TP2 - Fonctions

M. Tellene

Oct 2021

1 C'est à vous!

EXERCICE 1

Exercice etat.py TP sur les types, partie en plus

EXERCICE 2

Pour cet exercice, il faudra créer un fichier masquage.py

Écrire une fonction `masquage` qui prend en argument un mot entré par l'utilisateur et retourne un mot masqué à l'aide du caractère "*". Pour vous aider, penser à quelle fonction permet de récupérer la taille d'une chaîne de caractère et quel est l'opérateur de la répétition de chaîne de caractère.

Exemple de rendu :

```
>>> Entrer un mot : anonyme
*******
```

EXERCICE 3

Exercice inegalite_triangulaire.py TP sur les types, partie en plus

EXERCICE 4

Pour cet exercice, il faudra créer un fichier decodeur_octet.py
Vous allez réaliser un décodeur octet.

1. **Pour la première étape, nous allons juste passer l'octet en binaire.** Écrire une fonction `decodage` qui prend en argument un octet au format décimal (0-255) et qui retourne l'octet décoder en binaire.

   Si vous ne voyez pas comment faire, renseignez-vous sur la fonction `bin()`

   Attention, vous aurez sûrement comme résultat quelques choses comme ceci : `0bxxxxxxxx`. Nous voulons que notre décodeur affiche le résultat comme ceci : `xxxxxxxx`.

   Exemple de rendu :

   ```
   >>> decodage(42)
   '101010'
   ```

2. **Pour la deuxième étape, nous allons, en plus de renvoyer l'équivalent binaire, renvoyer l'équivalent hexadécimal et ASCII.** Modifier la fonction `decodage` pour qu'elle retourne en plus de la conversion binaire, la conversion en hexadécimal et la conversions en caractère ASCII.

   Si vous ne voyez pas comment faire, renseignez-vous sur la fonction `hex()` et `chr()`

   Attention, vous aurez sûrement comme résultat quelques choses comme ceci (pour l'hexadécimal) : `0xxx` (0x suivi de la conversion hexadécimale). Nous voulons que notre décodeur affiche le résultat comme ceci : `xx`.

   Exemple de rendu :

```
>>> decodage(42)
('101010', '2a', '*')
```

3. **Pour la troisième étape, nous allons ajouter le renvoie de l'octet passé en argument.** Modifier la fonction `decodage` pour qu'elle retourne en plus de la conversion binaire, hexadécimale et ASCII, l'octet entré en paramètre.

   Exemple de rendu :

```
>>> decodage(42)
('101010', '2a', '*', 42)
```

4. **Pour la quatrième étape, il faudra récupérer ce qui est renvoyé par la fonction dans une variable.** Modifier votre programme pour que les valeurs retournées par `decodage` soient récupérées dans une variable (disons `n`). Par conséquent, si à la fin du programme on écrit `print(n)`, cela devra afficher : `('101010', '2a', '*', 42)` (**si l'octet passé en paramètre est 42 !)**

EXERCICE 5

Pour cet exercice, il faudra créer un fichier discriminant.py

On souhaite écrire une fonction qui peut résoudre des équations d'ordre 2. Une équation d'ordre 2 est de la forme : $ax^2 + bx + c = 0$.

La méthode de résolution se passe en deux étapes :

— la première consiste à calculer $\Delta = b^2 - 4ac$

— la seconde étape consiste à trouver les solutions en fonction du signe de Δ. Si celui est positif et non nul alors les solutions seront :

$$x_1 = \frac{-b + \sqrt{\Delta}}{2a} \text{ et } x_2 = \frac{-b - \sqrt{\Delta}}{2a}$$

1. Écrire une fonction `calcul_delta` qui s'occupera de faire la première étape de la méthode. Cette fonction doit prendre en argument 3 nombres et retourne le Δ calculé
2. Sous votre fonction `calcul_delta`, écrire une fonction `resolution`. Cette fonction devra tout d'abord appeler `calcul_delta` et devra faire la seconde étape de la méthode.

Pour utiliser la racine carrée dans votre programme, ajouter la ligne : `from math import sqrt` **au tout début de votre programme (ligne 1).** `sqrt(delta)` **revient à faire** $\sqrt{\Delta}$

Exemple d'exécutions :

```
>>> resolution(1, 4, 3)
```