Introduction à la programmation – TP5

Objectif : Des fonctions en langage Python

Exercice 1 : une première fonction

Ecrire la fonction moyenne, qui étant donnés deux réels renvoie leur moyenne.
On utilisera cette fonction pour calucler la moyenne de 4.5 et 7.75.

Exercice 2 : Boucle somme

1. Écrire la fonction `masomme(n,p)` qui calcule et renvoie la somme des puissances p des entiers entre 0 et n : $masomme(n, p)$ doit renvoyer la valeur de S_n où $S_n = 1 + 2^p + 3^p + \ldots.. + n^p$.
2. Écrire la fonction `main()` qui demande à l'utilisateur de saisir les entiers n et p et renvoie la valeur de S_n où $S_n = 1 + 2^p + 3^p + \ldots.. + n^p$.

Exercice 3 : test de lancers de dés

On veut écrire des programmes pour « tester le lancer de dés ou de pièces aléatoire » : On n'oubliera pas le `from random import randint` en début de fichier

1. Écrire la fonction `alea(n)` qui renvoie un entier entre 1 et n (inclus)
2. - Écrire la fonction `lancer(n)` qui lance virtuellement (on représentera face par 1 et pile par 2) une pièce de monnaie n fois et compte au fur et à mesure puis renvoie le nombre de valeurs piles obtenues.
   - Écrire la fonction main qui demande le nombre de lancers à effectuer, lance l'expérience et affiche les résultats.

   Exemple :

```
combien de lancers ? 50
nombre de lancer pile pour ces 50 essais: 21
```

Exercice 4 : tests de calcul mental

1. Écrire la fonction `alea(n)` qui prend en argument une variable de type entier n et renvoie un nombre aléatoire compris entre 1 et n (inclus) .
2. Écrire la fonction `calcul(n)` qui prend en argument une variable de type entier `n`. Cette fonction détermine deux entiers aléatoirement choisis entre 1 et n (on utilisera la fonction précédente) et demande

à l'utilisateur de calculer la somme de ces deux nombres. Si la réponse est bonne le programme affiche `bravo` et renvoie 1, si la réponse est fausse le programme affiche la bonne réponse et renvoie 0.

3. Écrire la fonction `Serie_calcul(n,nb)` qui prend en argument deux variables de type entier (n et nb). Le programme va enchaîner nb questions comme celles de la question précédente, n étant la valeur maximum des nombres utilisés ; à la fin le programme doit afficher le nombre de bonnes réponses. Le programme doit fonctionner comme dans l'exemple.
4. Écrire la fonction `main()` pour lancer le programme précédent : le programme demande à l'utilisateur les valeurs de n et de nb.

```
valeur maximale des nombres  : 20
nombre de calculs  : 6
calculez  12 + 3 :15
bravo !
calculez  8 + 2 :10
bravo !
calculez  19 + 1 :21
c'est faux, il fallait trouver 20
calculez  12 + 10 :22
bravo !
calculez  8 + 7 :15
bravo !
calculez  7 + 20 :27
bravo !
score total pour les 6 calculs: 5
```

Exercice 5 : palindromes

1. Ecrire la fonction `retourne(ch)` qui, étant donné une chaîne de caractères, renvoie la chaîne écrite " à l'envers".
   Exemple : `retourne ("bonjour")` renvoie "ruojnob".
2. Ecrire une fonction qui teste si une chaîne de caractères est un palindrome (on rappelle qu'un palindrome est une chaîne qui se lit de la même façon de gauche à droite et de droite à gauche). On programmera cette fonction de deux façons différentes :
   1. En se servant de la fonction `retourne`
   2. En travaillant directement sur la chaîne .
   Comparer ces deux méthodes. Laquelle est la plus efficace ?
3. En déduire un programme qui affiche les années palindromiques de 1 à n. Une année est palindromique si la chaîne de caractères correspondante est un palindrome. On fera un affichage au fur et à mesure (exemples d'années palindromiques : 2, 66, 161, 1001, 1991)

Exercice 6 : Bonus – Nombres de Zuckerman

On rappelle que `n%p== 0` permet de tester que l'entier naturel `p` divise `n`.
On rappelle aussi les fonctions `str(n)` et `int(ch)` qui permettent de transformer un entier en chaîne de caractères et réciproquement.

1. Un entier positif est un nombre de Zuckerman s'il est divisible par chacun des chiffres qui le composent. Par exemple 36 est un nombre de Zuckerman car 3 et 6 divisent 36 mais 39 ne l'est pas car 9 ne divise pas 39. Attention tout nombre contenant un 0 dans son écriture n'est pas un nombre de Zuckerman.
   Ècrire la fonction `Zuckerman(nb)` qui renvoie `True` si `nb` est un nombre de Zuckerman et `False` sinon.
2. Ecrire la fonction `afficheZuckerman(debut, fin)` qui affiche les nombres de Zuckerman compris entre `debut` et `fin` (inclus). On utilisera évidemment la fonction de la question précédente.

```
>>> Zuckerman(15)
True
>>> Zuckerman(16)
False
>>> afficheZuckerman(10,100)
11 12 15 22 24 33 36 44 48 55 66 77 88 99
```