Les chaines de caractères (Série 4)

Exercices corrigés

Exercice 1 :

Un entier est dit distinct s'il est composé de chiffres distincts (différents).

Ecrire un programme python qui permet de saisir un entier n (n>0), puis de vérifier et d'afficher si cet entier est distinct ou non.

Exemple :

N=1273 est dit distinct car il est formé par les chiffres 1, 2, 7 et 3 qui sont tous distincts, donc, le programme affichera : ***cet entier est distinct***

N=1565 est dit non distinct car il est formé par les chiffres 1, 5, 6, 5 qui ne sont pas tous distincts (le chiffre 5 se répète deux fois, donc le programme affichera : ***cet entier est non distinct***

⇒ *Corrigé*

```
n=int(input('saisir un entier :  '))
etat=True
if n>0 :
        ch=str(n)
        for lettre in ch :
                if ch.count(lettre)>1 :
                        etat=False
                        break
        if etat==True :
                print(n,'  est distinct'
        else :
                print(n,' est non distinct')
else :
        print('saisir un entier positif')
```

Exercice 2 :

Ecrire un programme python qui permet d'afficher tous les entiers positifs de trois chiffres de la forme ***cdu*** *tel que, pour chaque entier, la somme de ses chiffres (****c+d+u****) est un diviseur du produit de ses chiffres (****c*d*u****)*

Exemple :

L'entier 514 vérifie cette propriété, en effet, ***(5+1+4) =10*** *est un diviseur de* ***(5*1*4) =20***

⇒ *Corrigé*

```
for i in range(100,1000) :
```

```
        ch=str(i)
        s=int(ch[0])+ int(ch[1])+ int(ch[2])
        p=int(ch[0])* int(ch[1])* int(ch[2])
        if p%s==0 :
                print(i)
```

Exercice 3 :

En arithmétique, un auto-nombre est un entier naturel N qui ne peut pas s'écrire sous la forme d'un nombre M ajouté à la somme des chiffres de M.

- ***Exemple :***
  - Pour N=21, n'est pas un auto nombre, puisqu'il peut être généré à partir de la somme d'un nombre M égal à 15 et les chiffres qui le constituent (1 et 5) c'est-à-dire 21=15+1+5.
  - Pour N=20, est un auto-nombre puisqu'il ne peut pas être généré à partir de la somme d'un nombre M et les chiffres qui le constituent.

Ecrire un programme permettant de vérifier si un entier naturel N strictement positif est un auto-nombre.

⇒ *Corrigé*

```
def auto_nombre(N) :
        etat=False
        for i in range(N):
                s=i
                M=str(i)
                for j in M:
                        s+=int(j)
                if s==N :
                        etat=True
                        break
        return etat
```

Exercice 4 :

Ecrire un programme Python qui permet de déterminer si un entier N de quatre chiffres vérifie la relation suivante :

N=somme des puissance $K^{ème}$ de ses chiffres, avec 1<=K<=5

- ***Exemple :***

Pour voir si le nombre n=1634 vérifie ou non cette propriété on commence par calculer la somme des chiffres à la puissance 1, puis à la puissance 2, puis à la puissance 3,… :

- $1^1+6^1+3^1+4^1=14$ est différent de 1634 alors on continue avec les chiffres à la puissance 2
- $1^2+6^2+3^2+4^2=62$ est différent de 1634 alors on continue avec les chiffres à la puissance 3
- $2^3+6^3+3^3+4^3=308$ est différent de 1634 alors on continue avec les chiffres à la puissance 4
- $1^4+6^4+3^4+4^4=1634$ est égal à 1634 alors on arrête le traitement et on affiche : n=1634 et K=4

Pour le nombre n=2114, voyons s'il vérifie ou pas la propriété :

- $2^1+1^1+1^1+4^1=8$ est différent de 2114 alors on continue avec les chiffres à la puissance 2
- $2^2+1^2+1^2+4^2=22$ est différent de 2114 alors on continue avec les chiffres à la puissance 2
- $2^3+1^3+1^3+4^3=74$ est différent de 2114 alors on continue avec les chiffres à la puissance 2
- $2^4+1^4+1^4+4^4=274$ est différent de 2114 alors on continue avec les chiffres à la puissance 2
- $2^5+1^5+1^5+4^5=1058$ est différent de 2114 alors on arrête le traitement et on affiche le message : n=2114 ne vérifie pas la propriété

⇒ *Corrigé*

```
def somme_chiffres(N) :
        NB=str(N)
        if len(NB) !=4 :
                print('le nombre doit contenir 4 chiffres ')
        else :

                s=int(NB[0])+ int(NB[1])+ int(NB[2])+ int(NB[3])
                puis=2
                while s<N and puis<=5:
                        s=int(NB[0])**puis+ int(NB[1]) **puis + int(NB[2]) **puis + int(NB[3]) **puis
                        puis+=1
                if s==N :
                        return True
                else :
                        return False
```

Exercice 5 :

Un nombre premier N est dit circulaire s'il vérifie la propriété suivante : chacune des rotations de ses chiffres d'un élément vers la droite, forme à son tour un nombre premier.

- ***Exemple :***

Si N=719 est un nombre premier circulaire car 719, 971 et 197 sont des nombres premiers avec :

- 971 est le nombre obtenu après une rotation des chiffres de 719 d'un élément vers la droite.
- 197 est le nombre obtenu après une rotation des chiffres de 971 d'un élément vers la droite

Si N=23, N n'est pas un nombre premier circulaire car il est premier mais 32 ne l'est pas.

Si N=6102, N n'est pas un nombre premier circulaire car il n'est pas premier.

Ecrire un programme Python permettant de chercher tous les nombre premiers circulaire se trouvant dans un intervalle [p,q] fournis par l'utilisateur.

⇒ *Corrigé*

```
def premier(n):
    if n==2 :
        return True
    etat=True
    for i in range(2, int(sqrt(n))+2):
        if n%i==0:
            etat=False
            break
    return etat
def circulaire(p,q):
    for i in range(p,q+1):
        if premier(i)==True:
            etat=True
            ch=str(i)
            k=0
            while k<len(ch) and etat==True:
                ch=ch[-1]+ch[:len(ch)-1]
                if premier(int(ch))==False:
                    etat=False
                    break
                k+=1
            if etat==True:
                print('nombre : ',i)
```

Exercice 6 :

On définit le poids d'une chaine comme étant la somme des produits de la position de chaque voyelle dans cette chaine par son rang dans l'alphabet français.

Si la chaine ne contient pas de voyelles alors son poids est égal à zéro.

N.B : les voyelles sont A, E, I, O, U, Y et leurs rangs respectifs sont :1, 5, 9, 15, 21, 25

- ***Exemple :***

*La chaine 'BONNE' contient 2 voyelles 'O' et 'E', sont poids est égal à 2*15+5*5=55*

*La chaine 'CHANCE' contient 2 voyelles 'A' et 'E', son poids est égal à : 3*1+6*5=33*

- ***Travail à faire :***

Ecrire un programme Python qui permet de lire une chaine non vide, composée seulement par des lettres alphabétiques majuscules puis calcule et affiche le poids de cette chaine.

⇒ *Corrigé*

```
def poids(s):
        s=s.upper()
        alpha='ABCDEFGHIJKLMNOPQRSTUVWXYZ'
        voy='AEIOUY'
        pos=1
        poids=0
        for Lettre in s:
                if Lettre in voy
                        poids+=pos*(alpha.find(Lettre)+1)
                pos+=1
        return poids
```

Exercice 7

On se propose d'écrire un programme Python permettant de déterminer et d'afficher un code à partir d'un entier N strictement positif et supérieur à 100, selon le principe suivant :

- Calculer la somme S des chiffres qui composent le nombre N
- Recommencer le calcul de la somme des chiffres de la somme obtenue S tant que celle-ci n'est pas comprise entre 1 et 9.

Le code sera le nombre formé par N auquel on place à sa gauche la dernière somme obtenue.

- ***Exemple :***

Pour N=9867, le programme affichera : le code est 39867

En effet :

Pour N=9867 :

- La 1ère somme S vaut 30 (car 9+8+6+7=30)
- La 2ème somme S vaut 3 (car 3+0=3)
- Etant donné que la derniers somme S, qui vaut 3, est comprise entre 1 et 9, le code sera 39867

⇒ *Corrigé*

```
def code_nombre(n):
    if n>=100:
        s=n
        while s<1 or s>9:
            nb=str(s)
            s=0
            for lettre in nb:
                s+=int(lettre)
        return str(s)+str(n)
```