LISTES PAR COMPRÉHENSION

La compréhension de liste permet la création de liste de manière concise.

Syntaxe

Méthode	Effet
`L=[k**2 for k in range(1,9)]`	affecte à la liste L la liste des carrés des entiers de 1 (inclus) à 9 (exclu). soit [1, 4, 9, 16, 25, 36, 49, 64]
`doubles=[2*x for x in L]`	affecte à la liste « doubles » la liste des doubles des nombres de la liste L.
`positifs=[x for x in L if x>0]`	affecte à la liste « positifs » la liste des nombres strictement positifs de la liste L.
`multiples=[x for x in L if x%3==0]`	affecte à la liste « multiples » la liste des nombres de la liste L qui sont multiples de 3.

Exercice 1

Exécuter à la main le script ci-dessous et indiquer les résultats affichés au fur et à mesure.

```
L1=[k for k in range (5)]
print(L1)
.....
L2=[i**2 for i in range (1,6)]
print(L2)
.....
L3=[3*x for x in L1]
print(L3)
.....
L4=[x for x in L2 if x<15]
print(L4)
.....
L5=[x for x in L1 if x%2==0]
print(L5)
.....
```

Exercice 2

1. Compléter l'instruction ci-dessous pour que la liste L1 contienne la liste 1,2,3,4,5,6.

   `L1=[ for k in range( .....)]`

2. Compléter l'instruction ci-dessous pour que la liste L2 contienne la liste des nombres qui s'écrivent sous la forme $2+5k$ avec $k \in \mathbb{N}$ et $k < 100$.

   `L2=[ for k in range(. ..)]`

3. Compléter l'instruction ci-dessous pour que la liste L3 contienne la liste des multiples de 7 de 0 à 70.

   `L3=[..... for k in .]`

4. Compléter l'instruction ci-dessous pour que la liste L4 contienne la liste des multiples de 3 de la liste L2.

   `L4=[.. ..... for x in . .... if . ...]`

5. Les nombres de Mersenne sont les nombres de la forme $2^n - 1$ où n est un entier naturel n non nul.
   Compléter l'instruction ci-dessous pour que la liste L5 contienne les dix premiers nombres de Mersenne.

   `L5=..... ..... .....`

Exercice 3

La légende se situe 3 000 ans av. J.C.
Le roi Belkib (Indes) promit une récompense fabuleuse à qui lui proposerait une distraction qui le satisferait. Lorsque le sage Sissa, fils du Brahmine Dahir, lui présenta le jeu d'échecs, le souverain, demanda à Sissa ce que celui-ci souhaitait en échange de ce cadeau extraordinaire.
Sissa demanda au roi de déposer un grain de riz sur la première case, deux sur la deuxième, quatre sur la troisième, et ainsi de suite pour remplir l'échiquier en doublant la quantité de grain à chaque case. Le roi accorda immédiatement cette récompense sans se douter de ce qui allait suivre.
Son conseiller lui expliqua qu'il venait de précipiter le royaume dans la ruine car les récoltes de l'année ne suffiraient pas à payer Sissa.

On note u_n le nombre de grains de riz déposés sur la n-ième case.

1. Donner u_1, u_2 et u_3.

.....

2. Pour tout entier naturel n non nul, exprimer u_{n+1} en fonction de u_n.

.....

3. On considère la fonction `riz(n)` ci-dessous où n est un entier naturel compris entre 1 et 64. Que renvoie cette fonction ?

```
1 def riz(n):
2     u=1
3     for i in range(2,n+1):
4         u=2*u
5     return(u)
```

.....

4. Compléter la fonction `somme1()` ci-dessous qui renvoie le nombre de grains de riz total que le roi devra donner à Sissa.

```
1 def somme1():
2     S=    .
3     for i in range(1,65):
4         S=. . . . . .
5     return(S)
```

5. Pour tout entier naturel n non nul, exprimer u_n en fonction de n.

.....

6. On souhaite calculer le nombre de grains de riz total que le roi devra donner à Sissa à l'aide d'une autre méthode utilisant les listes par compréhension.

   Choisir parmi les fonctions suivantes, celle(s) qui permet(tent) de renvoyer le résultat voulu.

```
1 def somme2():
2     L=[2*u for u in range(1,65)]
3     S=sum(L)
4     return(S)
```

```
1 def somme3():
2     L=[2**(n-1) for n in range(1,65)]
3     S=sum(L)
4     return(S)
```

```
1 def somme4():
2     L=[2**k for k in range(64)]
3     S=sum(L)
4     return(S)
```

```
1 def somme5():
2     L=[    for k in range(1,65)]
3     S=sum(L)
4     return(S)
```

7. Programmer les fonctions précédentes et déterminer le nombre total de grains de riz que le roi devra donner à Sissa.

....

Exercice 4 (Paradoxe de Saint-Pétersbourg)

On s'intéresse au jeu qui consiste à lancer en l'air une pièce de monnaie équilibrée et à regarder s'il apparaît « FACE » ou « PILE » lorsqu'elle retombe.
Si « FACE » apparaît, le joueur gagne 2 euros et le jeu s'arrête, sinon le joueur relance la pièce.
Si « FACE » apparaît lors du deuxième lancer, le joueur gagne 4 euros et le jeu s'arrête, sinon le joueur relance la pièce.
Si « FACE » apparaît lors du troisième lancer, le joueur gagne 8 euros et le jeu s'arrête, sinon le joueur relance la pièce.
Et ainsi de suite ....
Si le joueur n'obtient jamais « FACE », il ne gagne rien.

1. La fonction `jeu` écrite ci-dessous en Python renvoie le gain du joueur à l'issue d'une partie de ce jeu dans laquelle on se limite à n lancers au maximum avec n un entier naturel non nul.
   Pour cela, on simule n lancers de la pièce et on cherche le rang d'apparition du premier « FACE » s'il existe. (Dans la réalité, le jeu s'arrêterait dès l'obtention du premier « FACE ».)

   • L'obtention d'un « FACE » est modélisée par la valeur 1 et l'obtention d'un « PILE » est modélisée par la valeur 0.
   • Lorsque la liste L contient au moins une fois la valeur « x », l'instruction `L.index(x)` renvoie l'indice de la première apparition de la valeur « x » dans la liste « L ».

   Compléter la fonction ci-dessous.

```
1 from random import randint
2 def jeu(n):
3     L=[randint(0,1) for k in range(n)]
4     if        in L:
5         rang=L.index(.....)
6         return(       ..)
7     else:
8         return(.........)
```

2. Écrire une fonction `moyenne` qui prend en paramètre un entier naturel n non nul, simule 10 000 parties de ce jeu (limitées à n lancers au maximum par partie) et qui renvoie le gain moyen du joueur sur les 10 000 parties. On utilisera une liste par compréhension ainsi que la fonction `jeu` de la question 1.

3. On note X_n la variable aléatoire donnant le gain du joueur si on limite chaque partie à n lancers au maximum.
   Déterminer la loi de probabilité de X_n puis calculer son espérance.

4. Programmer les fonctions précédentes. Les résultats renvoyés par la fonction `moyenne` sont-ils cohérents avec l'espérance trouvée à la question précédente ? Expliquer.

Exercice 5 :

1. Quelles sont les valeurs affichées lors de l'exécution du programme suivant ?

```
1 L=[5,7,8,12]
2 for i in range(4):
3     print(L[i]*2)
```

2. Compléter les pointillés dans le programme suivant pour que soient affichées les valeurs 25, 49, 64 et 1.

```
1 L=[5,7,8,1]
2 for i in range(.    ..):
3     print(     ..)
```

3. Quelles sont les valeurs affichées lors de l'exécution du programme suivant ?

```
1 L=[78,89,56,47,55,21]
2 for i in range(6):
3     print(L[5-i])
```

4. Quelles sont les valeurs affichées lors de l'exécution du programme suivant ? .....

```
1 L=[5,7,8,1]
2 for i in range(4):
3     print(L[i]*2**i)
```

Exercice 6

On considère la fonction `facile()` qui prend en paramètre une liste de nombres `V`.

```
1 def facile(V):
2     for i in range(len(V)):
3         V[i]=V[i]*2+i
4     return V
```

On choisit $V = [2,4,-1,0]$.

1. Donner `len(V)`
2. Compléter le tableau d'étapes ci-dessous en donnant la liste obtenue à chaque étape :

Étapes :	V
Initialisation	[2,4,-1,0]
$i = 0$	
$i = 1$	
$i = 2$	
$i = 3$	

B. En itérant sur les éléments de la liste

Exercice 7 :

Quelles sont les valeurs affichées lors de l'exécution du programme suivant ?

```
1 L=[5,7,8,12]
2 for x in L:
3     print(x*2)
```

Exercice 8

On considère la fonction `kezako()` ci-dessous :

```
1 def kezako(valeur):
2     liste=[2,5,6,6,7,2,1,2,5,6,7,8,9,4,2,2]
3     nombre=0
4     for x in liste:
5         if x==valeur:
6             nombre=nombre+1
7     return(nombre)
```

1. On choisit valeur=6. Quelle est LA valeur renvoyée par la fonction ?
2. On choisit valeur=3. Quelle est LA valeur renvoyée par la fonction ?
3. Que renvoie, d'une façon générale, la fonction kezako ?

....

Exercice 9

Écrire en Python une fonction `dernier()` qui prend en paramètre une liste d'au moins deux nombres `L` et qui renvoie la somme du premier et du dernier élément de la liste.

Exercice 10

Les fonctions suivantes prennent en paramètre une liste `L`. Indiquer celles qui renvoient le produit des éléments de la liste.

```
1 def produit1(L):
2     p=1
3     for i in range(len(L)):
4         p=p*L[i]
5     return(p)
```

```
1 def produit2(L):
2     p=1
3     for i in range(len(L)):
4         p=p*i
5     return(p)
```

```
1 def produit3(L):
2     p=1
3     for x in L:
4         p=p*x
5     return(p)
```

```
1 def produit4(L):
2     p=1
3     for i in range(L):
4         p=p*L[i]
5     return(p)
```

LISTES PAR COMPRÉHENSION

La compréhension de liste permet la création de liste de manière concise.

Syntaxe

Méthode	Effet
`L=[k**2 for k in range(1,9)]`	affecte à la liste `L` la liste des carrés des entiers de 1 (inclus) à 9 (exclu). soit [1, 4, 9, 16, 25, 36, 49, 64]
`doubles=[2*x for x in L]`	affecte à la liste « doubles » la liste des doubles des nombres de la liste `L`.
`positifs=[x for x in L if x>0]`	affecte à la liste « positifs » la liste des nombres strictement positifs de la liste `L`.
`multiples=[x for x in L if x%3==0]`	affecte à la liste « multiples » la liste des nombres de la liste `L` qui sont multiples de 3.

Exercice 1

Exécuter à la main le script ci-dessous et indiquer les résultats affichés au fur et à mesure.

```
L1=[k for k in range (5)]
print(L1)
```
k ou 0 [0;1;2;3;4]

```
L2=[i**2 for i in range (1,6)]
print(L2)
```
i^2 ou 0 [1,4,9,16,25]

```
L3=[3*x for x in L1]
print(L3)
```
3k ou 0 [0,3,6,9,12,15]

```
L4=[x for x in L2 if x<15]
print(L4)
```
[1,4,9]

```
L5=[x for x in L1 if x%2==0]
print(L5)
```
[0,2,4]

Exercice 2

1. Compléter l'instruction ci-dessous pour que la liste `L1` contienne la liste 1,2,3,4,5,6.
   `L1=[` k ou k+2 `for k in range(`1,7 ou :...6.....`)]`
2. Compléter l'instruction ci-dessous pour que la liste `L2` contienne la liste des nombres qui s'écrivent sous la forme $2+5k$ avec $k \in \mathbb{N}$ et $k < 100$.
   `L2=[`2+5*k `. for k in range(..0,.100..)]`
3. Compléter l'instruction ci-dessous pour que la liste `L3` contienne la liste des multiples de 7 de 0 à 70.
   `L3=[..... for k in .]`
4. Compléter l'instruction ci-dessous pour que la liste `L4` contienne la liste des multiples de 3 de la liste `L2`.
   `L4=[.. ..... for x in . .... if . ...`
5. Les nombres de Mersenne sont les nombres de la forme $2^n - 1$ où n est un entier naturel n non nul.
   Compléter l'instruction ci-dessous pour que la liste `L5` contienne les dix premiers nombres de Mersenne.
   `L5=.....`

Exercice 3

La légende se situe 3 000 ans av. J.C.
Le roi Belkib (Indes) promit une récompense fabuleuse à qui lui proposerait une distraction qui le satisferait. Lorsque le sage Sissa, fils du Brahmine Dahir, lui présenta le jeu d'échecs, le souverain, demanda à Sissa ce que celui-ci souhaitait en échange de ce cadeau extraordinaire.
Sissa demanda au roi de déposer un grain de riz sur la première case, deux sur la deuxième, quatre sur la troisième, et ainsi de suite pour remplir l'échiquier en doublant la quantité de grain à chaque case. Le roi accorda immédiatement cette récompense sans se douter de ce qui allait suivre.
Son conseiller lui expliqua qu'il venait de précipiter le royaume dans la ruine car les récoltes de l'année ne suffiraient pas à payer Sissa.

On note u_n le nombre de grains de riz déposés sur la n-ième case.

1. Donner u_1, u_2 et u_3.

   1, 2, 4

2. Pour tout entier naturel n non nul, exprimer u_{n+1} en fonction de u_n.

   $u_{n+1} = u_n \times 2$

3. On considère la fonction `riz(n)` ci-dessous où n est un entier naturel compris entre 1 et 64. Que renvoie cette fonction ?

```
1 def riz(n):
2     u=1
3     for i in range(2,n+1):
4         u=2*u
5     return(u)
```

   le nombre de grains de riz sur la n^e case

4. Compléter la fonction `somme1()` ci-dessous qui renvoie le nombre de grains de riz total que le roi devra donner à Sissa.

```
1 def somme1():
2     S=1..
3     for i in range(1,65):
4         S=.........
5     return(S)
```

5. Pour tout entier naturel n non nul, exprimer u_n en fonction de n.

   .....

6. On souhaite calculer le nombre de grains de riz total que le roi devra donner à Sissa à l'aide d'une autre méthode utilisant les listes par compréhension.

   Choisir parmi les fonctions suivantes, celle(s) qui permet(tent) de renvoyer le résultat voulu.

```
1 def somme2():
2     L=[2*u for u in range(1,65)]
3     S=sum(L)
4     return(S)
```

```
1 def somme3():
2     L=[2**(n-1) for n in range(1,65)]
3     S=sum(L)
4     return(S)
```

```
1 def somme4():
2     L=[2**k for k in range(64)]
3     S=sum(L)
4     return(S)
```

```
1 def somme5():
2     L=[ v_k for k in range(1,65)]
3     S=sum(L)
4     return(S)
```

7. Programmer les fonctions précédentes et déterminer le nombre total de grains de riz que le roi devra donner à Sissa.

.....

Exercice 4 (Paradoxe de Saint-Pétersbourg)

On s'intéresse au jeu qui consiste à lancer en l'air une pièce de monnaie équilibrée et à regarder s'il apparaît « FACE » ou « PILE » lorsqu'elle retombe.
Si « FACE » apparaît, le joueur gagne 2 euros et le jeu s'arrête, sinon le joueur relance la pièce.
Si « FACE » apparaît lors du deuxième lancer, le joueur gagne 4 euros et le jeu s'arrête, sinon le joueur relance la pièce.
Si « FACE » apparaît lors du troisième lancer, le joueur gagne 8 euros et le jeu s'arrête, sinon le joueur relance la pièce.
Et ainsi de suite ....
Si le joueur n'obtient jamais « FACE », il ne gagne rien.

1. La fonction `jeu` écrite ci-dessous en Python renvoie le gain du joueur à l'issue d'une partie de ce jeu dans laquelle on se limite à n lancers au maximum avec n un entier naturel non nul.
   Pour cela, on simule n lancers de la pièce et on cherche le rang d'apparition du premier « FACE » s'il existe. (Dans la réalité, le jeu s'arrêterait dès l'obtention du premier « FACE ».)

   • L'obtention d'un « FACE » est modélisée par la valeur 1 et l'obtention d'un « PILE » est modélisée par la valeur 0.
   • Lorsque la liste L contient au moins une fois la valeur « x », l'instruction `L.index(x)` renvoie l'indice de la première apparition de la valeur « x » dans la liste « L ».

   Compléter la fonction ci-dessous.

```
1 from random import randint
2 def jeu(n):
3     L=[randint(0,1) for k in range(n)]
4     if    k  in L:
5         rang=L.index(.....)
6         return(      ..)
7     else:
8         return(.....)
```

2. Écrire une fonction `moyenne` qui prend en paramètre un entier naturel n non nul, simule 10 000 parties de ce jeu (limitées à n lancers au maximum par partie) et qui renvoie le gain moyen du joueur sur les 10 000 parties. On utilisera une liste par compréhension ainsi que la fonction `jeu` de la question 1.

3. On note X_n la variable aléatoire donnant le gain du joueur si on limite chaque partie à n lancers au maximum.
   Déterminer la loi de probabilité de X_n puis calculer son espérance.

4. Programmer les fonctions précédentes. Les résultats renvoyés par la fonction `moyenne` sont-ils cohérents avec l'espérance trouvée à la question précédente ? Expliquer.

Exercice 5 :

1. Quelles sont les valeurs affichées lors de l'exécution du programme suivant ? 10, 14, 16, 24

```
1 L=[5,7,8,12]
2 for i in range(4):
3     print(L[i]*2)
```

2. Compléter les pointillés dans le programme suivant pour que soient affichées les valeurs 25, 49, 64 et 1.

```
1 L=[5,7,8,1]
2 for i in range(..4..):
3     print(L[i]**2)
```

3. Quelles sont les valeurs affichées lors de l'exécution du programme suivant ? 21, 55, 47, 56, 89, 78

```
1 L=[78,89,56,47,55,21]
2 for i in range(6):
3     print(L[5-i])
```

4. Quelles sont les valeurs affichées lors de l'exécution du programme suivant ? .....

```
1 L=[5,7,8,1]
2 for i in range(4):
3     print(L[i]*2**i)
```

Exercice 6

On considère la fonction `facile()` qui prend en paramètre une liste de nombres `V`.

```
1 def facile(V):
2     for i in range(len(V)):
3         V[i]=V[i]*2+i
4     return V
```

On choisit $V = [2,4,-1,0]$.

1. Donner `len(V)`. 4
2. Compléter le tableau d'étapes ci-dessous en donnant la liste obtenue à chaque étape :

Étapes :	V
Initialisation	[2,4,-1,0]
$i = 0$	[4,4,-1,0]
$i = 1$	[4,9,-1,0]
$i = 2$	[4,9,0,0]
$i = 3$	[4,9,0,3]

B. En itérant sur les éléments de la liste

Exercice 7 :

Quelles sont les valeurs affichées lors de l'exécution du programme suivant ? 10, 14, 16, 24

```
1 L=[5,7,8,12]
2 for x in L:
3     print(x*2)
```

Exercice 8

On considère la fonction `kezako()` ci-dessous :

```
1 def kezako(valeur):
2     liste=[2,5,6,6,7,2,1,2,5,6,7,8,9,4,2,2]
3     nombre=0
4     for x in liste:
5         if x==valeur:
6             nombre=nombre+1
7     return(nombre)
```

1. On choisit valeur=6. Quelle est LA valeur renvoyée par la fonction ? ...3.....
2. On choisit valeur=3. Quelle est LA valeur renvoyée par la fonction ? .0.....
3. Que renvoie, d'une façon générale, la fonction kezako ?
   .....le nombre de fois on (valeur) est répété dans la liste.....

Exercice 9

Écrire en Python une fonction `dernier()` qui prend en paramètre une liste d'au moins deux nombres `L` et qui renvoie la somme du premier et du dernier élément de la liste.

Exercice 10

Les fonctions suivantes prennent en paramètre une liste `L`. Indiquer celles qui renvoient le produit des éléments de la liste.

```
1 def produit1(L):
2     p=1
3     for i in range(len(L)):
4         p=p*L[i]
5     return(p)
```

```
1 def produit2(L):
2     p=1
3     for i in range(len(L)):
4         p=p*i
5     return(p)
```

```
1 def produit3(L):
2     p=1
3     for x in L:
4         p=p*x
5     return(p)
```

```
1 def produit4(L):
2     p=1
3     for i in range(L):
4         p=p*L[i]
5     return(p)
```