Nous allons manipuler des tableaux et la structure que nous considérons pour l'instant est celle de type `list` que l'on désigne en parlant de *liste*.
Une liste est un ensemble ordonné (chaque élément à un numéro d'ordre) d'éléments (de types non nécessairement homogènes). Elles sont définies entre crochets et les éléments sont séparés par une virgule. Voici, au travers d'exemples, quelques façons de définir des listes :

```
>>> L = [1,2,3]   # définition directe
>>> list(range(5))   # définition à partir d'un objet itérable
[0, 1, 2, 3, 4]
>>> list(range(4)) + [1, 5, 42]   # par concaténation de listes
[0, 1, 2, 3, 1, 5, 42]
>>> [0]*5   # par concaténation de copies d'une même liste
[0, 0, 0, 0, 0]
>>> [2*k+1 for k in range(5)]   # en compréhension
[1, 3, 5, 7, 9]
>>> [x**2 for x in range(15) if x%3 == 0]   # en compréhension avec condition
[0, 9, 36, 81, 144]
```

On peut directement accéder à tous les éléments de la liste et les modifier :

```
>>> L = [2, 5, 42, 6, 3, -2, 4, 6, 8, 9, 12]
>>> L[2]
42
>>> L[-3]
8
>>> L[1:5] # sous-liste de l'élément n°1 à l'élément n°5 non compris
[5, 42, 6, 3]
>>> L[1:10:2] # de l'indice 1 à l'indice 10 non compris avec un pas de 2
[5, 6, -2, 6, 9]
>>> L[2:] # de l'indice 2 jusqu'à la fin
[42, 6, 3, -2, 4, 6, 8, 9, 12]
>>> L[2::3] # idem avec un pas
[42, -2, 8]
>>> L[:8] # du début jusqu'à l'indice 8 non compris
[2, 5, 42, 6, 3, -2, 4, 6]
>>> L[:8:2] # idem avec un pas
[2, 42, 3, 4]
>>> L[:-2:3] # fonctionne également avec des n° négatifs
[2, 6, 4]
>>> L[0] = 76; L # remplacement d'une valeur
[76, 5, 42, 6, 3, -2, 4, 6, 8, 9, 12]
```

Pour le moment, nous aurons également besoin de ce qui suit :

```
>>> L = [1, 6, 9, 23, 21, 0, 3]
>>> len(L)  # longueur de la liste
7
>>> L.append(5)   # ajout d'une valeur
>>> L.append(8)   # ajout d'une valeur
>>> L
[1, 6, 9, 23, 21, 0, 3, 5, 8]
>>> 5 in L
True
>>> 7 in L
False
```

Exercice 1

Écrire une fonction :

1. d'argument `n` et qui renvoie la liste formée par `n` zéros ;
2. d'argument `n` et qui renvoie la liste des `n` premiers carrés ;
3. d'argument `n` et qui renvoie la liste des `n` premiers carrés pairs ;
4. d'argument `n` et qui renvoie la liste des couples (i, j) avec $0 \leqslant i < n$ et $0 \leqslant j < n$;
5. d'argument `n` et qui renvoie la liste des couples (i, j) avec $0 \leqslant i < j < n$;
6. d'argument une liste d'entiers `L` et qui renvoie la liste formée par les termes d'indices pairs de `L` ;
7. d'argument une liste d'entiers `L` et qui renvoie la liste formée par les valeurs positives de `L`.

Exercice 2

1. Que réalise l'instruction : `J = list(range(32))` ?
2. Que donnent les instructions : `J[::4]` et `J[1::4]` ?
3. On exécute cinq fois l'instruction suivante, qu'observe-t-on ?

```
>>> J = J[0::4] + J[1::4] + J[2::4] + J[3::4]
```

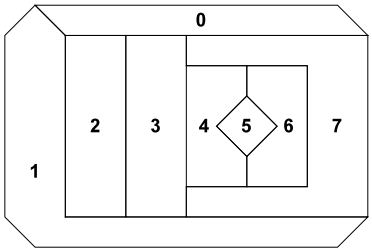
Exercice 3

Dans cet exercice, un segment $[a, b]$ (avec $a \leqslant b$) est représenté par la liste : `[a,b]`.

1. Deux segments sont disjoints si leur intersection est vide. Écrire une fonction `disjoints` de deux arguments `i1` et `i2` qui teste si les segments `i1` et `i2` sont disjoints. La fonction renvoie le booléen `True` s'ils sont disjoints, `False` sinon.
2. Écrire une fonction `intersection` de deux arguments `i1` et `i2` qui renvoie la liste correspondant à l'intersection de `i1` et `i2` (en renvoie la liste vide s'ils sont disjoints).
3. La fusion de deux segments a comme minimum le plus petit des minimums des deux segments, et comme maximum le plus grand des maximums des deux segments. Écrire une fonction `fusion` de deux arguments `i1` et `i2` et qui renvoie le segment correspondant à la fusion de `i1` et `i2`.

Exercice 4

Le dessin ci-dessous représente une carte avec 8 pays numérotés de 0 à 7.



1. Définir la liste `carte` de taille 7 telle que l'élément `carte[i]` la liste des numéros supérieurs à i des pays ayant une frontière commune avec le pays numéro i.
   Par exemple `carte[3]` est la liste `[4,7]`.
2. Écrire puis tester une fonction `voisins` de trois arguments, deux nombres entiers distincts i et j et une liste caractérisant les frontières communes d'un ensemble de pays, qui renvoie `True` si les pays numérotés i et j ont une frontière commune, et `False` sinon.