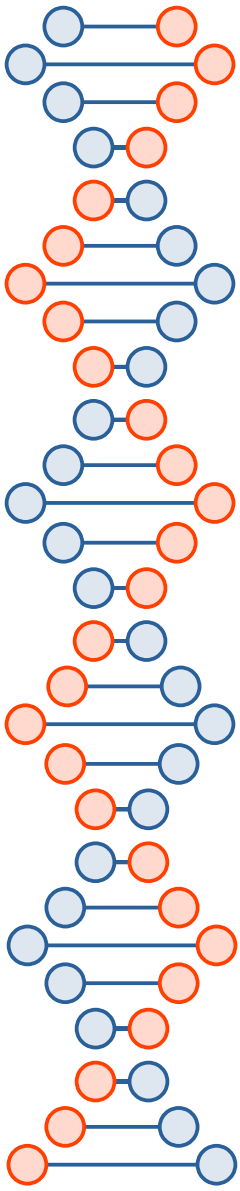


# Présentation du langage de programmation Python3

```
print('Bonjour à tous !')  
  
message = "Bonjour à tous!"  
print(message)  
  
salutation = "Bonjour"  
nom = "à tous !"  
print(salutation + ", " + nom + "!")  
  
nom = "à tous !"  
print(f"Bonjour, {nom}!")  
  
print("Bonjour,")  
print("à tous !")  
  
print("Bonjour, \nà tous!")
```



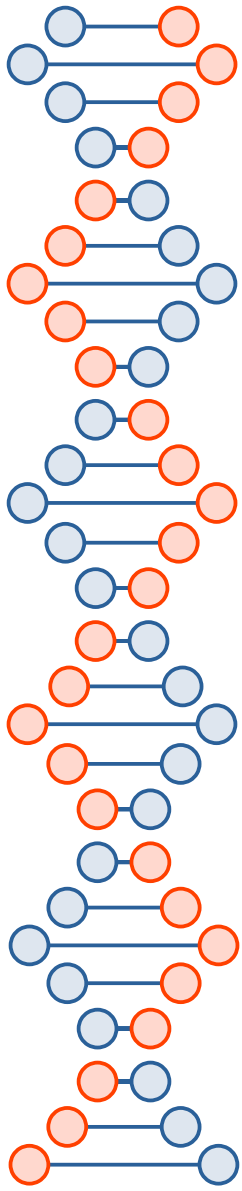


# Présentation du langage de programmation Python3



## *Les différents points abordés*

- 1 [Présentation rapide du langage python](#) : Pourquoi programmer en python? ;
- 2 [Avantages/ Inconvénients de Python3](#)
- 3 [Les différentes versions de Python](#): Version 2, version 3 , Micropython, Cpython
- 4 [Comment programmer](#) , les outils: IDE , Plugins, Les bases
- 5 [Un exemple de programme simple](#) : print(); Choix If ... else; Programme calculatrice
- 6 [Programmer en Ligne avec micropython](#) : Wokwi ....
- 7 [Python dans node-red](#) : Librairies , Exemples
- 8 [Mise en pratique sur votre PC](#): Avec l'aide de ChatGPT... ou pas !



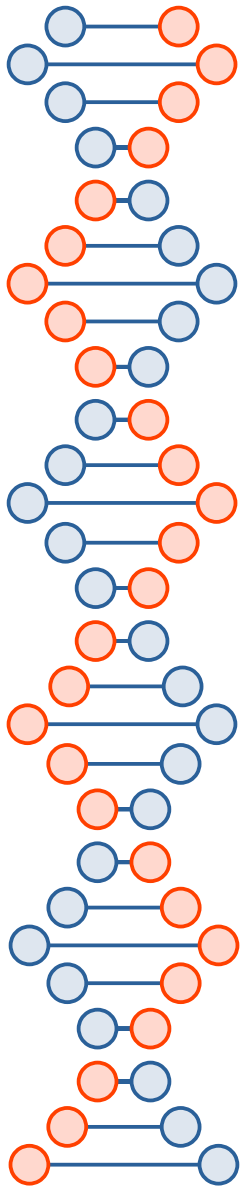
# Présentation du langage de programmation **Python3**



Présentation rapide du langage python ;

## Qu'est-ce que Python ?

- - **Python** est un langage de **programmation interprété**, de haut niveau, et largement utilisé. - Il a été conçu par Guido van Rossum et a été lancé en **1991**.
- - **Python** est connu pour sa **syntaxe simple et claire**, ce qui en fait un **langage populaire** pour les **débutants**.



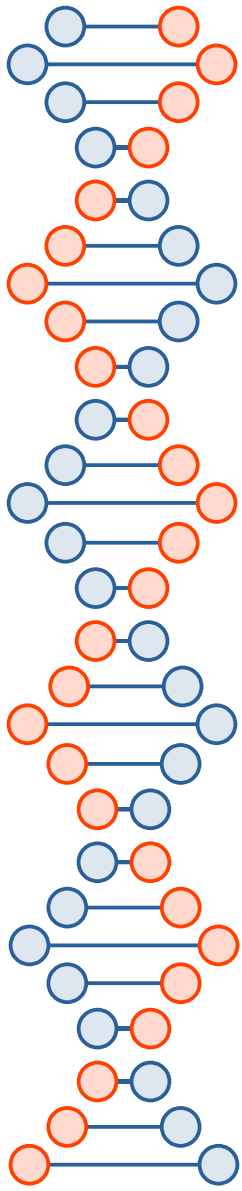
# Présentation du langage de programmation **Python3**



Pourquoi programmer en python? ;

## Pourquoi choisir Python ?

- - **Facilité d'apprentissage** et de lecture.
- - **Polyvalence** : utilisé dans divers domaines tels que le web, la science des données, l'intelligence artificielle, l'automatisation, etc.
- - **Grande communauté** et de nombreuses ressources.



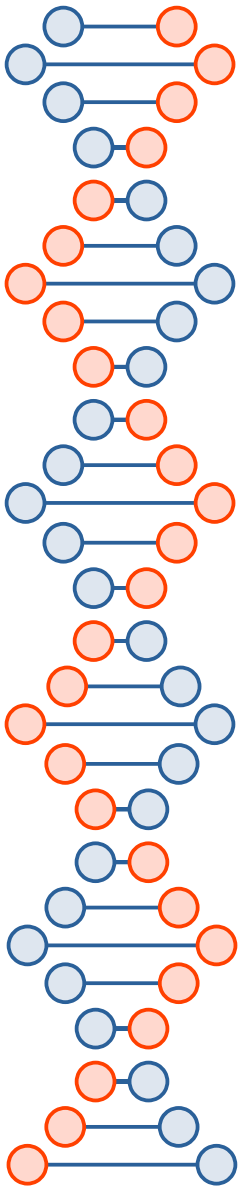
# Présentation du langage de programmation Python3



## Avantages/ Inconvénients

### Avantages de Python3

- **langages de programmation** le plus **populaire** aujourd'hui.
- **Facilité** d'apprentissage et sa lisibilité,
- Dispose de **bibliothèques** intégrées et d'outils tiers
- Permet de travailler dans **différents domaines** tels que l'intelligence artificielle, le traitement des données , le développement web, la robotique, l'apprentissage,...
- Étant un langage **open source (Licence GPL)** , **gratuit**, il est accessible à tous .
- Il est **orienté objets**



# Présentation du langage de programmation

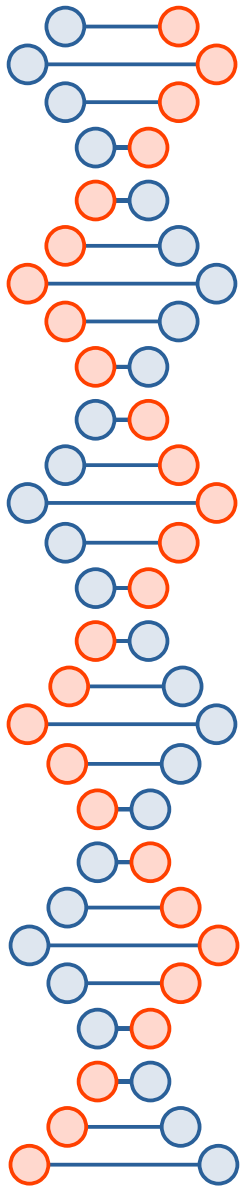
## Python3



Avantages/ Inconvénients

### Inconvénients de Python3

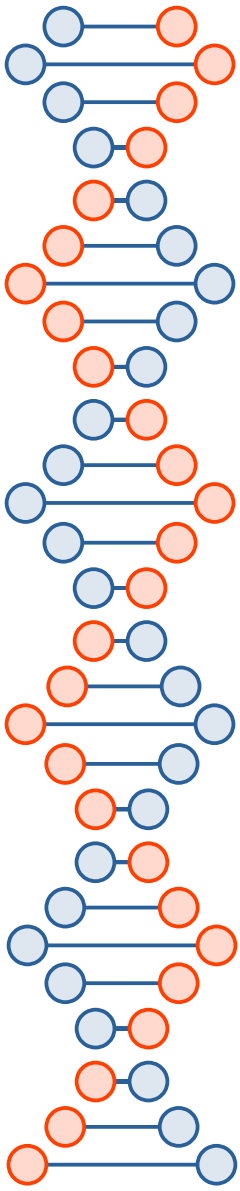
- **Vitesse limitée** par rapport à d'autres langages tels que C++ ou [Java](#).
- Pas idéal pour les appareils mobiles en raison de sa **forte consommation de mémoire**.
- Le Verrouillage Global d'interpréteur ([GIL](#)) : Le GIL **empêche l'exécution simultanée des threads** natifs dans un seul processus ( empêche la fuite de mémoire donc le piratage ou le plantage ...)



# Présentation du langage de programmation Python3

## Les différentes versions de Python:

- Python 2.7 arrêté en janvier 2020
- **Python 3.13** depuis le 7 octobre 2024 ( Fin 10/2029)
- **MicroPython** 1.24.1 , Python pour microcontrôleurs
- **Cpython** 3.13.2 CPython est un **langage compilé** ; il compile le code en **bytecode** avant de le transmettre à l'interpréteur. Tandis que Python est un langage interprété
- **Jython** 2.7.4 Jython est une implémentation de Python **écrite en Java**
- **IronPython** 3.4 IronPython est une implémentation de Python **écrite en C#**



# Présentation du langage de programmation Python3



## les outils: IDE

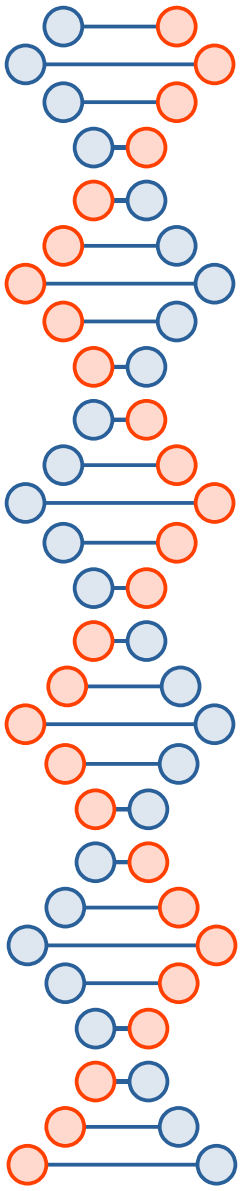
- -1- Thonny
- -2- Visual Studio Code
- -3- Anaconda- Jupyter
- -4- PyCharm Community Edition
- -5- Sublime Text
- -6- Notepad++
- -7- IDLE

```
Python 3.5.3 Shell - test_01.py - /home/pi/test_01.py (3.5.3)
File Edit Shell Debug Options File Edit Format Run Options Window Help
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[[GCC 6.3.0 20170324]] on linux
Type "copyright", "credits" or
"license()" for more informatio
n.
>>>
===== RESTART
T: /home/pi/test_01.py =====
guess a number (1-100): 55
nope
>>>
```

```
Python 3.5.3 Shell - test_01.py - /home/pi/test_01.py (3.5.3)
File Edit Shell Debug Options File Edit Format Run Options Window Help
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[[GCC 6.3.0 20170324]] on linux
Type "copyright", "credits" or
"license()" for more informatio
n.
>>>
===== RESTART
T: /home/pi/test_01.py =====
guess a number (1-100): 55
nope
>>>
```

```
Thonny - /home/pi/Thonny/Python3/Scripts/venv/bin/python3.5 test_01.py
1 import pygame
2 surfFond = pygame.Surface(400, 400)
3 surfFond.fill((255, 255, 255))
4 surfFond.set_colorkey((255, 255, 255))
5
6 class Menu:
7     """Création et gestion des boutons d'un menu"""
8     def __init__(self, application, *groupes):
9         self.application = application
10        self.groupes = groupes
11
12    def __call__(self, *args):
13        self.application.quit()
14
15    def __str__(self):
16        return "Menu"
17
18    def __repr__(self):
19        return "Menu"
20
21
22 if __name__ == '__main__':
23     pygame.init()
24     screen = pygame.display.set_mode((400, 400))
25     menu = Menu(application=pygame.display)
26     menu.show(screen)
27
28     while True:
29         for event in pygame.event.get():
30             if event.type == pygame.QUIT:
31                 pygame.quit()
32                 sys.exit()
33             elif event.type == pygame.MOUSEBUTTONDOWN:
34                 menu(event.pos)
```





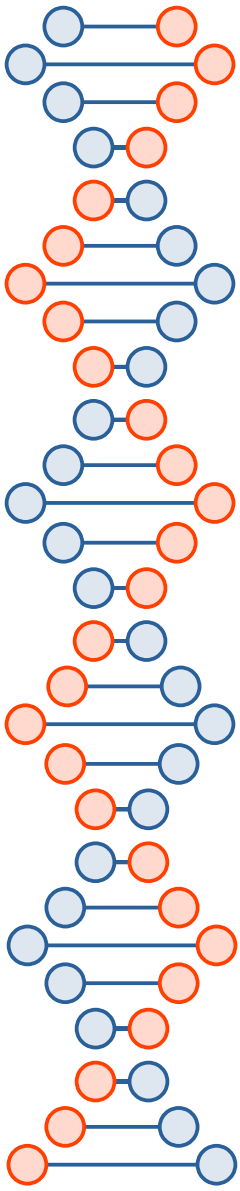
# Présentation du langage de programmation Python3



## Les Plugins

- **Les plugins** sont des extensions de logiciels qui permettent d'ajouter des fonctionnalités supplémentaires à une application existante.
- **Principaux Plugins ou Modules Python**
  - NumPy
  - Matplotlib
  - Django
  - TensorFlow
  - Tkinter
  - OpenCV
  - Pygame
  - PyInstaller
  - Pyserial
  - Time
  - Random
  - Scipy
  - paho-mqtt
  - Pillow

Installer un plugin pour python :  
***python -m pip install numpy***



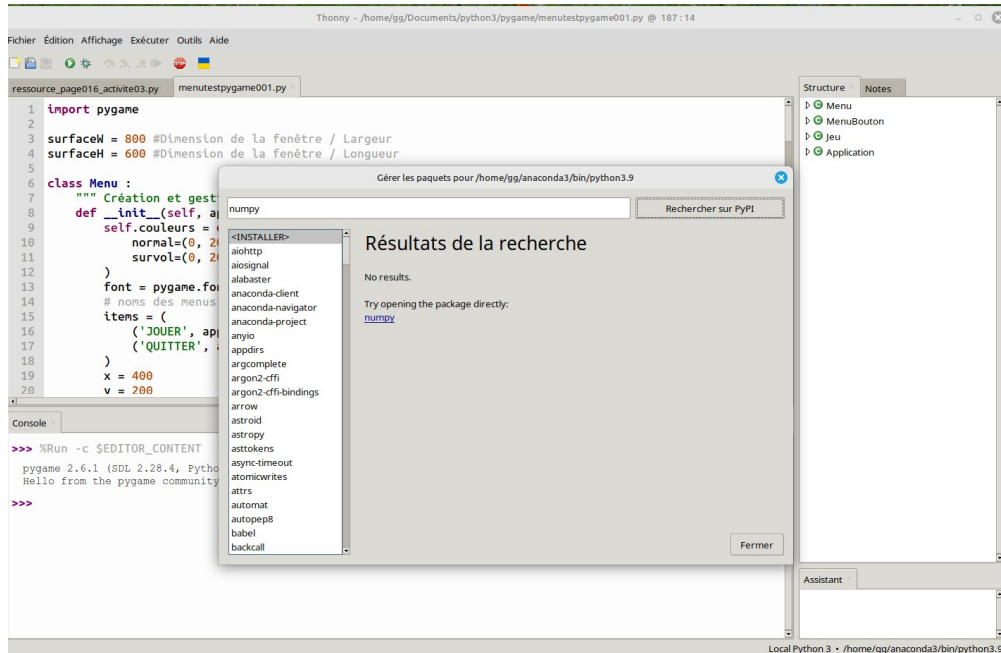
# Présentation du langage de programmation Python3



## Les Plugins

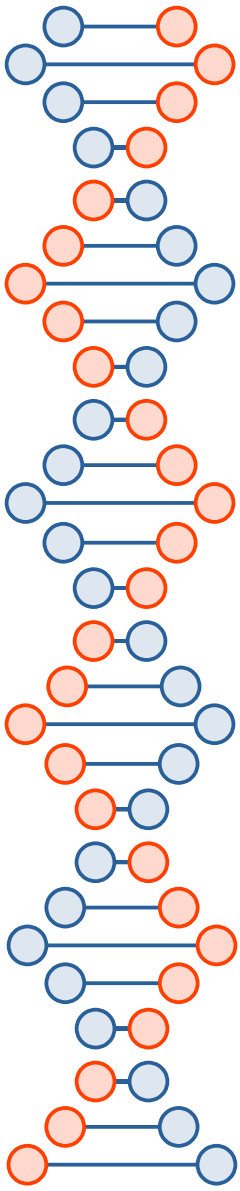
**Ajouter un plugin avec Thonny :**

Outils → Gérer les paquets → nom du paquet → rechercher sur PyPI → sélectionner le paquet → installer



<https://pypi.org/>

Utilisation de l'IDE Thonny  
Thonny doc



# Présentation du langage de programmation Python3



## Programmer avec l'IDE Thonny

### Affichage :

Ouvrir un éditeur de texte comme VSCode, PyCharm, IDLE (inclus avec l'installation de Python)

ou **Thonny- Exemple de code : "Bonjour à tous !" en Python.**

```
print("Bonjour à tous!") #print() est une fonction qui affiche du texte à l'écran.
```

### Variables :

Python est un langage dynamique, les variables n'ont pas besoin d'être déclarées avec un type explicite. - **Types de données de base :**

- **Entiers (int) :** 5 - **Flottants (float) :** 3.14 - **Chaînes de caractères (str) :** "Bonjour" - **Booléens (bool) :** True, False

- **nombres complexes :**  $z = 2 + 3j$ . Exemple de code python : `age = 25`    `nom = "Alice"`    `est_etudiant = True``

### Tests

Conditionnelles (**if, else**) : - Utilisées pour exécuter des blocs de code en fonction d'une condition.

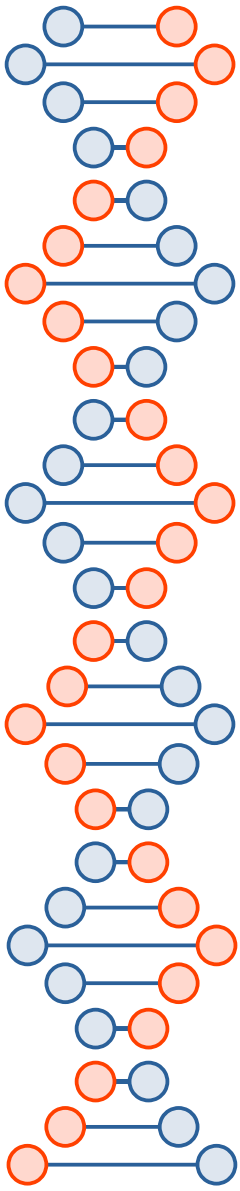
```
age = int( input ("entrez votre age ? "))
```

```
if age >= 18:
```

```
----print("Vous êtes majeur.") # Indentation de 4 espaces ou une tabulation ( ne pas mélanger )
```

```
else:
```

```
----print("Vous êtes mineur.")
```



# Présentation du langage de programmation Python3



## Programmer avec l'IDE Thonny

### Boucles for:

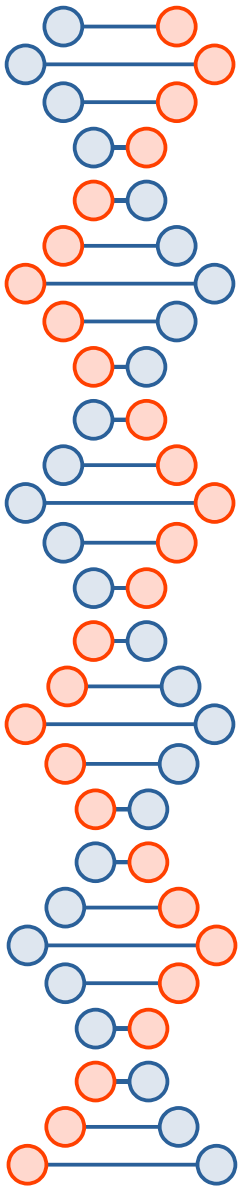
```
for i in [0, 1, 2, 3]:  
    print("i a pour valeur", i)  
ou  
for i in range(4):  
    print("i a pour valeur", i)  
ou parcourir une liste  
c = ["Marc", "est", "dans", "le", "jardin"]  
for i in range(len(c)):  
    print("i vaut", i, "et c[i] vaut", c[i])
```

### Boucles while :

```
x = 1  
while x < 10:  
    print("x a pour valeur", x)  
    x = x * 2  
print("Fin")
```

### Choisir entre for et while :

si on connaît avant de démarrer la boucle, le nombre d'itérations à exécuter, on choisit une boucle for. Au contraire, si la décision d'arrêter la boucle ne peut se faire que par un test, on choisit une boucle while.



# Présentation du langage de programmation Python3



## Programmer avec l'IDE Thonny

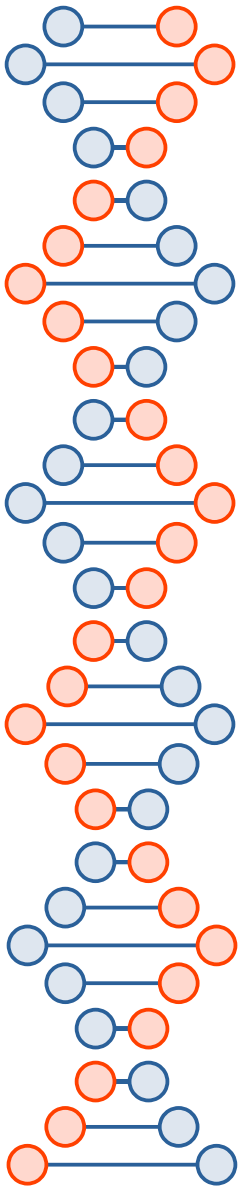
### Une fonction:

Lorsqu'une tâche doit être réalisée plusieurs fois par un programme avec seulement des paramètres différents.

```
def compteur(stop):  
    i = 0  
    while i < stop:  
        print(i)  
        i = i + 1  
compteur(4)  
compteur(2)
```

Si vous voulez découvrir la programmation Objets

<https://courspython.com/classes-et-objets.html>



# Présentation du langage de programmation Python3



## Programmer avec l'IDE Thonny

```
1 import time
2 print("Bonjour à tous !")
3 time.sleep(2)
4 message = "Bonjour à tous!"
5 print(message)
6 time.sleep(2)
7
8 salutation = "Bonjour"
9 nom = "à tous "
10 print(salutation + ", " + nom + "!")
11 time.sleep(2)
12 nom = "à tous !"
13 print(f"Bonjour, {nom}")
14 time.sleep(2)
15 print("Bonjour,")
16 print("à tous")
17 time.sleep(2)
18 print("Bonjour, à tous !")
19 time.sleep(2)
20
```

```
>>> %Run bonjouratous001.py
Bonjour à tous !
Bonjour à tous!
Bonjour, à tous !
Bonjour, à tous !
Bonjour,
à tous
Bonjour,
à tous!
>>>
```

```
import time

print("Bonjour à tous !")
time.sleep(2)

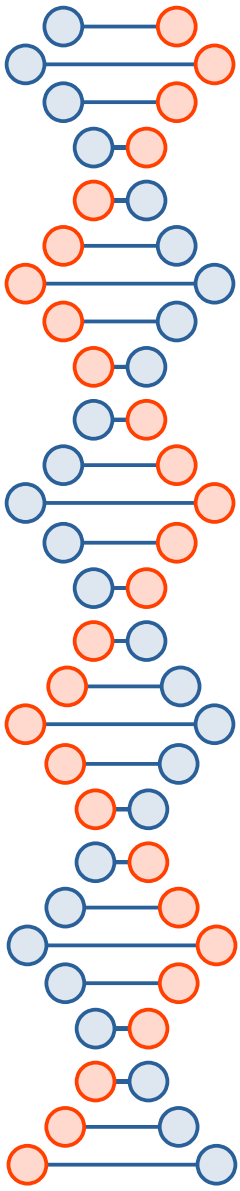
message = "Bonjour à tous!"
print(message)
time.sleep(2)

salutation = "Bonjour"
nom = "à tous "
print(salutation + ", " + nom + "!")
time.sleep(2)

nom = "à tous !"
print(f"Bonjour, {nom}!")
time.sleep(2)

print("Bonjour,")
print("à tous")
time.sleep(2)

print("Bonjour,\nà tous!")
time.sleep(2)
```



# Présentation du langage de programmation Python3



## Exemple de programme de balle que l'on déplace

```
from tkinter import *

# procédure générale de déplacement :
def avance(gd, hb):
    global x1, y1
    x1, y1 = x1 +gd, y1 +hb
    can1.coords(oval1, x1, y1, x1+30, y1+30)

# gestionnaires d'événements :
def depl_gauche():
    avance(-10, 0)

def depl_droite():
    avance(10, 0)

def depl_haut():
    avance(0, -10)

def depl_bas():
    avance(0, 10)

#----- Programme principal -----

# les variables suivantes seront utilisées de manière globale :
x1, y1 = 10, 10          # coordonnées initiales

# Création du widget principal ("maître") :
fen1 = Tk()
fen1.title("Exercice d'animation avec Tkinter")

# création des widgets "esclaves" :
can1 = Canvas(fen1,bg='dark grey',height=300,width=300)
oval1 = can1.create_oval(x1,y1,x1+30,y1+30,width=2,fill='red')
can1.pack(side=LEFT)
Button(fen1,text='Quitter',command=fen1.destroy).pack(side=BOTTOM)
Button(fen1,text='Gauche',command=depl_gauche).pack()
Button(fen1,text='Droite',command=depl_droite).pack()
Button(fen1,text='Haut',command=depl_haut).pack()
Button(fen1,text='Bas',command=depl_bas).pack()

#quit_button = tk.Button(root, text="Quitter", command=root.destroy)

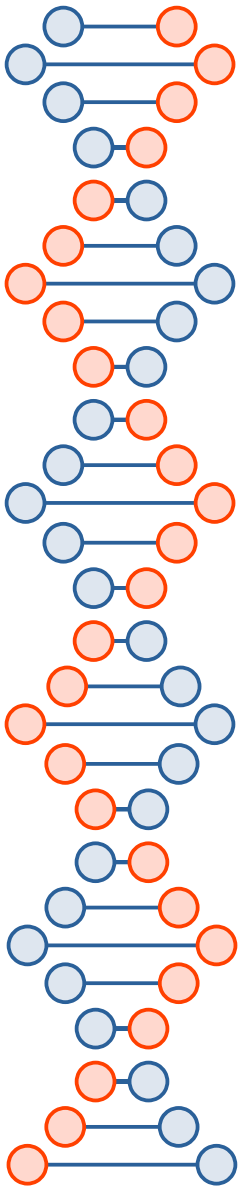
# démarrage du récepteur d'évènements (boucle principale) :
fen1.mainloop()
```

# Présentation du langage de programmation Python3

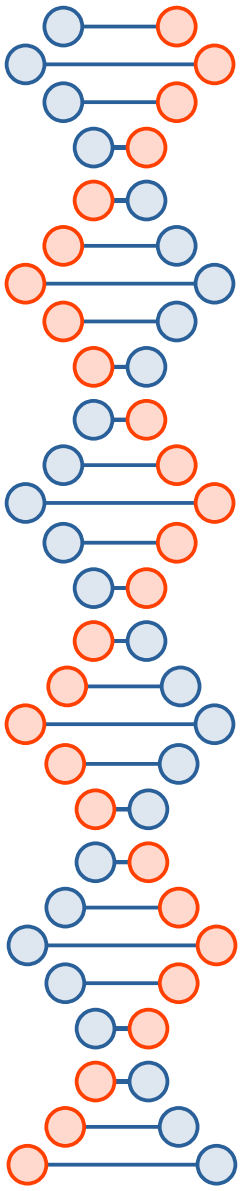


## Exemple de programme de balle que l'on déplace : Explications

Explications du programme :







# Présentation du langage de programmation Python3



## Exemples de programmes de balle qui rebondit

```
# Chutes et rebonds

from tkinter import *

def move():
    global x, y, v, dx, dv, flag
    xp, yp = x, y      # mémorisation des coord. précédentes
    # déplacement horizontal :
    if x > 385 or x < 15 : # rebond sur les parois latérales :
        dx = -dx        # on inverse le déplacement
    x = x + dx
    # variation de la vitesse verticale (toujours vers le bas):
    v = v + dv
    # déplacement vertical (proportionnel à la vitesse)
    y = y + v
    if y > 240:         # niveau du sol à 240 pixels :
        y = 240        # défense d'aller + loin !
        v = -v         # rebond : la vitesse s'inverse
    # on repositionne la balle :
    can.coords(balle, x-10, y-10, x+10, y+10)
    # on trace un bout de trajectoire :
    can.create_line(xp, yp, x, y, fill='light grey')
    # ... et on remet ça jusqu'à plus soif :
    if flag > 0:
        fen.after(50,move)
```

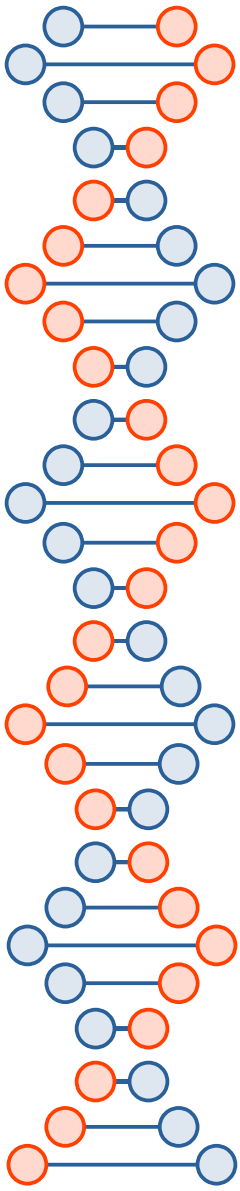
```
def start():
    global flag
    flag = flag +1
    if flag == 1:
        move()

def stop():
    global flag
    flag =0

# initialisation des coordonnées, des vitesses et du témoin d'animation :
x, y, v, dx, dv, flag = 15, 15, 0, 6, 5, 0

fen = Tk()
fen.title(' Chutes et rebonds')
can = Canvas(fen, width =400, height=250, bg="white")
can.pack()
balle = can.create_oval(x-10, y-10, x+10, y+10, fill='red')
Button(fen, text='Start', command =start).pack(side =LEFT, padx =10)
Button(fen, text='Stop', command =stop).pack(side =LEFT)
Button(fen, text='Quitter', command =fen.destroy).pack(side =RIGHT, padx
=10)
#quit_button = tk.Button(root, text="Quitter", command=root.destroy)

fen.mainloop()
```

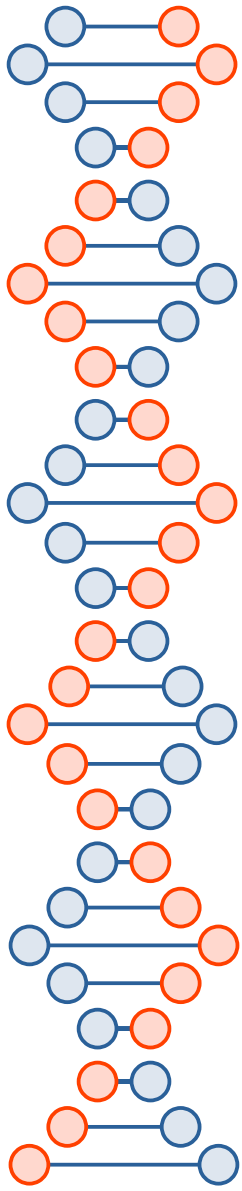


# Présentation du langage de programmation Python3



## Exemples de programmes de balle qui rebondit : Explications

Explications du programme :



# Présentation du langage de programmation Python3

## Python3

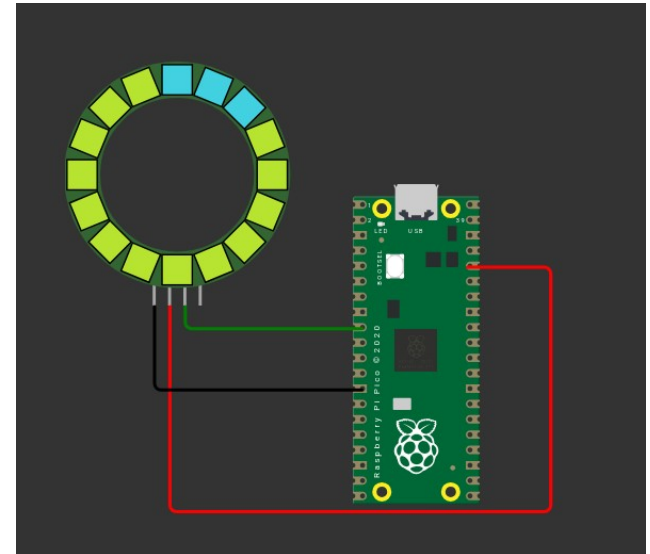


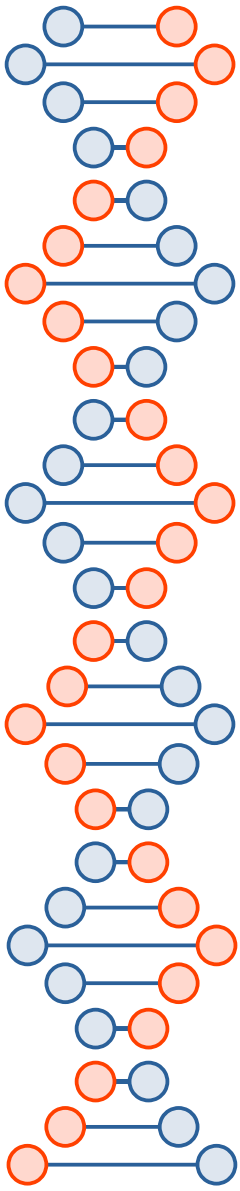
### Programmer en Ligne avec micropython : Wokwi ....

Pour les tests , on va utiliser le [site Wokwi](https://wokwi.com) qui permet de programmer en virtuel des microcontrolleur en micropython. Un exemple avec un anneau Neopixel

<https://wokwi.com/projects/314265138001609280>

```
1 import time
2 from neopixel import Neopixel
3 pixels = Neopixel(17, 0, 6, "GRB")
4
5 colors = [
6     (0xb6, 0xe4, 0x30),
7     (0x42, 0xd1, 0xe0),
8 ]
9
10 pixel_index = 0
11 color_index = 0
12 while True:
13     pixels.set_pixel(pixel_index, colors[color_index])
14     pixels.show()
15     #print(pixel_index, colors[color_index])
16     pixel_index += 1
17     if pixel_index == 16:
18         pixel_index = 0
19         color_index = (color_index + 1) % 2
20     time.sleep(0.1)
21
```





# Présentation du langage de programmation Python3



Programmer en Ligne avec micropython : Wokwi ....

Programme neopixel en micropython sur Wokwi

## Explications



# Présentation du langage de programmation Python3



Python dans node-red: Librairies

@background404/node-red-contrib-python-venv

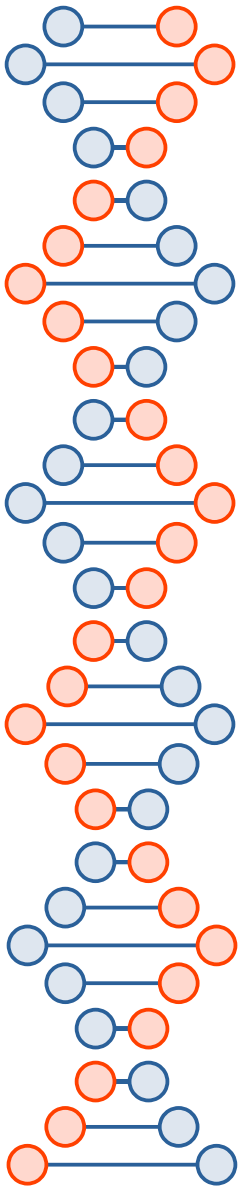
node-red-contrib-pythonshell

node-red-contrib-python-function-ps

The screenshot shows the 'Paramètres de l'utilisateur' (User Parameters) panel in Node-RED. It displays a list of installed Python libraries under the 'Environment' section. The search bar contains 'python'. The list includes:

- @background404/node-red-contrib-python-venv (version 0.5.3, 4 nodes)
- node-red-contrib-pythonshell (version 1.5.4, 1 node)
- node-red-contrib-python-function-ps (version 0.1.1, 1 node)

Each entry has a button labeled 'En cours d'utilisation' (In use).



# Présentation du langage de programmation Python3



Python dans node-red : Bibliothèques : exemples

node-red-contrib-python-function-ps

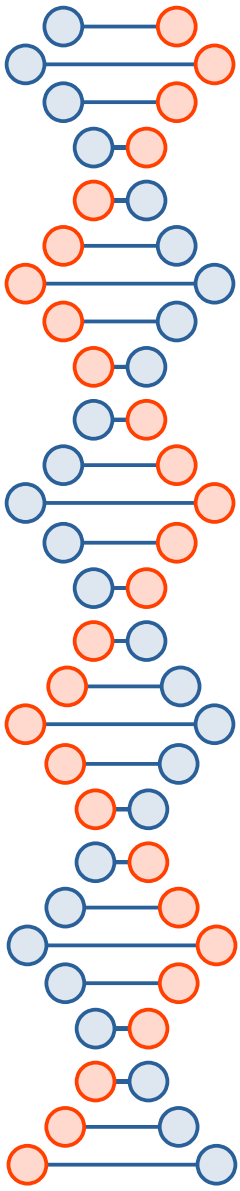
The screenshot shows the Node-RED interface. On the left, a workflow is visible on a grid background, consisting of three nodes: 'horodatage', 'python-function-ps', and 'debug 3'. The 'python-function-ps' node is highlighted, and its configuration panel is open on the right. The configuration panel includes a 'Name' field, 'Settings' tab, and 'Code' tab. The code in the 'Code' tab is as follows:

```
1 # user code here.  
2  
3 a = 1234  
4  
5 node.globals["a"] = a  
6 del node.globals["a"]  
7  
8 write_output("a", a)  
9 msg["payload"] = read_input("a")  
10  
11 return msg  
12
```

Below the code editor, there is a 'Code(Post)' tab. The output of the function is displayed in the bottom right corner of the interface, showing the following debug message:

```
25/02/2025 21:49:51 noeud: debug 3  
msg.payload : number  
1234
```

An orange arrow points from the 'return msg' line in the code editor to the '1234' output value in the debug console.



# Présentation du langage de programmation Python3



Python dans node-red : Librairies : exemples

node-red-contrib-pythonshell

The screenshot shows a Node-RED workflow with three nodes: 'horodatage', 'PythonShell', and 'debug 1'. The 'PythonShell' node is highlighted with a blue box. A blue arrow points from the 'Terminer' button in the configuration panel to the 'PythonShell' node, and a red arrow points from the 'Name' field to the code editor.

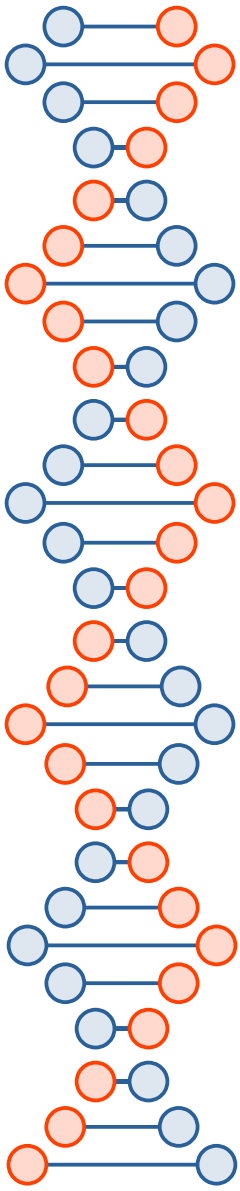
```
test001.py
for i in range(5):
    print(i)
    print(" ")
    print("ca passe ou ça
casse !!!!")
```

**Configuration Panel (Propriétés):**

- Name: test001
- Py file: /home/pi/test001.py
- Virtual Environment Path: /home/user/venv
- Continuous?:
- Stdin Input?:

**Output Log:**

```
25/02/2025 21:36:30 noeud: debug 1
msg.payload : string[46]
> "0..1..2..3..4..
ça casse !!!!"
```



# Présentation du langage de programmation Python3



Mise en pratique sur votre PC: Avec l'aide de ChatGPT...

Consigne à ChatGPT : Créer un programme qui trace un cercle et autour trace un rectangle en python.

```
programme trace un cercle et autour trace un rectangle en python

Voici un programme Python qui utilise la bibliothèque turtle pour dessiner un cercle avec un rectangle autour de ce cercle. Le rectangle entoure le cercle de manière à ce que le cercle soit parfaitement inscrit dans le rectangle.

Programme pour dessiner un cercle avec un rectangle autour :

python
import turtle

# Initialisation de la fenêtre turtle
fenetre = turtle.Screen()
fenetre.bgcolor("white")

# Initialisation de la tortue
t = turtle.Turtle()
t.speed(2)

# Fonction pour dessiner un cercle
def dessiner_cercle(rayon):
    t.penup()
    t.goto(0, -rayon) # Placer la tortue en bas du cercle
    t.pendown()
    t.circle(rayon) # Dessiner le cercle avec le rayon spécifié

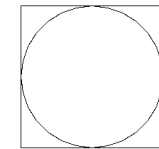
# Fonction pour dessiner un rectangle autour du cercle
def dessiner_rectangle(rayon):
    largeur = 2 * rayon # Le rectangle doit avoir la même largeur que le diamètre du cercle
    hauteur = 2 * rayon # Le rectangle doit avoir la même hauteur que le diamètre du cercle
    t.penup()
    t.goto(-rayon, rayon) # Placer la tortue au coin supérieur gauche du rectangle
    t.pendown()

for _ in range(2):
    t.forward(largeur)
    t.right(90)
    t.forward(hauteur)
    t.right(90)

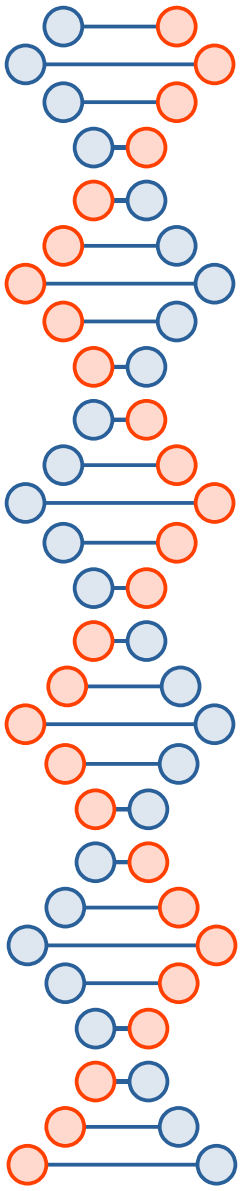
# Paramètre du cercle
rayon_cercle = 100
```

Programme

```
Fichier Edition Affichage Exécuteur Outils Aide
c:\users\p... \cours\python3\...
1 import turtle
2 # Initialisation de la fenêtre turtle
3 fenetre = turtle.Screen()
4 fenetre.bgcolor("white")
5
6
7 # Initialisation de la tortue
8 t = turtle.Turtle()
9 t.speed(2)
10
11 # Fonction pour dessiner un cercle
12 def dessiner_cercle(rayon):
13     t.penup()
14     t.goto(0, -rayon) # Placer la tortue en bas du cercle
15     t.pendown()
16     t.circle(rayon) # Dessiner le cercle avec le rayon spécifié
17
18 # Fonction pour dessiner un rectangle autour du cercle
19 def dessiner_rectangle(rayon):
20     largeur = 2 * rayon # Le rectangle doit avoir la même largeur que le diamètre du cercle
21     hauteur = 2 * rayon # Le rectangle doit avoir la même hauteur que le diamètre du cercle
22     t.penup()
23     t.goto(-rayon, rayon) # Placer la tortue au coin supérieur gauche du rectangle
24     t.pendown()
25
26     for _ in range(2):
27         t.forward(largeur)
28         t.right(90)
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```





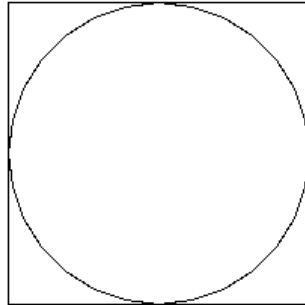


# Présentation du langage de programmation Python3



Créer un programme qui trace  
un cercle et autour trace un  
rectangle en python.

Programme



```
import turtle

# Initialisation de la fenêtre turtle
fenetre = turtle.Screen()
fenetre.bgcolor("white")

# Initialisation de la tortue
t = turtle.Turtle()
t.speed(2)

# Fonction pour dessiner un cercle
def dessiner_cercle(rayon):
    t.penup()
    t.goto(0, -rayon) # Placer la tortue en bas du cercle
    t.pendown()
    t.circle(rayon) # Dessiner le cercle avec le rayon spécifié

# Fonction pour dessiner un rectangle autour du cercle
def dessiner_rectangle(rayon):
    largeur = 2 * rayon # Le rectangle doit avoir la même largeur
    hauteur = 2 * rayon # Le rectangle doit avoir la même hauteur
    que le diamètre du cercle
    t.penup()
    t.goto(-rayon, rayon) # Placer la tortue au coin supérieur
    gauche du rectangle
    t.pendown()

    for _ in range(2):
        t.forward(largeur)
        t.right(90)
        t.forward(hauteur)
        t.right(90)

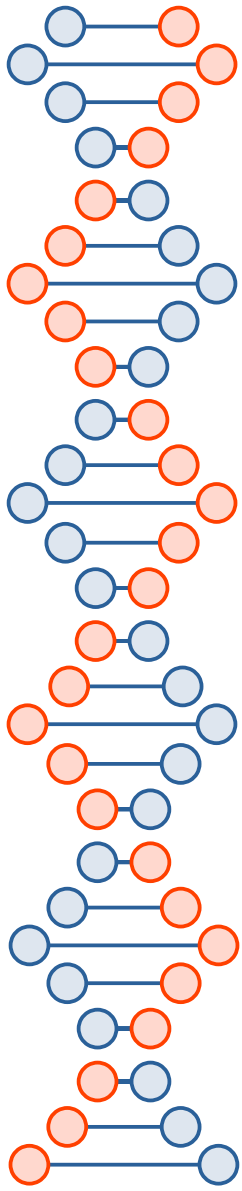
# Paramètre du cercle
rayon_cercle = 100

# Dessiner le cercle
dessiner_cercle(rayon_cercle)

# Dessiner le rectangle autour du cercle
dessiner_rectangle(rayon_cercle)

# Cacher la tortue après le dessin
t.hideturtle()

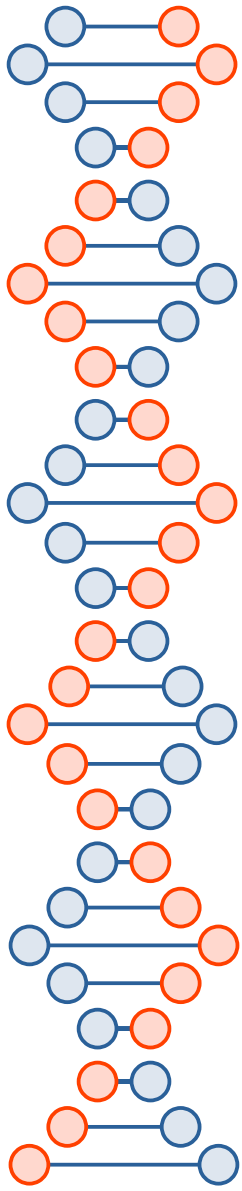
# Maintenir la fenêtre ouverte
turtle.done()
```



# Présentation du langage de programmation **Python3**



# A suivre



# Présentation du langage de programmation **Python3**



Fin

# Présentation du langage de programmation Python3



## L' IDE Python3 : IDLE

The screenshot shows the IDLE Python IDE interface. On the left is the 'IDLE Shell 3.12.3' window, and on the right is the 'test001.py - /home/gg/test001.py (3.12.3)' window.

```
Python 3.12.3 (main, Feb 4 2025, 14:48:35) [GCC 13.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/gg/test001.py =====
29
20
0
Le chiffre cherché est plus grand
20
1
Le chiffre cherché est plus grand
30
2
Le chiffre cherché est plus petit
29
2
Gagné en 3 essai(s) !
>>> KeyboardInterrupt
>>>
```

```
from random import*
compteur=0

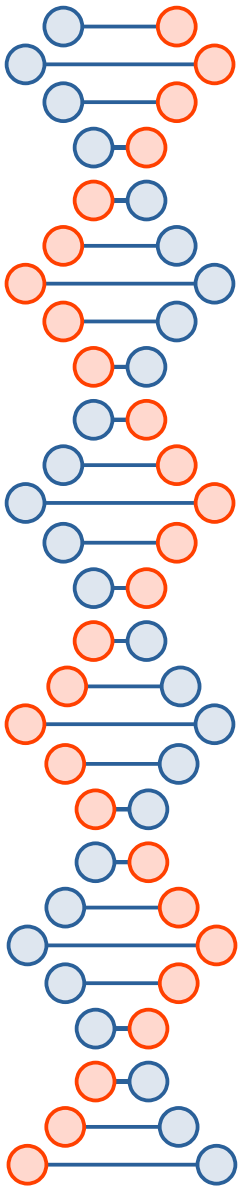
NB_ESSAIS_MAX = 6
n=randint(0,100)
i=-1
print(n)
while i!=n or compteur -1 == NB_ESSAIS_MAX:
    i=int(input())
    print(compteur)
    if i<n:
        print("Le chiffre cherché est plus grand")
        compteur=compteur+1
    elif i>n:
        print("Le chiffre cherché est plus petit")
    elif compteur -1 == NB_ESSAIS_MAX:
        print("Perdu ! Le secret était ", n )
        break
print("Gagné en", compteur + 1 , "essai(s) !")
```

sudo apt-get install idle

OU

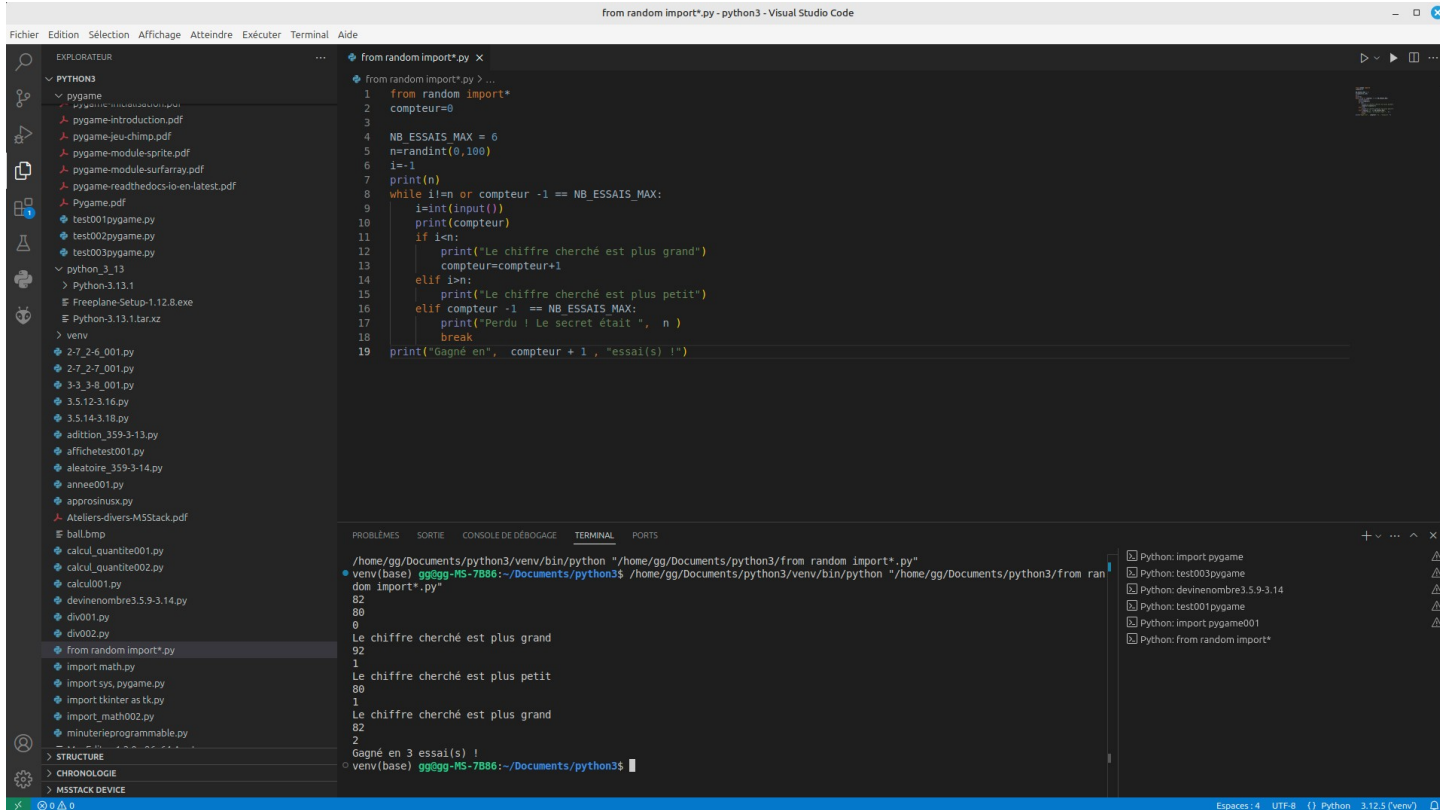
Python3.exe

Retour



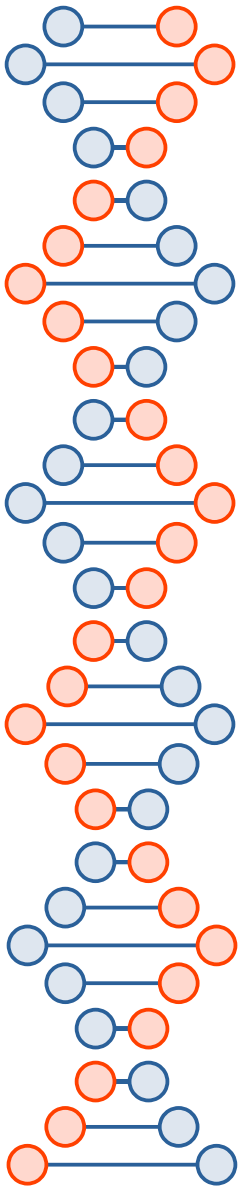
# Présentation du langage de programmation Python3

## L' IDE Python3 : Visual Studio Code



VisualStudioCode

Retour



# Présentation du langage de programmation Python3

L' IDE Python3 : Thonny

```
Thonny - /home/gg/Documents/python3/from random import*.py @ 19:48
Fichier  Édition  Affichage  Exécuter  Outils  Aide
from random import*.py
1  from random import*
2  compteur=0
3
4  NB_ESSAIS_MAX = 6
5  n=randint(0,100)
6  i=-1
7  print(n)
8  while i!=n or compteur -1 == NB_ESSAIS_MAX:
9      i=int(input())
10     print(compteur)
11     if i<n:
12         print("Le chiffre cherché est plus grand")
13         compteur=compteur+1
14     elif i>n:
15         print("Le chiffre cherché est plus petit")
16     elif compteur -1 == NB_ESSAIS_MAX:
17         print("Perdu ! Le secret était ", n )
18         break
19 print("Gagné en", compteur + 1 , "essai(s) !")

Console
58
52
0
Le chiffre cherché est plus grand
100
1
Le chiffre cherché est plus petit
57
1
Le chiffre cherché est plus grand
58
2
Gagné en 3 essai(s) !
>>> |

Assistant
The code in from random import*.py
looks good.

Local Python 3 • /home/gg/anaconda3/bin/python3.9
```

<https://thonny.org/>

Retour