TP 4 - Boucles

Récupérez le fichier `iutk.py`. Vous devez écrire un programme par exercice, avec comme nom **exoX.py** (X est le numéro de l'exo) et mettre en commentaires les tests que vous avez faits pour vérifier que votre programme fonctionne.

Exercice 1. Premières listes

1. Déclarer la liste `lst` suivante : `[14,7,6,12,2,3,3,10]` et l'afficher.
2. Ensuite, modifier la liste de telle façon que son dernier élément soit divisé par 2 et la ré-afficher.
3. Modifier la liste de telle façon que l'on retranche 1 à tous ses éléments et la ré-afficher.
4. Ensuite afficher tous les éléments de la liste sur des lignes différentes.
5. Puis faire afficher seulement les nombres pairs (sans fabriquer de nouvelle liste).
6. Ensuite, faire afficher 10 fois chaque élément de la liste sur la même ligne séparés par des espaces. Attention, la fonction `print('bonjour')` va à la ligne après avoir affiché `bonjour` ; pour éviter ce comportement, il faut utiliser `print('bonjour', end='')`.
7. Enfin, afficher chaque élément autant de fois que sa valeur. Par exemple, 14 sera affiché 14 fois, ...

A la fin de l'exercice, votre programme devra afficher exactement :

```
Question 1
[14, 7, 6, 12, 2, 3, 3, 10]
Question 2
[14, 7, 6, 12, 2, 3, 3, 5]
Question 3
[13, 6, 5, 11, 1, 2, 2, 4]
Question 4
13
6
5
11
1
2
2
4
Question 5
6
2
2
4
Question 6
13  13  13  13  13  13  13  13  13  13
6  6  6  6  6  6  6  6  6  6
5  5  5  5  5  5  5  5  5  5
11  11  11  11  11  11  11  11  11  11
1  1  1  1  1  1  1  1  1  1
```

```
2  2  2  2  2  2  2  2  2  2
2  2  2  2  2  2  2  2  2  2
4  4  4  4  4  4  4  4  4  4
Question 7
13  13  13  13  13  13  13  13  13  13  13  13  13
6  6  6  6  6  6
5  5  5  5  5
11  11  11  11  11  11  11  11  11  11  11
1
2  2
2  2
4  4  4  4
```

Exercice 2. Entrées

1. Voici un programme qui demande des entiers à l'utilisateur jusqu'à ce qu'il rentre `stop`.

```
1 while True:
2     valOuStop=input("valeur ou stop ")
3     if valOuStop=="stop":
4         break
5     a=int(valOuStop)
```

   Modifier le programme pour qu'il enregistre les entiers dans une liste `lst`.

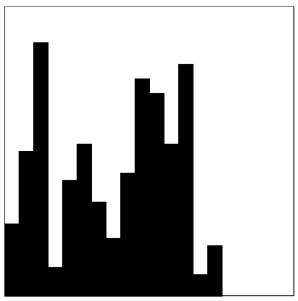
2. A la suite de la boucle qui crée la liste `lst`, afficher les nombres dans l'ordre où ils ont été rentrés sur la même ligne séparés par des espaces.
3. Ensuite construire une liste `lst2` contenant seulement les nombres pairs de `lst` et l'afficher avec la fonction `print`.
4. Ensuite construire une liste `lst3` contenant les entiers `lst` dans l'ordre inverse de leur saisie et l'afficher avec la fonction `print`.

Exercice 3. Dessiner une ville

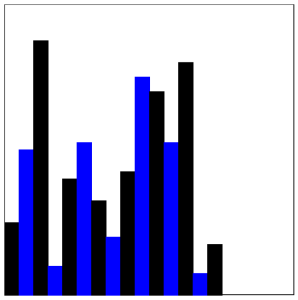
Pour cet exercice, vous pouvez ré-utiliser le fichier `dessin-init.py` fourni au TP précédent. Le but de l'exercice est de réaliser un programme dessinant une ville. Pour voir le résultat que l'on cherche à obtenir, récupérer le fichier `dessin-ville.pyc`. Vous pouvez le lancer en tapant :

```
python3 dessin-ville.pyc
```

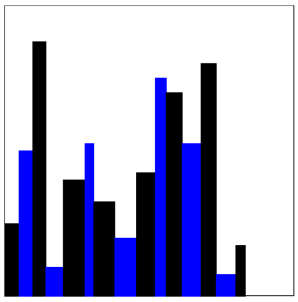
1. Déclarer la liste `lst` des hauteurs des immeubles dans la ville. Dans l'exemple : `lst=[100,200,350,40,160,210,130,80,170,300,280,210,320,30,70]`
2. Faire afficher une suite de rectangles de largeur fixe de 20 et dont les hauteurs sont données par la liste `lst`.



3. Modifier le programme pour que les rectangles soient alternativement coloriés en bleu et noir (pour ce faire, vous utiliserez une variable `couleur` qui vaudra soit “blue” soit “black”).



4. On souhaite maintenant pouvoir faire des rectangles de largeur variable. Pour cela créer la liste :
   `lstLargeurs=[20, 19, 18, 24, 30, 12, 29, 30, 26, 15, 22, 26, 21, 27, 13]`
   Modifier votre programme pour qu'il affiche les rectangles dont les hauteurs sont données par la liste `lst` et les largeurs par `lstLargeurs`. Par exemple, le premier immeuble est de hauteur 100 et de largeur 20.



5. Faites en sorte que la liste `lstLargeurs` soit créée en choisissant des valeurs aléatoires (entre 10 et 30, en utilisant `randint`. Il vous faudra rajouter `from random import randint` au début de votre programme). Afficher la liste `lstlargeurs` pour vérifier.
6. Faire en sorte que la largeur de la fenêtre corresponde exactement à la largeur de tous les immeubles.
7. (facultatif) Modifier votre programme pour qu'il affiche chaque rectangle avec un petit délai, en utilisant les fonction `sleep` et `miseAJour`.

Exercice 4. Facultatif

Récupérez le programme `balle-init.py` sur la page du cours. Ce programme simule le lancer d'une balle dont le vecteur de vitesse initial est donné par le premier clic dans la fenêtre. Le but de l'exercice est de réaliser un petit jeu dans lequel on a 3 tentatives pour toucher une cible.

Pour voir ce que l'on attend de vous, récupérer le fichier `balle.pyc`. Vous pouvez lancer le programme, en tapant :

```
python3 balle.pyc
```

1. Ouvrez et lisez le fichier `balle-init.py` afin de comprendre son fonctionnement. N'hésitez pas à ajouter des commentaires.
2. Utilisez la fonction `randint` vue au TP précédent pour placer une cible qui sera un carré rouge de coté 20 dans le quart inférieur droit de la fenêtre.
3. Ajoutez les tests permettant de savoir si la balle sort de la fenêtre, auquel cas on sort de la boucle et l'on affiche *Perdu* avec le code suivant :

```
1    texteCentre(largeurFenetre/2,hauteurFenetre/2,"Perdu","red")
```

4. Ajoutez les tests permettant de savoir si la balle touche la cible, auquel cas on sort de la boucle et l'on affiche *Gagné* avec le code suivant :

```
1    texteCentre(largeurFenetre/2,hauteurFenetre/2,"Gagne","red")
```

5. Modifiez votre code de manière à donner 3 essais au joueur.