type. Attention : l'indexation commence à 0

`NomListe=[élément1, élément2, élément3...]`

`NomListe[i]` → élément d'index i (lecture ou écriture)

`NomListe[0]` → premier élément

`NomListe[-1]` → dernier élément

- Principales fonctions :**

Longueur : `len()` → renvoie le nombre d'éléments

Ajout : `NomListe.append(x)` → ajoute x en fin de liste

Insertion : `NomListe.insert(i,x)` → insère x à l'index i

Suppression : `NomListe.pop()` → supprime le dernier élément

`NomListe.pop(i)` → supprime élément d'indexe i

```
L=[5,8,"Julie"] # Liste de 3 éléments
a=L[0] # Lecture: a vaut 5
L[1]=10 # Ecriture: L vaut [5,10,"Julie"]
b=L[-1] # b vaut 'Julie'
x= len(L) # x vaut 3
c=L[3] # erreur dépassement
L.append(3) # L vaut [5,10,'Julie',3]
L.insert(2,"P") # L vaut [5,10,'P','Julie',3]
L.pop() # L vaut [5,10,'P', Julie']
L.pop(2) # L vaut [5,10,'Julie']
```

- Tuples** → un tuple est une liste non modifiable

`NomTuple=(élément1, élément2, élément3,...)`

`NomTuple[i]` → élément d'indexe i en lecture seule

```
T=(4,8,10)
x=T[0] # x vaut 4
T[0]=5 # erreur, T non modifiable
```

- Liste de listes** → une liste peut contenir des listes !!

`NomListe[i][j]` désigne l'élément d'index j de la liste d'index i

```
Tab=[[4,2,9], [0,7,8]]
x=Tab[0] # x vaut [4,2,1] : index 0 de Tab
y=Tab[0][2] # y vaut 9 : index 2 de Tab[0]
```

Tests

- if test 1 : # un test est une valeur booléenne (logique)**

| bloc si test1 vérifié

elif test 2 : # (facultatif). Sinon si :

| bloc si test 1 non vérifié mais test2 vérifié

....

else : # (facultatif). Sinon

| bloc si aucun des tests précédent n'est vérifié

suite du programme

```
a=float(input("Donne un nombre"))
if a==0:
    texte = "nul"
elif a>0:
    texte= "positif"
else:
    texte = "négatif"
print(texte)
```

Boucle « Tant que »

- While test : # Tant que ...**

| Bloc répété tant

que test vérifié

suite du programme

```
# comparateurs
== # égal
!= # différent
> # supérieur
>= # sup ou égal
```

```
a=5
while a<10:
    print(a)
    a=a+2 # affiche 5,7 et 9

texte=input("12*5=?")
while texte!="60":
    print("C'est faux!")
    texte= input("12*5=?")
print("Correct")
```

Boucle « Pour... »

- For variable in liste : # Pour chaque ... dans... :**

| Bloc répété pour chaque valeur de la variable parcourant la liste

suite du programme

```
# -----
# Pour rompre une boucle
# -----
break # arrête la boucle
continue # passe au tour suivant
```

```
for k in [4,5,8]:
    x=k*k
    print(x)# affiche 16, 25 et 64

for i in range(3):
    print(i)# affiche 0,1 et 2

for i in range(2,5):
    print(i)# affiche 2,3 et 4

for i in range(2,10,3):
    print(i)# affiche 2,5 et 8
```

Logique : variables booléennes

- Une variable booléenne** ne prend que 2 valeurs `True`, `False`

- Opérateurs booléens**

`a or b` → vaut `True` si et seulement si l'un au moins vaut `True`

`a and b` → vaut `True` si et seulement si les deux valent `True`

`not a` → contraire de a : `True` si a `False`, `False` si a `True`

`a in Liste` → vaut `True` si et seulement si a élément de `Liste`

```
a=(1==2) # a booléen valant False
x=3
b=(x>0) # b booléen valant True
c=(a or b) # c vaut True
d=(a and b) # d vaut False
e= not a # e vaut True
L=[2,4,6]
f=(3 in L) # f vaut False
```

Procédures et fonctions ↗ Essentielles pour structurer un programme

Ce sont des sous-programmes autonomes avec leurs propres variables. Ils ne sont exécutés que lorsqu'ils sont appelés par le programme principal ou par une autre fonction

- Procédure (ou sous-programme)

```
def Nom(arg1, arg2,...):
    | bloc instructions
    # programme principal
    Nom (variable1,variable2...) # appel de la procédure
```

Les variables de 'passage' sont appelées arguments

- Fonction = procédure avec retour de valeur(s)

```
def Nom(arg1, arg2,...):
    | bloc instructions
    return x # x valeur ou liste de valeurs
    # programme principal
    a = Nom (valeur1,...) # appel + affectation de la valeur renvoyée
```

Arguments de la fonction

```
def bonjour(nom): # nom est argument
    # nom, text variables internes à la procédure
    text="Bonjour M."+nom+", comment va?"
    print(text) # action de la procédure
```

programme principal

x=input("votre nom?")

bonjour(x) # appel de la fonction

```
def pythagore(x,y): # x,y arguments de la fct
    # x,y et z variables internes à la fct
    z=(x**2+y**2)**0.5
    return z # valeur renournée par la fct
```

programme principal

x=pythagore(3,4) # appel + affectation valeur

x variable du programme principal

print(x) # affiche 5

Importation de librairies – Librairies utiles

Importer une librairie : plusieurs méthodes

- import MaLibrairie # Importation d'un ensemble de fcts
MaLibrairie.fonction1(var1,...) # appel d'une fonction
- import MaLibrairie as Lib # nom local de la librairie
Lib.fonction1(var1,...) # appel d'une fonction
- From MaLibrairie import fct1, fct2, # liste fcts utiles,
- From MaLibrairie import * # toutes les fonctions
fonction1(var1,...) # appel d'une fonction

```
import turtle # importe la librairie turtle
turtle.forward(50) # forward() fct de turtle
turtle.exitonclick() # fct de turtle
```

```
import math as m # m désigne la librairie math
x=m.sqrt(2) # racine carrée
y=m.sin(1.25) # sinus (angle en radian)
```

```
from math import sqrt, sin # seulement 2 fcts
x=sqrt(2) # pas de math.sqrt
y=sin(1.25)
```

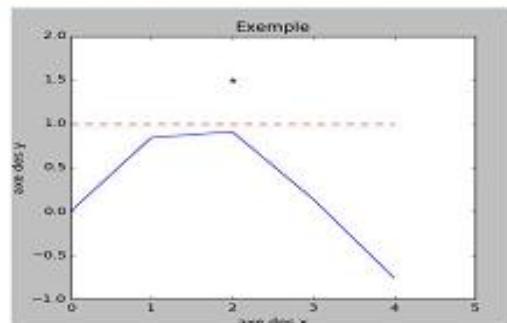
```
from math import * # toutes les fcts de math
x=sqrt(2) # pas de math.sqrt
y=sin(1.25)
```

Mathématiques

- Librairie math → fonctions mathématiques
`sqrt()` → racine carrée `sin()` → sinus(radian), etc...

```
from random import *
x=random() # décimal dans [0,1[
y=uniform(1,5) # décimal dans [1,5[
z=randint(1,4) # entier 1,2,3 ou 4
L=[2,4,'e']
t=choice(L) # 2,4 ou 'e'
```

```
import pylab as pl
from math import *
pl.axis([0,5,-1,2]) # fenêtre
pl.plot(2,1.5,'r+') # point rouge en +
pl.plot([0,4],[1,1],'r--') # segment (0,1) à (4,1) en ---
X=[0,1,2,3,4] # liste des x
Y=[sin(0),sin(1),sin(2),sin(3),sin(4)]
pl.plot(X,Y,'b-') # création polygone
pl.xlabel("axe des x") # Label axe des x
pl.ylabel("axe des y")
pl.title("Exemple") # titre du graphique
pl.show() # affichage du graphique
```



Nombres aléatoires

- Librairie random → génération de nombres aléatoires
`randint(a,b)` → entier dans [a, b]
`random()` → décimal (float) dans [0, 1]
`uniform(a,b)` → décimal (float) dans [a, b]
`choice(lista)` → élément de la liste lista

Graphiques mathématiques

- Librairie pylab combine deux librairies : pyplot et numpy pour les graphiques et calculs mathématiques
`point plot(x,y,'ro')` → point de coord (x,y) rouge et rond
`segment plot([x1,x2], [y1,y2], 'b-')` → segment de (x1,y1) à (x2,y2) en bleu et trait plein
`polygone plot(liste des x, liste des y, 'g--')` → polygone en vert et trait pointillé
`axes axis([xmin, xmax, ymin, ymax])`
`affichage show()` → affiche le graphique
`grille grid()` → affiche la grille
`label xlabel('texte')` → Label de l'axe des x
`ylabel('texte')` → Label de l'axe des y
`titre title('texte du titre')`
`sauvegarde savefig('nomfichier')` → sauve au format .png
- + info → http://matplotlib.org/users/pyplot_tutorial.html

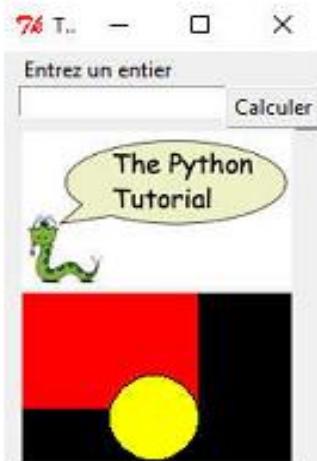
Fichiers textes

<ul style="list-style-type: none"> Ouverture nom = <code>open ("fichier.txt", "w")</code> → en mode lecture ou écriture, pointe en début de fichier <p>Variable qui identifie le fichier.</p> <p>(chemin +) nom fichier sur</p> <p>Mode <ul style="list-style-type: none"> • "r" lecture • "w" écriture • "a" ajout </p> <ul style="list-style-type: none"> Écriture nom.write("texte_") <ul style="list-style-type: none"> → Si le fichier n'existe pas, il est créé. → Si le fichier existe il est écrasé Lecture → ouvrir le fichier en mode lecture <ul style="list-style-type: none"> x= nom.read() → lecture du fichier entier x= nom.read(n) → lecture des n caractères suivants . x= nom.readline() → lecture ligne par ligne . 	<pre> Ecriture mode 'write': crée le fichier ou l'écrase si existe déf F=open("fich.txt","w") # pointe en début de fichier F.write("Début..") F.write("...suite de la liere ligne"+chr(10))#chr(10)->nlle ligne F.write("Deuxième ligne") F.close() # indispensable: enregistre le fichier #Écriture mode 'ajout' F=open("fich.txt","a") # pointe en fin de fichier F.write("...suite deuxième ligne") F.close() # indispensable: enregistre le fichier # Lecture (mode 'read') Fichier = open("fich.txt","r") # pointe au début du fichier t= Fichier.read() # lit tout le fichier et l'affecte à variable t Fichier.close() # indispensable: enregistre le fichier Fichier = open("fich.txt","r")# pointe au début du fichier t= Fichier.read(12) # lit les 12 premiers caractères du fichier t= Fichier.read(10) # lit les 10 caractères suivants Fichier.close()# indispensable pour libérer le fichier Fichier = open("fich.txt","r") # pointe au début du fichier t= Fichier.readline() # lit ligne par ligne Fichier.close()# indispensable: enregistre le fichier </pre>
Interfaces graphiques avec la librairie tkinter	

```

1 # Importation de la Librairie graphique tkinter
2 from tkinter import *
3 #
4 #-----#
5 # Création d'une fenêtre
6 #
7 Mafenetre= Tk() # Crédation d'une fenêtre
8 Mafenetre.title("Tutorial") # Titre de la fenêtre
9 Mafenetre.geometry("180x250") # Dimensions de la fenêtre
10 #
11 # Les "widgets" sont des composants que l'on positionne dans la fenêtre
12 #
13 Affichage = Label(Mafenetre, text='Entrez un entier')# création d'un Label
14 Affichage.place(x=10,y=80)# positionnement
15 # Widget "Entry" (zone de saisie de texte)
16 Saisie = Entry(Mafenetre) # création d'une zone de saisie
17 Saisie.place(x=10, y= 20) # positionnement
18 # Widget "Button" (bouton de commande)
19 def calcul ():# Procédure associée au click du bouton
20     v = Saisie.get() # récupère la valeur du widget entry "Saisie"
21     double = float(v) # (→ penser à la conversion en nombre)
22     MonTexte = "Le double de " + Saisie.get() + " = " + str(double)
23     Affichage.config(text=MonTexte) # le texte du label 'Affichage' est changé
24     # création et association du click à la procédure 'calcul'
25 bouton = Button( Mafenetre, text='Calculer', command=calcul) # création bouton
26 bouton.place(x=125,y=20) # positionnement
27 #
28 # Image dans un Label
29 from PIL import Image, ImageTk # importation des librairies images
30 monimage = Image.open("tutorial.png") # Chargement d'une image à partir de PIL
31 photo = ImageTk.PhotoImage(monimage) # Crédation d'une image compatible tkinter
32 LabelImage = Label( Mafenetre,image=photo) # label contenant une image
33 LabelImage.place(x=10,y=45) # positionnement
34 #
35 #
36 MonCanevas = Canvas(Mafenetre,bg="black",width = 150, height =100)
37 Rectangle = MonCanevas.create_rectangle(0,0,100,70,fill='red')#points opposés
38 Cercle = MonCanevas.create_oval(50,50,100,100,fill='yellow') #x-R, y-R, x+R, y+R
39 MonCanevas.place(x=10, y=140)
40 #
41 # IMPORTANT: la fonction mainloop i provoque le démarrage du réceptionnaire
42 # d'événements (clics, clavier,...) associé à la fenêtre
43 #
44 Mafenetre.mainloop()

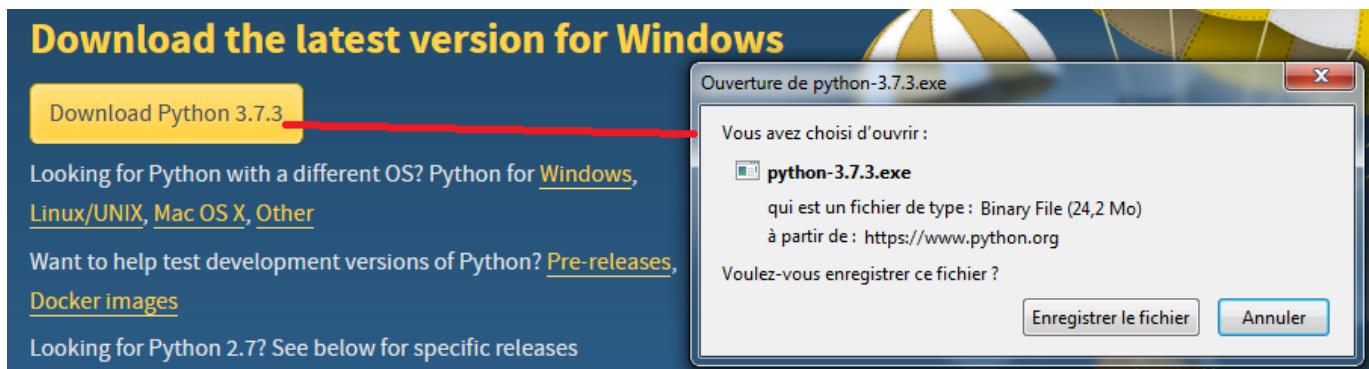
```



Installation de pycharm sous Windows

Installer python 3 en premier

<https://www.python.org/downloads/>



Vérification de python avec l'invite de commande

```
C:\Windows\system32\cmd.exe
Microsoft Windows [version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Tous droits réservés.

C:\Users\anthony>py
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Oct 24 2018, 16:44:53)
[MSC v.1911 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()

C:\Users\anthony>
```

Installer pycharm, choisir la version community

<https://www.jetbrains.com/pycharm/download/#section=windows>

The PyCharm download page features a large logo on the left and two main download options on the right. The "Professional" section is described as being for both scientific and web Python development, supporting HTML, JS, and SQL. It includes a "DOWNLOAD" button and a "Free trial" link. The "Community" section is described as being for pure Python development and is labeled as "Free, open-source". Both sections have "Windows", "macOS", and "Linux" tabs above them.

Installation de pycharm sous Linux

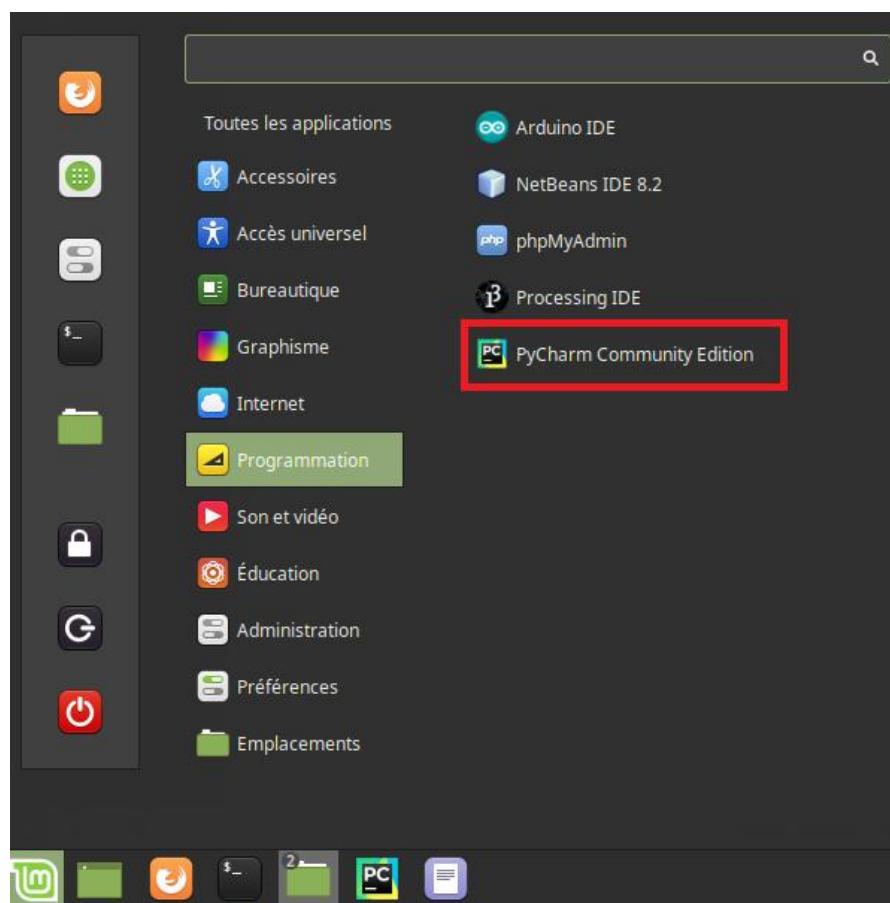
Dans le terminal :

```
sudo apt-get install python3-tk  
sudo apt-get install python3-pip  
sudo apt-get install python3-setuptools  
sudo apt-get install snapd  
sudo snap install pycharm-community --classic  
snap list  
nsi@nsi-LIFEBOOK-A555:~$ python3 -V  
Python 3.6.7
```

l'icone de lancement se trouve ici:

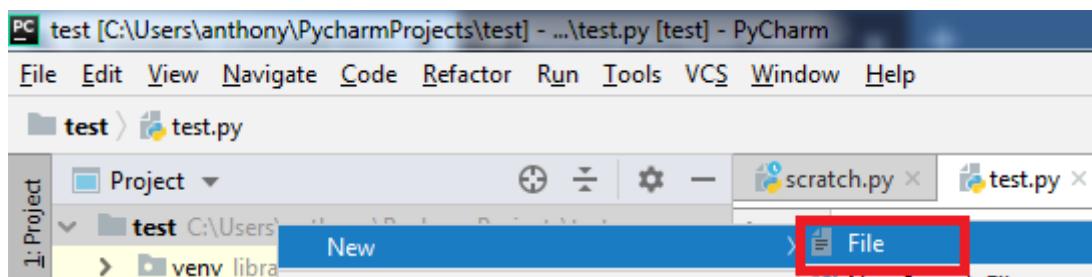
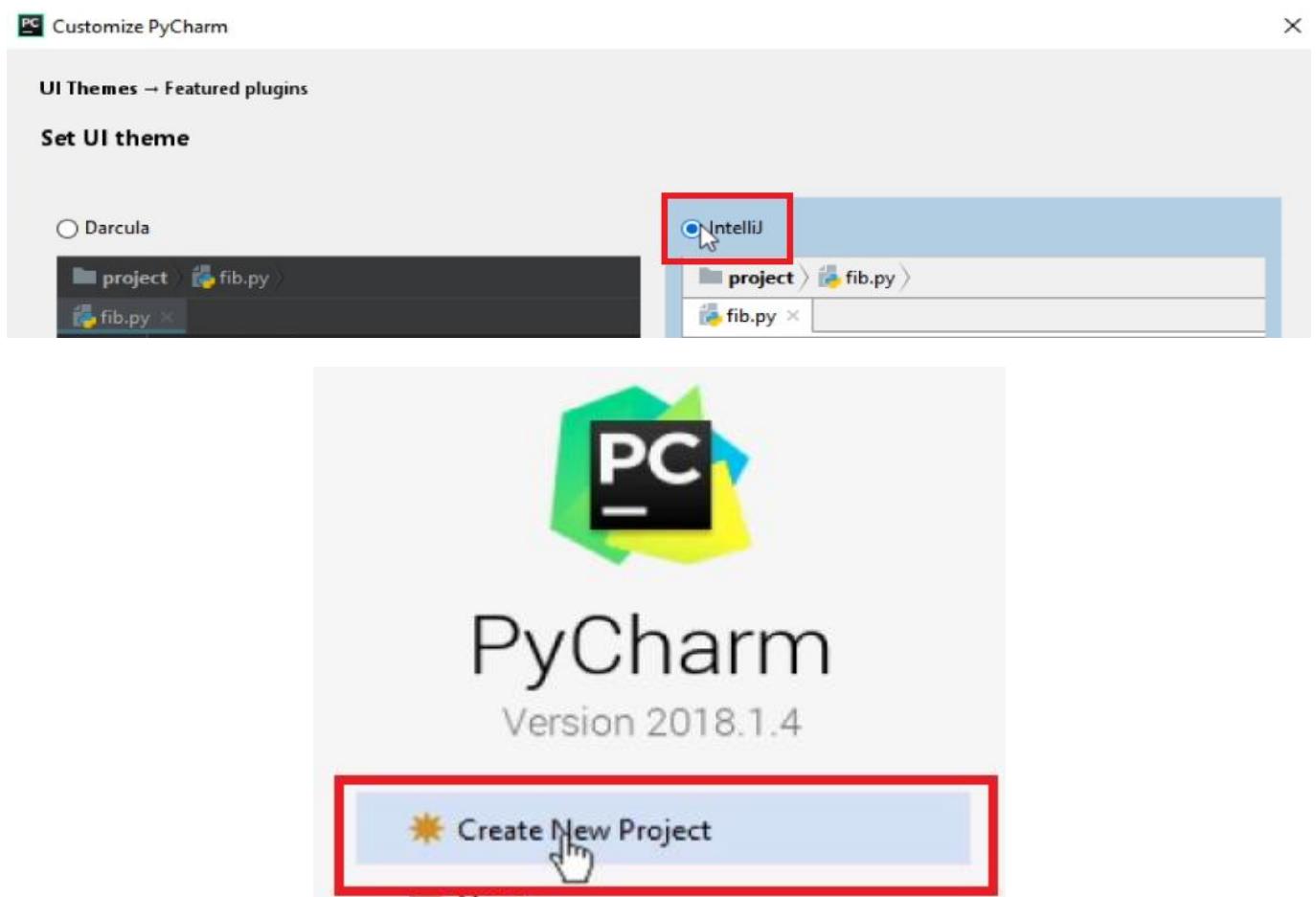
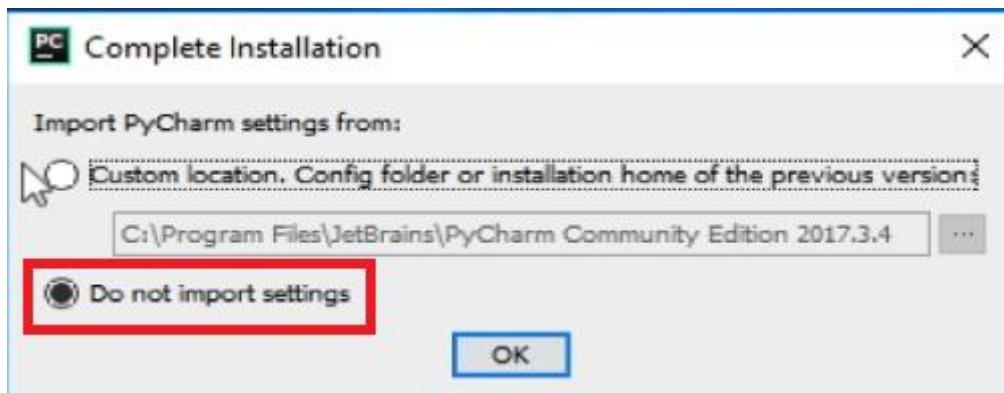
/var/lib/snapd/desktop/application

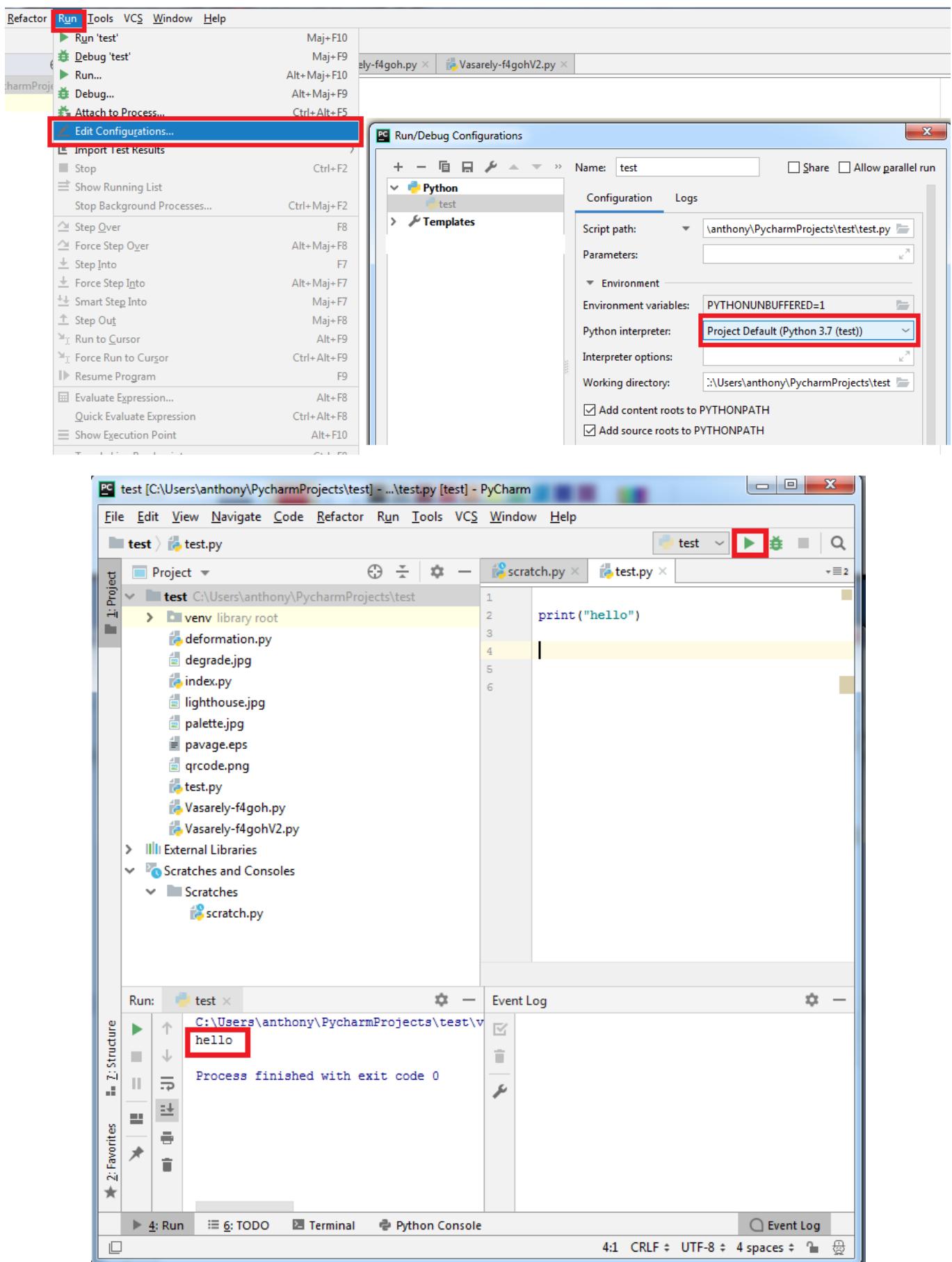
Copier le sur le bureau ou redémarrer le PC, un menu « programmation » apparaitra avec pycharm



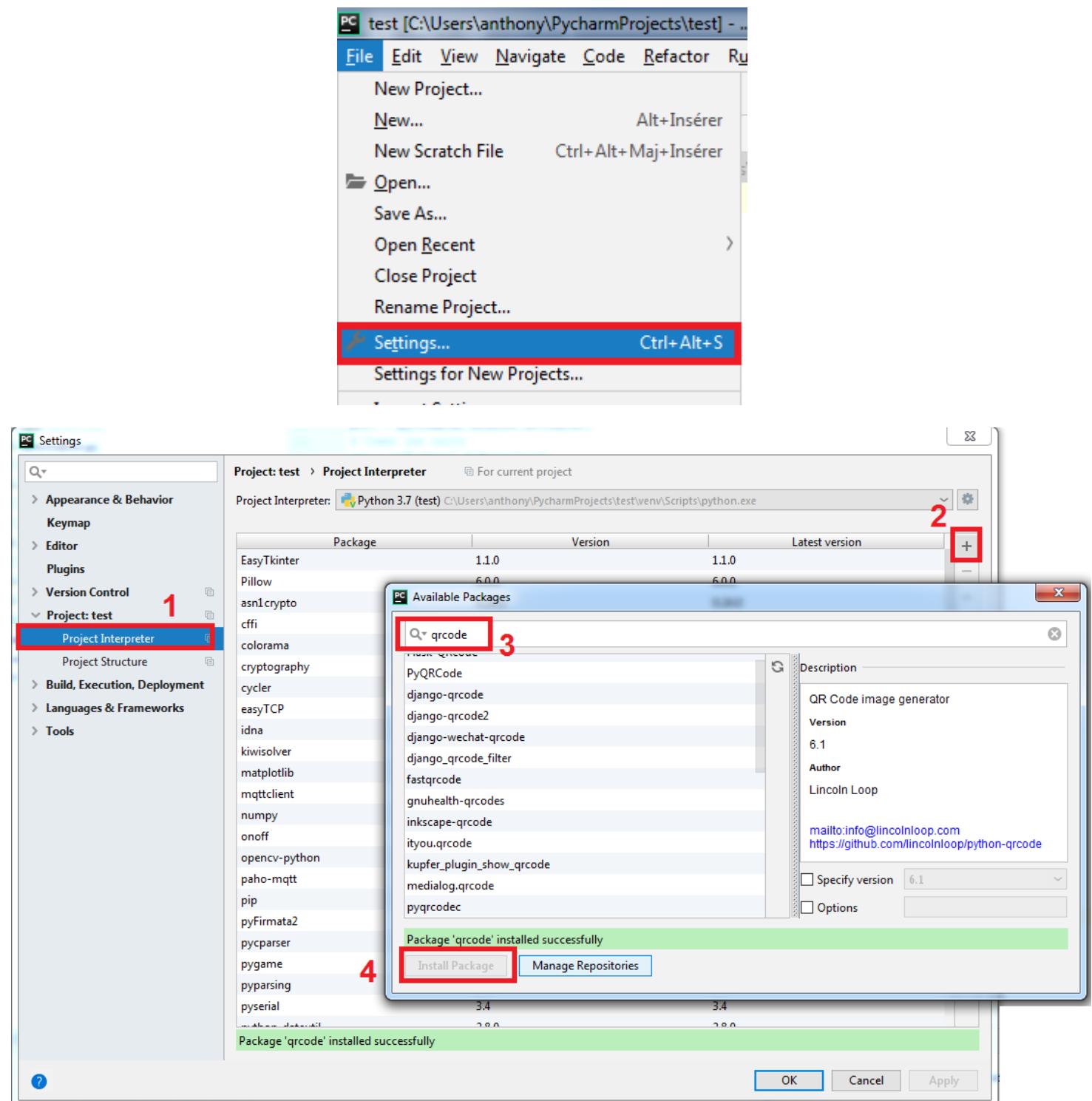
Configuration de pycharm

<https://www.youtube.com/watch?v=SZUNUB6nz3g>

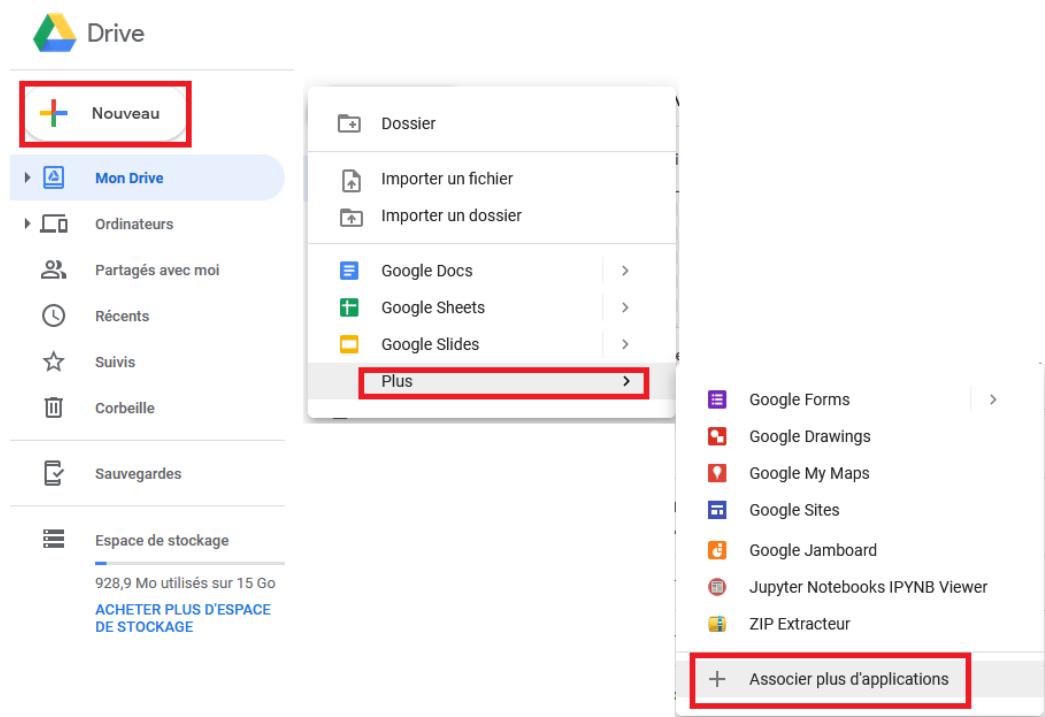




Installer des librairies (packages)



Créer un fichier jupyter notebook dans google Drive



Connecter des applications à Drive

Tous

Colaboratory
proposé par <https://colab.research.google.com>

A data analysis tool that combines code, output, and descriptive text into one collaborative document.

ÉVALUER Productivité ★★★★★ (618)

