

## MEMO PYTHON 3

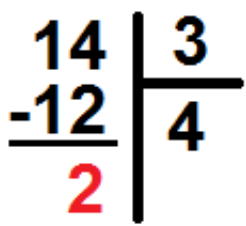
	Les types	Commentaire
Booléen	<pre>&gt;&gt;&gt; mon_booleen=True &gt;&gt;&gt; mon_booleen True &gt;&gt;&gt; type(mon_booleen) &lt;class 'bool'&gt;</pre>	Un booléen ne peut prendre que 2 valeurs True ou False.
Entier	<pre>&gt;&gt;&gt; valeur=1235 &gt;&gt;&gt; type(valeur) &lt;class 'int'&gt;</pre>	Un entier est appelé aussi entier
Float	<pre>nombre=12.58126 type(nombre) &lt;class 'float'&gt;</pre>	<pre>&gt;&gt;&gt; import sys &gt;&gt;&gt; sys.float_info sys.float_info(max=1.7976931348623157e+308 etc....</pre>
String	<pre>&gt;&gt;&gt; mot1="bonjour" &gt;&gt;&gt; mot2='bye' &gt;&gt;&gt; print(mot1,"/",mot2) bonjour / bye &gt;&gt;&gt; type(mot1) &lt;class 'str'&gt;</pre>	Une chaine de caractère peut être délimitée par des apostrophes (simple quotes) ou des guillemets (double quotes)
Complexe	<pre>&gt;&gt;&gt; z = 2+3j &gt;&gt;&gt; z.real 2.0 &gt;&gt;&gt; z.imag 3.0 &gt;&gt;&gt; z.conjugate() (2-3j) &gt;&gt;&gt; type(z) &lt;class 'complex'&gt;</pre>	<pre>&gt;&gt;&gt; from cmath import polar, rect &gt;&gt;&gt; cmath.polar(z) (3.6055512754639896, 0.982793723247329) &gt;&gt;&gt; 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	<pre>&gt;&gt;&gt; a=3.45 &gt;&gt;&gt; int(a) 3</pre>	La valeur a est affichée en valeur entière.
Integer-> float	<pre>&gt;&gt;&gt; b=5 &gt;&gt;&gt; c= float(b) print(c) 5.0</pre>	<pre># ceci est un commentaire su une ligne  """ ceci est un bloc de commentaire """</pre>
Int -> str	<pre>d=15 type(d) &lt;class 'int'&gt; e=str(d) print(e) 15 type(e) &lt;class 'str'&gt;</pre>	<pre>from random import * n = randint(1,6) print(n) 3 x = uniform(12, 18) print(x) 14.271572580135519 <a href="https://docs.python.org/3/library/random.html">https://docs.python.org/3/library/random.html</a></pre>
Str-> int ou float	<pre>int("bonjour") Traceback (most recent call last):   File "&lt;input&gt;", line 1, in &lt;module&gt; ValueError: invalid literal for int() with base 10: 'bonjour' int("10") 10</pre>	<pre>float("3.5") 3.5 float("hello") Traceback (most recent call last):   File "&lt;input&gt;", line 1, in &lt;module&gt; ValueError: could not convert string to float: 'hello'</pre>

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

	Calculs		Commentaires
Modulo	<pre>&gt;&gt;&gt; a=14 &gt;&gt;&gt; b=3 &gt;&gt;&gt; print(a%b) 2</pre>		Le modulo est le reste de la division entière
Décalage	<pre>&gt;&gt;&gt; a=5 &gt;&gt;&gt; a=a&lt;&lt;2 &gt;&gt;&gt; print(a) 20</pre>	<pre>&gt;&gt;&gt; a=a&gt;&gt;1 &gt;&gt;&gt; print(a) 10</pre>	
Incrémentation	<pre>&gt;&gt;&gt; a=5 &gt;&gt;&gt; a=a+1 &gt;&gt;&gt; print(a) 6</pre>	<pre>&gt;&gt;&gt; a=10 &gt;&gt;&gt; a+=5 &gt;&gt;&gt; print(a) 15</pre>	
Test	<pre>&gt;&gt;&gt; 1==2 False &gt;&gt;&gt; 1==1 True &gt;&gt;&gt; 3.5==3.5 True &gt;&gt;&gt; 1&gt;5 False</pre>	<pre>&gt;&gt;&gt; (1==1) and (3&gt;2) True &gt;&gt;&gt; (1==1) and (3&gt;5) False &gt;&gt;&gt; (1==2) or (10&gt;5) True &gt;&gt;&gt; not(1==2) True</pre>	
Test suite	<pre>&gt;&gt;&gt; 3!=4 True &gt;&gt;&gt; 3==4 False</pre>	<p><b>Attention :</b></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 &gt; 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 &lt;= 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>	<div>onj</div> <table><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> <div>Le chiffre 4 est exclu</div>	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>	<div>ne parlez pas si fort !</div> <div>NSI</div>														
Concaténation	<pre>w = "Washington" t= "Touchard" lycee= w+" "+t print (lycee) print (len(lycee))</pre>	<div>Washington Touchard</div> <div>19</div>														
Sortir d'un menu	<pre>chaine = "" while chaine.lower() != "q":     print("Tapez 'Q/q' pour quitter...")     chaine = input() print("Merci !")</pre>	<div>Tapez 'Q/q' pour quitter...</div> <div>s</div> <div>Tapez 'Q/q' pour quitter...</div> <div>q</div> <div>Merci !</div>														

	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>	<pre> saisir un nombre decimal: 2.3 6.8999999999999995 saisir un nombre entier: 5 25 saisir une chaine de caractères : hello il y a 5 caractere(s) dans votre chaine </pre>	
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, /)   identifier(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><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><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> Indice 4 exclu	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]												

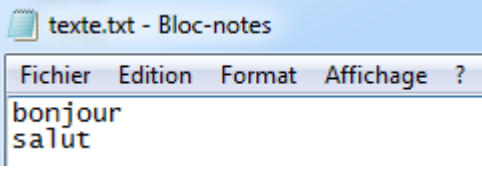
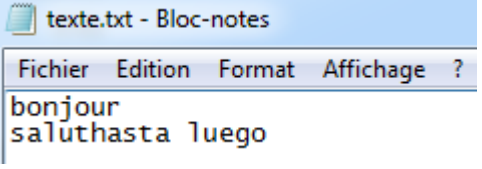
	Instructions	Résultat affiché sur la console
Initialisation(1)	<pre> liste=[] for i in range(0,10):     liste.append(0) print(liste) #ou tab=[0] *10 print(tab) </pre>	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Initialisation(2)	<pre> liste=[0 for i in range(0,10)] print(liste) liste=[i for i in range(0,10)] print(liste) liste=[i*2 for i in range(0,10)] print(liste) </pre>	<p>[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]</p> <p>[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]</p> <p>[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]</p>
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>	<pre> 1 2 3 4 5 6 7 8 9 </pre>

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>	<p>pi vaut plus ou moins 3.14  pi vaut plus ou moins 3.142  pi vaut plus ou moins 003.142</p> <p>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</p>
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>	<p>valeur : 1  La valeur était 3</p>
Tuple(1)	<pre>tuple=('a',1,5.6,[1,2,45]) print("longueur du tuple",len(tuple)) for i in tuple:     print(i)</pre>	<p>longueur du tuple 4</p> <p>a  1  5.6  [1, 2, 45]</p>
Tuple(2)	<pre>tuple_1 = (5, 2, 25, 56) tuple_2 = ("jack","tony") tuple_3 = tuple_1 + tuple_2 print(tuple_3)</pre>	<p>(5, 2, 25, 56, 'jack', 'tony')</p>
Tuple(3)	<pre>tuple_1 = ("jack","tony") tuple_2 = tuple_1 * 2 print(tuple_2)</pre>	<p>('jack', 'tony', 'jack', 'tony')</p> <p>Rem : un tuple est non mutable  (on ne peut pas changer le contenu)</p>

	Instructions	Résultat affiché sur la console
Tuple(3)	<pre>def recupere_resultats():     a=1     b=0.56     c=["toto",45]     return (a,b,c)  print(recupere_resultats())</pre>	(1, 0.56, ['toto', 45])
Tuple(4)	<pre>def recupere_resultats():     a=1     b=0.56     c=["toto",45]     return (a,b,c)  res=recupere_resultats() print(len(res)) for items in res:     print(items)</pre>	3 1 0.56 ['toto', 45]
Tuple(5)	<pre>def recupere_resultats():     a=1     b=0.56     c=["toto",45]     return (a,b,c)  res=recupere_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

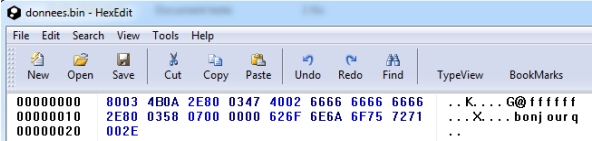
	Instructions	Résultat affiché sur la console
Dictionnaire(3)	<pre>stock = {'poires': 5, 'pommes': 10, 'fraises': 35} print('fraises' in stock) print('fraises' in stock.keys()) print(7 in stock.values()) print(5 in stock.values()) print(3 in stock.values())</pre>	<p>True True False True False</p>
Dict.(4)	<pre>stock = {'poires': 5, 'pommes': 10, 'fraises': 35}  for nom, num in stock.items():     print(nom, '-&gt;', num)</pre>	<p>poires -&gt; 5 pommes -&gt; 10 fraises -&gt; 35</p>
Dict.(5)	<pre>inventaire="" for nom in stock.keys():     inventaire += nom + ', ' print(inventaire)</pre>	<p>poires, pommes, fraises,</p>
Dict.(6)	<pre>for num in stock.values():     print (num)</pre>	<p>5 10 35</p>
Ensemble(1)	<pre>H = set([1,2,3,6,3,2,1,8,5]) H.add(4)    # un nouvel élément 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>	<p><math>H = \{1, 2, 3, 4, 5, 6, 8\}</math> <math>H = \{1, 2, 3, 4, 5, 6\}</math></p>
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 &amp; G)    #&amp; 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>	<p><math>H \cap G = \{1, 3\}</math> <math>H \cup G = \{1, 2, 3, 5, 6, 8, 10\}</math> <math>H \setminus G = \{8, 2, 5, 6\}</math> <math>H \ominus G = \{2, 5, 6, 8, 10\}</math></p>

	Instructions	Résultat affiché sur la console																																								
split	chaine="pensez au c++ après le python" nouvelle_chaine=chaine.split() print(nouvelle_chaine)	['pensez', 'au', 'c++', 'après', 'le', 'python']																																								
	chaine="alors,quoi de neuf" nouvelle_chaine=chaine.split(',') print(nouvelle_chaine)	['alors', 'quoi de neuf']																																								
Répétions	chaine="hello" print(chaine*3)	hellohellohello  Impossible à faire en C++																																								
join	liste=['scotty', 'say', 'hello', 'computer'] nouvelle_chaine=" ".join(liste) print(nouvelle_chaine)	scotty says hello computer																																								
count	chaine="hello lycée touchard" print(chaine.count('o'))	2  En effet il y a 2 'o' dans la chaine																																								
find	chaine="hello lycée touchard" print(chaine.find('tou'))	12																																								
		<table><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	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 \\ """	Hello World Aujourd'hui C'est "Dommage!" Hum \o/																																								

	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>	<pre>bonjour saluthasta luego</pre>

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
Écriture 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>	<pre>[['Personne', 'Age'], ['Peter', '22'], ['Tony', '45'], ['Bruce', '50']]</pre>

	Instructions	Résultat affiché sur la console
Fichier json (écriture)	<pre>import json  dico = {     "agrumes": {"oranges": 4, "citrons": 2, "pamplemuse": 54},     "salades": {"batavia": 2, "laitue": 9} }  f = open('fruits_legumes.json', 'w') json.dump(dico, f, indent=4) f.close()</pre>	<pre>{   "agrumes": {     "oranges": 4,     "citrons": 2,     "pamplemuse": 54   },   "salades": {     "batavia": 2,     "laitue": 9   } }</pre>
Fichier json (lecture)	<pre>import json  g = open('fruits_legumes.json', 'r') fruitslegumes = json.load(g) g.close() print(fruitslegumes)</pre>	<pre>{'agrumes': {'oranges': 4, 'citrons': 2, 'pamplemuse': 54}, 'salades': {'batavia': 2, 'laitue': 9}}</pre>

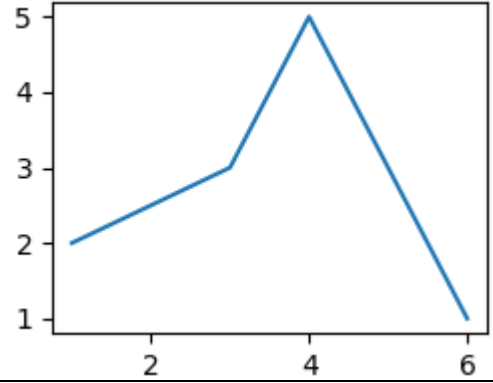
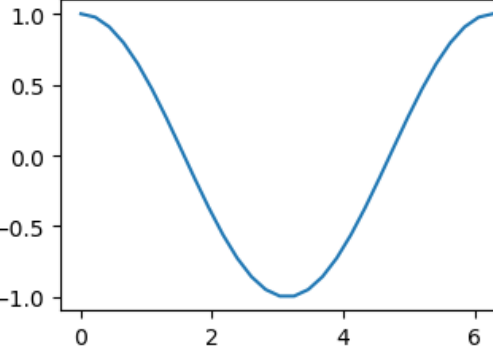
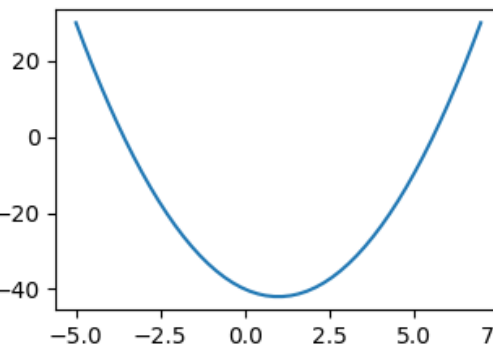
<https://jsonformatter.curiousconcept.com/>

	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]
filtre	<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


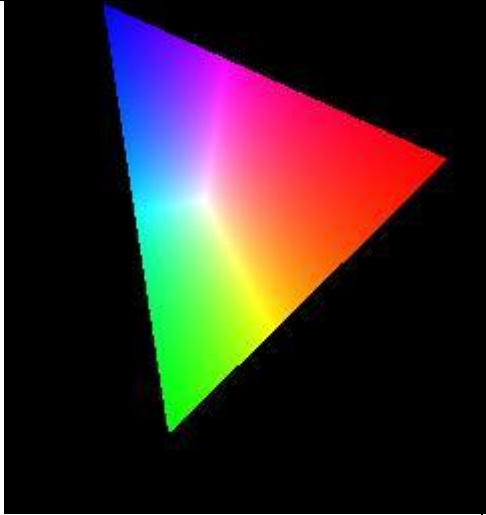
<https://matplotlib.org/2.0.2/Matplotlib.pdf>

	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/>

	Instructions	
Affichage d'une image	<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
Dégradé	<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>	 <p>(le temps de traitement est long)</p>

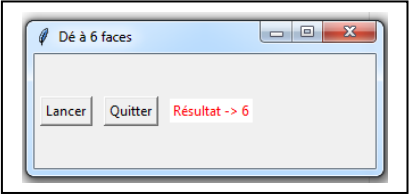
	Instructions
Génération d'une page web	<pre> #https://python-visualization.github.io/folium/quickstart.html import folium  m=folium.Map(     location=[ 47.995332, 0.204976],     tiles='OpenStreetMap',     zoom_start=20 )  folium.Marker(     location=[ 47.995653, 0.20267],     popup='Washington place',     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')</pre>
Résultat	

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

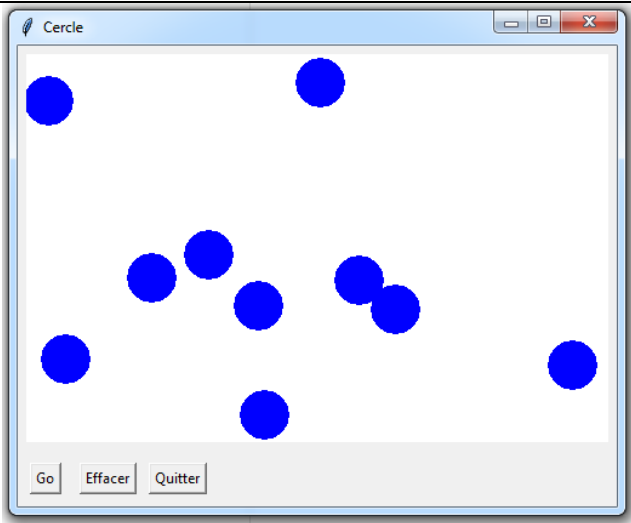
## Les fenêtres avec tkinter

[https://www.python-course.eu/tkinter\\_buttons.php](https://www.python-course.eu/tkinter_buttons.php)

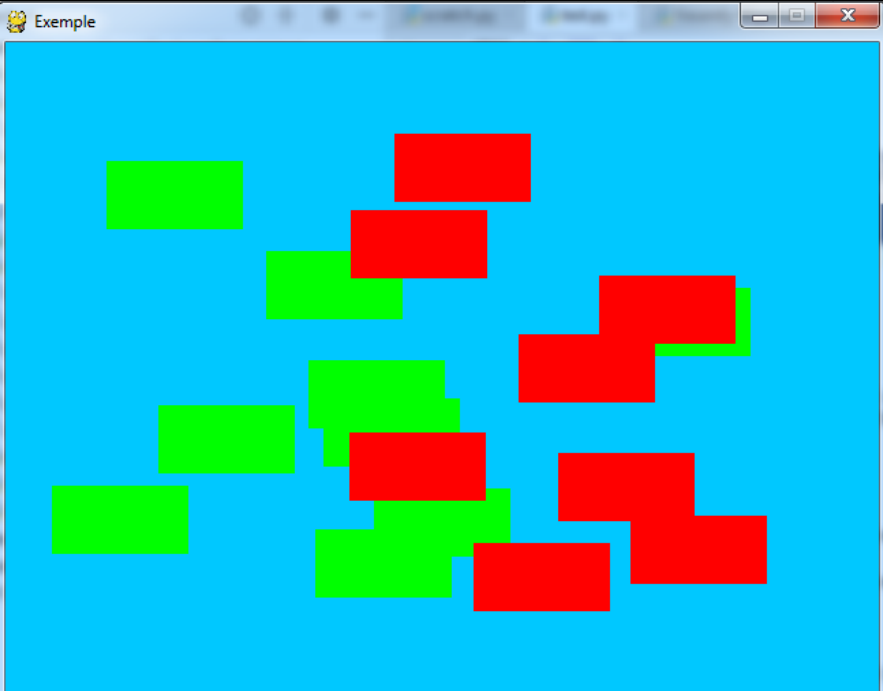
	Instructions
Boutons (1)	<pre> import tkinter as tk def write_slogan():     print("Tkinter is easy to use!") root = tk.Tk() frame = tk.Frame(root) frame.pack() button =  tk.Button(frame, text="QUIT", fg="red", command=quit) button.pack(side=tk.LEFT)  slogan = tk.Button(frame, text="Hello", command=write_slogan) slogan.pack(side=tk.LEFT)  root.mainloop() </pre> 

	Instructions
Boutons (2)	<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 -&gt; ' + 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 -&gt; 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	 <p>The screenshot shows a graphical user interface window titled "Cercle". The window contains a white rectangular canvas area. On the canvas, there are approximately 10 blue circles of varying sizes, representing random points. Below the canvas, there is a horizontal bar containing three buttons: "Go", "Effacer", and "Quitter". The "Go" button is on the left, "Effacer" is in the middle, and "Quitter" is on the right. The window has standard OS window controls (minimize, maximize, close) in the top right corner.</p>

## 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 fenetre (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	 <p>The screenshot shows a window titled 'Exemple' with a blue background. It contains several overlapping rectangles of two colors: red and green. The rectangles are scattered across the window, with some overlapping each other. The window has standard OS controls (minimize, maximize, close) in the top right corner.</p>

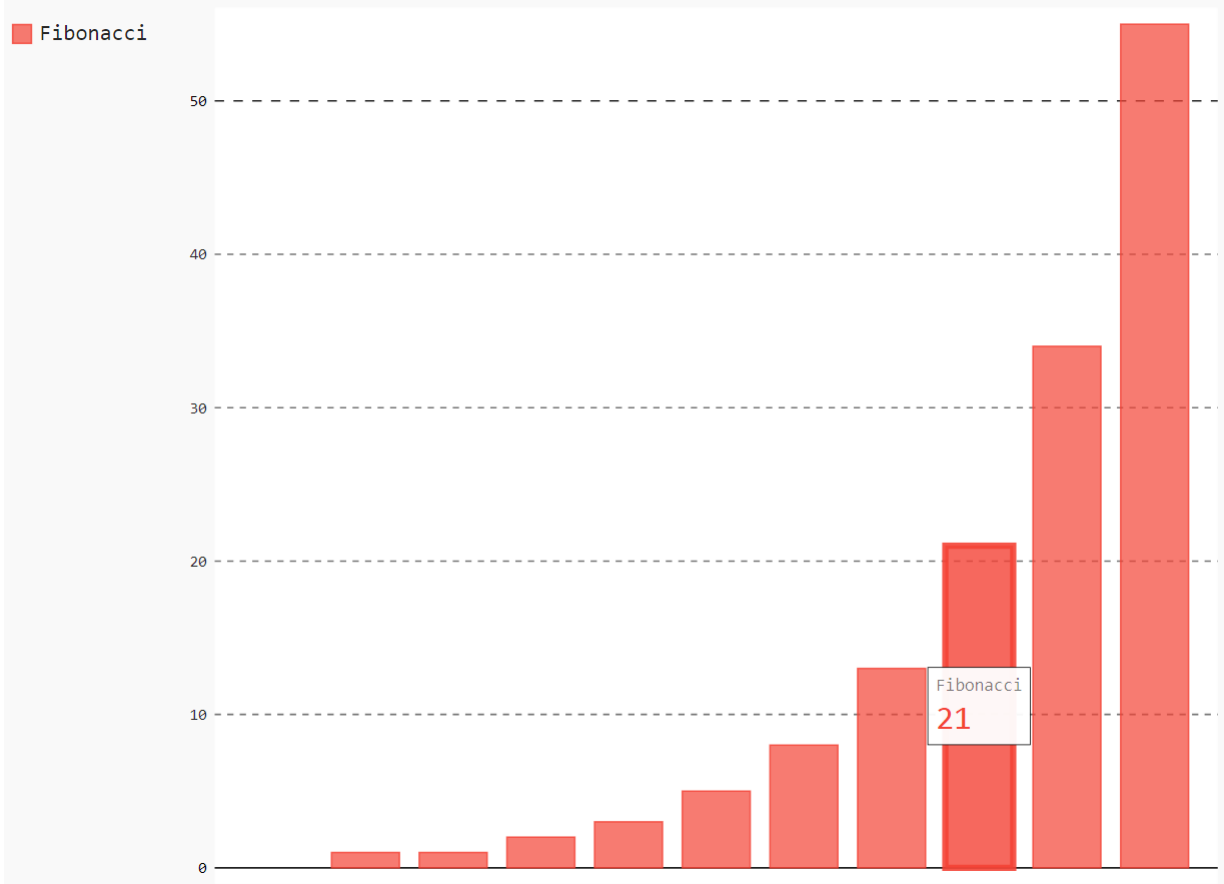
## Pyglet

	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	<div data-bbox="220 1088 820 1603">  </div> <div data-bbox="842 1088 1465 1603"> <p>Touche a</p> <p>Affichage d'un message</p> <p>Touche p</p> <p>Lecture d'un fichier mp3</p> </div>

A suivre...

## Pygal : Création de fichiers svg

[https://fr.wikipedia.org/wiki/Scalable\\_Vector\\_Graphics](https://fr.wikipedia.org/wiki/Scalable_Vector_Graphics)

	Instructions																								
Test	<pre>import pygal                                # First import pygal bar_chart = pygal.Bar()                    # 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	 <p>The bar chart displays the Fibonacci sequence. The y-axis is labeled from 0 to 50 in increments of 10. The x-axis represents the index of the sequence. The bars are red. A tooltip is shown for the 9th bar (index 9, value 21).</p> <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></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
Index	Value																								
0	0																								
1	1																								
2	1																								
3	2																								
4	3																								
5	5																								
6	8																								
7	13																								
8	21																								
9	34																								
10	55																								

	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 tarentule',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 tarentule', 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 tarentule', 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>

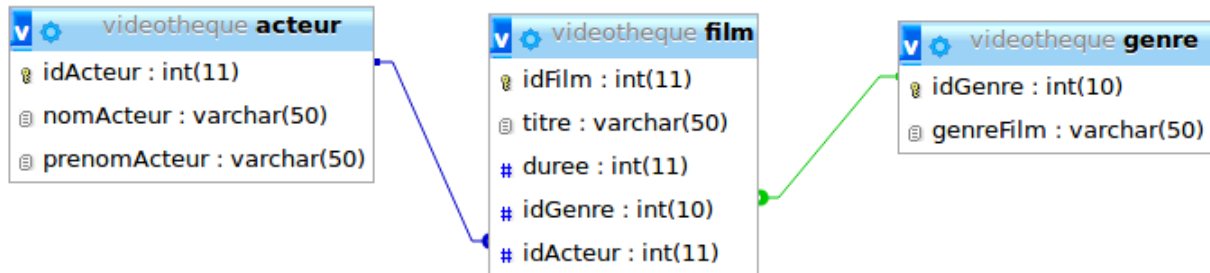
Afficher le contenu d'une base en ligne :

<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')

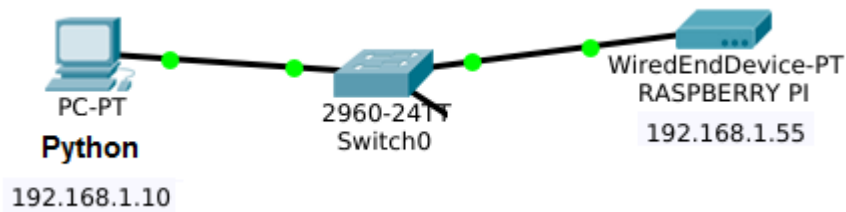
def requete(req):
    print("-----")
    c = conn.cursor()
    c.execute(req)
    rows = c.fetchall()
    for row in rows:
        print(row)

#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://philippes.ddns.net/documentation/Raspberry\\_pi/](https://philippes.ddns.net/documentation/Raspberry_pi/)

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

	Host	User	Password	Select_priv
<input type="checkbox"/> Modifier <input type="checkbox"/> Copier <input type="checkbox"/> Effacer	%	user1		Y
<input type="checkbox"/> Modifier <input type="checkbox"/> Copier <input type="checkbox"/> Effacer	localhost	phpmyadmin	*1844F2B11CCA3F3B31F573A1384F608BB6DE3DF9	N
<input type="checkbox"/> Modifier <input type="checkbox"/> Copier <input type="checkbox"/> Effacer	localhost	root	*1844F2B11CCA3F3B31F573A1384F608BB6DE3DF9	Y
<input type="checkbox"/> Modifier <input type="checkbox"/> Copier <input type="checkbox"/> Effacer	%	user2	*63D85DCA15EAF5C908FD2FAE50CCBC60C4EA2	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)
```

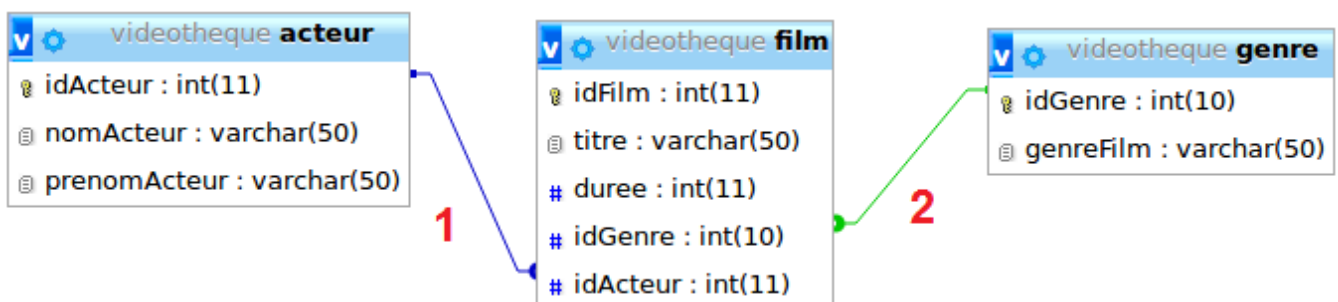
## Requete sql avec avec mysql.connector

	Instructions
Requête 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	<p>Star wars 1  Star wars 2  Alien 3  blade  new york 1997</p>

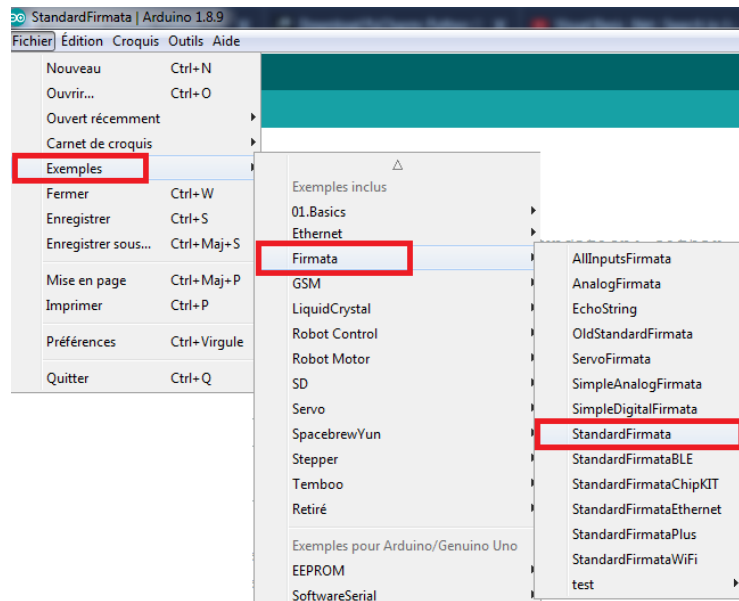
Un convertisseur sqlite to mysql est disponible ici

<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 firmadata 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.AUTODETECT
# Créer une carte
uno = pyfirmata2.Arduino(port)
```

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

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

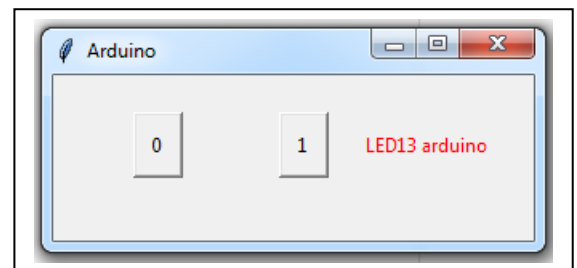
```
fenetre.title('Arduino')
fenetre.geometry('300x300')
```

```
# 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'])</pre>	
	<pre>[{'type': 'QRCODE', 'data': 'https://touchard-washington.paysdelaloire.e-lyco.fr/'}] https://touchard-washington.paysdelaloire.e-lyco.fr/</pre>	


## Opencv-python (avec ou sans webcam)

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

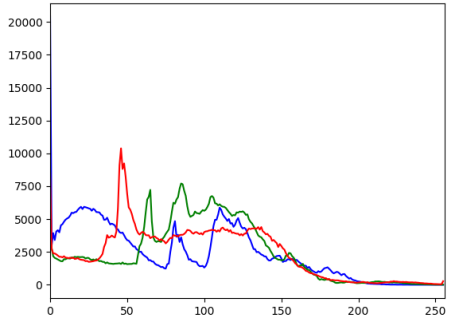




	Instructions	Résultat affiché
OpenCV(1)	<pre>import numpy as np import cv2 # Load an color image in grayscale img = cv2.imread('lighthouse.jpg',0) # Load an color image in color #img = cv2.imread('lighthouse.jpg',1)  cv2.imshow('image',img) k = cv2.waitKey(0) if k == 27:    # wait for ESC key to exit     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\\_misc.html](https://docs.opencv.org/3.1.0/d7/d1b/group_imgproc_misc.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) &amp; 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](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>	
Résultat affiché	<div style="display: flex; flex-wrap: wrap; justify-content: space-around;"> <div style="text-align: center;"> <p>Original Image</p>  </div> <div style="text-align: center;"> <p>Global Thresholding (v = 127)</p>  </div> <div style="text-align: center;"> <p>Adaptive Mean Thresholding</p>  </div> <div style="text-align: center;"> <p>Adaptive Gaussian Thresholding</p>  </div> </div>	



## Ethernet : Client pour serveur industriel type Sollae

[https://python.developpez.com/cours/TutoSwinnen/?page=page\\_20#L18.3](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 #break hum, peut mieux faire : événements
    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 bientôt
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

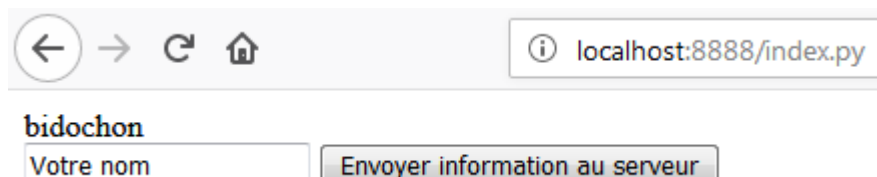
form = cgi.FieldStorage()
print("Content-type: text/html; charset=utf-8\n")

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



bidochon

Votre nom

Envoyer information au serveur

<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))

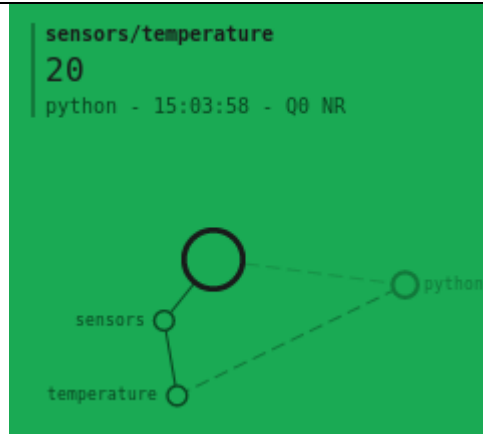
client = mqtt.Client("python") #id must be unique
client.username_pw_set("weatherSensors", "bme280Sensors") #login, password
client.on_message = on_message

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

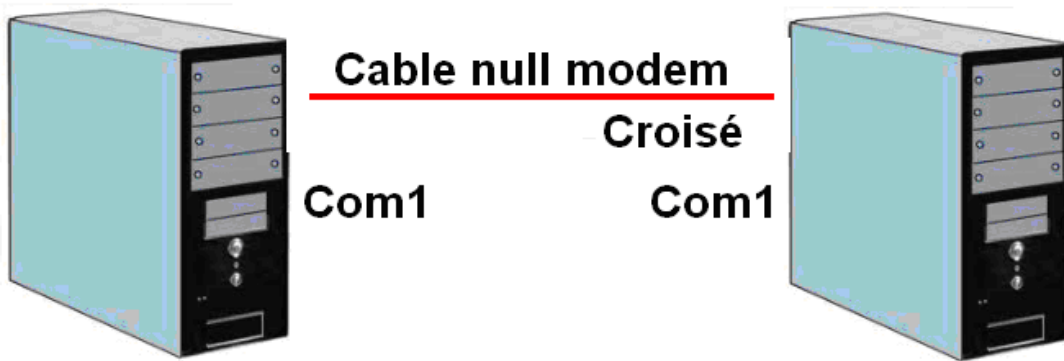
client.subscribe("/sensors/temperature")

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



## 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' b'0123456789'
```

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

## Encore du tri...

	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]&lt;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], '&lt;--&gt;', x[i], ':', x)         compteur+=1  print("Apres tri croissant : ", x) print("Nombre d'echanges : " , compteur) </pre>
Résultat	<pre> Tri par selection : Avant tri :  [128, 291, 9, 57, 107, 330, 5] 128 &lt;--&gt; 5 : [5, 291, 9, 57, 107, 330, 128] 291 &lt;--&gt; 9 : [5, 9, 291, 57, 107, 330, 128] 291 &lt;--&gt; 57 : [5, 9, 57, 291, 107, 330, 128] 291 &lt;--&gt; 107 : [5, 9, 57, 107, 291, 330, 128] 291 &lt;--&gt; 128 : [5, 9, 57, 107, 128, 330, 291] 330 &lt;--&gt; 291 : [5, 9, 57, 107, 128, 291, 330] Apres tri croissant :  [5, 9, 57, 107, 128, 291, 330] Nombre d'echanges :  6 </pre>

	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&gt;0 and k&lt;x[j-1]:         x[j]=x[j-1]         j-=1     x[j]=k     print(i+1,"premieres valeurs trieées: ",x)  print("Apres tri croissant : ", x) </pre>
Résultat	<pre> Tri par insertion : Avant tri :  [235, 130, 86, 199, 90, 179, 303] 1 premieres valeurs trieées:  [235, 130, 86, 199, 90, 179, 303] 2 premieres valeurs trieées:  [130, 235, 86, 199, 90, 179, 303] 3 premieres valeurs trieées:  [86, 130, 235, 199, 90, 179, 303] 4 premieres valeurs trieées:  [86, 130, 199, 235, 90, 179, 303] 5 premieres valeurs trieées:  [86, 90, 130, 199, 235, 179, 303] 6 premieres valeurs trieées:  [86, 90, 130, 179, 199, 235, 303] 7 premieres valeurs trieées:  [86, 90, 130, 179, 199, 235, 303] Apres tri croissant :  [86, 90, 130, 179, 199, 235, 303] </pre>
Version récursive	<pre> print("Tri par insertion version recursive : ") init_liste(); print("Avant tri : ",x)  def insere(t,j):     if j&gt;0 and t[j]&lt;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&lt;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]&gt;x[i+1]:             permutation=True             x[i],x[i+1]=x[i+1],x[i]     print("tri en cours : ",x)  print("Après tri croissant : ", x) print("Nombre passages : " , compteur) </pre>
Résultat	<pre> Tri à bulles : Avant tri :  [180, 285, 129, 249, 99, 130, 100] tri en cours :  [180, 129, 249, 99, 130, 100, 285] tri en cours :  [129, 180, 99, 130, 100, 249, 285] tri en cours :  [129, 99, 130, 100, 180, 249, 285] tri en cours :  [99, 129, 100, 130, 180, 249, 285] tri en cours :  [99, 100, 129, 130, 180, 249, 285] tri en cours :  [99, 100, 129, 130, 180, 249, 285] Après tri croissant :  [99, 100, 129, 130, 180, 249, 285] Nombre passages :  6 </pre>



## Recherche dichotomique

	Instructions
Recherche dichotomique	<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] &gt; 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 &lt; b :         if liste_triee[m] == element :             return m         elif liste_triee[m] &gt; 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	<pre> valeur à l'indice 3 valeur à l'indice 3 </pre>

## 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&gt;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&gt;=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.Changer_decouvert(500) print(compte.Retirer(150)) print(compte) </pre>
<pre> 100 100 True 50 False 50 </pre>	<pre> 100 True -50 </pre>

## 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&lt;=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])&gt;=valeur(chaine[1]):             return (valeur(chaine[0]))+calcul(chaine[1:])         else:             return calcul(chaine[1:])-valeur(chaine[0]) print(calcul('MDLIX'))</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

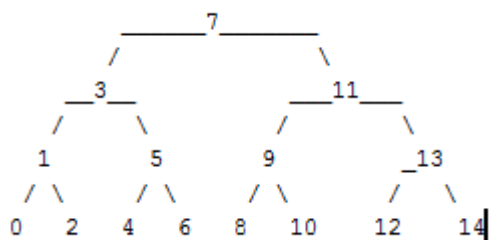
	Instructions	Résultat
Calcul simple	<pre>def calcul(x):     y=2*x     return y  print("pass test1",calcul(3)==6) print("pass test2",calcul(2)!=5)</pre>	pass test1 True pass test2 True
Recherche de caractère	<pre>def rechercher_caractere(chaine,car):     trouve=False     for c in chaine:         if (c==car):             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

## Les arbres binaires

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

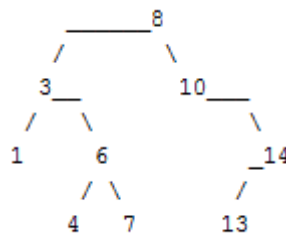
### Mode automatique

```
from binarytree import tree, bst, heap
my_bst = bst(height=3, is_perfect=True)
print(my_bst)
print(my_bst.properties)
```



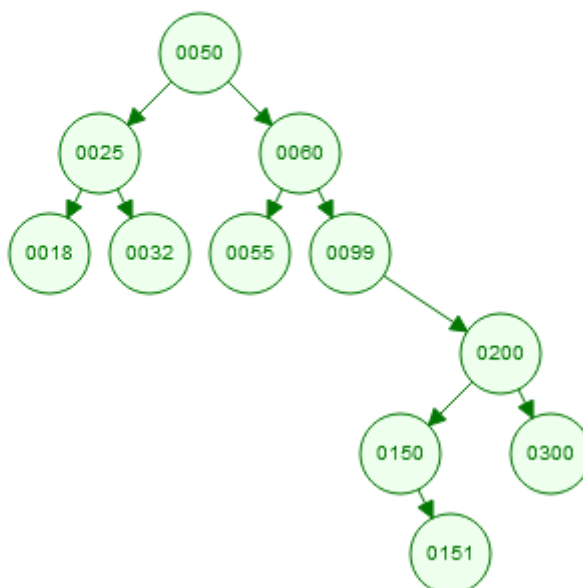
### Mode manuel

```
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)
```



```
{'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}
```

## Binary Search Tree



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

## Parcours préfixe, suffixe, infixe

	Instructions	suite
Parcours d'un arbre ( <a href="http://www.tepacap-lemans.fr/">http://www.tepacap-lemans.fr/</a> )	<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&gt;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:         afficherInfixe(a.filsGauche)     if a.filsDroit!=None:         afficherInfixe(a.filsDroit)  def afficherSuffixe(a):     if a.filsGauche!=None:         afficherInfixe(a.filsGauche)     if a.filsDroit!=None:         afficherInfixe(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,300) inserer(a,150) inserer(a,151)  print("Infixe :") afficherInfixe(a) print("Prefixe :") afficherPrefixe(a) print("Suffixe :") afficherSuffixe(a) </pre>
	Infixe : 18 25 32 50 55 60 99 150 151 200 300	<div> <div>Prefixe :</div> <div> 50  18  25  32  55  60  99  150  151  200  300 </div> </div> <div> <div>Suffixe :</div> <div> 18  25  32  55  60  99  150  151  200  300  50 </div> </div>



## 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&gt;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  print("minimum :",minimum(a)) print("maximum :",maximum(a)) print("nb noeuds :",nbnoeuds(a)) </pre>	
	<pre> minimum : 18 maximum : 300 nb noeuds : 11 </pre>	

## 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

```

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 verifier < 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)

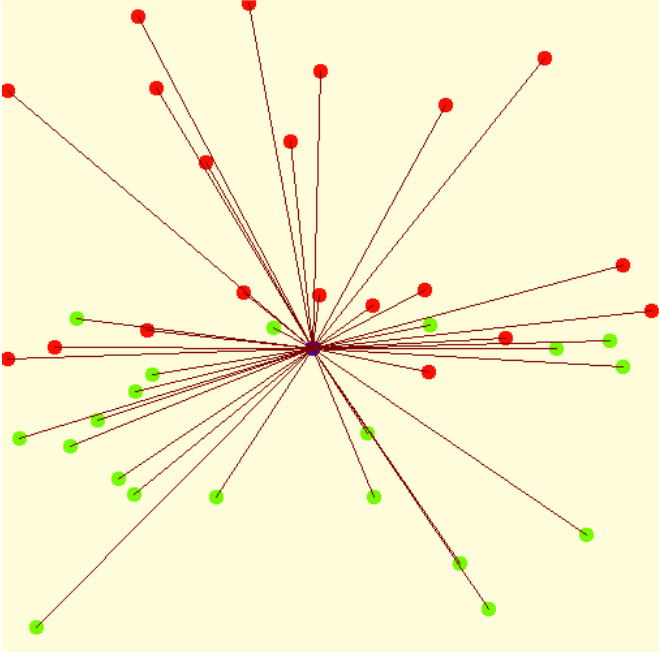
#génère un intrus
def creationintrus(debut,fin):
    x = random.randint(int(debut * largeur), int(fin * largeur))
    y = random.randint(int(debut * hauteur), int(fin * hauteur))
    return x,y

#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])

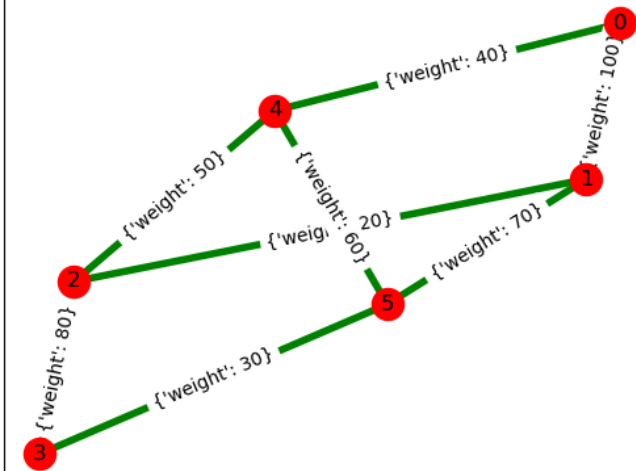
```

	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&gt;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>

Instructions	
Régression linéaire	<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 a 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	<div data-bbox="183 1467 853 1948"> </div> <div data-bbox="869 1456 1548 1948"> <pre> nombre de valeurs dans le fichier csv 96 ----- Y= 1.213547253908358 * X + - 4.211504005424086 Y = 20.38709883129833 pour X = 20.27 </pre> </div>

	Instructions
Graphe 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	<div data-bbox="204 1366 842 1836"> </div> <div data-bbox="874 1350 1559 1850"> <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> </div>

	Instructions
Graphe orienté (2)	<pre> #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 </pre>
Résultat	<div data-bbox="188 1411 861 1937"> </div> <div data-bbox="877 1411 1548 2016"> <pre> [[ 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}) <b>plus court chemin ['a', 'd', 'g']</b> Name: Type: DiGraph Number of nodes: 7 Number of edges: 8 Average in degree: 1.1429 Average out degree: 1.1429 </pre> </div>

	Instructions
Graphe (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-Moore

[https://fr.wikipedia.org/wiki/Algorithme\\_de\\_Boyer-Moore](https://fr.wikipedia.org/wiki/Algorithme_de_Boyer-Moore)

	Instructions
Boyer-Moore	<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 &lt; 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 = 'abacaabadcabacabaabb' 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	<pre> Text:  abacaabadcabacabaabb 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 </pre>



## Fonctions prédéfinies

- ▷ `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 valeur 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 *i*ème 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 *i*ème élément reprend le *i*ème 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 car. 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 string
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 booléen qui vaut False
b=(2*3==6) # b booléen 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  
`Chaîne[i]` → renvoie le caractère de rang i  
astuces :  
`MaChaîne[-1]` → dernier caractère  
`MaChaîne[-2]` → avant dernier caractère, etc...  
`MaChaîne[i:j]` → caractères indexés de i à j-1  
Attention : on en 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[L-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'index *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'index *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
- Génération de listes d'entiers**  
`range(a)` → tous les entiers de [0 ; a[  
`range(a, b)` → tous les entiers de [a ; b[  
`range(a, b, p)` → tous les entiers de [a ; b[ de *p* en *p*

```
-----
# 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

- 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 retournée

Les variables de  
'passage' sont  
appelées arguments

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 retournée par la fct

# programme principal

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

# x variable du programme principal

print(x) # affiche 5

## Importation de bibliothèques – Bibliothèques utiles

## Importer une bibliothèque : 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)
```

## Mathématiques

- Librairie math → fonctions mathématiques

sqrt() → racine carrée sin() → sinus(radian), etc...

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

## 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(maList) → élément de la liste maList

```
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'
```

## Graphiques mathématiques

- Librairie pylab combine deux bibliothèques : 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

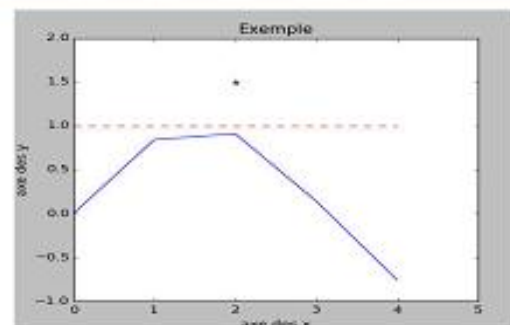
ylabel('texte') → Label de l'axe des y

titre title('texte du titre')

sauvegarde savefig('nomfichier') → sauve au format .png

+ info → [http://matplotlib.org/users/pyplot\\_tutorial.html](http://matplotlib.org/users/pyplot_tutorial.html)

```
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
```



## Fichiers textes



- **Ouverture** `nom = open("fichier.txt", "w")` → en mode lecture ou écriture, pointe en début de fichier

Variable  
qui identifie  
le fichier.

(chemin +)  
nom  
fichier sur

Mode

- "r" lecture
- "w" écriture
- "a" ajout

Écriture mode 'write': crée le fichier ou l'écrit si existe déjà  
`F=open("fich.txt", "w")` # pointe en début de fichier  
`F.write("Début...")`  
`F.write("...suite de la liere ligne"+chr(10))` #chr(10)→nllie ligne  
`F.write("Deuxième ligne")`  
`F.close()` # indispensable: enregistre le fichier

- **Écriture** `nom.write("texte...")`  
 → Si le fichier n'existe pas, il est créé.  
 → Si le fichier existe il est écrasé
- **Lecture** → ouvrir le fichier en mode lecture  
`x= nom.read()` → lecture du fichier entier  
`x= nom.read( n )` → lecture des n caractères suivants.  
`x= nom.readline()` → lecture ligne par ligne.

#Écriture mode 'ajout'  
`F=open("fich.txt", "a")` # pointe en fin de fichier  
`F.write("...suite deuxième ligne")`  
`F.close()` # indispensable: enregistre le fichier

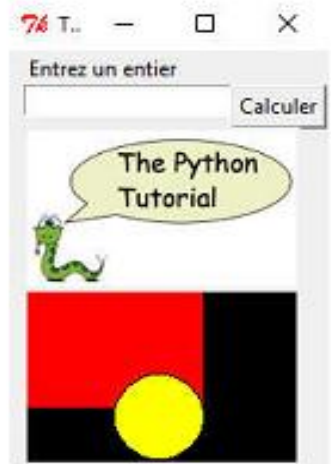
# Lecture (mode 'read')  
`Fichier = open("fich.txt", "r")` # pointe au début du fichier  
`t= Fichier.read()` # lit tout le fichier et l'affecte à variable t  
`Fichier.close()` # indispensable: enregistre le fichier

`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

### Interfaces graphiques avec la librairie tkinter

```
1 # Importation de la librairie graphique tkinter
2 from tkinter import *
3 #-----
4 # Création d'une fenêtre
5 #-----
6 Mafenetre= Tk() # Création d'une fenêtre
7 Mafenetre.title("Tutoriel") # Titre de la fenêtre
8 Mafenetre.geometry("180x250") # Dimensions de la fenêtre
9 #-----
10 # Les "widgets" sont des composants que l'on positionne dans la fenêtre
11 #-----
12 # Widget "Label" (zone d'affichage de texte)
13 Affichage = Label(Mafenetre, text='Entrez un entier') # création d'un Label
14 Affichage.place(x=10,y=0) # 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 =2*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 # Image dans un Label
28 from PIL import Image, ImageTk # importation des librairies images
29 monimage = Image.open("tutorial.png") # Chargement d'une image à partir de PIL
30 photo = ImageTk.PhotoImage(monimage) # Création d'une image compatible tkinter
31 LabelImage = Label( Mafenetre, image=photo) # Label contenant une image
32 LabelImage.place(x=10,y=45) # positionnement
33 #-----
34 # Un Canvas est une zone de la fenêtre pour images ou dessins
35 #-----
36 MonCanvas = Canvas(Mafenetre,bg="black",width = 150, height =100)
37 Rectangle = MonCanvas.create_rectangle(0,0,100,70,fill='red') #points opposés
38 Cercle = MonCanvas.create_oval(50,50,100,100,fill='yellow') #x-R, y-R, x+R, y+R
39 MonCanvas.place(x=10, y=140)
40 #-----
41 # IMPORTANT: la fonction mainloop i provoque le démarrage du récepteur
42 # d'événements (clics,clavier,...) associé à la fenêtre
43 #-----
44 Mafenetre.mainloop()
```

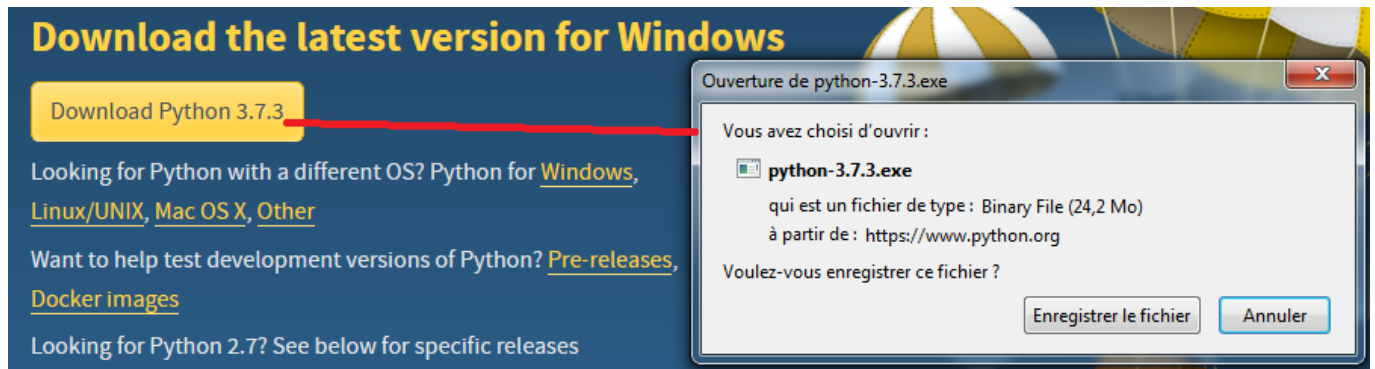




## Installation de pycharm sous Windows

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

C:\Users\anthony>py
Python 3.7.2 (tags/v3.7.2:9a3ffc0492,
<Intel>1 on win32
Type "help", "copyright", "credits" or
>>> exit()

C:\Users\anthony>
```

Installer pycharm, choisir la version community

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



Version: 2019.1.2  
Build: 191.7141.48  
Released: May 8, 2019

[System requirements](#)  
[Installation Instructions](#)  
[Previous versions](#)

## Download PyCharm

Windows macOS Linux

### Professional

For both Scientific and Web Python development. With HTML, JS, and SQL support.

DOWNLOAD

Free trial

### Community

For pure Python development

DOWNLOAD

Free, open-source

## Installation de pycharm sous Linux

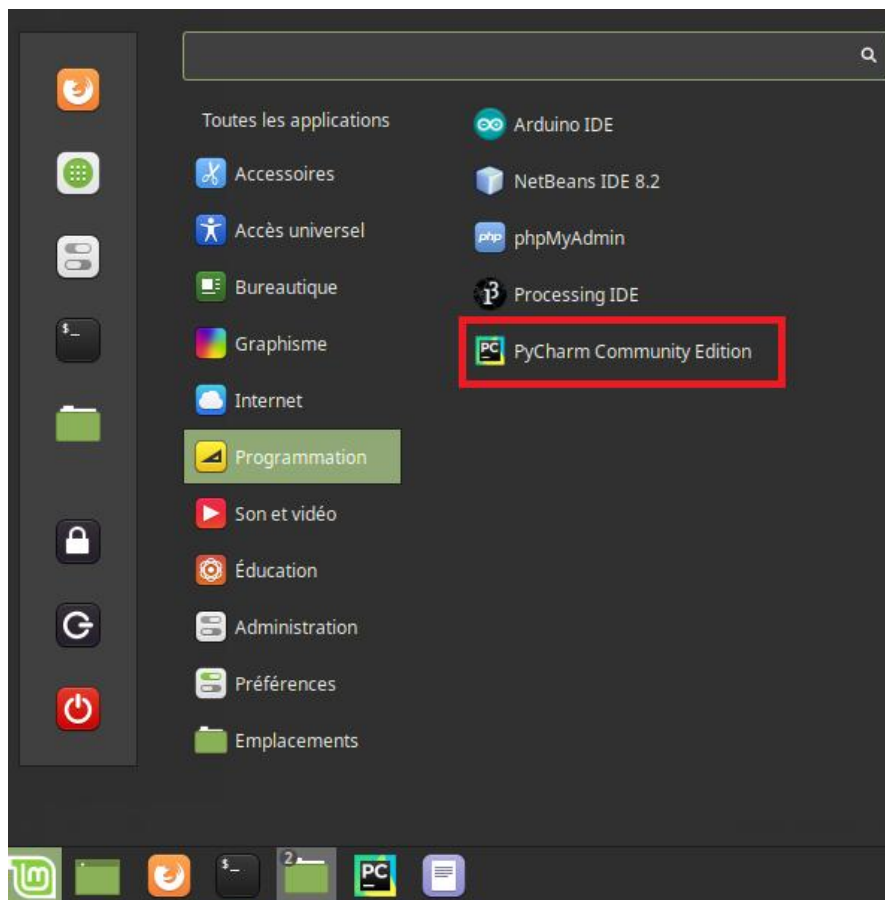
Dans le terminal :

```
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-LIFEBLOCK-A555:~$ python3 -V
Python 3.6.7
```

l'icone de lancement se trouve ici:

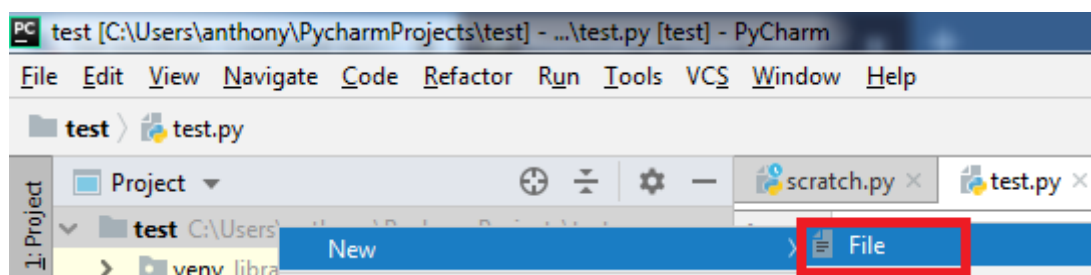
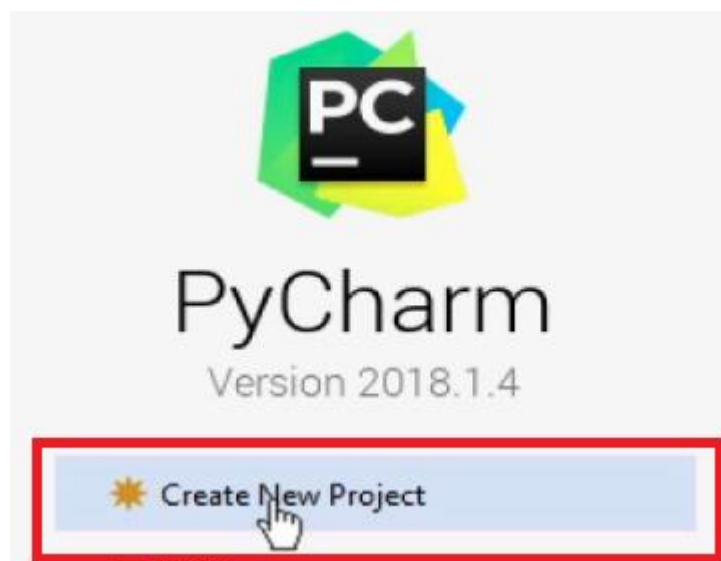
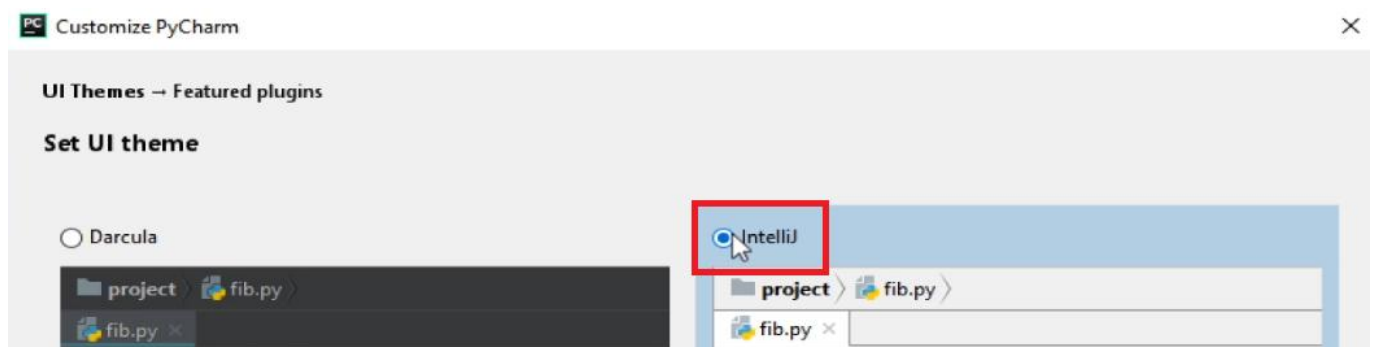
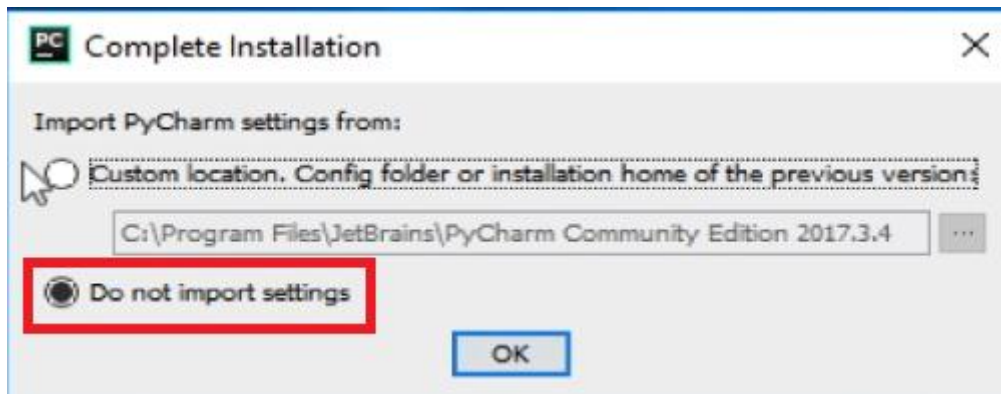
/var/lib/snapd/desktop/application

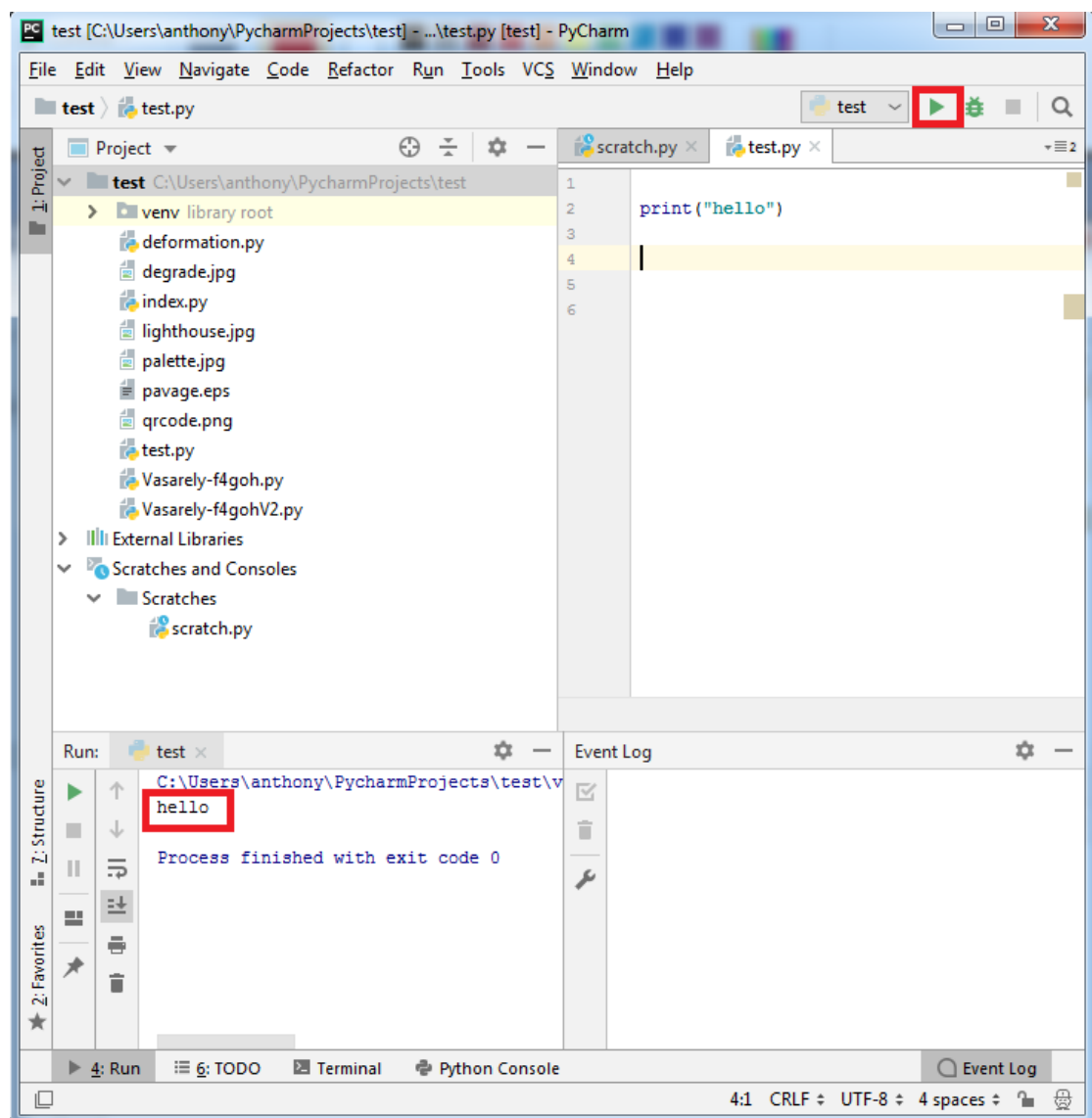
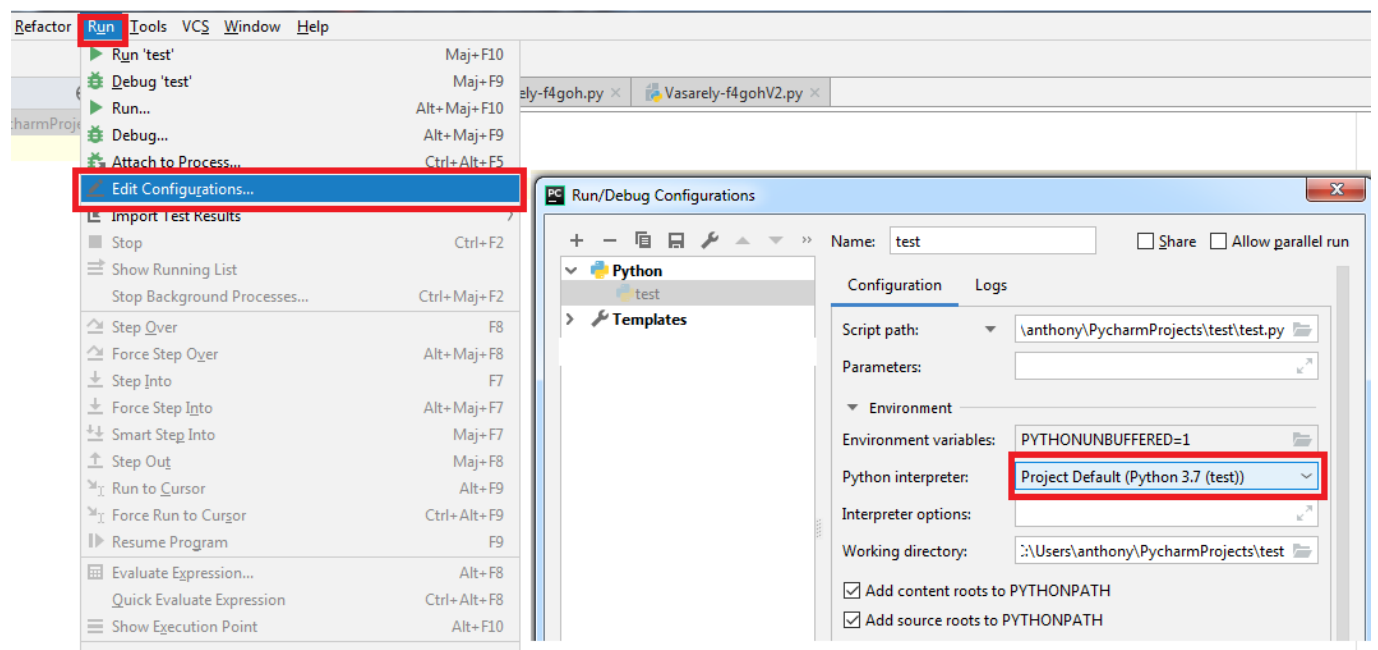
Copier le sur le bureau ou redémarrer le PC, un menu « programmation » apparaîtra avec pycharm



## Configuration de pycharm

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





## Installer des librairies (packages)

