UNIVERSITÉ ABDELMALEK ESSAADI
FACULTÉ DES SCIENCES ET TECHNIQUES DE TANGER
DÉPARTEMENT GÉNIE INFORMATIQUE

POLYCOPIÉ DES TDs CORRIGÉS

PROGRAMMATION ORIENTÉE OBJETS EN PYTHON

Exercices corrigés

MASTERS

MOBIQUITÉ ET BIG DATA & SYSTÈMES INFORMATIQUES ET MOBILES

PR. ABDERRAHIM GHADI

DÉPARTEMENT GÉNIE INFORMATIQUE

2018 — 2021

Table des matières

EXERCICES

SÉRIE 1

Exercice 1

Écrire un programme Python permettant d'obtenir la version de Python que vous utilisez.

Exercice 2

Demander un entier à l'utilisateur. Selon que le nombre est pair ou impair, afficher un message approprié à l'utilisateur. (Indication : comment un nombre pair/impair réagit-il différemment lorsqu'il est divisé par 2 ?)

Exercice 3

Créer un programme qui demande à l'utilisateur d'entrer son nom et son âge. Afficher un message qui lui est adressé indiquant l'année où il aura 100 ans.

Exercice 4

Soit la liste suivante : a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89] . écrire un programme qui affiche tous les éléments de la liste qui sont inférieurs à 15.

Exercice 5

Créez un programme qui demande à l'utilisateur un nombre et qui affiche ensuite une liste de tous les diviseurs de ce nombre.

Exercice 6

Prenez deux listes, disons par exemple ces deux-là :

```
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
```

et écrire un programme qui renvoie une liste qui ne contient que les éléments communs entre les listes (sans doublons). Assurez-vous que votre programme fonctionne sur deux listes de tailles différentes.

Remarque :
On peut créer un liste aléatoire comme suit :

```
import random
a = range(1, random.randint(1,30))
b = range(1, random.randint(10,40))
```

Exercice 7

Demander à l'utilisateur une chaîne de caractères et indiquez-lui si cette chaîne est un palindrome ou non. (Un palindrome est une chaîne qui se lit de la même façon à l'endroit et à l'envers).

Exercice 8

Soit la liste enregistrée dans une variable : a = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]. Donner une ligne de Python qui prend cette liste a et qui en fait une nouvelle liste qui ne contient que les éléments pairs de cette liste.

SÉRIE 2

Exercice 1

Générer un nombre aléatoire entre 1 et 9 (y compris 1 et 9). Demander à l'utilisateur de deviner le nombre, puis dites-lui s'il a deviné trop bas, trop haut ou exactement juste :
Continuer le jeu jusqu'à ce que l'utilisateur tape "exit".
Afficher à la fin du jeu le nombre de devinettes .

Exercice 2

Demander à l'utilisateur un nombre et déterminer si le nombre est premier ou non.

Exercice 3

Écrire un programme qui prend une liste de nombres (par exemple, a = [5, 10, 15, 20, 25]) et qui fait une nouvelle liste des premiers et derniers éléments de la liste donnée. Pour vous entraîner, écrire ce code à l'intérieur d'une fonction.

Exercice 4

Écrire un programme qui demande à l'utilisateur combien de nombres de Fibonacci il doit générer et qui les génère ensuite. (Rappel : la séquence de Fibonacci est une séquence de nombres où le nombre suivant dans la séquence est la somme des deux nombres précédents dans la séquence.).

Exercice 5

Écrivez un programme (fonction) qui prend une liste et renvoie une nouvelle liste qui contient tous les éléments de la première liste sans les doublons.

Exercice 6

Écrire un programme qui demande à l'utilisateur la longueur et la largeur d'un rectangle et calcule l'aire.
On peut écrire une fonction qui attend la longueur et la largeur comme arguments et retourne l'aire du rectangle.

Exercice 7

Écrire un programme qui calcule le reste de la division de deux entiers demandés à l'utilisateur.
On peut écrire une fonction qui attend le dividende et le diviseur comme arguments et retourne le reste de la division entière.

Exercice 8

Écrire un programme qui calcule le périmètre d'un cercle après avoir demandé son rayon.
On peut écrire une fonction qui attend comme argument le rayon et retourne le périmètre du cercle.

SÉRIE 3

Exercice 1

1. Écrire un programme Python qui accepte une séquence de nombres séparés par des virgules (ou autre séparateur) et générer une liste et un tuple avec ces nombres.
2. Écrire un programme Python permettant d'extraire et d'afficher l'extension d'un fichier.

Exercice 2

Écrire un programme Python pour calculer le nombre de jours entre deux dates.

Exercice 3

Écrire un programme qui peut calculer la factorielle d'un nombre donné.

Exercice 4

Écrire un programme qui demande deux chiffres à l'utilisateur et affiche le plus grand.

Exercice 5

Écrire un programme qui calcule la racine carré d'un nombre demandé à l'utilisateur.

Exercice 6

Écrire un programme qui calcule les racines d'un polynôme du second degré.

Exercice 7

Soit la chaîne de caractères p = "L'enseignement devrait être ainsi : celui qui le reçoit le recueille comme un don inestimable mais jamais comme une contrainte pénible."

- Créez le dictionnaire des fréquences de cette chaîne
  (avec l'association clef = caractère, valeur = nombre d'apparitions du caractère dans la chaîne) (Utilisez dict, str.count, set).
  Créez la liste correspondante à ce dictionnaire (Utilisez les list en compréhension).
- Triez cette liste par ordre décroissant de fréquences (Utilisez list.sort).

Exercice 8

Un nombre est dit de Armstrong s'il est égal à la somme des cubes de ses chiffres (par exemple, 371 = 33 + 73 + 13).
Générez la liste des nombres de Armstrong inférieurs à 1000.

SÉRIE 4

Exercice 1

Suite de Conway : Le premier terme de la suite de Conway est posé comme égal à 1. Chaque terme de la suite se construit en annonçant le terme précédent, c'est-à-dire en indiquant combien de fois chacun de ses chiffres se répète :
u_0=1 ce terme comporte simplement un « 1 ». Par conséquent, le terme suivant est u_1 =11
Celui-ci est composé de deux « 1 » : u_2=21 . En poursuivant le procédé :

u_3=1211
u_4=111221
u_5=312211
u_6=13112221
Et ainsi de suite.

Quel est la valeur de u_{26} ?

Exercice 2

Soit le dictionnaire suivant :

```
inventaire = {
   or : 500,
   poche" : ["silex", "ficelle", "pierre précieuse"],
   "sac à dos" : ["xylophone", "poignard", "rouleau de lit", "pain"]
}
```

Écrire un programme qui réalise les taches suivantes :

- Ajoutez une clé à l'inventaire appelée "poche".
- Définissez la valeur de "poche" comme étant une liste composée des chaînes "coquillage", "baie étrange" et "peluche".
- Trier() les articles de la liste stockés sous la clé "sac à dos".
- Ensuite, supprimer "poignard" de la liste des articles stockés sous la clé "sac à dos".
- Ajoutez 50 au nombre stocké sous la clé "or".

Exercice 3

1. Écrire une fonction créer_tuple() qui accepte un nombre variable d'arguments et les affiche tous.
2. Écrire une fonction somme_tous() pour accepter un nombre variable d'arguments et afficher la somme de tous les éléments présents dans celle-ci.
3. Écrire une fonction Opération() pour accepter deux arguments et réaliser la somme, le différence, le produit et la division entre ces deux arguments.

Exercice 4

La fonction zip() est une fonction intégrée en Python. Elle prend des éléments en séquence à partir d'un certain nombre de collections pour établir une liste de tuples, où chaque tuple contient un élément de chacune des collections. Cette fonction est souvent utilisée pour regrouper les éléments d'une liste qui a le même index.

```
>>> A1=[1,2,3]
>>> A2="XYZ"
>>> list(zip(A1,A2))
[(1, 'X'), (2, 'Y'), (3, 'Z')]

>>> L1=['Laptop', 'Desktop', 'Mobile']
>>> L2=[40000, 30000, 15000]
>>> L3=tuple((list(zip(L1,L2))))
>>> L3
(('Laptop', 40000), ('Desktop', 30000), ('Mobile', 15000))
```

Considérons deux listes, à savoir les listes L1 et L2.
Ici, L1 contient une liste de couleurs et L2 contient leur code couleur comme L1=["Noir", "Blanc", "Gris"] L2=[255,0,100]
Affichez le contenu comme suit :

("Noir",255)
("blanc",0)
("Gris",100)

Exercice 5

L'opérateur * est utilisé dans la fonction zip(). L'opérateur * décompresse une séquence en position arguments. Un exemple simple de l'opérateur * sur les arguments de position est donné ci-dessous.

```
>>> X=[("HP",50000),("DELL",30000)]
>>> Marque,Prix=zip(*X)
>>> print(Marque)
('HP', 'DELL')
>>> print(Prix)
(50000, 30000)
```

→ Écrire un programme permettant de calculer la transposée d'une matrice (Utiliser la fonction zip et les tuples).

$$M=\begin{pmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{pmatrix} \quad \rightarrow \quad T=\begin{pmatrix}1 & 3 & 5\\ 2 & 4 & 6\end{pmatrix}$$

Exercice 6

Créer un dictionnaire de quatre manières différentes.
Remarque : L'opérateur % est utilisé pour substituer des valeurs d'un dictionnaire, dans une chaîne, par un nom.

Exercice 7

1. Écrire un programme permettant d'attribuer des notes aux étudiants et afficher toutes les notes en utilisant les méthodes keys() et get() du dictionnaire.
2. Écrire une fonction permettant de compter la fréquence des caractères en utilisant la méthode get() du dictionnaire.

Exercice 8

Écrire un programme permettant de passer une liste à une fonction, calculer le nombre total de nombres positifs et négatifs de la liste. Enfin, afficher le résultat dans un dictionnaire.

CORRECTIONS

SÉRIE 1 - Correction

Exercice 1

Écrire un programme Python permettant d'obtenir la version de Python que vous utilisez.

```
>>> import sys
>>> print("Python version")
Python version
>>> print (sys.version)
3.7.6 (default, Jan  8 2020, 19:59:22)
[GCC 7.3.0]
```

Exercice 2

Demander un entier à l'utilisateur. Selon que le nombre est pair ou impair, afficher un message approprié à l'utilisateur. (Indication : comment un nombre pair/impair réagit-il différemment lorsqu'il est divisé par 2 ?)

```
num = int(input("Entrer un chiffre "))
mod = num % 2
if mod != 0:
    print("Vous avez choisi un nombre impair.")
else:
    print("Vous avez choisi un nombre pair.")
```

Exercice 3

Créer un programme qui demande à l'utilisateur d'entrer son nom et son âge. Afficher un message qui lui est adressé indiquant l'année où il aura 100 ans.

```
name = input("Quel est votre nom ?: ")
age = int(input("Quel âge avez-vous ? "))
year = str((2020 - age)+100)
print(name + " aura 100 ans en " + year)
```

Exercice 4

Soit la liste suivante : a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89] . écrire un programme qui affiche tous les éléments de la liste qui sont inférieurs à 15.

```
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
print([num for num in a if num < 15])
```

Exercice 5

Créez un programme qui demande à l'utilisateur un nombre et qui affiche ensuite une liste de tous les diviseurs de ce nombre.

```
num = int(input("Veuillez choisir un nombre :"))
listRange = list(range(1,num+1))
ListDivisor = []
for number in listRange:
    if num % number == 0:
        ListDivisor.append(number)

print(ListDivisor)
```

Ou

```
n=int(input("donner un chiffre : "))
l=list(range(1,n+1))
print([i for i in l if n%i==0])
```

Exercice 6

Prenez deux listes, disons par exemple ces deux-là :

a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]

b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]

et écrire un programme qui renvoie une liste qui ne contient que les éléments communs entre les listes (sans doublons). Assurez-vous que votre programme fonctionne sur deux listes de tailles différentes.

Solution 1

```
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
c=[]
for i in a:
    if i in b:
        if i not in c:
            c.append(i)
c.sort()
print (c)
```

Solution 2

```
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
c = []
def overlap(v1,v2,v3):
    for i in v1:
        if i in v2:
            if i not in v3:
                val3.append(i)
overlap(a,b,c)
c.sort()
print(c)
```

Solution 3 en utilisant SET

```
a = {1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89}
b = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}
c=[]
for i in a:
    if i in b:
        c.append(i)
c.sort()
```

```
print (c)
```

Remarque :
On peut créer un liste aleatoire comme suit :

```
import random
a = range(1, random.randint(1,30))
b = range(1, random.randint(10,40))
```

Exercice 7

Demander à l'utilisateur une chaîne de caractères et indiquez-lui si cette chaîne est un palindrome ou non. (Un palindrome est une chaîne qui se lit de la même façon à l'endroit et à l'envers).

```
wrd=str(input("Veuillez saisir un mot : "))
rvs=wrd[::-1]
print(rvs)
if wrd == rvs:
   print("Ce mot est un palindrome")
else:
   print("Ce mot n’est pas un palindrome")
```

```
def reverse(word):
        x = ''
        for i in range(len(word)):
                x += word[len(word)-1-i]
        return x

word = input('give me a word:\n')
x = reverse(word)
if x == word:
        print('Ce mot est un palindrome')
else:
        print('Ce mot n’est pas un palindrome')
```

Exercice 8

Soit la liste enregistrée dans une variable : a = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]. Donner une ligne de Python qui prend cette liste a et qui en fait une nouvelle liste qui ne contient que les éléments pairs de cette liste.

```
a = [e for e in a if e % 2 == 0]
```

SÉRIE 2 - Correction

Exercice 1

Générer un nombre aléatoire entre 1 et 9 (y compris 1 et 9). Demander à l'utilisateur de deviner le nombre, puis dites-lui s'il a deviné trop bas, trop haut ou exactement juste :
Continuer le jeu jusqu'à ce que l'utilisateur tape "exit".
Afficher à la fin du jeu le nombre de devinettes .

```
import random

number = random.randint(1,9)
d = 0
count = 0

while d != number and d != "exit":
    d = input("Quelle est votre supposition?")

    if d == "exit":
        break

    d = int(d)
    count += 1

    if d < number:
        print("Trop faible!")
    elif d > number:
        print("Trop élevé!")
    else:
        print("vous l'avez trouvé")
        print("Et il ne vous a fallu que ",count,"essais!")
```

Exercice 2

Demander à l'utilisateur un nombre et déterminer si le nombre est premier ou non.

```
e = int(input('Donner un nombre : '))
a = [x for x in range(2, e) if e % x == 0]

def is_prime(n):
        if e > 1:
                if len(a) == 0:
                        print(e ,"est un nombre premier")
                else:
                        print(e, "n'est pas premier")
        else:
                print(e," n'est pas premier")

is_prime(e)
```

Exercice 3

Écriver un programme qui prend une liste de nombres (par exemple, a = [5, 10, 15, 20, 25]) et qui fait une nouvelle liste des premiers et derniers éléments de la liste donnée. Pour vous entraîner, écriver ce code à l'intérieur d'une fonction.

```
a = [5, 10, 15, 20, 25]
def list_ends(a_list):
    return [a_list[0], a_list[len(a_list)-1]]
print(a)
print(list_ends(a))
```

Exercice 4

Écrire un programme qui demande à l'utilisateur combien de nombres de Fibonacci il doit générer et qui les génère ensuite. (Rappel : la séquence de Fibonacci est une séquence de nombres où le nombre suivant dans la séquence est la somme des deux nombres précédents dans la séquence.).

```
def fibonacci():
    num = int(input("How many numbers that generates?:"))
    i = 1
    if num == 0:
        fib = []
    elif num == 1:
        fib = [1]
    elif num == 2:
        fib = [1,1]
    elif num > 2:
        fib = [1,1]
        while i < (num - 1):
            fib.append(fib[i] + fib[i-1])
            i += 1
    return fib
print(fibonacci())
```

Exercice 5

Écrivez un programme (fonction) qui prend une liste et renvoie une nouvelle liste qui contient tous les éléments de la première liste sans les doublons.

```
def dedupe_v1(x):
  y = []
  for i in x:
    if i not in y:
      y.append(i)
  return y

#this one uses sets
def dedupe_v2(x):
    return list(set(x))
```

```
a = [10,2,3,4,3,2,1]
print(a)
print(dedupe_v1(a))
print(dedupe_v2(a))
```

Exercice 6

Écrire un programme qui demande à l'utilisateur la longueur et la largeur d'un rectangle et calcule l'aire.

```
largeur = float(input("largeur = "))
print("largeur = ", largeur)

longueur = float(input("largeur = "))
print("longueur = ", longueur)

# affiche le résultats
print("Aire du rectangle=",largeur*longueur)
```

On peut écrire une fonction qui attend la longueur et la largeur comme arguments et retourne l'aire du rectangle.

```
def aire_rectangle(largeur, longueur):
    """ Calcul de l'aire d'un rectangle """
    return largeur * longueur
largeur = float(input("largeur = "))
print("largeur = ", largeur)
longueur = float(input("largeur = "))
print("longueur = ", longueur)
# arriche le résultats
print("Aire du rectangle = ", aire_rectangle(largeur, longueur))
```

Exercice 7

Écrire un programme qui calcule le reste de la division de deux entiers demandés à l'utilisateur.

```
dividende = int(input("entrer le dividende : "))
print("dividende = ", dividende)
diviseur = int(input("entre le diviseur : "))
print("diviseur = ", diviseur)
reste = dividende % diviseur
print("reste = ", reste)
```

On peut écrire une fonction qui attend le dividende et le diviseur comme arguments et retourne le reste de la division entière.

```
def reste(dividende, diviseur):
    """ Calcul du reste de la division de deux entiers """
    reste = dividende % diviseur
    print("reste = ", reste)
```

Exercice 8

Écrire un programme qui calcule le périmètre d'un cercle après avoir demandé son rayon.

```
from math import pi
```

```
rayon = float(input("entrer le rayon : "))
print("rayon = ", rayon)
# affichage du résultat
print("périmètre = ", 2 * pi * rayon)
```

On peut écrire une fonction qui attend comme argument le rayon et retourne le périmètre du cercle.

```
def perimetre(rayon):
    """ Calcul du périmètre d'un cercle """
    return 2 * pi * rayon
```

SÉRIE 3 - Correction

Exercice 1

1. Ecrire un programme Python qui accepte une séquence de nombres séparés par des virgules (ou autre séparateur) et générer une liste et un tuple avec ces nombres.

```
>>> values = input("Saisissez quelques nombres séparés par des virgules: ")
Saisissez quelques nombres séparés par des virgules: 1,2,3,4,5
>>> l = values.split(",")
>>> t = tuple(l)
>>> print('List : ',l)
List :  ['1', '2', '3', '4', '5']
>>> print('Tuple : ',t)
Tuple :  ('1', '2', '3', '4', '5')
```

2. Ecrire un programme Python permettant d'extraire et d'afficher l'extension d'un fichier.

```
>>> filename = input("Saisissez le nom du fichier : ")
Saisissez le nom du fichier : python3.pdf
>>> print ("L'extension du fichier est : " + repr(f_ext[-1]))
L'extension du fichier est : 'pdf'
```

Exercice 2

Écrivez un programme Python pour calculer le nombre de jours entre deux dates.

```
>>> from datetime import *
>>> l_date = date(2020, 8, 17)
>>> f_date = date(2010, 10, 22)
>>> delta = l_date - f_date
>>> print(delta.days)
3587
```

Exercice 3

Écrire un programme qui peut calculer la factorielle d'un nombre donné.

```
def fact(x):
   if x == 0:
      return 1
   return x * fact(x - 1)
x=int(input("Donner un chiffre : "))
print("Le factoriel de {} est {} : ".format(x,fact(x)))
```

Exercice 4

Écrire un programme qui demande deux chiffres à l'utilisateur et affiche le plus grand.

```
x = float(input("entrer x : "))
print("x = ", x)

y = float(input("entrer y : "))
print("y = ", y)

# test entre x et y
if x > y:
    print("x est plus grand")
    print("le plus grand = ", x)
elif y > x:
    print("y est plus grand")
    print("le plus grand = ", y)
else:
    print("x et y sont égaux")
    print("x = ", x, "\t y = ", y)
```

Exercice 5

Écrire un programme qui calcule la racine carré d'un nombre demandé à l'utilisateur.

```
from math import sqrt

def racine(x):
    """ Calcul de la racine carré de x si x est positif. """

    if x >= 0:
        print("RACINE(x) = ", sqrt(x))
    else:
        print("x est négatif")

if __name__ == "__main__":
    # lecture de x
    x = float(input("entrer x : "))
    print("x = ", x)

    racine(x)
```

Exercice 6

Écrire un programme qui calcule les racines d'un polynôme du second degré.

```
from math import sqrt

def racine_trinome(a, b, c):
    """ Calcul des racines réelles d'un polynome de degré 2 """

    # Calcul du discriminent
    delta = b**2 - 4. * a * c
    print("delta = ", delta)

    # Test du discriminent
    if delta > 0:
        print("L'équation a deux solutions")
        print("x1 = ", (-b - sqrt(delta)) / (2. * a))
        print("x2 = ", (-b + sqrt(delta)) / (2. * a))
    elif delta < 0.:
        print("L'équation a deux solutions complexes")
    else:
        print("L'équation a une seule solution")
        print("x  = ", -b / (2.0 * a))

if __name__ == "__main__":
    print("On va résoudre l'équation a*x^2 + b*x + c = 0")

    # lecture des variables
    a = float(input("entrer a : "))
    print("a = ", a)

    b = float(input("entrer b : "))
    print("b = ", b)

    c = float(input("entrer c : "))
    print("c = ", c)

    racine_trinome(a, b, c)
```

Exercice 7

Soit la chaîne de caractères p = “L'enseignement devrait être ainsi : celui qui le reçoit le recueille comme un don inestimable mais jamais comme une contrainte pénible.”

- Créez le dictionnaire des fréquences de cette chaîne
  (avec l’association clef = caractère, valeur = nombre d’apparitions du caractère dans la chaîne) (Utilisez dict, str.count, set).
  Créez la liste correspondante à ce dictionnaire (Utilisez les list en compréhension).
- Triez cette liste par ordre décroissant de fréquences (Utilisez list.sort).

```
>>> p = "L'enseignement devrait être ainsi : celui qui le reçoit le recueille comme un don inestimable mais jamais comme une contrainte pénible."
>>> freq = dict((x,p.count(x)) for x in set(p))
>>> l= list((x,freq[x]) for x in freq)
>>> def fr(t) :
>>>     return t[1]
>>> l.sort(key=fr, reverse = True)
>>> print(l)
[(' ', 8), ('e', 5), ('u', 5), ('i', 3), ('o', 2), ('a', 1), ('c', 1), ('b', 1), ('d', 1),('g', 1), ('f', 1), ('h', 1), ('k', 1), ('j', 1), ('m', 1), ('l', 1), ('n', 1), ('q', 1),('p', 1), ('s', 1), ('r', 1), ('t', 1), ('w', 1), ('v', 1), ('y', 1), ('x', 1), ('z', 1)]
```

Exercice 8

Un nombre est dit de Armstrong s'il est égal à la somme des cubes de ses chiffres (par exemple, 371 = 33 + 73 + 13).
Générez la liste des nombres de Armstrong inférieurs à 1000.

```
>>> def armstrong() :
>>>    r = [] # au départ, la liste résultat est vide
>>>    for x in range(1000) : # pour chaque nombre <1000
>>>       s = str(x) # créez la chaîne correspondante à ce nombre
>>>       t = 0 # init calcul somme des cubes des chiffres
>>>       for c in s : # pour chaque chiffre de la chaîne
>>>          t += int(c)**3 # ajouter à t le cube de ce chiffre
>>>       if t == x : # si la somme des cubes est égale au nombre
>>>          r += [x] # l'ajouter à la liste
>>>     return r # retourner la liste résultat

>>> armstrong()
```

SÉRIE 4 - Correction

Exercice 1

Suite de Conway : Le premier terme de la suite de Conway est posé comme égal à 1. Chaque terme de la suite se construit en annonçant le terme précédent, c'est-à-dire en indiquant combien de fois chacun de ses chiffres se répète :
u_0=1 ce terme comporte simplement un « 1 ». Par conséquent, le terme suivant est u_1 =11
Celui-ci est composé de deux « 1 » : u_2=21 . En poursuivant le procédé :

u_3=1211
u_4=111221
u_5=312211
u_6=13112221
Et ainsi de suite.

Quel est la valeur de u_{26} ?

```
def prochainConway(s):
   c=s[0]
```

```
    n=1
    r=""
    for i in s[1:] :
        if i==c :
            n+=1
        else :
            r=r+str(n)+str(c)
            c=i
            n=1
    r=r+str(n)+str(c)
    return r
u='1'
for i in range(1,27) :
    u = prochainConway(u)
    print(i,"\n")
    print(u,"\n")
else :
    print(len(u))
```

U_{26} :

```
3113112221131112311332111213122112311311123112111331121113122112132113121113222112311311221112131221123113112221121113311 2
1113112221121113122113121113222112132113213221232112111312111213322112311311222113111221221113122112132113121113222112311 3
1122211311123113322112111331121113112221121113122113111231133221121113122113121113222123211211131211121332211213211321322 1
1331121321132213211231132132211211131221232112111312212221121123222112311311222113111231133211121312211231131112311211133 1
1211131221121321133112132113212221221113122113121113222123211211131211121332211213211321322113311213211322132112311321322 1
1211131221232112111312212221121123222112132113213221133112132123123112111311222112132113311213211231232112311311222112111 3
1221131112311332211213211321223112111311222112132113212221132221222112112322211231131122211311123113321112131221123113111 2
3112111331121113122112132113121113222112311311222113111231133221132231131122211311123113322112111312211312111322111213122 1
1231131112311211232221121321132132211331221122311311222112111312211311123113322112132113213221133122211332111213112221133 2
1132211221121332211211131221131211132221232112111312111213111213211231132132211211131221232112111312211213111213122112132 1
1321322112311311222113311213212322211211131221131211221321123113213221121113122113121113222112131112131221121321131211132 2
2112132113213221133112132123222112311311222113111231132231121113112221121321133112132112211213322112111312211312111322212 3
1122212213211321322112311311222113311213212322211211131221131211132221232112111312111213322112131112131221121321131211132 2
2112132113213221232112111312111213322112132113213221133112132123123112111311222112132113311213211221121332211231131122211 3
1112311332111213122112311311123112111331121113122112132113121113222112311311222113111221221113122112132113121113222112132 1
1321322113312221133211121332211213211321322113223113112221131112311332211211131221131211132221232112111312212321123113112 21

1322112311322111312211213211321321112133221231132211321222 1
```

Exercice 2

Soit le dictionnaire suivant :

inventaire = {
    or : 500,
    poche" : ["silex", "ficelle", "pierre précieuse"],
    "sac à dos" : ["xylophone", "poignard", "rouleau de lit", "pain"]
}

Écrire un programme qui réalise les taches suivantes :

- Ajoutez une clé à l'inventaire appelée "poche".
- Définissez la valeur de "poche" comme étant une liste composée des chaînes "coquillage", "baie étrange" et "peluche".
- Trier() les articles de la liste stockés sous la clé "sac à dos".
- Ensuite, supprimer "poignard" de la liste des articles stockés sous la clé "sac à dos".
- Ajoutez 50 au nombre stocké sous la clé "or".

```
>>> inventaire={'or' : 500, 'poche' : ['silex', 'ficelle', 'pierre précieuse'],'sac à dos' : ['xylophone', 'poignard', 'rouleau de lit', 'pain']}
```

```
>>> inventaire['poche']=['coquillage','baie étrange','peluche']
>>> inventaire['sac à dos'].sort()
>>> inventaire['sac à dos'].remove('poignard')
>>> inventaire['or']=inventaire['or']+50
>>> print(inventaire)
{'or': 550, 'poche': ['coquillage', 'baie étrange', 'peluche'], 'sac à dos': ['pain', 'rouleau de lit', 'xylophone']}
```

Exercice 3

1. Écrire une fonction creer_tuple() qui accepte un nombre variable d'arguments et les affiche tous.
2. Écrire une fonction somme_tous() pour accepter un nombre variable d'arguments et afficher la somme de tous les éléments présents dans celle-ci.

```
>>> def creer_tuple(*args):
...     print(args)
...
>>> creer_tuple(1,2,3,4)
(1, 2, 3, 4)
>>> creer_tuple('a','b')
('a', 'b')

>>> def somme_tous(*args):
...     t=()
...     s=0
...     for i in args:
...             s=s+i
...     print(s)
...
>>> somme_tous(10,20,30,40)
100
>>>



>>> def Operations(a,b):
...     sum1=a+b
...     diff=a-b
...     prod=a*b
...     div=a/b
...     return sum1,diff,prod,div
...
>>> s=Operations(20,10)
>>> print(s)
(30, 10, 200, 2.0)
>>> print(type(s))
<class 'tuple'>
```

Exercice 4

Considérons deux listes, à savoir les listes L1 et L2.
Ici, L1 contient une liste de couleurs et L2 contient leur code couleur comme L1=["Noir", "Blanc", "Gris"] L2=[255,0,100]
Affichez le contenu comme suit :

("Noir",255)
("blanc",0)
("Gris",100)

```
>>> L1=["Noir", "Blanc", "Gris"]
>>> L2=[255,0,100]
>>> for couleur, code in zip(L1,L2):
...     print((Couleur, code))
...
('Noir', 255)
('Blanc', 0)
('Gris', 100)
```

Exercice 5

→ Écrire un programme permettant de calculer la transposée d'une matrice (Utiliser la fonction zip et les tuples).

$$M=\begin{pmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{pmatrix} \quad \rightarrow \quad T=\begin{pmatrix}1 & 3 & 5\\ 2 & 4 & 6\end{pmatrix}$$

```
>>> Matrice=[(1,2),(3,4),(5,6)]
>>> x=zip(*Matrice)
>>> T=tuple(x)
>>> T
((1, 3, 5), (2, 4, 6))
```

Exercice 6

Créer un dictionnaire de quatre manières différentes.

```
>>> D1={'Prénom':'Mohamed','Age':22}
{'Prénom': 'Mohamed', 'Age': 22}

>>> D2={}
>>> D2['Prénom']='Mohamed'
>>> D2['Age']=22
>>> D2
{'Prénom': 'Mohamed', 'Age': 40}

>>> D3=dict(Prénom='Mohamed',Age=22)
>>> D3
```

```
{'Prénom': 'Mohamed', 'Age': 22}

>>> D4=dict([('Prénom','Mohamed'),('Age',22)])
>>> D4
{'Prénom': 'Mohamed', 'age': 40}

>>> P="Mon prénom est %(Prénom)s et j'ai %(Age)d ans"%D4
>>> print(P)
Mon prénom est Mohamed et j'ai 22 ans
>>>
```

Dans l'exemple ci-dessus, nous avons créé des dictionnaires de quatre manières différentes.

- Nous pouvons choisir la première façon si nous connaissons à l'avance tout le contenu d'un dictionnaire. Nous pouvons utiliser la deuxième si nous devons ajouter un champ à la fois.
- La troisième façon nécessite toutes les clés de la chaîne.
- La quatrième méthode est bonne si nous voulons construire les clés et les valeurs au moment de l'exécution.

Exercice 7

1. Écrire un programme permettant d'attribuer des notes aux étudiants et afficher toutes les notes en utilisant les méthodes keys() et get() du dictionnaire.

```
Notes={"Zaid":10,"Mohamed":14,"Ali":12}

>>> for key in Notes.keys():
...     print(key,",Notes.get(key,0))
...
Zaid  10
Mohamed  14
Ali  12
```

2. Écrire une fonction permettant de compter la fréquence des caractères en utilisant la méthode get() du dictionnaire.

```
>>> def Occurence(S):
...     D=dict()
...     for C in S:
...             if C not in D:
...                     D[C]=1
...             else:
...                     D[C]=D.get(C,0)+1
...     return D
...
>>> H=Occurence("FSTT DE TANGER")
>>> print(H)
{'F': 1, 'S': 1, 'T': 3, ' ': 2, 'D': 1, 'E': 2, 'A': 1, 'N': 1, 'G': 1, 'R': 1}
```

Exercice 8

Écrire un programme permettant de passer une liste à une fonction, calculer le nombre total de nombres positifs et négatifs de la liste. Enfin, afficher le résultat dans un dictionnaire.

Exemple :

Entrée: L=[1,-2,-3,4]

Sortie: {‘Négatif’: 2, ‘Positif’: 2}

```
>>> def PosNeg(L):
...     D={}
...     D["Pos"]=0
...     D["Neg"]=0
...     for x in L:
...             if x>0:
...                     D["Pos"]+=1
...             else:
...                     D["Neg"]+=1
...     print(D)

>>> L=[1,-2,-3,4,5,7,-4,-9]
>>> PosNeg(L)
{'Pos': 4, 'Neg': 4}
>>> L=[1,-2,-3,4,5,7,-4,-9,-9,-8]
>>> PosNeg(L)
{'Pos': 4, 'Neg': 6}
```