Programmation Python Partie 1 : Les fonctions

Fonctions

Une fonction en Python est définie avec le mot clé `def`, suivi par le nom de la fonction, la signature entre parenthèses `()`, et un `:`.

Exemples:

In [111]:
```
def fonction0():
    print("test")
```

In [112]:
```
fonction0()
```

```
test
```

Ajout d'une documentation :

In [113]:
```
def fonction1(s):
    """
    Affichage d'une chaine et de sa longueur
    """
    print("%s est de longueur %d" % (s, len(s)))
```

In [114]:
```
help(fonction1)
```

```
Help on function fonction1 in module __main__:

fonction1(s)
    Affichage d'une chaine et de sa longueur
```

Illustration du typage dynamique

In [115]:
```
fonction1([1, 2, 3])
```

```
[1, 2, 3] est de longueur 3
```

In [116]:
```
fonction1("test")
```

```
test est de longueur 4
```

In [117]:
```
print(fonction1("test"))  # ne retourne rien soit None
```

```
test est de longueur 4
None
```

Les fonctions

Fonctions
Paramètres

Retourner une valeur avec `return`:

In [118]:
```
def square(x):
    """
    Returne le carré de x.
    """
    return x ** 2
```

In [119]:
```
print(square(4))
```

```
16
```

Les fonctions

Fonctions
Paramètres

Retourner plusieurs valeurs:

In [120]:
```
def powers(x):
    """
    Retourne les premières puissances de x.
    """
    return x ** 2, x ** 3, x ** 4
```

In [121]:
```
out = powers(3)
print(type(out))
print(len(out))
print(out[1])
```

```
<type 'tuple'>
3
27
```

In [122]:
```
x2, x3, x4 = powers(3)
print(x2, x3)
```

```
(9, 27)
```

Arguments par défaut

Il est possible de fournir des valeurs par défaut aux paramètres:

In [123]:
```
def ma_fonction(x, p=2, debug=False):
    if debug:
        print("evalue ma_fonction avec x = %s "
              "et l'exposant p = %s" % (x, p))
    return x ** p
```

Les paramètres p et debug peuvent être omis:

In [124]:
```
ma_fonction(5)
```

Out[124]: 25

In [125]:
```
ma_fonction(5, 3)
```

Out[125]:
```
125
```

In [126]:
```
ma_fonction(5, debug=True)
```

```
evalue ma_fonction avec x = 5 et l'exposant p = 2
```

Out[126]:
```
25
```

On peut expliciter les noms de variables et alors l'ordre n'importe plus:

In [127]:
```
ma_fonction(p=3, debug=True, x=7)
```

```
evalue ma_fonction avec x = 7 et l'exposant p = 3
```

Out[127]:
```
343
```

Attention les paramètres ne sont pas copiés:

In [128]:
```
def fonction1(l):
    l += l
    return None

a = [1, 2]
fonction1(a)
print a  # a est modifé
```

```
[1, 2, 1, 2]
```

Pour éviter le problème il faut une copie:

In [129]:
```
def fonction1(l):
    l = list(l)
    l += l
    return None

a = [1, 2]
fonction1(a)
print a  # a est modifé
```

```
[1, 2]
```