

# Fiche Aide Python

## Table des matières

<b>I</b>	<b>Exprimer une condition</b>	<b>1</b>
<b>II</b>	<b>Blocs conditions</b>	<b>2</b>
II.1	Syntaxe . . . . .	2
II.2	Exemple . . . . .	2
II.3	Mises en garde . . . . .	2
<b>III</b>	<b>Boucles</b>	<b>2</b>
III.1	Les boucles tant que . . . . .	2
III.1.1	Syntaxe . . . . .	2
III.1.2	Exemple . . . . .	2
III.1.3	Mise en garde . . . . .	2
III.2	Les boucles for . . . . .	2
III.2.1	Syntaxe . . . . .	2
III.2.2	Mise en garde . . . . .	2
III.2.3	Exemple . . . . .	2
<b>IV</b>	<b>Demander à l'utilisateur</b>	<b>2</b>
IV.1	Du texte . . . . .	2
IV.2	Un nombre . . . . .	3
<b>V</b>	<b>Les fonctions</b>	<b>3</b>
V.1	Syntaxe . . . . .	3
V.2	Mises en garde . . . . .	3
V.3	Exemple . . . . .	3
<b>VI</b>	<b>Autre</b>	<b>3</b>
VI.1	Tester si $a$ est divisible par $b$ . . . . .	3
VI.2	Définir un nombre aléatoire . . . . .	4

## I Exprimer une condition

1. Pour tester si  $a$  est égal à  $b$  :  $a==b$
2. Pour tester si  $a$  est plus strictement plus grand que  $b$  :  $a>b$
3. Pour tester si  $a$  est plus strictement plus petit que  $b$  :  $a<b$
4. Pour tester si  $a$  est plus strictement plus grand ou égal à  $b$  :  $a>=b$

5. Pour tester si  $a$  est plus strictement plus petit ou égal à  $b$  :  $a<=b$

6. Pour tester si  $a$  est différent de  $b$  :  $a!=b$

## II Blocs conditions

### II.1 Syntaxe

```
if <condition> :  
    <code>  
elif <condition> :  
    <code>  
else :  
    <code>
```

Traduction :

1. if veut dire «si»
2. elif veut dire «sinon si»
3. else veut dire «sinon».

### II.2 Exemple

```
age = 18  
if age > 18 :  
    print("Vous êtes majeur")  
elif age == 18 :  
    print("Vous venez d'être  
    ↪ majeur")  
else :  
    print("Vous êtes mineur")
```

### II.3 Mises en garde

1. On commence toujours par if, puis, si besoin, on place un bloc elif, puis, on termine par un else, là encore si besoin
2. Pas de condition après else, on rentre dans ce bloc dès que toutes les autres conditions données par les blocs if et elif ne sont pas vérifiées

## III Boucles

### III.1 Les boucles tant que

#### III.1.1 Syntaxe

```
while <condition> :  
    <code>
```

Traduction :

1. while signifie «Tant que»

#### III.1.2 Exemple

```
nombre = 1  
while nombre*nombre < 99 :  
    nombre = nombre + 1  
# à la fin de la boucle, la  
↪ variable nombre vaut 9, car  
↪ 10*10 > 99
```

#### III.1.3 Mise en garde

1. Vérifier que votre condition n'est pas respectée indéfiniment, sinon vous obtenez une **boucle infinie**.

### III.2 Les boucles for

#### III.2.1 Syntaxe

```
for <variable> in range(<nombre>):  
    <code>
```

Traduction :

1. for signifie «Pour»
2. in signifie «dans»
3. range signifie «dans l'intervalle»

#### III.2.2 Mise en garde

1. range(9) correspond à tous les nombres entiers entre 0 et 8. Cela nous donne bien 9 répétitions de code, mais en commençant à 0.
2. Le nom de la variable après votre for n'importe pas, mais elle augmente toute seule de 1 entre différent passage de boucle.

#### III.2.3 Exemple

```
for i in range(11):  
    print(i*i)  
# Affiche tous les carrés  
↪ d'entiers de 0 à 10
```

## IV Demander à l'utilisateur

### IV.1 Du texte

```
texte = input("message affiché")
```

## IV.2 Un nombre

```
nombre = int(input("message  
→ affiché"))
```

# V Les fonctions

## V.1 Syntaxe

```
def <nom de la fonction>(<var1>,  
→ <var2>, ...):  
    <code>  
    return <nom de variable>
```

Traduction :

1. `def` est un raccourci pour définition qui signifie «Définition»
2. `return` signifie «Retourner»

## V.2 Mises en garde

1. Une fonction peut avoir aucun argument, un seul, ou bien plusieurs.
2. Le mot clé `return` est optionnel, mais si vous ne le mettez pas, vous ne pourrez pas récupérer de résultat.
3. Pour appeler une fonction, on a besoin de son nom, puis entre parenthèses, les arguments qu'on lui envoie (regardez les exemples).

## V.3 Exemple

```
def affichage(messages):  
    print("=====")  
    print(message)  
    print("=====")  
  
affichage("Coucou !")  
  
def rappel():  
    print("Pas besoin d'argument")  
rappel()  
# Affiche : "Pas besoin  
→ d'argument"  
essai = rappel()  
# essai vaut None  
  
def calcul(variable):  
    if variable > 3 :  
        return variable * 3  
    else :  
        return variable * 2  
  
res = calcul(3)  
# res vaut 9  
autreRes = calcul(2)  
# autreRes vaut 4  
  
def minimum(a, b):  
    if a > b :  
        return b  
    else :  
        return a  
petit = minimum(13, 193)  
# petit vaut 13
```

# VI Autre

## VI.1 Tester si $a$ est divisible par $b$

```
if a % b == 0 :  
    print("a est divisible par b")  
else :  
    print("a n'est pas divisible  
→ par b")
```

## VI.2 Définir un nombre aléatoire

```
from random import randint  
  
nombre = randint(1, 100)  
  
# La variable nombre contient un nombre aléatoire entre 1 et 100
```