

Ce document recense des fonctions du cours et des TP que vous devez savoir écrire rapidement et sans erreur. Lorsque plusieurs versions d'une même fonction sont données, toutes les versions sont à connaître.

1 Entiers et flottants

- Renvoie la valeur absolue d'un flottant :

```
def abs(x):
    if x >= 0:
        return x
    else:
        return -x
```

- Renvoie $\sum_{i=1}^n i$:

```
def somme_entiers(n):
    s = 0
    for i in range(1,n+1):
        s += i # s = s + i
    return s
```

- Indique si n est premier :

```
def est_premier(n):
    if n <= 1:
        return False
    for d in range(2,n):
        if n % d == 0:
            return False
    return True
```

- Pour $n \geq 0$, renvoie u_n où :
$$\begin{cases} u_0 = 4 \\ \forall n \geq 0 : u_{n+1} = u_n^2 - u_n + 1 \end{cases}$$

```
def get_u(n):
    u = 4
    for i in range(1,n+1):
        u = u*u - u + 1
    return u
```

- Pour $n \geq 2$, renvoie le plus grand entier $d < n$ tel que d divise n :

```
def plus_grand_div(n):
    for d in range(n-1, 0, -1):
        if n % d == 0:
            return d
```

- Pour $n \geq 0$, renvoie F_n où :
$$\begin{cases} F_0 = 0, F_1 = 1 \\ \forall n \geq 0 : F_{n+2} = F_{n+1} + F_n \end{cases}$$

```
def fibo(n):
    x = 0; y = 1
    for i in range(n):
        tmp = y
        y = y + x
        x = tmp
    return x
```

- Renvoie la somme des chiffres d'un nombre

<pre># Version 1 def somme_chiffres(n): s = 0 while n > 0: s = s + (n % 10) n = n//10 return s</pre>	<pre># Version 2 def somme_chiffres(n): s = 0 for c in str(n): s = s + int(c) return s</pre>
---	--

2 Listes

- Renvoie la somme des éléments de L :

<pre># Version 1 def somme(L): s = 0 for i in range(len(L)): s += L[i] return s</pre>	<pre># Version 2 def somme(L): s = 0 for e in L: s += e return s</pre>
---	--

- Renvoie le maximum des éléments de L (déclenche une erreur si L est vide) :

```
def maximum(L):
    assert L != []
    maxi = L[0]
    for i in range(1, len(L)):
        if maxi < L[i]:
            maxi = L[i]
    return maxi
```

- Indique si e appartient à L :

```
def appartient(L, e):
    for a in L:
        if a == e:
            return True
    return False
```

- Indique si les éléments de L sont rangés par ordre croissant :

```
def est_croissante(L):
    for i in range(len(L)-1):
        if L[i] > L[i+1]:
            return False
    return True
```

- Renvoie la liste $[1^2, 2^2, 3^2, \dots, n^2]$:

<pre># Version 1 def liste_carres(n): L = [] for i in range(1, n+1): L.append(i*i) return L</pre>	<pre># Version 2 def liste_carres(n): return [i*i for i in range(1, n+1)]</pre>
---	---

- Renvoie une liste contenant les éléments pairs de L :

<pre># Version 1 def elem_paires(L): P = [] for e in L: if e % 2 == 0: P.append(e) return P</pre>	<pre># Version 2 def elem_paires(L): return [e for e in L if e % 2 == 0]</pre>
---	--

3 Chaînes de caractères

- Indique si s est un palindrome.

<pre># Version 1 def est_palindrome(s): n = len(s) for i in range(n//2): if s[i] != s[n-1-i]: return False return True</pre>	<pre># Version 2 def est_palindrome(s): return s == s[::-1]</pre>
--	---