

2ème Série d'exercices

Rappel: Opérateurs de comparaison en Python

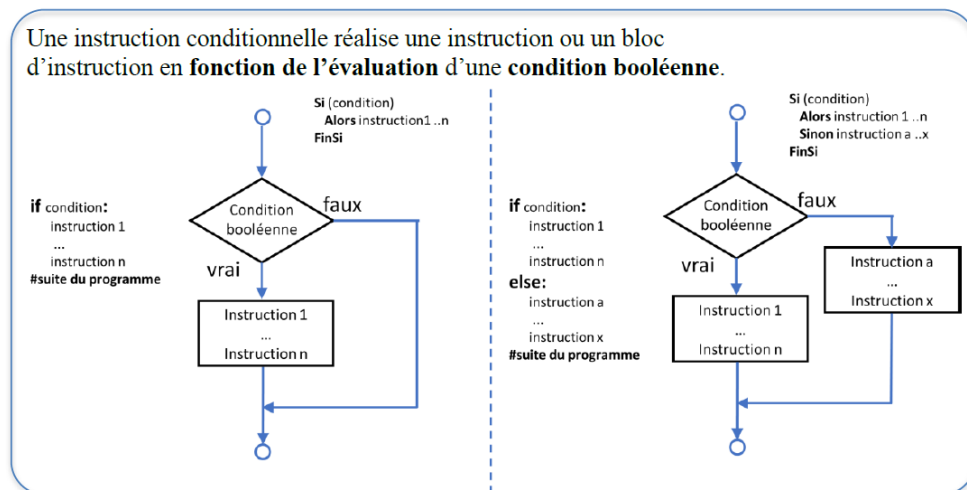
Opérateur	Signification
<code>x == y</code>	x est égal à y
<code>x != y</code>	x est différent de y
<code>x > y</code>	x est plus grand que y
<code>x < y</code>	x est plus petit que y
<code>x >= y</code>	x est plus grand ou égal à y
<code>x <= y</code>	x est plus petit ou égal à y

Exercice 1: exercice1_comp.py

Quels sont les résultats de ces évaluations ?

<code>print(10 > 10)</code>	
<code>print(10 >= 10)</code>	
<code>print(7 == 7)</code>	
<code>print(7 != 7)</code>	
<code>print(7 != 8)</code>	

Partie I : introduction instruction conditionnelle



[Vidéo Structures conditionnelles](#)



Exercice 2: exercice2_comp.py

Que fait le programme suivant ?

```
x = 10
if x > 4:
    print(x, " est plus grand que 4")
print("fin")
```

Exercice 3: exercice3_comp.py

Que fait le programme suivant ?

```
m = 5
if m > 0:
    print(m, " est plus grand que 0")
    print(m, " est strictement positif")
print("fin")
```

Exercice 4: exercice4_comp.py

Demandez à l'utilisateur deux nombres entiers ou décimaux **x** et **y**

- Si **x** est égal à **y**, on affiche dans la console : « les nombres sont égaux ».
- Sinon si **x** est supérieur à **y**, on affiche : « le premier nombre est supérieur au deuxième ».
- Sinon, on affiche : « le premier nombre est inférieur au deuxième ».

Exercice 5: exercice5_comp.py

Écrire un programme qui indique « Il fait chaud » si la température saisie par l'utilisateur est supérieure ou égale à 22°C.

Exercice 6: exercice6_pile_face.py

Pile ou face.

La machine tire un nombre entier pseudo-aléatoire entre 1 et 10 :

```
import random
```

```
nombre = random.randint(1, 10) # génère un nombre entre 1 et 10
```

Si le nombre est plus petit ou égal à 5 : écrire « PILE ».

Si le nombre est plus grand que 5 : écrire « FACE »

forPartie I : introduction itération

Une itération est la **répétition** d'un **bloc d'instructions**.

Boucle bornée :

Le nombre de répétitions est **connu** ou peut être déterminé à l'avance.

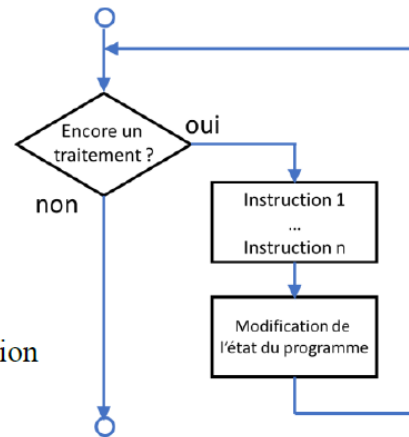
La structure généralement utilisée une boucle **for**.

Boucle non bornée :

Le nombre de répétitions **n'est pas connu** à l'avance.
L'arrêt de la répétition est engendré par une modification de l'état du programme.

Par exemple une condition devenant fausse.

La structure généralement utilisée une boucle **while**



[Vidéo boucle for ... in range\(...\)](#)



L'intervalle de la boucle est décrit par **range**(début, fin, pas) ou **range**(nbItérations).

Exercice 7: exercice7_boucle.py

Que fait le programme suivant ?

Quels sont les nombres affichés ?

```
for i in range(5):  
    print(i)  
print("fin")
```

Exercice 8: exercice8_boucle.py

Que fait le programme suivant ?

```
for j in range(10, 16):  
    print(10*j)  
print("fin")
```

Exercice 9: exercice9_boucle.py

Que fait le programme suivant ?

```
for k in range(0, 10, 2):  
    print(k)  
print("fin")
```

Exercice 10: exercice10_boucle.py

Que fait le programme suivant ?

```
for c in "Hello world":  
    print(c)  
print("fin")
```

Exercice 11: exercice11_carre.py

Écrivez un script permettant de réaliser la figure ci-après. On définit une variable SYMBOLE = "#"

```
#####  
#####  
#####  
#####  
#####
```

Exercice 12: exercice12_triangle.py

Écrivez un script permettant de réaliser la figure ci-après :

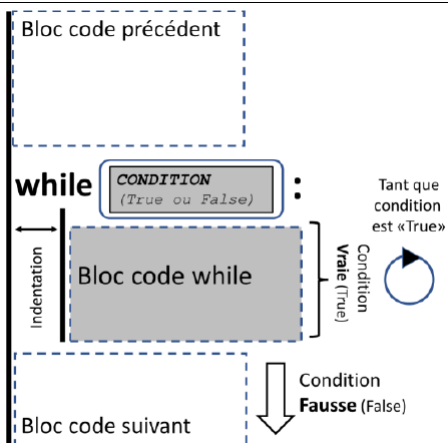
```
#  
##  
###  
####  
#####  
#####  
#####  
#####
```

Tant que la **CONDITION** est vraie (True)

On exécute le bloc code while

Quand la **CONDITION** devient fausse (False)

On exécute le Bloc code suivant



[Vidéo boucle while](#)



Exercice 13: exercice13_while.py

Que fait le programme suivant ?

```
i = 0
while i < 10 :
    i = i + 1
    print(i)
print("fin")
```

Exercice 14: exercice14_while.py

Que fait le programme suivant ?

```
j = 0
s = 0
while j < 10 :
    j = j + 1
    s = s + j
    print(j, s)
print("fin")
```

Exercice 15: exercice15_while.py

Que fait le programme suivant ?

Reconnaissez-vous la fonction mathématique correspondante ?

```
i = 0
fac = 1;
n = int(input("Nombre entier : "))
while n > i:
    i = i + 1
    fac = fac * i
print(fac)
```

Exercice 16: exercice16_somme.py

Écrire un programme qui calcule la somme des nombres saisis. L'utilisateur saisi ses nombres jusqu'à qu'il écrive -1 , ce qui déclenche l'affichage de la somme.

```
somme = 0
nombre = int(input("Entrer un entier : "))
while nombre != -1:
    .....
    .....
.....
```

Vidéo Utiliser les Fonctions

Vidéo Nombre Premier avec exit()

Vidéo Nombre Premier avec Fonction

Exercice 17: exercice17_pair.py

Écrivez un programme qui liste les 100 premiers nombres pairs. Il faut définir une fonction avec le mot clé **def** :

def nombrePair(*nombre*) :

nombre est le paramètre de la fonction. Dans le corps de la fonction les instructions sont indentées vers la droite. [Un nombre est pair si le reste de sa division par 2 est égal à 0. Pour calculer le reste, on utilise l'opérateur modulo %].

def nombrePair(*nombre*) :

```
    reste = .....  
    if reste ..... :  
        return True  
    return .....
```

```
print(nombrePair(10))  # invocation de la fonction nombrePair(nombre), la valeur 10 est copié  
                      # dans le paramètre nombre
```

Exercice 18: exercice18_premier.py

Écrivez un programme qui liste les 100 premiers nombres premiers. Il faut définir la fonction :

def nombrePremier(*nombre*):

nombre est le paramètre de la fonction. Dans le corps de la fonction les instructions sont indentées vers la droite. [Un nombre est premier s'il est divisible uniquement par 1 et par lui-même. Si le reste de la division du nombre par 2, 3, 4, ..., nombre - 1 est égal à zéro, alors le nombre n'est pas premier].

def nombrePremier(*nombre*) :

```
    for diviseur in range(2, .....):  
        reste = %  
        if reste ..... :  
            return .....  
    return True
```

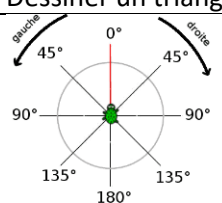
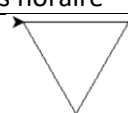
```
print(nombrePremier(13))      # invocation de la fonction nombrePremier(nombre)  
                             # la valeur 13 est copié dans le paramètre nombre
```

```
print(nombrePremier(14))  
print(nombrePremier(9))
```

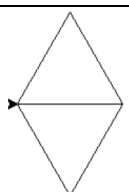
Turtle

Instruction	Résultat/Action
from turtle import *	insérer l'instruction au début du programme pour importer le module <i>turtle</i>
forward(avec une distance) fd(distance)	la tortue avance de ... unités/pixels
back(avec une distance) bk(distance)	la tortue recule de ... unités/pixels
right(avec un angle) rt(angle)	la tortue tourne à droite de l'angle en degrés
left(avec un angle) lt(angle)	la tortue tourne à gauche de l'angle en degrés
up()	relève le crayon
down()	baisse le crayon, recommence à dessiner

Exercice 19: exercice19_triangle.py

Dessiner un triangle de côté 100 la tortue tourne en sens horaire	
	

Exercice 20: exercice20_lostage.py

Dessiner un losange de côté 100 pixel	
---------------------------------------	---

Exercice 21: exercice21_polygone.py

Développez une fonction qui dessine un polygone régulier quelconque. Les paramètres de la fonction sont : nombreCotes, largeurCote. On demande à l'utilisateur les nombre des côtés du polygone et la largeur du côté du polygone

```
from turtle import *
```

```
nombreCotes = int(input("Quelle est le nombre de côtés ? "))
```

```
largeur = int(input("Quelle est la largeur ? "))
```

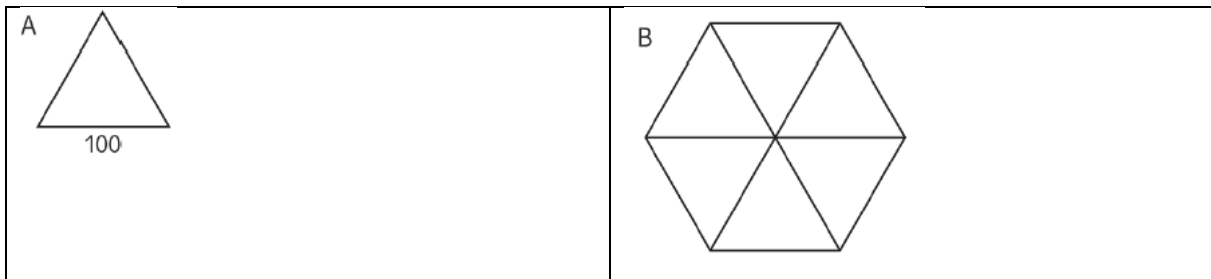
Exercice 22: exercice22_maisons.py

Développez un programme qui vous demande combien de maisons vous souhaitez dessiner. Dès que vous avez saisi un nombre, le programme dessine autant de maisons que vous lui avez demandées. Pour chaque maison, l'ordinateur demande la taille qu'elle doit avoir.

```
from turtle import *
## fonction qui dessine un carré et en dessus un triangle
def maison(taille):
    #dessine un carré
    for i in range(4):
        forward(taille)
        right(360/4)
    #dessine un triangle
    for i in range( ):
        forward( )
        left( )
## fonction qui dessine plusieurs maisons
def plusieurs_maisons(taille, nb_maisons):
    for i in range(nb_maisons)):
        maison(taille)      # appel à la fonction maison
        # relève le crayon  # la fonction est up()
        # la tortue avance  # forward
        # baisse le crayon  # la fonction est down()
# l'ordinateur demande la saisie -> Nombre de maisons :
# l'ordinateur demande la saisie -> Taille de maisons :
done()
```

Exercice 23: exercice23_hexagone.py

Définir une fonction triangle100() pour dessiner un triangle équilatéral de côté 100, comme dans l'illustration A. Utiliser triangle100() pour dessiner une forme comme dans l'illustration B.



Exercice 24: exercice24_croix.py

Définir une commande carre_ouvert() qui dessine un carré ouvert comme dans l'illustration A. Écrire un programme qui utilise cette nouvelle fonction carre_ouvert() pour dessiner une croix comme dans l'illustration.

