Tableaux indexés ou listes en Python

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

In [1]:

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

Out[1]:

```
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 :

In [2]:

```
T[0] # Premier élément de T
```

Out[2]:

```
4
```

In [3]:

```
# Compléter
T[-1]  # Dernier élément de T
```

Out[3]:

```
1
```

In [4]:

```
# Compléter
T[1]  # Deuxième élément de T
```

Out[4]:

```
7
```

Une fonction et une méthode associées

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

In [5]:

```
# Compléter
len(T) # Taille de la liste T
```

Out[5]:

```
4
```

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

In [6]:

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

Out[6]:

```
[4, 7, 8, 1, 2]
```

Parcourir les éléments d'une liste

En utilisant les index :

In [7]:

```
# Compléter
taille = len(T)  # Taille de la liste
for index in range(taille):
    print(T[index], end=', ')   # Affiche chaque élément de la liste
```

```
4, 7, 8, 1, 2,
```

En parcourant chaque élément

In [8]:

```
# Compléter
for element in T:
    print(element, end=', ')
```

```
4, 7, 8, 1, 2,
```

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 :

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

Exercice 1

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

u_0	u_1	u_2	u_3	u_4	u_5	u_6
10	5	16	8	4	2	1

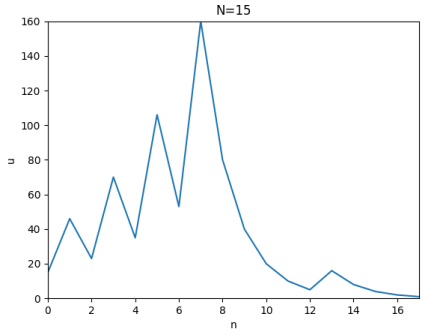
2. Reprendre le travail pour $u_0 = 21$.

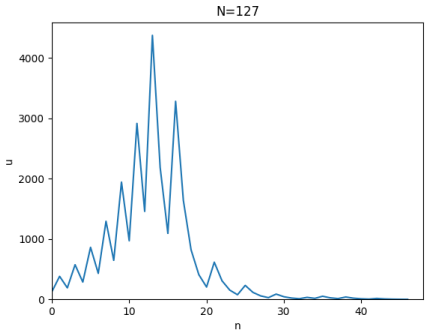
u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7
21	64	32	16	8	4	2	1

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.





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.

```
Saisir N
 U ← N
Tant que U > 1 faire
   Si U%2 = 0 alors
      U ← U / 2
   Sinon
      U ← 3xU+1
   Afficher U
Afficher U
```

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.

In [9]:

```
# À compléter
def collatz_indice(N : int)-> int:
    """ Retourne le premier indice de la suite de Collatz du nombre N pour
    lequel le terme vaut 1
    """
    # Les instructions assert sont utilisées pour vérifier les préconditions.
    # Une telle instruction se compose d'une condition (une expression booléenn
e)
    # 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 co
ndition est satisfaite.
    # Si c'est le cas, elle ne fait rien et sinon elle arrête immédiatement l'ex
écution du programme
    # en affichant éventuellement la phrase qui lui est associée.
    assert type(N) == int, "la variable N doit être de type int"
    assert N > 0, "la variable doit être un entier naturel non nul"
    # On commence la fonction
    indice = 0
    while N != 1:
        if N%2 == 0:
            N = N//2
        else:
            N = 3*N+1
        indice = indice + 1
    return indice
```

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

In [10]:

```
## Pour 3.2
collatz_indice(3.2)
```

```
---------------------------------------------------------------------
--------
AssertionError                            Traceback (most recent ca
ll last)
<ipython-input-10-0f55fb6d59a2> in <module>()
      1 ## Pour 3.2
----> 2 collatz_indice(3.2)

<ipython-input-9-8c916511c26e> in collatz_indice(N)
     10     # Si c'est le cas, elle ne fait rien et sinon elle arrê
te immédiatement l'exécution du programme
     11     # en affichant éventuellement la phrase qui lui est ass
ociée.
---> 12     assert type(N) == int, "la variable N doit être de type
int"
     13     assert N > 0, "la variable doit être un entier naturel
 non nul"
     14     # On commence la fonction

AssertionError: la variable N doit être de type int
```

In [11]:

```
### Pour -16
collatz_indice(-16)
```

```
---------------------------------------------------------------------
--------
AssertionError                            Traceback (most recent ca
ll last)
<ipython-input-11-ad29afdead03> in <module>()
      1 ### Pour -16
----> 2 collatz_indice(-16)

<ipython-input-9-8c916511c26e> in collatz_indice(N)
     11     # en affichant éventuellement la phrase qui lui est ass
ociée.
     12     assert type(N) == int, "la variable N doit être de type
int"
---> 13     assert N > 0, "la variable doit être un entier naturel
 non nul"
     14     # On commence la fonction
     15     indice = 0

AssertionError: la variable doit être un entier naturel non nul
```

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}
15	46	23	70	35	106	53	160	80	40	20	10	5	16	8	4

In [12]:

```
N = 15
print(f"Temps de vol pour N = {N} : {collatz_indice(N)}")
N = 127
print(f"Temps de vol pour N = {N} : {collatz_indice(N)}")
```

```
Temps de vol pour N = 15 : 17
Temps de vol pour N = 127 : 46
```

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]` .

In [14]:

```
def collatz_liste(N : int)-> list:
    """     Retourne une liste contenant les premiers termes de la uite de Colla
tz
    du nombre N tronquée au premier terme égal à 1
    """
    assert type(N) == int, "la variable N doit être de type int"
    assert N > 0, "la variable doit être un entier naturel non nul"
    # On commence la fonction
    U = [N]
    while N != 1:
        if N%2 == 0:
            N = N//2
        else:
            N = 3*N+1
        U.append(N)
    return U
```

In [15]:

```
## Quelques tests :
N = 15
print(f"Liste pour N = {N} : {collatz_liste(N)}")
N = 21
print(f"Liste pour N = {N} : {collatz_liste(N)}")
```

```
Liste pour N = 15 : [15, 46, 23, 70, 35, 106, 53, 160, 80, 40, 20,
10, 5, 16, 8, 4, 2, 1]
Liste pour N = 21 : [21, 64, 32, 16, 8, 4, 2, 1]
```

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.

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.

In [16]:

```
def vol_altitude(N : int) -> int:
    """
    Retourne le temps de vol en altitude de la suite de Collatz.
    On utilise la fonction collatz_liste() (assert inutile)
    """
    indice = 0
    U = collatz_liste(N)
    while U[indice+1] > N:
        indice = indice + 1
    return indice
```

In [17]:

```
## Quelques tests :
N = 15
print(f"Temps de vol en altitude pour N = {N} : {vol_altitude(N)}")
N = 127
print(f"Temps de vol en altitude pour N = {N} : {vol_altitude(N)}")
```

```
Temps de vol en altitude pour N = 15 : 10
Temps de vol en altitude pour N = 127 : 23
```

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.

In [18]:

```
def altitude_max(N : int) -> int:
    """
    Retourne l'altitude maximale de la suite de Collatz.
    On utilise la fonction collatz_liste()
    """
    maximum = N
    U = collatz_liste(N)
    for element in U:
        if element > maximum:
            maximum = element
    return maximum
```

In [19]:

```
## Quelques tests :
N = 15
print(f"Altitude maximale pour N = {N} : {altitude_max(N)}")
N = 127
print(f"Altitude maximale pour N = {N} : {altitude_max(N)}")
```

```
Altitude maximale pour N = 15 : 160
Altitude maximale pour N = 127 : 4372
```

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
```

In [20]:

```
entier = int(input("Saisir un entier : "))
vol_max = [0,0]
vol_altitude_max = [0,0]
max_altitude_max = [0,0]
for i in range(3, entier+1):
    if collatz_indice(i) > vol_max[0]:
        vol_max = [collatz_indice(i), i]
    if vol_altitude(i) > vol_altitude_max[0]:
        vol_altitude_max = [vol_altitude(i), i]
    if altitude_max(i) > max_altitude_max[0]:
        max_altitude_max = [altitude_max(i), i]
print(f"Le temps de vol maximal est {vol_max[0]} pour N = {vol_max[1]}")
print(f"Le temps de en altitude maximal est {vol_altitude_max[0]} pour N = {vol_
altitude_max[1]}")
print(f"L'altitude maximale est {max_altitude_max[0]} pour N = {max_altitude_max
[1]}")
```

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