Devoir-Maison 3 - CORRECTION

1. Créer une fonction ***Estpremiers(n)*** qui reçoit en paramètre un nombre entier naturel n et qui renvoie :
   — True si n est premier
   — False sinon
   .
   Rappel : un nombre premier est un entier naturel qui admet exactement deux diviseurs distincts entiers et positifs. Attention, 1 n'est pas un nombre premier.

```
def estpremier(n):
    cpt=0
    for i in range(1,n+1):
        if n%i==0:
            cpt=cpt+1
    return(cpt==2)
```

2. Créer une fonction ***Listepremiers(n)*** qui reçoit en paramètre un nombre entier naturel n et qui renvoie une liste de tous les nombres premiers inférieurs ou égaux à n.
   Par exemple, decompose(30) retourne :

   [2, 3, 5, 7, 11, 13, 17, 19, 23, 29].

```
def Listepremiers(n):
    L=[]
    for i in range(n+1):
        if Estpremier(i):
            L.append(i)
    return(L)
```

3. Créer une fonction ***Factorielle(n)*** qui reçoit en paramètre un nombre entier naturel n et qui renvoie la factorielle de n (sans utiliser de fonctions prédéfinies).

```
def Factorielle(n):
    if n==0:
        return(1)
    elif n==1:
        return(1)
    else:
        p=1
        for i in range(2,n+1):
            p=p*i
        return(p)
```

4. On se propose de programmer une fonction cosinus. On admet la formule suivante :

$$cos(x) = \sum_{k=0}^{+\infty} \frac{(-1)^k x^{2x}}{(2k)!}$$

Numériquement, on ne peut pas faire la somme de 0 à l'infini. On s'arrêtera donc à un entier n :

$$cos(x) \approx \sum_{k=0}^{n} \frac{(-1)^k x^{2k}}{(2k)!}$$

(a) Écrire une fonction ***Cosinus(x,n)***qui prend comme arguments un réel x et un entier naturel n et qui renvoie le cosinus de x calculé en prenant en compte n termes de la somme définie plus haut.

(b) Faire afficher le pourcentage d'erreur pour $x = \frac{\pi}{4}$ et $n = 5$.

```
def Cosinus(x,n):
    s=0
    for k in range(n):
        s=s+((-1)**k)*(x**(2*k))/Factorielle(2*k)
    return(s)

from math import *
rac=sqrt(2)/2
approx=Cosinus(pi/4,5)
pourcentage=(abs((rac-approx)/rac))*100
print(pourcentage)
```

On obtient pour pourcentage d'erreur : $3.4643631319841066 \times 10^{-6}$