Les listes en Python

Objectifs : savoir écrire des programmes Python comme dans le cours

I. Exemple 1

1°) Liste des carrés des 10 premiers entiers naturels

• 1ère façon :

```
L = [ ] #liste vide
for i in range(10):
    L.append(i**2)
print(L)
```

On peut aussi utiliser l'instruction `L+[i**2]`.

Attention à la place du `print(L)`.

Si l'on met le `print` à l'intérieur de la boucle, on obtient les listes successives ce qui permet d'ailleurs de comprendre ce qui se passe avec la fonction `append`.

```
L = [ ]
for i in range(10):
    L.append(i**2)
    print(L)
```

Si l'on met le `print` à l'extérieur de la boucle, on a seulement la liste finale.

• 2e façon :

On peut en utilisant une liste par compréhension.

```
L=[i**2 for i in range(10)]
print(L)
```

L'écriture peut se lire comme la liste L dont les éléments sont de la forme i^2 avec $i \in [\![0\,;9]\!]$ (intervalle d'entiers de 0 à 9).

On peut aisément faire le lien avec l'écriture des ensembles en compréhension. L'ensemble des carrés des dix premiers entiers naturels s'écrit $E = \left\{i^2, i \in [\![0\,;9]\!]\right\}$.

2°) Liste des carrés des *n* premiers entiers naturels

n est un entier naturel supérieur ou égal à 1.

```
def carrés(n):
    L=[i**2 for i in range(n)]
    return L
```

II. Exemple 2

1°) Liste des carrés des entiers pairs de 0 à 30

```
L = [ ]
for i in range(16):
    L.append((2*i)**2)
print(L)
```

```
L=[(2*i)**2 for i in range(16)]
print(L)
```

```
L=[i**2 for i in range(31) if i%2==0]
print(L)
```

On peut écrire une condition. On notera que l'on utilise == pour le test, le signe = étant réservé à l'affectation.

`i%2` désigne le reste de la division euclidienne de i par 2.

2°) Liste des carrés des entiers pairs de 0 à *n*

```
def carrés(n):
    L=[i**2 for i in range(n+1) if i%2==0]
    return L
```

III. Quelques fonctions à connaître sur les listes

max(L) renvoie la plus grande valeur de la liste.

min(L) renvoie la plus petite valeur de la liste.

len(L) renvoie la taille (c'est-à-dire le nombre d'éléments) de la liste (len abréviation de length).

sum(L) renvoie la somme des éléments de la liste.

remove : enlever

append : ajouter

Exercices

1 1°) Écrire un programme Python qui permet d'obtenir la liste des cubes des 10 premiers entiers naturels.
2°) Écrire une fonction Python qui prend pour argument un entier naturel quelconque *n* supérieur ou égal à 1 et qui renvoie la liste de des cubes des *n* premiers entiers naturels.

2 1°) Écrire un programme Python qui permet d'obtenir la liste de tous les entiers naturels impairs inférieurs ou égaux à 10.
2°) Écrire une fonction Python qui prend pour argument un entier naturel quelconque *n* et qui renvoie la liste de tous les entiers naturels impairs inférieurs ou égaux à *n*.

3 Pour tout entier naturel *n*, on pose $u_n = \sum_{k=0}^{k=n} (-1)^k k$.

1°) Écrire un programme Python qui affiche la liste des 30 premiers termes de la suite.
2°) Écrire une fonction Python qui prend pour argument un entier naturel quelconque *n* et qui renvoie la liste de tous les valeurs de u_k pour *k* inférieurs ou égaux à *n*.

4 Écrire une fonction Python qui prend pour argument une liste L de réels quelconques et qui renvoie la liste des carrés des éléments de L.

5 Écrire une fonction Python qui prend pour argument une liste L d'entiers relatifs et qui renvoie la liste des nombres pairs de L.

6 1°) Écrire une fonction Python qui prend pour argument une liste L d'entiers relatifs et qui renvoie la liste des restes de la division euclidienne par 3 de tous les éléments de L.
2°) Écrire une fonction Python qui prend pour arguments une liste L d'entiers relatifs ainsi qu'un entier naturel non nul *a* et qui renvoie la liste des restes de la division euclidienne par *a* de tous les éléments de L.

On peut faire pareil avec les quotients.

7 Pour tout entier naturel *n*, on pose $S_n = \sum_{k=0}^{k=n} \frac{1}{k!}$.

Écrire une fonction Python qui prend pour argument un entier naturel quelconque *n* et qui affiche la valeur de S_n.
On pourra utiliser directement la fonction prédéfinie fact à importer de la bibliothèque math (donc commencer par « from math import fact »).

8 Écrire une fonction Python qui prend pour arguments une liste L et un réel *a* et qui renvoie la liste obtenue en ajoutant *a* à tous les éléments de L.

On peut faire pareil en multipliant tous les éléments de L par *a*.

9 Écrire une fonction Python qui prend pour arguments deux entiers relatifs *a* et *b* tels que $a \leqslant b$ et un entier naturel *n* et qui renvoie la liste des multiples de *n* compris entre *a* et *b* au sens large.

10 Écrire une fonction Python qui prend pour argument une liste L d'entiers relatifs et qui renvoie la liste des couples d'éléments de L premiers entre eux.
On pourra utiliser la fonction gcd de la bibliothèque math qui donne le PGCD de deux entiers naturels.
On commencera donc le programme par `from math import gcd`.

11 Écrire une fonction Python qui prend pour arguments deux listes L1 et L2 de réels et qui renvoie la liste des couples $(x\,;y)$ avec *x* dans L1 et *y* dans L2.

12 Écrire une fonction Python qui prend pour argument une liste L de réels et une fonction *f* et qui renvoie la liste des éléments des images par *f* des éléments de L.

13 Écrire une fonction Python `somme(L)` qui prend pour argument une liste L de réels et qui renvoie la somme des éléments de L (évidemment sans utiliser la commande `sum`).
Écrire une fonction Python `produit(L)` qui prend pour argument une liste L de réels et qui renvoie le produit des éléments de L.

14 Écrire une fonction Python qui prend pour argument une liste L d'entiers relatifs et qui renvoie la liste des éléments de L divisibles par 3 mais pas par 5.

15 1°) Écrire une fonction Python qui prend pour argument une liste L de réels et qui renvoie ceux qui sont entiers relatifs.
On rappelle la condition nécessaire et suffisante pour qu'un réel soit un entier relatif.

Un réel *x* est un entier relatif si et seulement si il est égal à sa partie entière.

On utilisera la fonction `floor(x)` qui renvoie la partie entière d'un réel `x`.
On commencera donc le programme par `from math import floor`.
2°) Écrire une fonction Python qui prend pour argument une liste L de réels et qui renvoie ceux qui sont entiers naturels.

16 Écrire une fonction Python qui prend pour argument une liste L de réels et qui renvoie Vrai si tous les éléments de la liste sont des entiers relatifs et Faux dans le cas contraire.

17 Écrire une fonction Python qui prend pour argument une liste L de réels de longueur supérieure ou égale à 3 et qui renvoie Vrai si les nombres de la liste sont, dans l'ordre, en progression arithmétique et Faux dans le cas contraire.

Idem pour une liste de réels non nuls dont on désire savoir s'ils sont en progression géométrique.

18 Écrire une fonction Python qui prend pour argument une liste L de réels de longueur supérieure ou égale à 2 et qui renvoie Vrai si tous les éléments de L sont deux à deux distincts et Faux dans le cas contraire.

19 Écrire une fonction Python qui prend pour argument un réel *x* strictement positif et un réel M positif ou nul et qui renvoie la liste L de multiples entiers positifs de *x* inférieurs ou égaux à M.

Solutions

2

1°)
Il y a deux façons.

1ère façon :

```
L=[2*i+1 for i in range(5)]
print(L)
```

2e façon :

```
L=[i for i in range(11) if i%2==1]
print(L)
```

2°)

Liste des entiers naturels impairs inférieurs ou égaux à un entier naturel n

Il y a deux façons :

```
def ex(n) :
    L=[2*i+1 for i in range(n+1)]
    return L
```

```
def ex(n) :
    L=[i for i in range(n+1) if i%2==1]
    return
```

```
def ex(n) :
    L=[2*i+1 for i in range(n+1)]
    return L
```

3

1ère façon :

```
u=0
L=[u]
for k in range(31):
    u=u+((-1)**k)*k
    L.append(u)
print(L)
```

```
def terme(n) :
    u=0
    for k in range(n+1):
        u=u+((-1)**k)*k
    return u

L=[terme(i) for i in range(30)]
print(L)
```

2e façon :

```
def terme(n) :
    u=sum([((-1)**k)*k for k in range(n+1)])
    return u

L=[terme(i) for i in range(30)]
print(L)
```

On peut créer une fonction.

```
def liste(n) :
    u=0
    L=[u]
    for k in range(n+1):
        u=u+((-1)**k)*k
        L.append(u)
    return L
```

4 Écrire une fonction Python qui prend pour argument une liste L de réels quelconques et qui renvoie la liste des carrés des éléments de L.

Les deux façons sont à connaître, autrement dit, il faut savoir écrire les deux programmes sans hésitations.

1ère façon :

```
def carré(L):
    M=[x**2 for x in L]
    return(M)
```

2e façon :

```
def carré(L):
    M=[0]
    for i in range(len(L)):
        M.append(L[i]**2)
    return M
```

5 Écrire une fonction Python qui prend pour argument une liste L d'entiers relatifs et qui renvoie la liste des nombres pairs de L.

```
def pairsliste(L):
    M=[x for x in L if x%2==0]
    return(M)
```

6 1°) Écrire une fonction Python qui prend pour argument une liste L d'entiers relatifs et qui renvoie la liste des restes de la division euclidienne par 3 de tous les éléments de L.

```
def restes1(L):
    M=[x%3 for x in L]
    return(M)
```

2°) Écrire une fonction Python qui prend pour argument une liste L d'entiers relatifs ainsi qu'un entier naturel non nul *a* et qui renvoie la liste des restes de la division euclidienne par *a* de tous les éléments de L.

```
def restes2(L,a):
    M=[x%a for x in L]
    return(M)
```

10

```
from math import gcd

def prem(L):
    M=[(a,b) for a in L for b in L if gcd(a,b)==1]
    return M
```

```
prem([2,3,4,6,11,23,19])
[(2, 3), (2, 11), (2, 23), (2, 19), (3, 2), (3, 4), (3, 11), (3, 23), (3, 19),
(4, 3), (4, 11), (4, 23), (4, 19), (6, 11), (6, 23), (6, 19), (11, 2), (11, 3),
(11, 4), (11, 6), (11, 23), (11, 19), (23, 2), (23, 3), (23, 4), (23, 6), (23,
11), (23, 19), (19, 2), (19, 3), (19, 4), (19, 6), (19, 11), (19, 23)]
```

11 Écrire une fonction Python qui prend pour arguments deux listes L1 et L2 de réels et qui renvoie la liste des couples $(x\,;y)$ avec *x* dans L1 et *y* dans L2.

```
def couples(L1,L2):
    L=[(a,b) for a in L1 for b in L2]
    return L
```

13

```
def somme(L):
    S=0
    for i in range(len(L)):
        S=S+L[i]
    return S
```

```
def prod(L):
    P=1
    for i in range(len(L)):
        P=P*L[i]
    return P
```

8 Écrire une fonction Python qui prend pour arguments une liste L et un réel *a* et qui renvoie la liste obtenue en ajoutant *a* à tous les éléments de L.

1ère façon :

```
def somme(L,a):
    M=[]
    for i in range(len(L)):
        M.append(L[i]+a)
    return M
```

2e façon :

```
def somme(L,a):
    M=[x+a for x in L]
    return M
```

Exemple :

L'appel de la fonction `somme([-1,3,5,7],2)` renvoie la liste `[1,5,7,9]`.

14 Démontrer

```
def exo(L):
    M=[x for x in L if x%3==0 and x%!=0]
    return M
```

Exemple :

L'appel de la fonction `exo([5,3,15,-27,6])` renvoie la liste `[3,-27,6]`.

15

1°)

```
from math import floor
def exo(L):
    M=[x for x in L if floor(x)==x]
    return M
```

2°)

```
from math import floor
def exo(L):
    M=[x for x in L if floor(x)==x and x>=0]
    return M
```

16

Écrire une fonction Python qui prend pour argument une liste L de réels et qui renvoie Vrai si tous les éléments de la liste sont des entiers relatifs et Faux dans le cas contraire.

```
from math import floor
def exo(L):
    r=True
    for i in range(len(L)):
        if floor(L[i])!=L[i]
            r=False
            return r
    return r
```

17

On utilise une variable booléenne `r`.

```
def exo(L):
    r=True
    for i in range(1,len(L)-1):
        if L[i]-L[i-1]!=L[i+1]-L[i]:
            r=False
            return r
    return r
```

Exemple :

L'appel de la fonction `exo([1,3,5])` renvoie `True`.