



Math93.com

# TD n°1 - Algorithmes

## Fonctions (d'Euler)

### Thèmes

Écriture de fonctions numériques simples, écriture de fonctions comportant plusieurs arguments (dont une autre fonction), documentation des fonctions par les *docstrings*, construction de listes par compréhension, tracés via le module matplotlib.



[http://www.math93.com/images/pdf/algo\\_Python/Td\\_Algorithmes\\_seconde\\_Fonctions\\_1.pdf](http://www.math93.com/images/pdf/algo_Python/Td_Algorithmes_seconde_Fonctions_1.pdf)

### Exercice 1. Tableau de valeurs et graphique



#### def nom\_fonction(paramètres) :

*def nom\_fonction(paramètres)* : définit une nouvelle fonction, les deux points entraînent une indentation délimitant la déclaration de la fonction. Le bloc peut servir à effectuer une série d'actions, mais le plus souvent il se termine par *return* pour renvoyer une ou plusieurs valeurs.



#### Code Python

```
def f(x):  
    return x**2 - x + 41 # x**2 retourne le carré de x, c'est la notation puissance.
```

1. La fonction définie ci-dessus renvoie l'image de la variable  $x$  par la fonction  $f$  définie par  $f(x) = x^2 - x + 41$ . Tester le programme avec des valeurs en écrivant dans la console (*la fenêtre noire*) directement  $f(1)$  ou  $f(2)$  par exemple.

```
>>> f(1)  
=> 41  
>>> f(2)  
=> 43  
>>> f(3)  
=> 47  
>>>
```

2. Liste en compréhension.

On veut maintenant calculer les valeurs de la fonction  $f$  pour  $x$  entier variant de 0 à 19 par exemple.



#### range(début,fin,pas)

*range(début,fin,pas)* : Génère une liste d'entiers, les paramètres *début* et *pas* sont optionnels.

- Dans l'intervalle  $[0; fin[$  si un seul paramètre est renseigné.  
 $L=range(5)$  va créer la liste  $[0, 1, 2, 3, 4]$  de 5 termes, le premier sera  $L[0]=0$ , le dernier  $L[4]=5$ .
- Dans l'intervalle  $[début; fin[$  si 2 paramètres sont renseignés.  
 $L=range(1,5)$  va créer la liste  $[1, 2, 3, 4]$ , le premier terme sera  $L[0]=1$  et le dernier  $L[3]=5$ .
- Dans l'intervalle  $[début; fin[$  mais de *pas* en *pas*, si les 3 paramètres sont renseignés.



#### Code Python

```
valeursx = [ x for x in range(20) ] # permet d'itérer pour x de 0 à 19  
valeursf = [ f(x) for x in range(20) ]  
# on pouvait aussi écrire valeursf = [ f(x) for x in valeursx ]
```

Pour visualiser le résultat, écrire simplement valeursx et valeursf dans la console.

**Remarque**

S'inspirant de l'écriture mathématique d'un ensemble en compréhension, par exemple  $\{x \mid x \in \llbracket 0 ; 19 \rrbracket\}$ , Python propose une syntaxe utile pour la création d'une liste en compréhension : `[ x for x in range(20) ]`.

3. Écrire une fonction `listeimages(f, n)` (de paramètres `f` et `n`) qui renvoie la liste des images des entiers de 0 à `n` par la fonction `f`. Attention, il faut donc `n + 1` valeurs!



**Remarque**

Le texte entre triple apostrophe `'''`, juste sous la première ligne de la définition de la fonction `listeimages(f, n)`, s'appelle la *docstring* de la fonction. C'est le texte qui apparaît si on demande de l'aide : sous la forme `help(listeimages)`



**Code Python**

```
def listeimages(f, n) :
    '''renvoie la liste des images des entiers de 0 à n par la fonction f.'''
    return [ f(x) for x in ..... ] # A compléter
```

Vous devez obtenir par exemple :

```
>>> help(listeimages)
Help on function listeimages:

listeimages(f, n)
    Renvoie la liste des images des entiers de 0 à n par la fonction f .

>>> listeimages(f,5)
=> [41, 41, 43, 47, 53, 61]
```

4. Définissez une autre fonction numérique, `g` de votre choix puis tester votre fonction `listeimages` en tapant dans la console `listeimages(f, 10)` ou `listeimages(g, 10)` par exemple.



**Code Python**

```
def g(n) :
    return ..... # Une autre fonction de votre choix
```

5. Notez le commentaire à gauche de la fonction `listeimages` ! Que faire pour l'éviter?

```
10 def listeimages(f,n):
    '''renvoie la liste des images des entiers de 0 à n par la fonction f .'''
    Redefining name 'f' from outer scope (line 3)
```

6. **Un tableau de valeurs.**

On souhaite obtenir un tableau, enfin ici une liste, avec les valeurs de `x` et les images, pour `x` entier variant de `a` à `b`.

On va alors créer une fonction `tableauvaleurs(f, a, b)`, dont les paramètres sont `f`, `a` et `b`.



**Code Python**

```
def tableauvaleurs(f, a, b) :
    '''renvoie un couple (valeursdex,images) constitué d'une liste d'abscisses et d'une liste des images associées par la fonction f. La variable x prend les valeurs entières de a à b.'''
    assert a < b # cette condition permet de vérifier que a est inférieur à b
    valeursdex = # À compléter.
    images = # À compléter.
    return (valeursx,images)
```

Pour visualiser le résultat, écrire simplement `tableauvaleurs(f,2,6)` ou `tableauvaleurs(g,1,30)` (par exemple ...) dans la console de droite.

Vous devez obtenir par exemple :

```
> tableauvaleurs(f,2,6)
=> ([2, 3, 4, 5, 6], [43, 47, 53, 61, 71])
>
```

## 7. Un affichage.

On peut facilement tracer les nuages de points correspondants, en utilisant les fonctions du module `matplotlib`. On importe le sous-module `pyplot` de `matplotlib` qu'on renomme au passage `plt`, par commodité. Sur un Python installé, pas de problème par contre sur repl ou les logiciels online, le graphique doit être converti en image.

### Code Python

```
import matplotlib.pyplot as plt
fig = plt.figure() # nécessaire seulement sur repl.
(vx, vy) = tableauvaleurs(f,0,10)
plt.plot(vx, vy, '.', color='red') # le 3e argument '.' permet de préciser qu'on veut un nuage de points, le 4e est explicite ...
fig.savefig('graph.png') # ou simplement plt.show() sur un Python installé comme anaconda
```

### Remarque sur `plt.plot(vx, vy, '.', color='red')`

Si on omet le troisième argument, `matplotlib` va relier deux points consécutifs par un segment de droite.

## Exercice 2. Complément en assignments sur repl

8. On peut construire plusieurs tracés sur une même figure en enchaînant simplement plusieurs instructions `plt.plot`.

Proposer le tracé des nuages de points (entre 0 et 30) correspondants à la fonction  $f$  et à la fonction  $g$  définie sur  $\mathbb{R}$  par  $g(x) = x^2 - 79x + 1601$ . Les points de  $f$  seront en vert et ceux de  $g$  en bleu.

## Exercice 3. Compléments : Une fonction du légendaire Euler

Le légendaire mathématicien suisse Léonhard EULER(1707-1783), proposait la formule suivante pour obtenir des nombres premiers : pour tout entier naturel  $n$ ,

$$f(n) = n^2 - n + 41$$



Léonhard EULER(1707-1783)

9. Le site math93 propose la liste des 168 nombres premiers inférieurs à 1 000 et un lien vers le site `compoasoe` qui donne la liste des nombres premiers inférieurs à 1 000 milliards. A l'aide de la fonction `tableauvaleurs(f, a, b)`, trouver la plus petite valeur entière dont l'image n'est pas un nombre premier par la fonction  $f$ , ... si elle existe!

## 10. Compléments.

La fonction  $g : x \mapsto x^2 - 79x + 1601$  est aussi assez remarquable.

Reprendre les questions précédentes avec cette fonction et déterminer la plus petite valeur entière dont l'image n'est pas un nombre premier par la fonction  $g$ , ... si elle existe!

*Remarque :* On peut donc se demander s'il existe une fonction polynomiale (à coefficient entiers) donnant tous les nombres premiers ... la réponse est non, mais il vous faudra attendre un peu pour démontrer ce magnifique résultat!

**Prolongements**

- Tests de primalité : [www.math93.com/images/pdf/algo\\_Python/Td\\_Algorithmes\\_TD\\_Test\\_Primalite.pdf](http://www.math93.com/images/pdf/algo_Python/Td_Algorithmes_TD_Test_Primalite.pdf)  
*Le test de primalité des images, c'est à dire la vérification que l'entier image soit premier ou pas, fut ici fastidieuse. On pourrait sans doute demander à l'ordinateur de le faire pour nous ... pour cela, rendez-vous au TD d'arithmétique intitulé : test de primalité.*
- Td Fonctions 2 : [www.math93.com/images/pdf/algo\\_Python/Td\\_Algorithmes\\_Fonctions\\_TD2.pdf](http://www.math93.com/images/pdf/algo_Python/Td_Algorithmes_Fonctions_TD2.pdf)  
*Pour des tableaux de valeurs plus précis permettant une résolution d'équations par balayage.*