TP 3: Les boucles et les listes

Les bonus sont des bonus .... à traiter à la fin !

Nous rappelons la syntaxe associée à la création d'une fonction, d'une boucle for/while, d'une instruction conditionelle :

— Fonction :

```
def nomdelafonction(variable):
        #corps de la fonction
        return(retourdelafonction}
```

— Boucle for :

```
for variable in objetiterable:
        #opérations
```

— Boucle while :

```
while conditiondarret:
        #opérations
```

— Instruction conditionelle :

```
if condtion1:
        #retour
elif condition2:
        #retour
else:
        #retour
```

EXERCICE 1: Des nombres premiers

1. Afficher la liste des multiples de 7 inférieurs à 100.
2. Justifier succinctement que la fonction suivante teste la primalité d'un nombre entier : La fonction `isprime` suivante :

```
def isprime(n):
        s=int(sqrt(n))
        t=True
        i=2
        while i<=s and t==True:
             if n% i==0:
                  t=False
             i+=1
        return(t)
```

3. Afficher la liste des nombres premiers inférieurs à 100.
4. Afficher la liste des 100 premiers nombres premiers.
5. On appelle nombres premiers jumeaux les couples de nombres premiers de la forme $(i, i+2)$. Afficher les nombres premiers jumeaux, avec i inférieur à 3000.
6. Nombres de Fermat : il paraît que Fermat pensait que tout nombre du type $2^{2^n}+1$ est premier. Qu'en pensez-vous ? Déterminer le plus petit k tel que $2^{2^k}+1$ n'est pas premier.
7. BONUS : la répartition des nombres premiers (Hadamard-De La Vallée-Poussin 1890) Tester la validité du théorème de répartition des nombres premiers : $\pi(n) \underset{n\to+\infty}{\sim} \dfrac{n}{\ln(n)}$.
   Ce qui se traduit par $\pi(n)\dfrac{\ln(n)}{n} \underset{n\to+\infty}{\longrightarrow} 1$ où $\pi(n)$ est le nombre de nombres premiers inférieurs à n.
   Que peut-on en conclure ?

EXERCICE 2: Approximation de racine de 2

L'objectif de cet exercice est d'aborder les problèmes d'approximations. Nous aborderons ces questions en cours ultérieurement.

Nous considérons la suite récurrente définie par $u_0 = 1$ et pour tout n dans $\mathbb{N}$,

$$u_{n+1} = \frac{1}{2}(u_n + \frac{2}{u_n})$$

3. Afficher la liste des 100 premières valeurs de la suite u.
4. Afficher la liste des 10 premières valeurs de la suite $u_n^2 - 2$.
   Nous pourrions justifier que la suite u converge vers $\sqrt{2}$.
5. Afficher le premier rang n tel que u_n est proche de racine de 2 à 10^{-5} près.
   (attention : la première étape de cette question revient à identifier une condition d'arrêt ...... qui ne peut pas être $|u_n - \sqrt{2}| < 10^{-5}$. Chercher une condition sous la forme $|u_n^2 - 2| <???$)
6. BONUS : Adapter la suite pour proposer une valeur approchée de $\sqrt{q}$ avec $q \in \mathbb{N}$.

EXERCICE 3: Les coefficients du binôme

Les deux questions suivantes sont indépendantes

1. (a) Ecrire une fonction fact qui a n retourne $n!$.
   (b) Proposer une fonction d'argument n et p permettant de retourner $\binom{n}{p}$ à l'aide de la question précédente.
2. Proposer une fonction d'argument n permettant de retourner la liste des coefficients binomiaux $\binom{n}{p}$ pour tout p entre 0 et n à l'aide d'une formule liant $\binom{n}{p}$ et $\binom{n}{p+1}$.

EXERCICE BONUS: Conjecture de Syracuse

Nous définissons sur $\mathbb{N}$ la fonction de Syracuse par : $s(n) = n/2$ lorsque n est pair et $s(n) = 3n + 1$ sinon.
Une conjecture très célèbre affirme que, pour tout entier $n \in \mathbb{N}^*$, s atteint la valeur 1 en un nombre fini d'étapes.
c'est à dire en appliquant successivement s nous obtenons 1. Ce qui se traduit par :

$$\forall n \in \mathbb{N},\ \exists N \in \mathbb{N}^*, s^N(n) = \underbrace{s(s(s(\dots s}_{\text{N fois}}(n)\dots))) = 1$$

1. Définir une fonction `syracuse`.
2. Proposer une fonction NbrEtapes admettant $n \in \mathbb{N}^*$ comme argument et renvoyant le nombre de fois qu'il est nécessaire d'appliquer la fonction syracuse pour atteindre pour la première fois 1. (ex : $n = 10$ le résultat est 6).
3. Afficher la liste des NbrEtapes(n) pour les n entre 1 et 1000.

EXERCICE BONUS : Palindromes et nombres de Lychrel

Une liste $[l_1, \cdots, l_n]$ est un palindrome lorsque

$$[l_1, \cdots, l_n] = [l_n, \cdots, l_1]$$

1. Ecrire une fonction palindrome qui détermine si une liste est ou non un palindrome.
2. Proposer une fonction (celle vue en cours) donnant la liste des chiffres de l'écriture décimale de n.
3. Proposer un fonction NbrPal d'argument N qui détermine le nombre de palindrome inférieur à N.
4. Un algorithme pour obtenir un palindrome ? ? ?
   Une première étape :
   On part d'un nombre N :
   — Si N est un palindrome, le processus s'arrête.
   — Si N n'est pas un palindrome, on renvoie le nombre qui est la somme de N et de l'entier N "retournée".
   Exemple : $122 \to 122 + 221$
   Nous notons f la fonction associée à l'étape précédente.
   La question est la suivante : est-ce qu'un nombre fini d'itérations suffit pour obtenir un palindrome ? ou combien d'étapes sont nécessaires ?
   Lorsqu'il est fini, nous appellerons la hauteur palindromique de N le nombre d'étapes effectuées.
   Par exemple, en base 10, le nombre 11 est de hauteur 0 et le nombre 10 de hauteur 1, 124 est de hauteur 1, 59 de hauteur 3.
5. Ecrire une fonction `hauteur` d'arguments n qui retourne la hauteur palindromique de n ? (question piège : quelle est la hauteur palindromique de 195 ? 196 ?)