# Correction Examen Atelier Programmation Python MBA Session Principale 2018

January 2, 2019

## 1 Module : Atelier de programmation

### 1.1 Exercice 1 : (sur les suites numériques)

Pour tout n > 0, on pose $U_n = \frac{1}{n^2}$

#### 1.1.1 a) Ecrire une fonction qui renvoie le n-ième terme de $U_n$

```
In [1]: # Version 1
        def U(n):
            return 1.0/n**2

        # Version 2
        def U(n):
            resultat = 1./(n*n)
            return resultat
```

#### 1.1.2 b) Écrire une fonction qui renvoie les premiers termes de la suite $U_n$

```
In [2]: # Version 1 en appelant la fonction U
        def n_premiers_termes(n):
            L = []
            for i in range(1,n+1):
                L.append(U(i))
            return L

        # Version 2 en appelant la fonction U
        def n_premiers_termes(n):
            L = [None] * n

            for i in range(1,n+1):
                L[i] = U(i)

            return L

        # Version 3 sans appel de la fonction U
```

```
        def n_premiers_termes(n):
            L = [0]*n

            for i in range(1,n+1):
                L[i] = 1./(i**2)

            return L

        # Version 4 en deux lignes
        def n_premiers_termes(n):
            return [U(i) for i in range(1,n+1)]

        # Version 5 en une seule ligne
        def n_premiers_termes(n): return [U(i) for i in range(1,n+1)]
```

#### 1.1.3 c) Ecrire une fonction qui renvoie le n-ième terme de $U_n$

```
In [3]: # Version 1
        def somme(n):
            L = n_premiers_termes(n)

            S = 0 # initialisation de S
            for i in range(n):
                S = S + L[i]

            return S

        # Version 2
        def somme(n):

            return sum(n_premiers_termes(n))
```

#### 1.1.4 d) Écrire une fonction qui représente graphiquement les n premiers termes. (indication : on pourrait utiliser le module matplotlib)

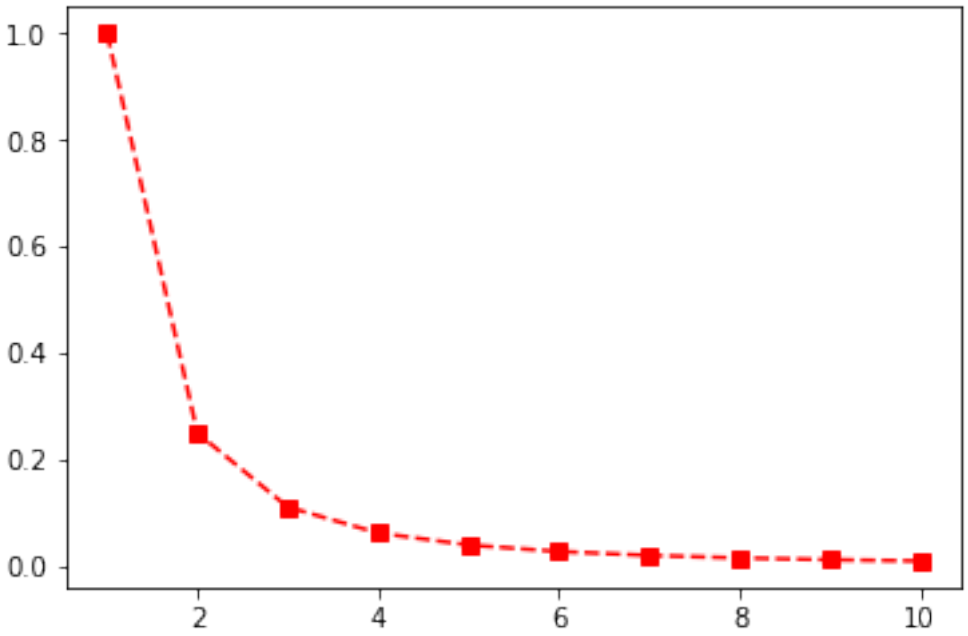
```
In [5]: import matplotlib.pyplot as plt

        n = 10 # par exemple

        indices = [i for i in range(1,n+1)]

        L = n_premiers_termes(n)

        plt.plot(indices,L,'rs--')
        plt.show()
```




## 1.2 Exercice 2 : (sur les listes)

### 1.2.1 a) Écrire un programme demandant la saisie de deux nombres, rappelle leurs valeurs puis affiche leur somme.

```
In [6]: a = input("Donner le premier nombre")

        a = int(a)

        b =  int(input("Donner le deuxième nombre"))

        print("La somme de a et de b est ",a+b)
```

```
Donner le premier nombre 14
Donner le deuxième nombre 569


La somme de a et de b est  583
```

### 1.2.2 b) Écrire un programme qui :

- crée une liste vide a
- demande la saisie d'un nombre de départ, d'un nombre d'arrivée et d'un pas
- ajoute à la liste a les trois nombres
- ajoute à la liste a une autre liste, constituée à partir des nombres saisis (nombre de départ, nombre maximal, pas d'augmentation)

- affiche la liste
- affiche les pas derniers éléments de la liste

```
In [7]: a = []

In [8]: depart = int(input("Donner un nombre de départ"))
        arrivee = int(input("Donner un nombre d'arrivée"))
        pas = int(input("Donner un pas"))

Donner un nombre de départ 3
Donner un nombre d'arrivée 26
Donner un pas 1


In [9]: a.append(depart)
        a.append(arrivee)
        a.append(pas)

In [10]: b = a.copy()
         a.append(b)

In [11]: print(a)

[3, 26, 1, [3, 26, 1]]


In [12]: a[2]

Out[12]: 1

In [13]: a[3][2]

Out[13]: 1
```

#### 1.2.3 c)

- Initialiser la liste a à la valeur [1,1,1,3,1,4,1,1,3,6,1,2,3,1,4]
- afficher a
- afficher le nombre de 3 présents dans a
- afficher la position du dernier 3
- retirer le dernier 3 de la liste
- afficher a
- afficher le nombre de 3 présents dans a
- afficher la position du dernier 3

```
In [14]: a = [1,1,1,3,1,4,1,1,3,6,1,2,3,1,4]

In [15]: print(a)

[1, 1, 1, 3, 1, 4, 1, 1, 3, 6, 1, 2, 3, 1, 4]
```

```
In [16]: # Le nombre de 3 présents dans a
         # on utilise la fonction count
         a.count(3)
```

```
Out[16]: 3
```

```
In [17]: # C'est astucieux
         # il suffit d'inverser la liste
         # puis d'appeler la fonction index
         # pour déterminer l'indice du dernier 3
         # dans la liste inversée
         # enfin on applique un calcul mathématique
         # pour trouver l'indice correct

         print("l'indice du dernier 3 est : ")
         len(a) - a[::-1].index(3) - 1
```

```
l'indice du dernier 3 est :
```

```
Out[17]: 12
```

```
In [18]: # On supprime un élément à partir de son indice avec la fonction pop()
         a.pop(12)
```

```
Out[18]: 3
```

```
In [19]: print(a)
```

```
[1, 1, 1, 3, 1, 4, 1, 1, 3, 6, 1, 2, 1, 4]
```

```
In [20]: a.count(3)
```

```
Out[20]: 2
```

```
In [21]: print("l'indice du dernier 3 est : ")
         len(a) - a[::-1].index(3) - 1
```

```
l'indice du dernier 3 est :
```

```
Out[21]: 8
```

### 1.3 Exercice 4 (Un test)

```
In [22]: chaine = input("Veuillez saisir une chaîne de caractères")
```

```
Veuillez saisir une chaîne de caractères on va à l'école
```

```
In [23]: if 'e' in chaine:
             print("La chaîne saisie contient la lettre e")
         else:
             print("La chaîne saisie ne contient pas la lettre e")
```

```
La chaîne saisie contient la lettre e
```

## 1.4 Exercice 5 (une boucle for)

### 1.4.1 crée la liste a de tous les nombres entiers inférieurs à 100

```
In [24]: a = [i for i in range(100)]
```

### 1.4.2 demande à l'utilisateur de saisir un nombre diviseur

```
In [25]: # Version 1 basique
         diviseur = int(input("Donner un nombre diviseur"))
```

```
Donner un nombre diviseur 5
```

```
In [26]: # Version 2 plus élaborée
         diviseur = int(input("Donner un nombre diviseur"))
         while diviseur <= 0:
                 diviseur = int(input("Donner un nombre diviseur"))
         print(diviseur)
```

```
Donner un nombre diviseur 5
```

```
5
```

### 1.4.3 crée à partir de a la liste de tous les nombres inférieurs à 100 qui sont multiples de diviseur

```
In [27]: L = []
         for nombre in a:
             if nombre % diviseur == 0:
                 L.append(nombre)
```

```
In [28]: print(L)
```

```
[0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95]
```

## 1.5 Exercice 6 (une boucle while)

- crée une liste des nombres entiers compris entre 3 et 15 inclus
- demande à l'utilisateur des nombres et tant que ces nombres sont présents dans la liste, les en retire
- si le nombre n'est pas dans la liste, le programme s'arrête en affichant la liste initiale, puis la liste finalement obtenue.

```
In [29]: # Création de la liste initiale
         a = [i for i in range(3,16)]

In [30]: # On réalise une copie
         b = a[:]

In [31]: nombre = int(input("Donner un entier"))

         while nombre in b:
             b.remove(nombre)
             nombre = int(input("Donner un entier"))

         print("La liste initiale est ",a)
         print("La liste finale est ",b)
```

```
Donner un entier 3
Donner un entier 15
Donner un entier 10
Donner un entier 19


La liste initiale est  [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
La liste finale est  [4, 5, 6, 7, 8, 9, 11, 12, 13, 14]
```

## 1.6 Exercice 7: Une fonction!

Définir une fonction prenant comme paramètres l'année puis le numéro de mois (par exemple, 3 pour mars) et qui renvoie le nombre de jours du mois correspondant. Les valeurs par défaut sont 2016 pour l'année et 4 pour le mois.

```
In [32]: def nombre_jours(annee=2016,mois=4):
             mois31 = [1,3,5,7,8,10,12]
             mois30 = [4,6,9,11]
             mois28 = [2]
             if mois in mois31:
                 return 31
             elif mois in mois30:
                 return 30
             else:
                 return 28
```

## 1.7 Exercice 8: Deux fonctions!!

Définir les fonctions suivantes : - a) f(x) = $x^2$ - b) g(x,y) = $x^4 + y^4$

```
In [33]: def f(x):
             return x**2

In [34]: def g(x,y):
             return x**4 + y**4
```

## 1.8 Exercice 9: (Un algorithme de tri par sélection)

### 1.8.1 a) Écrire une fonction abmax renvoyant le plus grand de deux nombres a et b.

```
In [35]: def abmax(a,b):
             if a > b:
                 return a
             else:
                 return b
```

### 1.8.2 b) Écrire une fonction vmax de paramètre d'entrée un vecteur v renvoyant le plus grand élément de v.

```
In [36]: def vmax(v):
             variable = v[0]
             for i in range(1,len(v)):
                 if v[i] > variable:
                     variable = v[i]
             return variable

In [37]: # Test
         a = [2,55,4,66,8,9]
         print(vmax(a))
```

```
66
```

### 1.8.3 c) Écrire une fonction permut de paramètres entrée un vecteur v ainsi que deux entiers i et j permutant les éléments v[i] et v[j], le vecteur initial étant ainsi modifié.

```
In [38]: # Version 1
         # On effectue une permutation avec python
         def permut(v,i,j):
             v[i], v[j] = v[j], v[i]
             return v

In [39]: # Version 2
         # On effectue une permutation classique
         # via une variable auxiliaire
         def permut(v,i,j):
```

```
            auxiliaire = v[i]
            v[i] = v[j]
            v[j] = auxiliaire
            return v
```

```
In [40]: # Test
         a = [2,55,4,66,8,9]
         print("la liste initiale",a)
         print("la liste permutée",permut(a,0,-1))
```

```
la liste initiale [2, 55, 4, 66, 8, 9]
la liste permutée [9, 55, 4, 66, 8, 2]
```

#### 1.8.4 d) Écrire une fonction ind_du_max de paramètres d'entrée un vecteur v et un entier k renvoyant i_pg l'indice du plus grand élément parmi ceux dont l'indice est supérieur ou égal à k.

```
In [41]: def ind_du_max(v,k):
             i_pg = k

             for i in range(k,len(v)):
                 if v[i] > v[i_pg]:
                     i_pg = i

             return i_pg
```

```
In [42]: # Test
         b= [22, 5, 10, 1 , 8, 4]
         print(ind_du_max(b,0)) # k = 0
         print(ind_du_max(b,1)) # k = 1
         print(ind_du_max(b,3)) # k = 3
```

```
0
2
4
```

#### 1.8.5 e) Tester la fonction triselection à l'aide du vecteur c= [0, 1, 2, 3, -8 ,-7]....

```
In [43]: def triselection(v):
             L=len(v)
             for i in range(L-1) :
                 i_pg=ind_du_max(v, i)
                 permut(v ,i, i_pg)
                 print (i, i_pg, v) # enlever si vecteurs trop longs
```

```
In [44]: # Test
         c= [0, 1, 2, 3, -8 ,-7]
```

```
In [45]: triselection(c)

0 3 [3, 1, 2, 0, -8, -7]
1 2 [3, 2, 1, 0, -8, -7]
2 2 [3, 2, 1, 0, -8, -7]
3 3 [3, 2, 1, 0, -8, -7]
4 5 [3, 2, 1, 0, -7, -8]
```