Exercices sur les boucles itératives (suite)

Exercice n° 1

Dans cet exercice, on utilisa la fonction « factorielle » d'un nombre N qui s'écrit N ! et qui est définie par :

N ! = N × (N-1) × (N-2) × (N-3) ×…..× 3 × 2 × 1

Ecrire la fonction Python `Fact` qui permette de calculer à l'aide d'une boucle for, la factorielle d'un nombre **entier** N (l'appel à cette fonction se fera comme `Nfac = Fact(N)`). Cette fonction retournera le calcul $N!$ si N ≤ 50 et -1 dans le cas contraire.

Dans une course hippique avec Q chevaux au départ, on suppose que tous les chevaux possèdent la même chance de gagner. Dans cette hypothèse le nombre de combinaisons possibles pour avoir les 3 premiers chevaux dans l'ordre est :

$$N_{ordre} = \frac{Q\ !}{(Q-3)!}$$

et le nombre de combinaisons possibles pour avoir les 3 premiers chevaux dans le désordre est :

$$N_{desordre} = \frac{Q\ !}{3! \times (Q-3)!}$$

Un parieur qui mise sur 3 chevaux « au hasard » a donc 1 chance sur N_{ordre} de trouver le tiercé dans l'ordre et 1 chance sur $N_{desordre}$ de le trouver dans le désordre.

Ecrire le programme qui, en fonction du nombre de chevaux partants (à faire saisir par l'utilisateur et filtré entre 10 et 30), calcule et affiche en pourcentage les probabilités de trouver les trois premiers chevaux dans l'ordre et dans le désordre. On utilisera évidement la fonction `Fact` précédente.

Exercice n° 2

On suppose la loi d'évolution, semaine après semaine, de 2 espèces animales (proie – prédateur) régit par l'équation :

$$\begin{cases} \boldsymbol{u_{n+1} = (1+a).u_n - b.u_n.v_n} \\ \boldsymbol{v_{n+1} = c.u_n.v_n + (1-d).v_n} \end{cases}$$

où $\boldsymbol{u_n}$ et $\boldsymbol{v_n}$ sont les populations à la semaine n et (a,b,c,d) un quadruplet de paramètres propre à la population étudiée.

Ecrire un programme qui suive l'évolution de la population modélisée par le quadruplet (0.08, 0.0002, 0.0003, 0.075). La population initiale sera prise au hasard (entre 25 et 200) en utilisant la fonction aléatoire `random.random()`

On mémorisera pour chaque population les valeurs minimales et maximales atteintes ainsi que les semaines où ces extremums ont été atteints.

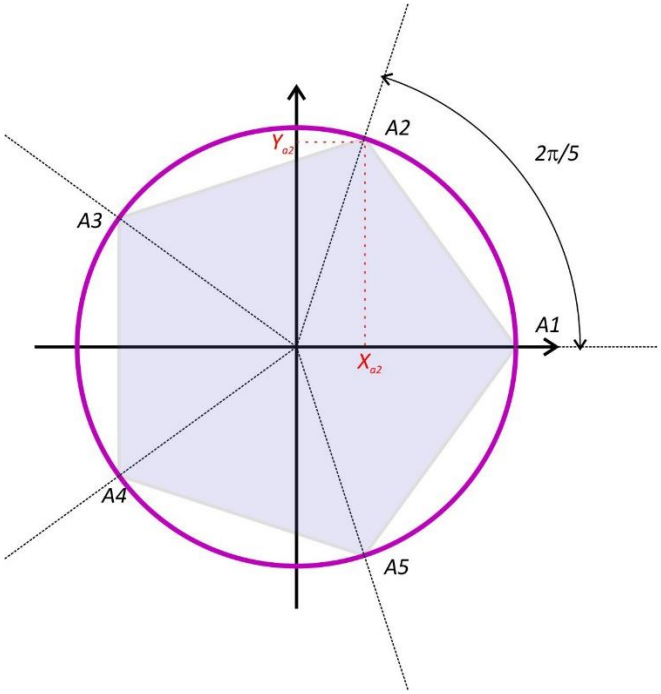
Exemple de résultat en forçant les conditions initiales à u0 = 225 et v0 = 115

```
Au bout de  499  semaines la population de Un est de  1703.61  et celle de Vn est est de  18.35
Le minimum de Un a été atteint à la semaine  380  avec la valeur de  0.5
Le maximum de Un a été atteint à la semaine  499  avec la valeur de  1703.6
Le minimum de Vn a été atteint à la semaine  474  avec la valeur de  0.7
Le minimum de Vn a été atteint à la semaine  496  avec la valeur de  1359.9
```

Problème 1 : Approximation d'un cercle

On souhaite approximer le cercle unité (rayon =1) par un polynôme à N sommets afin d'en calculer le périmètre (qui, rappelons-le, fait 2π) par des morceaux de droite.

La figure ci-dessous illustre ce principe d'approximation avec un pentagone (polynôme à 5 sommets)



Données mathématiques :

Les N sommets $\boldsymbol{A_i}$ $(i = 1, \dots N)$ du polynômes ont pour coordonnées $X_{A_i} = cos(\frac{2(i-1)\pi}{N})$ et $Y_{A_i} = sin(\frac{2(i-1)\pi}{N})$

Pythagore nous rappelle que la longueur d'un segment de droite [AB] est : $L = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$

A faire :

- Ecrire une fonction qui retourne la longueur du segment de droite entre les 2 points qui seront passés en argument d'entrée
- Ecrire le calcul l'approximation du cercle par le polynôme à N sommets, N étant un nombre saisi par l'utilisateur (strictement supérieur à 2).
- Inclure ce calcul dans une boucle for allant de 3 à N et qui calcule donc les différentes approximations. Faire les affichages qui conviennent pour que l'on observe la convergence vers 2π (plus il y a de sommets plus on s'approche du cercle). Dans cette étape on doit avoir 2 boucles for imbriquées.
- Modifier ce programme pour qu'il calcule le nombre de sommets nécessaires pour approximer le périmètre du cercle unité avec une précision donnée (par exemple 10^{-5})

Exercice n°3 : sur les chaines

Ecrire un programme qui affiche le nombre de caractères qui composent une chaine de caractères saisie par l'utilisateur.

Ecrire un programme qui retourne la place de la 3ième voyelle d'une chaine de caractères saisie par l'utilisateur.

Modifier le programme précédent pour qu'il affiche le caractère qui précède la 3ième voyelle de la chaine.

Exercice N°4 : sur les liste de nombres

Le résultat au dernier partiel est donné par la liste suivante :

```
Liste_de_Nombre = [11 3 12 5  4 16 17 11 14 7 9 12 14 6 12 5 12 7]
```

Ecrire et appeler la fonction qui retourne combien d'élèves qui ont une note strictement inférieure à la moyenne de la classe.

Exercice n° 5 : Polynôme

On considère la polynôme d'ordre 5

$$P(x) = -0.18x^5 - 3.41x^4 + 12.43x^3 + 242.62x^2 - 123.2$$

On décide de coder ce polynôme par la liste de ces coefficients :

```
Co =[-0.18 -3.41 12.43  242.62 0 -123.2]
```

Ecrire la fonction qui s'appellera par `Val =Polyval(Co,Nombre)` et qui permette d'évaluer le polynôme P(x) codé par `Co` au point `Nombre`. On utilisera obligatoirement une (et une seule) boucle for. Cette fonction doit pouvoir fonctionner avec un polynôme quelconque (en valeur et en dimension).

Ecrire un programme qui demande à l'utilisateur la saisie successive (boucle for) de 5 nombres flottants compris entre -10 et 10 et qui affiche lequel de ces 5 nombres donne la valeur de P(x) la plus grande en valeur absolue.

Problème n°2 : Histogramme

La fonction suivante :

```
def Lire(Fiche) :
    try :
        Flux= open(Fiche,'r',encoding='UTF-8')
        texte=Flux.read()
        Flux.close()
    except :
         texte=''
         print('Erreur d\'ouverture du fichier ',Fiche)
    return(texte)
```

permet de lire un fichier texte présent sur le disque et de retourner une chaine de caractères initialisée avec contenu de ce fichier.

Ecrire un programme qui lise un fichier texte, qui dénombre et affiche le nombre de mots de 1 lettre, de 2 lettres,…., de 7 lettres, ainsi que le nombre de mots de longueur supérieure à 7 lettres. On considère qu'un mot est une suite continue de lettres appartement à l'alphabet.

Exemple : « *Bonjour les étudiants ! Merci pour vos @ mails* » est une phrase contenant 2 mots de 3 lettres (*les, vos*), 1 mot de 4 lettres (*pour*), 2 mots de 5 lettres (*Merci, mails*) 1 mot de 7 lettres (*bonjour*) 1 mot de plus de 7 lettres (*étudiants*).

Question supplémentaires : comment inclure dans la longueur des mots, ceux contenant les caractères accentués ?

Nota : le fichier rimbaud.txt est joint à ce sujet pour vous tester vos programmes mais tout autre fichier texte fera très bien l'affaire.