TP N°0 :
ETUDE DE LA SUITE DE SYRACUSE

OBJECTIFS DU TP

Dans ce TP, (comme dans les suivants) veiller à ce que toutes les fonctions écrites contiennent :

- Une spécification précise, annotations et commentaires, jeux de tests ;
- Une signature : spécification des données attendues en entrée, et fournies en sortie.

Pour ce TP, vous aurez besoin des fonctions du module `matplotlib.pyplot` pour le tracé de graphes (notamment : `plot` et `show`).

```
import matplotlib.pyplot as plt
```

I. PRESENTATION DU PROBLEME

Q1. Vérifier que la conjecture de Syracuse est valable pour $N = 5$.

La suite de Syracuse de 5 est :

$$u_0 = 5 \quad ; \quad u_1 = 16 \quad ; \quad u_2 = 8 \quad ; \quad u_3 = 4 \quad ; \quad u_4 = 2 \quad ; \quad u_5 = 1$$

Q2. Montrer que si la conjecture de Syracuse est vraie, toute suite de Syracuse est périodique au-delà d'un certain rang.

S'il existe un rang n tel que $u_n = 1$, alors $u_{n+1} = 4$; $u_{n+2} = 2$; $u_{n+3} = 1$: donc la suite est périodique de période 3 à partir du rang n.

II. Vol de la suite de Syracuse

Q3. Ecrire une fonction `syracuse` prenant comme argument un entier N, et retournant une liste contenant le col de la suite de Syracuse de N.

```
def syracuse (N:int)->[int]:
    """
  Détermine le vol associé à la suite de Syracuse de N

  Entrée : N est de type int c'est le terme u0 de la suite

  Sortie : Renvoie une liste [int] contenant le 'vol' de la suite de
  Syracuse (le vol étant les termes de la suite de u_0 =N jusqu'à u_n =1)

  === jeux de tests ===
  >>> syracuse(5)
  [5,16,8,4,2,1]

  >>> syracuse(12)
  [12,6,3,10,5,16,8,4,2,1]

    """

    vol=[N]

    while N>1 : #il est aussi possible d'utiliser n!=1
        if N%2==0 :
            N = N//2
        else :
            N = 3*N+1
        vol.append(N)
    return vol
```

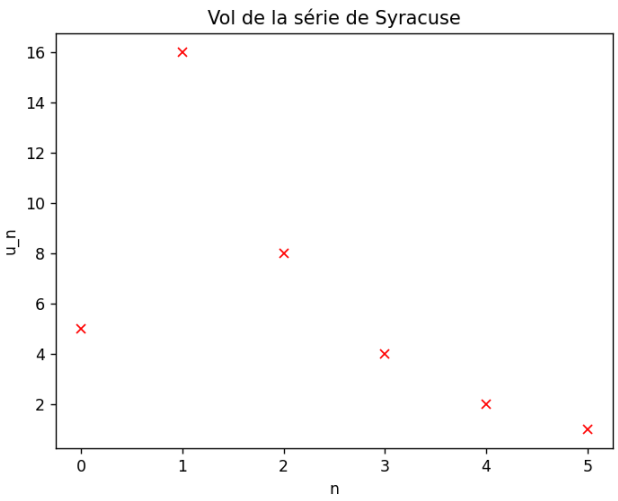
Q4. Proposer une suite d'instruction permettant de déterminer le vol de 42.

```
>>> syracuse(42)
[42,21,64,32,16,8,4,2,1]
```

Q5. Ecrire une fonction `graphevol` prenant comme argument un entier N, et affichant le vol de la suite de Syracuse de N sous la forme d'un graphique $u_n = f(n)$. La fonction `graphevol` ne retourne rien. On utilisera la fonction `syracuse`.

```
def graphevol(N:int):
    """
    Affiche une graphique u_n = f(n) contenant le vol de la suite
    Cette fonction utilise la fonction syracuse définie précédemment
    Fonction qui nécessite l'importation de matplotlib as plt
    Entrée : N (int) est le terme u0 de la suite

    Sortie : ne retourne rien

    === jeux de tests ===

    >>> graphevol(5)

    Affiche le graphique formé par le nuage de points ((0,5), (1,16),
(2,8), (3,4), (4,2), (5,1))

    """
    Y=syracuse(N)

    X=[]

    for n in range (len(Y)):
        X = X+[n]

    plt.plot(X,Y,'rx')
    plt.xlabel('n')
    plt.ylabel('u_n')
    plt.title('Vol de la série de Syracuse')
    plt.show()

    return None
```

Pour N = 5, on obtient, le graphique ci-contre :



III. QUELQUES CARACTERISTIQUES DE LA SUITE DE SYRACUSE

Q6. Déterminer le vol pour $N = 5$, puis le temps de vol, le temps de vol en altitude et l'altitude maximale pour cette suite.

Pour N=5, les premiers termes de la suite sont : [5,16,8,4,2,1].
Le temps de vol est donc 5. Le temps de vol en altitude est 2. L'altitude maximale est 16.

Q7. Ecrire une fonction `tempsvol` prenant comme argument un entier N, et retournant le temps de vol de la suite de Syracuse de N. On utilisera la fonction `syracuse`.

```
def tempsvol(N:int)-> int:
    """
   Détermine le temps de vol de la suite de Syracuse de N

   Entrée : N (int) est le terme u_0 de la suite de Syracuse considérée
   Sortie : le temps de vol de la suite de Syracuse de N - soit le rang n
(minimal) à partir duquel u_n vaut 1.

   === jeux de tests ===
   >>> tempsvol(5)
   5
   >>> tempsvol(12)
   9
   """

   vol = syracuse(N)
   return len(vol)-1
```

Q8. Ecrire une fonction `tempsvolalt` prenant comme argument un entier N, et retournant le temps de vol en altitude de la suite de Syracuse de N. On utilisera la fonction `syracuse`.

```
def temps_vol_alt (N:int)->int :
   """
   Détermine le temps de vol en altitude de la suite de Syracuse de N -
soit le rang n minimal tel que u_n+1 <u_0

   Entrée : N est le terme u_0 de la suite de Syracuse considérée
   Sortie : temps de vol en altitude de la suite de Syracuse de N
   === jeux de tests ===
   >>>temps_vol_alt(5)
   2
   >>>temps_vol_alt(12)
   0
     """
     rang = 0
     vol= syracuse(N)

     while vol[rang] >= N:
         rang = rang+1

     return rang-1
```

Q9. Ecrire une fonction `altmax` prenant comme argument un entier N, et retournant l'altitude maximale de la suite de Syracuse de N. On utilisera la fonction `syracuse`.

```
def altmax (N:int)->int :
    """
    Détermine l'altitude maximale de la suite de syracuse de N - soit la
valeur maximale atteinte durant le vol.

    Entrée : N (int) est le terme u_0 de la suite de Syracuse considérée
    Sortie : valeur maximale prise par u_n pendant le vol

    === jeux des tests ===
    >>> altmax(5)
    16
      """

      vol=syracuse(N)
      max=N

      for u in vol :

          if u>max :
              max=u

      return max
```

IV. ETUDE DE LA SUITE DE SYRACUSE POUR $N \in [\![1, 10\,000]\!]$

Q10. Ecrire une fonction `volmax` prenant comme arguments deux entiers A et B, et retournant l'entier N pour lequel la suite de Syracuse possède un temps de vol supérieur à celui de tous les entiers compris entre A et B. On utilisera la fonction `tempsvol`.

```
def volmax(A:int, B:int)-> int :
    """
    Détermine l'entier N compris entre A et B pour lequel la suite de
Syracuse à le temps de vol le plus important.
      Cette fonction utilise la fonction tempsvol

    Entrées :  A et B (int) sont les bornes de l'intervalle fermé d'entiers
N pour lesquels on souhaite étudier la suite de Syracuse

    Sortie : entier N compris entre A et B, temps de vol maximal de la suite
dans l'intervalle considéré

    === jeux de tests ===
    >>> volmax(4,8)
    7
      """

    N = A
    temps_max = 0
```

```
  for k in range (A,B+1):

    T = tempsvol(k)

    if T > temps_max :
      N = k
      temps_max = T

  return N
```

Q11. Proposer une suite d'instructions permettant de déterminer quel est l'entier possédant le temps de vol le plus grand sur l'intervalle $[\![1,\ 10\,000]\!]$.

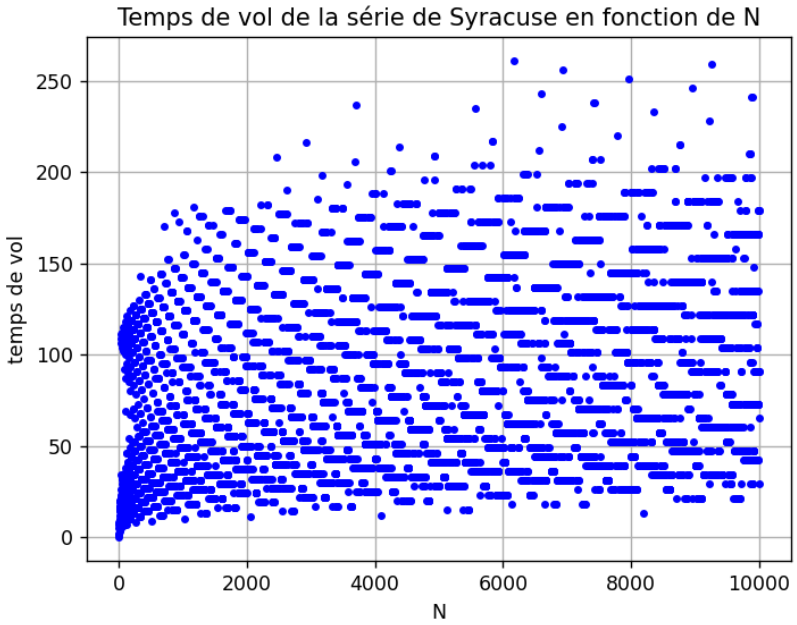
```
>>> volmax(1,10000)
6171
```

Q12. Ecrire une fonction `graphe_tempsvol` prenant comme arguments deux entiers A et B, et affichant le temps de vol pour chacune des suites de Syracuse de N, pour N compris entre A et B. La fonction `graphe_tempsvol` ne retourne rien. On utilisera la fonction `tempsvol`.

```
def graphe_tempsvol (A:int, B:int):
  """
  Affiche le graphique représentant le temps de vol de la suite de Syracuse
  de N en fonction de N, sur l'intervalle fermé [A,B]
  Cette fonction utilise la fonction tempsvol

  Entrées : A et B (int) sont les bornes de l'intervalle fermé d'entiers
  N pour lesquels on souhaite étudier la suite de Syracuse

  Sortie : Ne renvoie rien

  === jeux de tests ===
  >>> graphe_tempsvol(4,8)
  Affiche le nuage de points ((4,2), (5,5), (6,8), (7,16), (8,3))

  """

  T = []
  n=[]

  for N in range (A,B+1):
      T= T + [tpsvol(N)]
      n = n + [N]

  plt.plot(n,T,'b.')
  plt.xlabel('N')
  plt.ylabel("temps de vol")
  plt.title("Temps de vol de la série de Syracuse en fonction de N")
  plt.grid(True)
  plt.show()
```

Q13. Proposer une suite d'instructions permettant d'afficher le graphe représentant le temps de vol N en fonction de N, pour N compris entre 1 et 10 000.

```
>>> graphe_tempsvol(1,10000)
```



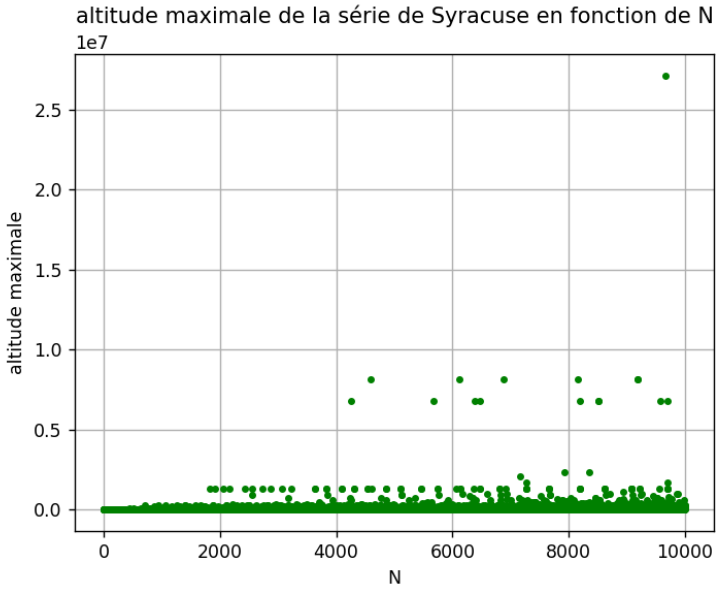
Q14. Écrire une fonction `graphe_altmax` prenant comme arguments deux entiers A et B, et affichant l'altitude maximale pour chacune des suites de Syracuse de N, pour N compris entre A et B. La fonction `graphe_altmax` ne retourne rien. On utilisera la fonction `altmax`.

```
def graphe_altmax (A:int,B:int):
    """
    Affiche le graphe de l'altitude maximale de la suite de Syracuse N en
fonction de N sur l'intervalle d'entiers [A,B]
    Cette fonction utilise la fonction altmax


    Entrées : A et B (int) sont les bornes de l'intervalle fermé d'entiers
    N pour lesquels on souhaite étudier la suite de Syracuse

    Sortie : Ne renvoie Rien

    === jeux de tests ===

    >>>graphe_altmax(4,8)
   Affiche le nuage de points ((4,4), (5,16), (6,16), (7,52), (8,8))

    """
    alt = []
    n=[]

    for N in range (A,B+1):
        alt = alt + [altmax(N)]
        n = n + [N]
```

```
    plt.plot(n,alt,'g.')
    plt.xlabel('N')
    plt.ylabel("altitude maximale")
    plt.title("altitude maximale de la série de Syracuse en fonction de
N")
    plt.grid(True)
    plt.show()
```

Q15. Proposer une suite d'instructions permettant d'afficher le graphe représentant l'altitude maximale de N en fonction de N, pour N compris entre 1 et 10 000.

```
>>> graphe_altmax(1,10000)
```



Ce n'est pas très lisible, il est possible de zoomer en ajoutant la ligne suivante à la fonction :

```
 plt.axis([1,10000,1,100000])
```

On obtient alors :