FORMATION PYTON

LES FONCTIONS

LES PROCÉDURES

Une procédure est un morceau de code qui s'exécute chaque fois qu'elle est appelée

```
def  nom_procédure (liste de paramètres):
    Code de la procédure
```

Exemple de procédure :

```
def carre(v1):
    v2=v1*v1
    print('le carre est ', v2)
```

Maintenant, à chaque fois que vous voulez lancer la procédure, il suffit d'appeler :

```
carre(9)
```

Exemple de procédure qui appelle une autre procédure

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

def double_compteur():
    compteur()
    compteur()
```

LES PROCÉDURES

procédure avec paramètres

Avec un seul paramètre

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

Avec plusieurs paramètres

```
def compteur_complet(start, stop, step):
    i = start
    while i < stop:
        print(i)
        i = i + step
```

```
compteur_complet(1, 7, 2)
```

Le langage Python permet de définir des fonctions à nombre variable d'arguments.

```
def add( *arguments ):
    total = 0
    for value in arguments:
        total += value
    return total
```

```
print(add(2,4))
```
→ 6

```
print(add(2,4,4,10))
```
→ 20

A chaque fois que nous définissons des variables à l'intérieur du corps d'une procédure, ces variables ne sont accessibles qu'à la procédure elle-même.

```
def test():
    b = 5
    print(a, b)
a = 2
b = 7
test()
print(a, b)
```

Affichage après exécution :

```
2 5
2 7
```

Une « vraie » fonction (au sens strict) doit en effet renvoyer une valeur lorsqu'elle se termine. Une « vraie » fonction peut s'utiliser à la droite du signe égale dans des expressions telles que y = sin(a).

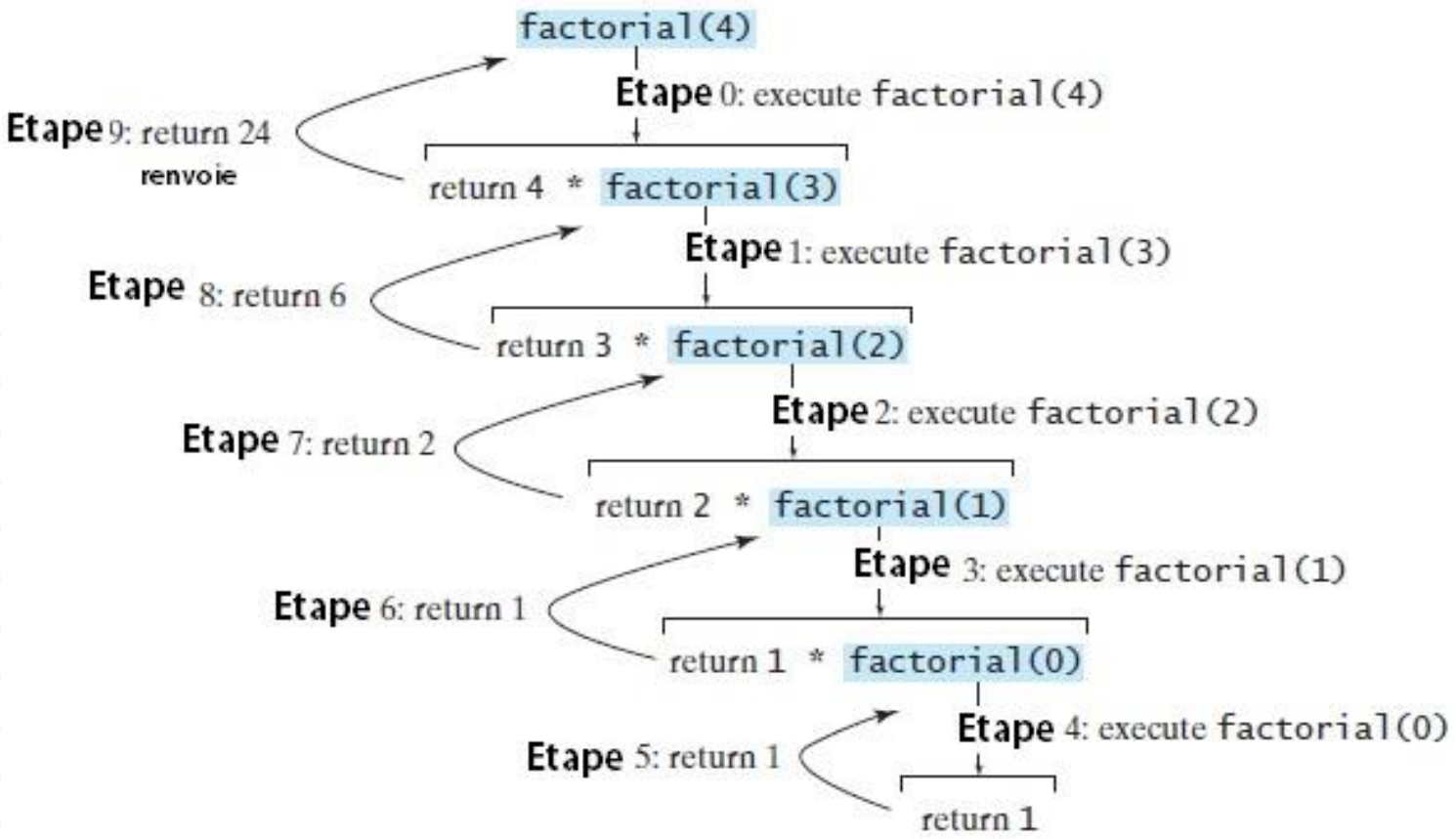
```
def cube(w):
    return w**3
```

```
>>> cube(3)
27
>>> a = cube(4)
>>> a
64
```

Définition: une fonction qui fait appel à elle-même.

Certains des avantages de l'utilisation de la récursivité sont:

- **La récursivité ajoute de la simplicité lors de l'écriture de code, ce qui facilite le débogage.**
- **La récursivité réduit la durée d'exécution d'un algorithme en fonction de la longueur de l'entrée.**
- **La récursivité est également préférée lors de la résolution de problèmes très complexes,**




```
def factorial(n):
    if(n == 0):
            return 1
    else:
        return n*factorial(n-1)
```