Exercices sur les listes

Dans cette feuille d'exercice la plupart des exercices demandent de créer une fonction. On se propose donc de les réunir dans le module `ModuleListe` et de créer un script `TestModuleListe` pour effectuer les tests d'essais de ces fonctions.

Exercice n° 1

Ecrire la fonction `CreaTabInt(N,Min,Max)` qui construit une liste de `N` entiers compris entre `Min` et `Max`. La fonction retournera la liste. On utilisera la fonction `randint(a,b)` du module `random` pour engendrer une à une les valeurs nécessaires à la création de la liste.

L'algorithme sera basé sur la création d'une liste vide (`Loli = list()`) puis par ajout successif des nouveaux éléments (voir les diapos de cours)

Ecrire ensuite la fonction `CreaTabFloat(N,Min,Max)` qui, en se basant sur la fonction `random()` du module `random`, créera une liste de `N` valeurs réelles comprises entre `Min` et `Max`.

Exercice n° 2

A partir de la fonction précédente, écrire la fonction `CreaTabIntUnique(N,Min,Max)` qui fera la même chose que mais qui s'assurera que la liste ne contienne pas de doublons. Pour cela il est possible d'utiliser le test `if(Val in Loli)` qui retourne `TRUE` si `Val` est présent dans `Loli`.

On vérifiera lors du test d'appel à la fonction que la liste retournée contient bien les `N` éléments demandés.

Exercice n° 3

Ecrire le fonction `MaxiTab(Maliste)` qui retourne la valeur et la position de l'élément le plus grand de la liste passée en argument.

On pourra sans peine faire une version similaire `MinTab(Maliste)` pour rechercher l'élément le plus petit.

Exercice n° 4

Ecrire la fonction `TriTab(Maliste)` qui retourne une copie ordonnée par valeur décroissante de la liste passée en argument.

Il est rappelé que pour faire une copie complète et indépendante d'une liste existante, il est nécessaire d'utiliser la syntaxe `Copie = Originale[:]` , avec `Originale` une liste existante et `Copie` le nom choisi pour ce clone.

Ecrire le programme suivant :

- En utilisant la fonction `CreaTabFloat(N,Min,Max)` de l'exercie n°1, créer un tableau de 100 valeurs réelles comprises entre -10 et +10
- Trier le tableau avec `TriTab`
- Demander une valeur à l'utilisateur comprise entre le minimum et le maximum du tableau créer (qui ne sont pas forcément -10 et 10)
- Insérer la valeur saisie dans le tableau « à sa place », c'est-à-dire en conservant la décroissance de la liste
- Afficher la place où cette valeur a été insérée ainsi que la valeur qui précède et celle qui suit dans ce tableau.

Exercice n° 5

Le module **Grafe.py** qui vous est fourni utilise la bibliothèque **matplotlib.pyplot** pour faire des représentations graphiques simples de listes. Essayez pour vous faire une idée de son contenu :

```
F = [1,10,20,30,40,12]
Grafe.Courbe_1D(F)
X = [1,2,3,4,5,6,7]
Y = [1,4,9,16,25,36,49]
Grafe_Point2D(X,Y)
```

Si cela vous intéresse le lien suivant : *https://matplotlib.org/api/pyplot_summary.html* vous sera utile si vous voulez comprendre et modifier le module **Grafe** (mais ce n'est ni nécessaire ni obligatoire !)

Dans cet exercice on vous demande de tracer la représentation graphique continue (fonction Grafe.Courbe2D) du polynôme de degré 3 :

$$F(x) = a.x^3 + b.x^2 + c.x + d$$

Le quadruplet de nombres réels (a,b,c,d) sera demandé à l'utilisateur. On utilisera 200 valeurs pour x, qui seront comprises entre -10 et +10.

On pourra utiliser la fonction **Polyval** (exercice n°5 sur les boucle for (suite)) pour calculer la valeur $F(x)$.

Exercice n° 6 (pour s'amuser…)

Faites la même chose avec la fonction :

$$F(x,r) = \begin{cases} \left\|x^{2/3}\right\| + \sqrt{1-x^2} & \text{si r=0} \\ \left\|x^{2/3}\right\| - \sqrt{1-x^2} & \text{si r=1} \end{cases}$$

- $\left\|x^{2/3}\right\|$ pourra s'écrire en python comme :

```
r = x ** (2 / 3)
N = math.sqrt(r.real**2 + r.imag**2)
```

- **x** sera une liste de 1000 valeurs, toutes comprises entre -1 et 1.
- **r** sera prendra la valeur 0 ou 1 aléatoirement à chaque appel de la fonction.
- On utilisera la fonction **Grafe.Point2D()** pour effectuer le tracé.