IC Devoir 2 - Mercredi 13 décembre - 1h

Exercice 1

1. Compléter la fonction suivante pour qu'elle retourne la somme des diviseurs propres (positifs et différents de lui-même) d'un entier naturel.

```
1 def sd(n):
2     s=0
3     for i in range(1,n):
4         if ...:
5             ...
6     return(s)
```

Par exemple, `sd(20)` revoie 22.
Rappel :
- `%` : reste de la division à quotient entier
- `//` : quotient de la division à quotient entier

Par exemple : `14%5` donne 4 tandis que `14//5` donne 2 car $14 = 2\times 5+4$

2. Dénombrer le nombre de boucles dans la structure répétitive de `sd()`.
3. Proposer une amélioration réduisant ce nombre et précisez la nouvelle quantité obtenue à l'unité près.
4. Un entier naturel n est dit **parfait** lorsque la somme de ses diviseurs propres est égale à n.
Par exemple : 6 est parfait car $6 = 1+2+3$.

Utilisant la fonction `sd()`, donner une fonction `parfait(n)` qui retourne la liste des nombres parfaits inférieurs à `n`.
5. Deux entiers naturels sont dits **amis** si la somme des diviseurs propres de l'un et égale à l'autre et vice versa.
Par exemple : 220 et 284 car `sd(220)=284` et `sd(284)=220`.

Utilisant la fonction `sd()`, donner une fonction `amis(n)` qui retourne la liste des couples de nombres amis dont l'un est inférieurs à `n`.

Exercice 2 *Recherche d'un mot* **Prog**

On s'intéresse à une fonction `recherche(mot,texte)` qui prend en entrées deux chaînes de caractères et retourne la liste des positions initiales de toute insertion de la première dans la deuxième : cela revient à chercher un mot dans un texte.

`recherche('aa','aaaabaa')` retourne `[0, 1, 2, 5]`

➢ L'algorithme est :

Entrée : mot, texte : deux chaînes de caractères

Initialiser `L` à une liste vide
Pour toute les positions possibles `i` dans `texte` faire
  Si `mot` est écrit dans `texte` à partir de la position `i` alors
    Ajouter `i` à `L`
  FinSi
FinPour

Sortie : `L`

1. Que retourne l'instruction `recherche('12','12321232123')` ?
2. Proposer un script de la fonction `recherche(mot,texte)` qui comporte des tests d'égalité entre deux caractères (mais entre deux listes).

Exercice 3 *Traitement d'image*

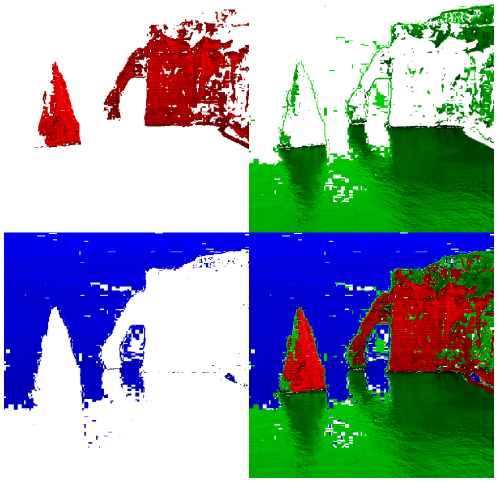
1. Identifier la transformation que fait la fonction suivante où la variable `f_in` est le nom de l'image initiale et `f_out` celle du nom de l'image transformée.

```
def transfo(f_in,f_out):
    f=plt.imread(f_in)
    l,c,d=np.shape(f)
    g=np.zeros((l,c,3))
    for i in range(l):
        for j in range(c):
            g[i,j,:3]=f[i,-j-1,:3]
    plt.imsave(f_out,g)
```

2. Écrire une fonction `quadri(f_in,f_out)` qui retourne une image composée de 4 reproductions de l'image initiale définies par :
- en haut à gauche, la reproduction est transformée par

$$[r,v,b] \mapsto \begin{cases} [r,0,0] & \text{si } r > \max(v,b) \\ [1,1,1] & \text{sinon} \end{cases}$$

Autrement dit, on opère un filtre sur le *rouge dominant* : le pixel conserve la teinte rouge et annule les autres, si la teinte rouge est plus grande que celles verte et bleue, sinon le pixel devient blanc.
- en haut à droite, idem avec le vert
- en bas à gauche, idem avec le bleu
- en bas à droite, un superposition des 3 premières images : seule la teinte dominante est conservée.



Lycée Saint-Exupery – maffre.saintexupery@gmail.com – MPSI

Exercice 4 *Liste*
Quelle propriété est testée par la fonction `T` sachant que `L` est une liste de nombres :

```
1 def T(L):
2     for i in range(len(L)-1):
3         if L[i]<L[i+1]:
4             return False
5     return True
```

Exercice 5 *Année bissextile*
Une année est bissextile si :
- elle est divisible par 4, mais pas par 100
- ou si elle l'est par 400.

Par exemple, 2014 et 2100 ne sont pas bissextiles, mais 2000 et 2020 le sont.
Écrire le script d'une fonction, `bissextile(n)`, qui retourne `True` si `n` est bissextile et `False` sinon. On pourra inclure une instruction de type `assert` pour vérifier que `n` est un entier naturel.

Exercice 6 *Chaîne de caractères*
Considérons `c` une chaîne de caractères uniquement composée de lettres minuscules non accentuées (pas de ponctuation, pas d'espace non plus). Donner les trois fonctions suivantes, vous pourrez utiliser les méthodes associées au type `str` :
1. `nblc(c)` donne le nombre de couple de lettres consécutives du texte qui sont aussi consécutives dans l'alphabet (dans le même ordre).
Exemple `nblc('abdef')` renvoie 3 pour les couples `ab`, `de` et `ef`.

A compléter

```
1 def nblc(c):
2     abc='abcdefghijklmnopqrstuvwxyz'
3     nb=...
4     for k in range(...):
5         i=...
6         if c[k+1]==abc[(i+1)%26]:
7             nb=...
8     return nb
```

2. `nbs(c)` donne le nombre de sandwichs, c'est-à-dire le nombre de lettres entourées directement par une même lettre.
Exemple `nbs('ababb')` renvoie 2 pour `aba` et `bab`.
3. `lplrc(c)` donne la longueur de la plus longue répétition consécutive d'une lettre.
Exemple `nblc('aabbbba')` renvoie 4.

Exercice 7 *Dichotomie* **Prog**
1. Donner le script d'une fonction `dicho(f,a,b,p)` qui met en place l'algorithme de dichotomie pour déterminer une racine de la fonction f sur $[a,b]$ à la précision p.
2. Donner les instructions qui utilise `dicho` pour trouver une solution de l'équation $x^3+2x-1=0$ sur $[0;1]$ à 10^{-5} près.

IC Devoir 2- Proposition de solutions

Solution 1 1. Compléter la fonction suivante pour qu'elle retourne la somme des diviseurs propres (positifs et différents de lui-même) d'un entier naturel.

```
1 def sd(n):
2     s=0
3     for i in range(1,n):
4         if n%i==0: s=s+i
5     return(s)
```

2. Le nombre de boucles est $n-1$.
3. Une amélioration consiste à travailler jusqu'à la racine carrée du nombre. Chaque diviseur propre d donne un autre diviseur $\frac{n}{d}$: tous les diviseurs propres sont trouvés.
Une petite discussion permet de traiter le cas particulier où n est un carré parfait. La nouvelle quantité est $\lfloor\sqrt{n}\rfloor$.

```
def sd2(n):
    if n<2:
        return(0)
    s,i=1,2
    while i*i<n:
        if n%i==0: s=s+i+n//i
        i=i+1
    if i*i==n: s=s+i
    return(s)
```

4. Recherche des nombres parfaits :

```
def parfait(n):
 return([i for i in range(2,n+1) if sd(i)==i])
```

5. Recherches des nombres amis :

```
def amis(n):
    l=[]
    for i in range(2,n+1):
        j=sd(i)
        if sd(j)==i and i<j: l.append([(i,j)])
    return(l)
```

Solution 2 **Recherche d'un mot**

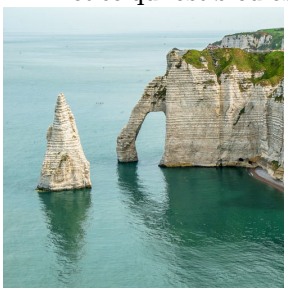
- `recherche('12','12321232123')` retourne `[0,4,8]`.

Recherche d'un mot

```
def recherche(mot,texte):
    L=[]
    for i in range(0,len(texte)-len(mot)+1):
        j=0
        while j<len(mot) and texte[i+j]==mot[j]:
            j=j+1
        if j==len(mot): L.append(i)
    return L
```

Solution 3 Traitement d'image

1. La transformation est une permutation des couleurs :
   - ce qui est rouge est devenu bleu
   - ce qui est vert est devenu rouge
   - et ce qui est bleu est devenu vert.



2. Un script de la fonction `quadri` est

```
def quadri(f_in,f_out):
    f=plt.imread(f_in)
    l,c,d=np.shape(f)
    g=np.ones((2*l,2*c,3))
    e=np.ones((l,c,3))
    # Le rouge
    h=np.ones((l,c,3))
    for i in range(l):
        for j in range(c):
            if f[i,j,0]>max(f[i,j,1],f[i,j,2]):
                h[i,j]=np.array([f[i,j,0],0,0])
                e[i,j]=np.array([f[i,j,0],0,0])
    g[:l,:c]=h
    # Le vert
    h=np.ones((l,c,3))
    for i in range(l):
        for j in range(c):
            if f[i,j,1]>max(f[i,j,0],f[i,j,2]):
                h[i,j]=np.array([0,f[i,j][1],0])
                e[i,j]=np.array([0,f[i,j][1],0])
    g[:l,c:]=h
    # Le bleu
    h=np.ones((l,c,3))
    for i in range(l):
        for j in range(c):
            if f[i,j,2]>max(f[i,j,0],f[i,j,1]):
                h[i,j]=np.array([0,0,f[i,j][2]])
                e[i,j]=np.array([0,0,f[i,j][2]])
    g[l:,:c]=h
    # La superposition
    g[l:,c:]=e[:,:,:3]
    plt.imsave(f_out,g)
```

Solution 4 Liste

Le programme définit une fonction qui teste si la liste est ordonnée par ordre décroissant : du plus grand au plus petit.
En effet, on parcours la liste de gauche à droite jusqu'à l'avant dernier élément :
1. s'il existe i tel L[i]<L[i+1] alors la fonction retourne False
2. si pour tout i, L[i]≥L[i+1] alors la fonction retourne True.

```
>>> L=[5,4,3,2,0]
>>> print(T(L))
True
>>> LL=[1,5,2,0]
>>> print(T(L))
False
```

Solution 5 Test pour année bissextile :

```
def bissextile(n):
    assert type(n)==int and n>=0,'ENC'
    return (n%4==0 and n%100!=0) or n%400==0
```

Autre version :

```
def bissextile2(n):
    assert type(n)==int and n>=0,'ENC'
    if n%400==0:
        return True
    elif n%4==0 and n%100!=0:
        return True
    else:
        return False
```

Solution 6 Chaîne de caractères

```
1 def nblc(c):
2     abc='abcdefghijklmnopqrstuvwxyz'
3     nb=0
4     for k in range(len(c)-1):
5         i=abc.index(c[k])
6         if c[k+1]==abc[(i+1)%26]:
7             nb=nb+1
8     return nb
```

```
def nbs(c):
    nb=0
    for k in range(1,len(c)-1):
        if c[k-1]==c[k+1]:
            nb=nb+1
    return nb
```

```
def lplrc(c):
    L,l=1,1
    for k in range(1,len(c)):
        if c[k]==c[k-1]:
            l=l+1
        else:
            if l>L:
                L=l
            l=1
    if l>L:
        L=l
    return L
```

Solution 7 Dichotomie

```
def dicho(f,a,b,p):
    while b-a>p:
        c=(a+b)/2
        if f(c)*f(a)<=0: b=c
        else: a=c
    return (a+b)/2
```

⇛ Application à la résolution de $x^3+2x-1=0$ sur $[0;1]$

```
>>> dicho(lambda t:t**3+2*t-1,0,1,1e-5)
0.4533958435058594
```