Initiation à l'algorithmique et programmation

L3 2023-2024

Travaux Pratiques – Révision

Voici quelques exercices qui permettront de pratiquer les listes, les functions et les boucles en Python :

Exercice 1. *Nombres pairs*

Créez une fonction qui returne le nombres pairs dans l'intervalle donné.

Exemple :

```
»»» intervalle = [2, 10000]
»»» n = nbr_pairs(intervalle)
»»» print(n)
4999
```

Exercice 2. *Divination*

Vous allez devoir écrire un programme jeu.py proposant à l'utilisateur de deviner un entier entre 1 et 10 en trois essais.

- Commencez par choisir un nombre au hasard. Pour cela, vous pouvez utiliser la fonction randint :
  ```
  from random import randint
  print(randint (0 ,2)) # affiche un entier aleatoire dans [0,2]
  ```
- Puis demandez un entier entre 1 et 10 à l'utilisateur et comparez-le à l'entier aléatoire choisi.
- Faites en sorte que l'utilisateur aie 3 essais et que le programme s'arrête quand il a trouvé, ou quand son nombre d'essais est écoulé, en affchant un message approprié.
- Testez votre programme avec diverses valeurs.
- Que se passe-t-il si l'utilisateur rentre un flottant ?
- Que se passe-t-il si l'utilisateur se trompe et rentre un nombre qui n'est pas entre 1 et 10 ? Modifiez votre programme pour qu'il redonne sa chance à l'utilisateur dans ce cas-là.

Exercice 3. *Distance de Hamming*

La distance de Hamming entre deux mots est une notion utilisée dans de nombreux domaines. Elle est définie, pour deux mots de **même longueur**, comme le nombre de positions où les deux mots ont un caractère différent. Écrire une fonction hamming qui calcule la distance de Hamming entre deux mots lorsqu'ils ont la même longueur, et qui renvoie -1 sinon.

Exemple :

```
»»» hamming("aaba", "aaha")
1
»»» hamming("poire", "pomme")
2
»»» hamming("stylo", "bouteille")
-1
```

Exercice 4. *Trois mais pas cinq*

Ecrire un programme qui prend un entier positif n en entrée, et affche tous les nombres de 1 à n qui sont divisibles par 3 mais pas divisibles par 5.

Exercice 5. *Doublons*

Soit la liste de nombres liste `[5, 1, 1, 2, 5, 6, 3, 4, 4, 4, 2]`. À partir de liste, créez une nouvelle liste sans les doublons, triez-la et affichez-la.

Exemple :

```
>>> l = [5, 1, 1, 2, 5, 6, 3, 4, 4, 4, 2]
>>> sans_doublons(l)
[1, 2, 3, 4, 5, 6]
```

Exercice 6. *Entrées*

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

```
while True:
    valOUstop = input("tapez une valeur ou stop")
    if valOUstop == "stop":
        break
    a = int(valOUstop)
```

- Modifier le programme pour qu'il enregistre les entiers dans une liste `lst`.
- A la suite de la boucle qui crée la liste lst, affcher les nombres dans l'ordre où ils ont été rentrés sur la même ligne séparés par des espaces.
- Ensuite construire une liste `lst2` contenant seulement les nombres pairs de `lst` et l'afficher avec la fonction `print`.
- Ensuite construire une liste `lst3` contenant les entiers `lst` dans l'ordre inverse de leur saisie et l'afficher avec la fonction `print`.

Exercice 7. *Tri de liste*

Soit la liste de nombres `[8, 3, 12.5, 45, 25.5, 52, 1]`. Triez les nombres de cette liste par ordre croissant, sans utiliser la fonction `sort()`. Les fonctions et méthodes `min()`, `.append()` et `.remove()` vous seront utiles.

Exemple :

```
>>> l = [8, 3, 12.5, 45, 25.5, 52, 1]
>>> sort_list(l)
[1, 3, 8, 12.5, 25.5, 45, 52]
```

Exercice 8. *Le nombre mystère*

Trouvez le nombre mystère qui répond aux conditions suivantes :

- Il est composé de 3 chiffres.
- Il est strictement inférieur à 300.
- Il est pair.
- Deux de ses chiffres sont identiques.
- La somme de ses chiffres est égale à 7.

On vous propose d'employer une méthode dite *"brute force"*, c'est-à-dire d'utiliser une boucle et à chaque itération de tester les différentes conditions.

Exercice 9. *Dessin des étoiles*

1. Créez un programme qui imprime un carré d'étoiles. L'utilisateur doit entrer un nombre entier qui correspond au nombre de lignes qui seront utilisées pour dessiner le carré.
   Exemple : `» n = 4;`
   ****
   ****
   ****
   ****
2. Créez un autre programme qui imprime une inverse pyramide d'étoiles sur l'écran. L'utilisateur doit entrer un nombre entier qui correspond au nombre de lignes qui seront utilisées pour dessiner la pyramide.
3. Créez un autre programme qui imprime une pyramide d'étoiles sur l'écran. L'utilisateur doit entrer un nombre entier qui correspond au nombre de lignes qui seront utilisées pour dessiner la pyramide.
4. Essayez maintenant de dessiner un clepsydre.
5. **BONUS :** Essayez de dessiner un cœur en utilisant des étoiles !