Mémo : Python et Algorithmique

Mathématiques de base et Python

Mathématique	Python
$x \times 5$	`x*5`
x^5	`x**5`
$a = b$ (test)	`a == b`
$a \neq b$	`a != b`
$a \leq b$	`a <= b`
$a \geq b$	`a >= b`
$a \leq b \leq c$	`a <= b <= c`
$a < b < c$	`a < b < c`
$a = b = c$	`a == b == c`
VRAI	`True`
FAUX	`False`
NON	`not`
ET	`and`
OU	`or`
$a < b$ **ET** $b < c$	`a < b and b < c`
$a = b$ **ET** $b = c$	`a == b and b == c`
$a = b$ **OU** $b = c$	`a == b or b == c`
$\frac{3}{2}$	`3/2`
$\left\lfloor \frac{3}{2} \right\rfloor$ (partie entière)	`int(3/2)`
« toto »	`"toto"`
« toto » . « tata » = « tototata »	`"toto" + "tata" → "tototata"`
$33 + \frac{5 \times x + 40 \times y}{3}$	`33+(5*x+40*y)/3`
3 est-il divisible par 2 ?	`3 % 2 == 0`
Liste $lst = [5,6,7,8,9]$	`lst = [5, 6, 7, 8, 9]`
$x \in lst$?	`x in lst`
Taille de la liste *lst*	`len(lst)`

Exemples à connaître

Fonction et appel de fonction

```
def f(x):
    return x+1

print(f(1))
```

Fonction en appelant d'autres

```
def a(x):
    return x+1
```

```
def b(x):
    return x**3
```

```
def c(x):
    return a(x) + b(x)
```

Condition si / sinon

```
def d(x):
    if x==0:
        return True
    else:
        return False
```

Condition si / sinon si / sinon

```
def e(x):
    if x<0:
        print("negatif")
    elif x==0:
        print("zero")
    else:
        print("positif")
```

Encadrement

```
def f(x, a, b):
    if a <= x <= b:
        return True
    else:
        return False
```

Encadrement (bis)

```
def f(x, a, b):
    if a <= x and x <= b:
        return True
    else:
        return False
```

Utilisation d'un « ou » logique

```
def f(x, a, b):
    if x == a or x == b:
        return True
    else:
        return False
```

Différence entre fonction et procédure

```
def fonction(x, y):
    return x+y
```

```
def procedure(x, y):
    print(x+y)
```

Notion de pseudo-code

```
FONCTION toto(x) :
    SI x=="toto" :
        RETOURNER VRAI
    SINON:
        RETOURNER FAUX
    FIN SI
```

```
def toto(x):
    if x=="toto":
        return True
    else:
        return False
```

Boucle « pour » : afficher les entiers de 0 à n-1

```
PROCEDURE afficher(n) :
    POUR i = 0, ..., n-1 :
        AFFICHER(i)
    FIN POUR
```

```
def afficher(n):
    for i in range(n):
        print(i)
```

Boucle « tant que » : afficher les entiers de 0 à n-1

```
PROCEDURE afficher(n) :
    i = 0
    TANT QUE i <= n-1 :
        AFFICHER(i)
        i = i+1
    FIN TANT QUE
```

```
def afficher(n):
    i = 0
    while i <= n-1 :
        print(i)
        i = i+1
```

Afficher les éléments d'une liste

```
PROCEDURE afficher(lst) :
    n = taille(lst)
    POUR i = 0, ..., n-1 :
        AFFICHER(lst[i])
    FIN POUR
```

```
def afficher(lst):
    n = len(lst)
    for i in range(n):
        print(lst[i])
```

Somme des éléments d'une liste

```
def somme(lst):
    n = len(lst)
    res = 0
    for i in range(n):
        res = res + lst[i]
    return res
```

Affichage simultané de deux listes

```
def afficher(lst1, lst2):
    n = len(lst1)
    for i in range(n):
        print(lst1[i])
        print(lst2[i])
```

Somme de deux listes terme à terme dans une troisième

```
def somme(lst1, lst2):
    n = len(lst1)
    res = [0]*n
    for i in range(n):
        res[i] = lst1[i] + lst2[i]
    return res
```

Affichage du résultat :

```
>>> res = somme([1,2,3], [3,4,5])
>>> print(res)
[4, 6, 8]
```

Afficher le k-ième élément d'une liste

```
def k_ieme(lst, k):
    # indice bien défini ?
    if 0 <= k < len(lst):
        print(lst[k])
```

Afficher le k-ième élément d'une chaîne de caractères

```
def k_ieme(s, k):
    # indice bien défini ?
    if 0 <= k < len(s):
        print(s[k])
```

Maximum des éléments d'une liste

```
def maxi(lst):
    n = len(lst)
    res = lst[0]
    for i in range(n):
        if lst[i] > res:
            res = lst[i]
    return res
```

Indice d'un élément maximum d'une liste

```
def indice_maxi(lst):
    n = len(lst)
    maxi_ = lst[0]
    res = 0
    for i in range(n):
        if lst[i] > maxi_:
            maxi_ = lst[i]
            res = i
    return res
```

Liste contenant les entiers pairs d'une liste passée en argument

```
def pairs(lst):
    n = len(lst)
    lst2 = []
    for i in range(n):
        if lst[i] % 2 == 0:
            lst2.append(lst[i])
    return lst2
```

Liste donnant la réduction modulo 2 des éléments d'une liste passée en argument

```
def mod_2(lst):
    n = len(lst)
    lst2 = [0]*n
    for i in range(n):
        lst2[i] = lst[i] % 2
    return lst2
```

Afficher les caractères d'une chaîne de caractères

```
def afficher(s):
    n = len(s)
    for i in range(n):
        print(s[i])
```

Symbole mathématique : somme ∑

$\sum_{i=0}^{n-1} i = 0+1+2+3+...+(n-1)$	`def somme(n):`
	`  res = 0`
	`  for i in range(n):`
	`  res = res + i`
	`  return res`

$\sum_{i=0}^{n-1} f(i) = f(0)+f(1)+f(2)+f(3)+...+f(n-1)$	`def somme_f(n, f):`
	`  res = 0`
	`  for i in range(n):`
	`  res = res + f(i)`
	`  return res`

Symbole mathématique : produit ∏

$\prod_{i=1}^{n-1} i = 1 \times 2 \times 3 \times ... \times (n-1)$	`def produit(n):`
	`  res = 1`
	`  for i in range(1, n):`
	`  res = res * i`
	`  return res`

$\prod_{i=0}^{n-1} f(i) = f(0) \times f(1) \times f(2) \times f(3) \times ... \times f(n-1)$	`def produit_f(n, f):`
	`  res = 1`
	`  for i in range(n):`
	`  res = res * f(i)`
	`  return res`

Formules mathématiques

Utilisation des fonctions du module math de Python

```
>>> from math import *
>>> pi
3.141592653589793
>>> cos(3)
-0.9899924966004454
>>> cos(pi)
-1.0
```

Quelques fonctions du module math de Python

Mathématiques	Python
$\ln(3)$	`log(3)`
$\log_{10}(3)$	`log10(3)`
$\log_{2}(3)$	`log2(3)`
e^{3}	`exp(3)`