1. **Test :**

Compléter la fonction suivante qui renvoie le prix d'une monture de lunettes vendu par un opticien qui fait la publicité suivante : "Si vous avez moins de 60 ans, recevez une remise d'un pourcentage égal à votre âge et si vous avez 60 ans ou plus, recevez une remise égale à votre âge".
Par exemple, un client de 28 ans aura 28% de remise et mamie qui a 74 ans aura 74€de remise.

```
def prixmonture(x,n) :
    """ Cette fonction renvoie le prix final d'une monture coûtant x euros après réduction éventuelle pour un client âgé de n ans """
    if n<60 :
        return x*(1-n/100)
    else :
        return x-n
```

Bernard a 59 ans. Il doit s'acheter une nouvelle monture. Expliquer ce que fait cette fonction Python.

```
def achat() :
    """ Cette fonction renvoie le prix minimal pour lequel un client de 59 ans paye moins cher qu'un client de 60 ans """
    p=60 # Prix minimum d'une paire de chaussure dans ce magasin
    while prixmonture(p,59)>prixmonture(p,60) :
        p+=1
    return p
```

Que doit faire Bernard si le prix de la monture choisie est inférieure à la valeur retournée par la fonction ?
Et s'il est supérieur ? Il peut l'acheter maintenant.

2. **Suites récurrentes :**

On considère la suite récurrente définie par $u_0 = 2$ et $\forall n \in \mathbb{N},\ u_{n+1} = -\frac{2}{3}u_n^2 + \frac{u_n}{2} - 1$

On veut connaître pour un n donné la valeur de u_n ainsi que $S = \sum_{k=0}^{n} u_k$.

Ecrire une fonction Python qui remplit ces deux objectifs.

```
def recusomme(n) :
    """ Cette fonction renvoie u_n et sum_{k=0}^{n} u_k pour un entier n donné """
    u=2 # initialisation
    s=2 # initialisation
    for k in range(1,n+1) :
        u=-2/3*u**2+u/2-1 # calcul du terme suivant de la suite
        s+=u # on rajoute ce terme à la somme
    return u,s
```

3. **Autour des listes de nombres :**

Ecrire une fonction produitliste qui renvoie la valeur du produit des éléments d'une liste donnée :

```
def produitliste(L) :
    """ Cette fonction calcule et renvoie le produit des éléments d'une liste L donnée """
    n=len(L) # taille de la liste L
    x=1 # initialisation d'un produit à 1
    for i in range(n) : # i va prendre les valeurs de 0 à n-1
        x=x*L[i]
    return x
```

Compléter la fonction suivante qui détermine le minimum d'une liste L donnée :

```
def minimumliste(L) :
    """ Cette fonction détermine et renvoie le minimum d'une liste de nombres L donnée """
    n=len(L) # taille de la liste L
    mini=L[0] # minimum provisoire
    for k in range(1,n+1) :
        if mini>L[k] :
            mini=L[k] # nouveau minimum provisoire
    return mini
```

4. **Entiers et factorions :** Expliquer ce que fait la fonction suivante :

```
def liste_entier(n) :
    """ Cette fonction prend en entrée un entier n et renvoie la liste des chiffres utilisés
    dans l'écriture de n en base 10 """
    x=str(n)
    p=len(x)
    L=p*[0]
    for k in range(p) :
        L[k]=int(x[k])
    return(L)
```

Que donne : liste_entier(145) ? Cela renvoie $[1, 4, 5]$

Ecrire la fonction factorielle qui étant donné un entier n renvoie n !, puis compléter les fonctions suivantes :

```
def factorielle(n) :
    """ Cette fonction renvoie n ! """
    f=1 # initialisation
    for k in range(1,n+1) :
        f=f*k
    return f
```

```
def factorion(n) :
    """ Cette fonction teste si un nombre est un factorion, ie si ce nombre est égal à la somme des
    factorielles des chiffres qui le compose. Elle renverra True dans ce cas, False sinon. """
    L=liste_entier(n) # On transforme n en la liste de ses chiffres
    S=0 # initialisation d'un somme
    for c in L : # on parcourt L
        S+=factorielle(c)
    if S==n :
        return True
    else :
        return False
```

Que donne factorion(125) ? False, car $1! + 2! + 5! = 1 + 2 + 120 = 123 \neq 125$

Et factorion(145) ? True, car $1! + 4! + 5! = 1 + 24 + 120 = 145$

```
def Listes_factorion(N) :
    """ Cette fonction renvoie la liste des factorions inférieurs ou égaux à N. """
    L=[ ] # initialisation de la liste vide
    for n in range(1,N+1) :
        if factorion(n) : # si n est un factorion
            L.append(n) # on le rajoute à la liste
    return L
```