1 Notion de listes

Exercice 1 :
Qu'affiche le programme suivant ?

```
1 s="Quelle␣question␣est-ce␣?"
2 print(s[0:16]+s[-1])
```

Exercice 2 :
Qu'affiche le programme suivant ?

```
1 L=[4,2,9]
2 n=L[0]+L[1]
3 print(n)
```

Exercice 3 :
Qu'affiche le programme suivant ?

```
1 L=[3,1,4,5,9]
2 for x in L:
3   print(x)
```

Exercice 4 :
Qu'affiche le programme suivant ?

```
1 L=[9,0,3]
2 M=[9,9,7]
3 N=L+M
4 print(N)
```

Exercice 5 :

1. Que font les programmes suivants :

```
1 s="je␣vais␣travailler␣ce␣soir"
2 for i in range(1,10):
3     print(s)
```

et

```
1 s="je␣vais␣travailler␣ce␣soir"
2 print(s*10)
```

Le chaîne spéciale $\backslash n$ permet, dans une chaîne de caractères, d'aller à la ligne.

Essayer:

```
1 s="je␣vais␣travailler␣...␣\n␣...␣ce␣soir\n\n"
2 print(s*10)
```

2. Écrire un programme qui affiche les lignes ci-dessous, avec 5 lignes, puis 10 lignes, puis n lignes, n étant demandé à l'utilisateur:

```
1 *
2 **
3 ***
4 ****
5 *****
6 ...
```

3. Modifier ce programme pour qu'il affiche maintenant le "sapin" ci-dessous, à 5 lignes, puis 10 lignes, puis n lignes, n étant demandé à l'utilisateur:

```
1     *
2    ***
3   *****
4  *******
5 *********
6 ...
```

Exercice 6 :

1. Quels sont les affichages successifs du programme suivant ?

```
1 s="je␣vais␣travailler␣ce␣soir␣"
2 print(s[3])
3 print(s[3:7])
4 print(len(s))
5
6 for i in range (len(s)):
7     print(s[i])
```

2. Compléter le programme précédent de manière à ce qu'il compte le nombre de "a" dans la chaîne `s` précédente.

3. Reprendre la question précédente pour compter et afficher le nombre de mots.

Exercice 7 :
Le programme suivant permet de décomposer les chiffres qui composent un nombre: le nombre n est converti en chaîne de caractères.
Cette chaine `s` peut alors être manipulée comme un tableau. Qu'affiche le programme suivant ?

```
1 n=412
2 s=str(n)
3 print(s[2])
4
5 for i in s:
6     print(i)
7
8 print(s[0]+s[1]+s[2])
9 print(int(s[0])+int(s[1])+int(s[2]))
```

1. Ecrire un programme qui, à un nombre donnée (ou demandé à l'utilisateur), retourne la somme des chiffres qui le compose.
   Par exemple, pour $n = 412$, le programme retourne $4 + 1 + 2 = 7$.

2. Ecrire un programme qui, à un nombre donné (ou demandé à l'utilisateur), retourne des carrés des chiffres qui le compose.
   Par exemple, pour $n = 412$, le programme retourne $4^2 + 1^2 + 2^2 = 21$.

3. Un nombre heureux est un nombre entier qui, lorsqu'on ajoute les carrés de chacun de ses chiffres, puis les carrés des chiffres de ce résultat et ainsi de suite jusqu'à l'obtention d'un nombre à un seul chiffre égal à 1.
   Par exemple, 7 et 13 sont des nombres heureux, en effet :
   $7^2 = 49$ puis $4^2 + 9^2 = 97$ puis $9^2 + 7^2 = 130$ puis $1^2 + 3^2 + 0^2 = 10$ puis $1^2 + 0^2 = 1$.
   De même pour 13.
   Ecrire un programme qui, à un nombre donné (ou demandé à l'utilisateur), retourne s'il est heureux ou non.

Exercice 8 :
Une cuve d'eau a une contenance de 1000L. En moyenne, la cuve d'eau se remplit de 3L d'eau par jours. Ecrire une fonction Python `remplir()` qui renvoie le nombre de jours qu'il faudra pour remplir la cuve.

Exercice 9 :

1. Ecrire un programme qui construit une liste de 5 nombres entiers aléatoires compris entre 1 et 6.

2. Compléter le programme précédent pour qu'il affiche de plus :
   - "paire" : si 2 chiffres sont identiques
   - "brelan" : si 3 chiffres sont identiques
   - "carré" : si 4 chiffres sont identiques
   - "yams" : si les 5 chiffres sont identiques

3. Modifier le programme précédent pour compter, sur 1000 tirages de 5 chiffres, le nombre de paires, de brelans, de carrés et de yams obtenus.

Exercice 10 :
Un altimètre est un instrument de mesure permettant de déterminer la distance verticale entre un point et une hauteur de référence. Lors d'une randonnée en montagne, Alice étalonne son altimètre à son point de départ, puis mesure à chaque heure l'altitude relative atteinte.

À la fin de sa randonnée, elle obtient une liste d'entiers naturels qu'elle range dans une liste Python nommée `alt` :

```
1 alt = [300, 500, 600, 1000, 800, 900, 500, 600, 200, 0]
```

La randonnée d'Alice a duré 10 heures, elle a atteint une altitude relative de 300m au bout d'une heure, de 500m aubout de deux heures, etc...

Alice souhaite désormais calculer certaines valeurs relatives à son parcours.

1. Quelle fonction déjà existante en Python lui permet de calculer la durée de sa randonnée ?

2. Définir une fonction Python nommée `altmax`, qui prend en argument une liste `alt` et qui retourne l'altitude relative maximale atteinte lors de sa randonnée.
   Par exemple, dans le cas précédent, la fonction devra renvoyer la valeur `1 000`.

3. Alice cherche maintenant à savoir à quel moment son ascension a été la plus rapide en calculant le dénivelé maximal réalisé en une heure. Elle cherche donc la plus grande différence entre deux emplacements consécutifs de la liste. Par exemple, pour la liste précédente, le dénivelé maximal est égal à 400 et a été réalisé entre la troisième et la quatrième heure.
   Définir en Python une fonction nommée `denivmax` prenant en argument une liste `alt` et retournant le dénivelé maximal réalisé en une heure durant sa randonnée.

4. Modifier la fonction précédente pour qu'elle retourne non pas le dénivelé maximal mais l'heure à laquelle débute la réalisation de ce dénivelé. Pour la liste donnée en exemple, cette fonction devra donc retourner la valeur 3.

5. Définir une fonction nommée `denivtotal` retournant la somme des dénivelés positifs réalisés durant cette randonnée. Pour la liste donnée en exemple, cette fonction devra retourner la valeur `1 200`.

6. Enfin, on appelle sommet toute altitude relative strictement supérieure à l'altitude qu'elle précède et à l'altitude qui lui succède dans la liste `alt`. Dans l'exemple qui nous sert à illustrer cet exercice, la randonnée d'Alice présente trois sommets de valeurs 1000, 900 et 600 atteints à la quatrième, sixième et huitième heure de marche.
   Rédiger fonction `sommets` retournant la liste des sommets de la randonnée.

Exercice 11 :
Onconsidère un tableau de bits dont les éléments sont égaux aux entiers 0 ou 1. Rédiger en Python une fonction calculant le nombre maximal de 0 consécutifs présents dans ce tableau.

Exercice 12 :

1. Dans le module `numpy.random` se trouve une fonction nommée `randint`. Utiliser cette fonction pour créer un tableau de 100 entiers tirés au hasard entre 0 et 99.

2. Calculer le nombre d'entier entre 0 et 99 qui n'appartiennent pas à ce tableau.

3. Recommencer cette expérience un grand nombre de fois pour évaluer le nombre moyens d'éléments d'absents (la valeur théorique est d'environ 36,6).

Exercice 13 :
On propose dans cet exercice plusieurs méthodes pour générer la liste des nombres premiers inférieurs ou égaux à 1 000.

1. Première méthode
   Définir une fonction qui prend en paramètre un entier n et retourne le plus petit nombre p strictement supérieur à n, puis utiliser cette fonction pour générer la liste demandée.

2. Deuxième méthode
   On se propose maintenant de générer cette liste en utilisant le principe du crible d'Erathostène. L'algorithme procède par élimination :
   Partant de la liste des entiers de 2 à 1 000, on supprime de celle-ci les multiples de 2.
   L'entier suivant, 3, est forcément premier. On élimine à son tour ses multiples de la liste et on procède ainsi jusqu'à parcourir toute la liste.

3. Troisième méthode
   Bien que cela ne soit pas très efficace, on peut enfin générer cette liste par compréhension en générant successivement les trois listes :

   - $l_1 = \{n,\ 2 \leq n \leq 1\ 000\}$;
   - $l_2 = \{(n, \mathcal{D}_n),\ n \in l_1\}$ où $\mathcal{D}_n = \{k \in [|1, n|],\ k|n\}$ est la liste des diviseurs de n ;
   - $l_3 = \{n,\ (n, \mathcal{D}_\setminus) \in l_2 \ \text{ et } \ \text{Card}(\mathcal{D}_n) = 2\}$.

Exercice 14 :
Dans la prison centrale de Sikinia, il y a 100 cellules numérotées de 1 à 100. Les portes des cellules peuvent être dans deux états : ouvertes ou fermées. On peut passer d'un état à un autre en faisant faire un demi-tour au bouton de la porte.
Au moment où commence l'histoire, toutes les cellules sont fermées et occupées.
Pour féter le vingtième anniversaire de la république de Sikinia, le président décide d'une amnistie. Il donne au directeur les ordres suivants :

Tournez successivement d'un demi-tour les boutons :

- *de toutes les portes ;*
- *puis d'une porte sur deux à partir de la deuxième ;*
- *puis d'une porte sur trois à partir de la troisième ;*
- *puis d'une porte sur quatre à partir de la quatrième ;*
- *et ainsi jusqu'à la dernière cellule.*

Quelles sont les numéros des cellules dont les occupants seront libérés ?
Conjecturer et prouver le résultat dans le cas général de n cellules.

Exercice 15 :
On considère n personnages agées en cercle et un entier positif $m \leq n$. En commançant par une personne désignée, on fait le tour du cercle en éliminant une personne toutes les m.

Après chaque élimination, on continue à compter avec le cercle qui reste.
Le processus se poursuite jusqu'à ce que les n personnes aient été éliminées.
L'ordre dans lequel les personnes sont éliminées du cercle définit la permutation de Josephus.
Par exemple, la permutation de Josephus pour $n = 7$ et $m = 3$ est $[3, 6, 2, 7, 5, 1, 4]$.

Définir une fonction Python qui prend en paramètres les entiers n et m et qui retourne la permutation de Josephus correspondante.

Exercice 16 :
Une permutation est un arrangement ordonné d'objets.
Par exemple, 3124 est une permutation des chiffrs 1, 2, 3 et 4.
Si on classe par ordre croissant toutes les permutations des chiffres 0, 1 et 2 on obtient:

012 ; 021 ; 102 ; 120 ; 201 ; 210

Quelle est la millionième permutation des chiffres 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 lorsqu'on les range par ordre croissant ?

Exercice 17 :
Un nombre de Hamming est un entier naturel non nul dont les diviseurs premiers sont inclus dans $\{2, 3, 5\}$:

1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, 18, 20, 24, 25, 27, 30, ...

Lorsqu'on les ordonne par ordre croissant, quel est le 2000ième nombre de Hamming?