BASES DE PYTHON

print() — afficher

```
print("Bonjour")      # texte
print(3 + 2)          # calcul
print("x =", x)       # texte + variable
```

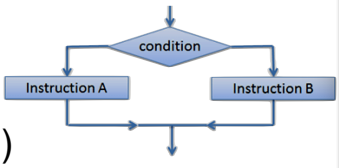
Variables

```
x = 5        # entier
pi = 3.14    # flottant
nom = "Ana" # chaîne
ok = True    # booléen
Age = input  ("entrer votre âge »)
```

= affecte une valeur (ne pas confondre avec ==)

Conditions

```
if age >= 18:
    print("Majeur")
elif age >= 16:
    print("Apprenti")
else:
    print(« Mineur »)
```



Opérateurs : == != < <= > >=

Listes

```
L = [10,20,30]
print(L[0])     # 10
L.append(40)    # ajoute
L[1] = 25       # modifie
n = len(L)      # taille
```

M

É

M

O

P

Y

T

H

O

N

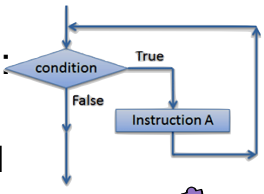
Boucle for

```
for v in L:
    print(v)

for i in range(0,5):
    print(i)
```

Boucle while

```
c = 0
while c < 3:
    print(c)
    c = c + 1
```



Pièges fréquents

- Indentation obligatoire
- = ≠ ==
- Conversions : int(), float(), str()

Notes

FONCTION EN PYTHON





La syntaxe Python pour la définition d'une fonction est la suivante :

```
def nom_fonction(liste de paramètres):
      bloc d'instructions
```

Exemple de fonction sans paramètre

```
def compteur3():
    i = 0
    while i < 3:
        print(i)
        i = i + 1

print("bonjour")
compteur3()
compteur3()
```



Notez bien les parenthèses, les deux-points, et l'indentation du bloc d'instructions qui suit la ligne d'en-tête (c'est ce bloc d'instructions qui constitue le corps de la fonction proprement dite).

Après la définition de la fonction, on trouve le programme principal qui débute par l'instruction print("bonjour"). Il y a ensuite au sein du programme principal, l'appel de la fonction grâce à compteur3().

```
def compteur3():
    i = 0
    while i < 3:
        print(i)
        i = i + 1

def double_compteur3():
    compteur3()
    compteur3()

print("bonjour")
double_compteur3()
```



Une première fonction peut donc appeler une deuxième fonction, qui elle-même en appelle une troisième, etc.

Créer une nouvelle fonction offre l'opportunité de donner un nom à tout un ensemble d'instructions. De cette manière, on peut simplifier le corps principal d'un programme, en dissimulant un algorithme secondaire complexe sous une commande unique, à laquelle on peut donner un nom explicite.

Fonction avec paramètre

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

compteur(4)
compteur(2)
```

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

a = 5
compteur(a)
```