6 Fonctions

Remarque. Les exemples de fonctions donnés dans ce résumé n'ont qu'un intérêt pédagogique. □

◇ Il n'est pas nécessaire qu'une fonctions possède des paramètres ou retourne une valeur.

```
def f():
    print "Hello"
f()
```

```
Hello
```

```
def f():
    u=1
    print "Dans f : u =", u
f()
print "Hors de f : u =", u
```

◇ Les variables définies au cours d'une fonction n'ont pas d'existence à l'extérieur : on parle de *variables locales.*

```
Dans f : u = 1 Hors de f : u =
```

◇ Si une variable u existe en dehors de la fonction, sa *valeur* peut être utilisée par la fonction (on dit alors que u est une *variable globale*).

```
u=1
def f():
    print "Dans f : u =", u
f()
print "Hors de f : u =", u
```

```
Dans f : u = 1 Hors de f : u = 1
```

```
u=1
def f():
    u=2
    print "Dans f : u =", u
f()
print "Hors de f : u =", u
```

◇ La valeur de u (variable *globale*) peut être modifiée dans la fonction, mais cette modification restera *locale.*

```
Dans f : u = 2 Hors de f : u = 1
```

⚠ On peut contourner ce comportement avec le mot clé **global** (utilisation très fortement déconseillée).

```
u=1
def f():
    global u
    u=2
    print "Dans f : u =", u
f()
print "Hors de f : u =", u
```

```
Dans f : u = 2 Hors de f : u = 2
```

◇ Les paramètres d'une fonction se comportent comme des *variables locales*.

```
def f(u):
    u=1
    print "Dans f : u =", u
u=2
f(u)
print "Hors de f : u =", u
```

```
Dans f : u = 1 Hors de f : u = 2
```

◇ Si on a besoin d'une fonction qui retourne plusieurs valeurs, on écrit ces valeurs entre parenthèses et séparées par des virgules dans l'instruction **return**.

Exemple Python. On considère les suites (u_n) et (v_n) définies par :

$$u_0 = 1 \text{ et } v_0 = 2$$
$$\forall n \in \mathbb{N},\ u_{n+1} = u_n + v_n \text{ et } v_{n+1} = u_n v_n$$

Écrire une fonction qui calcule u_n et v_n en fonction de n. □

◇ On présente ci-contre différentes manières d'utiliser cette fonction.

Remarque. La phrase qui se trouve entre triples guillemets est la *documentation* de la fonction (on parle de *docstring* en PYTHON). Elle ne joue qu'un rôle informatif et n'intervient pas dans les calculs effectués (explications données en classe). □

```
def u_et_v(n):
    """ Calcule u_n et v_n
    n : entier positif"""
    u=1
    v=2
    for k in range(n):
        t=u+v
        v=u*v
        u=t
    return (u,v)
print u_et_v(3)
```

```
(11, 30)
```

```
(x,y)=u_et_v(3)
print x
```

```
11
```

```
print y
```

```
30
```