Un-premier-travail-sur-les-tableaux-indexes-listes-en-Python-

October 14, 2019

Tableaux indexés ou listes en Python

0.1 Type construit

Un tableau indexé est un type construit car c'est un objet dont les éléments sont des objets de types de base (ou éventuellement construit) :

Exemple

[2]:
```
T = [4, 7, 8, 1]
type(T)
```

[2]:
```
list
```

- La variable `T` est un objet de type `list` en Python dont les éléments sont des objets de type `int`.
- Pour accéder aux éléments de `T`, on utilise son index :

[3]:
```
T[0] # Premier élément de T
```

[3]:
```
4
```

[ ]:
```
# Compléter
T[...]  # Dernier élément de T
```

[ ]:
```
# Compléter
T[...]  # Premier élément de T
```

0.2 Une fonction et une méthode associées

La fonction `len` prend pour paramètre une liste et retourne sa taille : le nombre de ses éléments :

[ ]:
```
# Compléter
len(...) # Taille de la liste T
```

La méthode `append` s'applique aux objets de type `list` et ajoute un élément à la fin de la liste :

```
[ ]: # Compléter
T.append(...)  # On ajoute l'entier 2 à la fin de la liste
T
```

0.3 Parcourir les éléments d'une liste

0.3.1 En utilisant les index :

```
[ ]: # Compléter
taille = ...  # Taille de la liste
for index in range(taille):
    print(....)   # Affiche chaque élément de la liste
```

0.3.2 En parcourant chaque élément

```
[ ]: # Compléter
for element in ...:
    print(...)
```

0.4 La suite de Collatz

Sur le site **wikipedia.org**, on trouve les informations suivantes :

La suite de Syracuse(ou de Collatz) d'un nombre entier $N > 0$ est définie par récurrence, de la manière suivante :

$$u_0 = N \text{ et pour tout entier naturel } n : u_{n+1} = \begin{cases} \frac{u_n}{2} \text{ si } u_n \text{ est pair,} \\ 3u_n + 1 \text{ si } u_n \text{ est impair.} \end{cases}$$

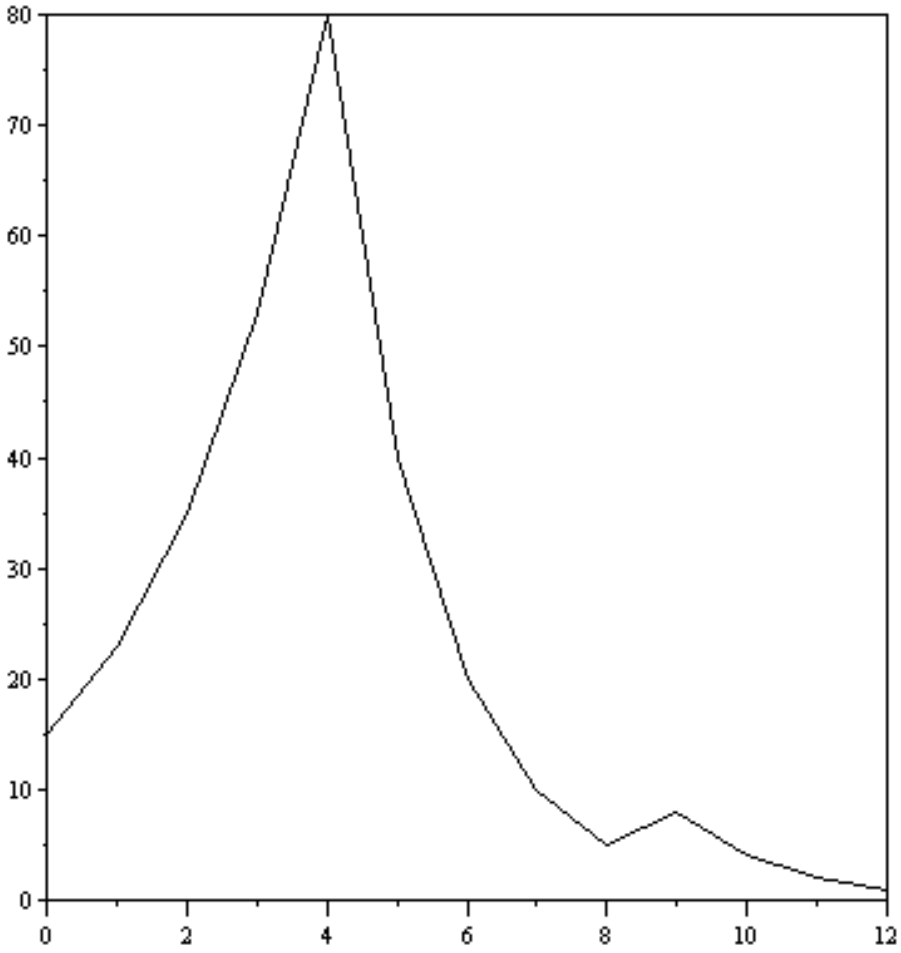
Exercice 1 **1.** Calculer, à la main, les dix premiers termes de la suite de Collatz lorsque $u_0 = 10$.

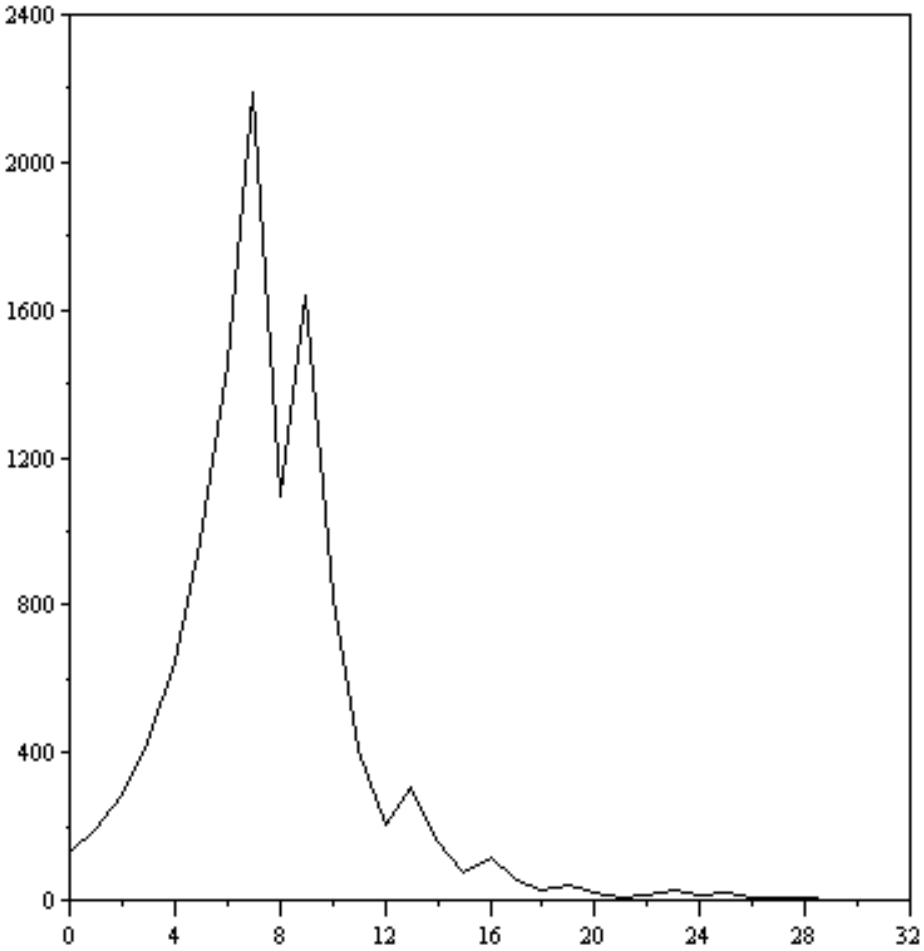
2. Reprendre le travail pour $u_0 = 21$.

0.4.1 Conjecture de Syracuse

La conjecture de Syracuse affirme que pour tout N, il existe un indice n tel que $u_n = 1$.

L'observation graphique de la suite pour $N = 15$ et pour $N = 127$ montre que la suite peut s'élever assez haut avant de retomber. Les graphiques font penser à la chute chaotique d'un grêlon ou bien à la trajectoire d'une feuille emportée par le vent. De cette observation est né tout un vocabulaire imagé : on parlera du vol de la suite.


80
70
60
50
40
30
20
10
0
0
2
4
6
8
10
12


2400
2000
1600
1200
800
400
0
0
4
8
12
16
20
24
28
32

0.4.2 Exercice 2

1. Écrire un algorithme (en langage courant) affichant tous les termes de la suite de Collatz jusqu'à afficher le premier terme égale à 1.

2. À l'aide de cet algorithme, compléter la fonction `collatz_indice` suivante qui prend pour paramètre un entier naturel `N` différent de zéro et retourne en sortie le plus petit indice `n` tel que $u_n = 1$: cet entier s'appelle le **temps de vol** de la suite de Collatz.

[ ]:
```
# À compléter
def collatz_indice(...):
    """ Écrire ici ce que fait la fonction (on parle de spécification)
    ...
    """
    # Les instructions assert sont utilisées pour vérifier les préconditions.
    # Une telle instruction se compose d'une condition (une expression booléenne)
    # éventuellement suivie d'une virgule et d'une phrase en langue naturelle,
    # sous forme d'une chaine de caractères. L'instruction assert teste si sa condition est satisfaite.
    # Si c'est le cas, elle ne fait rien et sinon elle arrête immédiatement l'exécution du programme
    # en affichant éventuellement la phrase qui lui est associée.
    assert type(N) == ..., "la variable N doit être de type ..."
    assert ... > ..., "la variable ..."

    # On commence la fonction
    indice = 0
    while ...:
        if ...:
            N = ...
        else:
            N = ...
        indice = ...
    return ...
```

3. Vérifier que les préconditions fonctionnent pour 3.2 et -16.

[ ]:
```
## Pour 3.2
...
```

[ ]:
```
### Pour -16
...
```

4. Vérifier que le **temps de vol** de la suite est 17 pour $u_0 = 15$ et 46 pour $u_0 = 127$:

u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}	u_{11}	u_{12}	u_{13}	u_{14}	u_{15}	u_{16}	u_{17}	u_{18}	u_{19}	u_{20}	
15	46	23	70	35	106	53	160	80	40	20	10	5	16	8	4	2	1	4	2	1	...

5. Écrire la fonction `collatz_liste` ayant comme paramètre d'entrée un entier naturel N et retournant en sortie la partie de la suite de Syracuse tronquée au premier terme égal à 1. Par exemple : `collatz(15)` retourne la liste `[15,46,23,70,35,106,53,160,80,40,20,10,5,16,8,4,2,1]`.

[ ]:
```
def collatz_liste(...):
    ...
```

[ ]:
```
## Quelques tests :
print(...)
```

0.4.3 Pour allez plus loin

On définit alors : - **le temps de vol en altitude :** c'est le plus petit indice n tel que $u_{n+1} \leq u_0$. Il est de 10 pour la suite de Syracuse 15 et de 23 pour la suite de Syracuse 127 ; - **l'altitude maximale :** c'est la valeur maximale de la suite.
Elle est de 160 pour la suite de Syracuse 15 et de 4 372 pour la suite de Syracuse 127.

0.4.4 Exercice 3

1. Écrire une fonction `vol_altitude` utilisant la fonction `collatz_liste`, ayant comme paramètre d'entrée un entier naturel `N` et retournant le temps de vol en altitude de la suite de Collatz du nombre N.

[ ]:
```
def vol_altitude(N):
    ...
```

[ ]:
```
# Pour tester
...
```

2. Écrire une fonction `altitude_max`, utilisant la fonction `collatz_liste`, ayant comme paramètre d'entrée un entier naturel `N` et retournant l'altitude maximale de la suite de Collatz du nombre N.

[ ]:
```
def altitude_max(N):
    ...
```

[ ]:
```
# Pour tester
```

3. Écrire un programme demandant un entier à l'utilisateur et affichant les valeurs maximales des paramètres définis précédemment ainsi que les entiers correspondants, pour tout les entiers inférieurs ou égaux à l'entier saisit. Par exemple, on obtiendrait :

```
Saisir un entier : 200
Le temps de vol maximal est 124 pour N = 171
Le temps de vol en altitude maximal est 95  pour N = 27
L'altitude maximale est 9232 pour N = 27
```

[ ]:
```

```