TP 1

Révisions : fonctions, conditionnelles, boucles

5 septembre 2018

Créez un dossier `infoPSI` dans un dossier à votre nom. Enregistrez un fichier `VotreNom-TP1.py` dans le dossier `infoPSI`. Faire **tous les exercices dans ce fichier** (à chaque exercice va correspondre une fonction). Lorsque vous testez une fonction, mettre les appels à la fonction directement **dans votre programme** puis les mettre en commentaire avec `#` avant de passer à l'exercice suivant (**ne pas les effacer**). A la fin du TP, vous devez enregistrer votre fichier `VotreNom-TP1.py` sur votre session, puis le **copier** sur le NAS (**ne pas enregistrer directement depuis spyder sur le NAS**).

Important :

— respectez les noms qu'on vous demande de donner aux fonctions.
— testez vos fonctions pour les vérifier (avec plusieurs valeurs pertinentes !).
— laissez en commentaire au moins un appel par fonction
— **ne pas utiliser de liste dans ce TP**

Exercice 1 : Bonjour

Écrire un programme qui demande à l'utilisateur son prénom, puis qui affiche un message lui disant bonjour en le saluant de son prénom. Exemple d'exécution :

```
Quel est votre prénom ? Jean
Bonjour Jean
```

Transformer votre programme en une fonction `bonjour` et tester votre fonction.

Exercice 2 : Conversion

Un euro vaut environ 9.13 couronnes suédoises. Écrire une fonction `conversion` qui prend en argument un prix en euros et renvoie le prix en couronnes suédoises correspondant. Tester votre fonction.

Exercice 3 : Examen

Écrire une fonction `examen` qui demande à l'utilisateur d'entrer sa note sur 20 à l'examen, puis qui affiche un message indiquant si l'élève est reçu à l'examen ou non. Un élève est reçu à l'examen si sa note est supérieure ou égale à 10. Si la note n'est pas cohérente (note négative ou supérieure à 20), la fonction doit simplement afficher un message d'erreur. Tester votre fonction. Exemple d'exécution :

```
Entrez votre note : 11
Votre note est 11, vous avez réussi l'examen.
```

Exercice 4 : Note

Écrire une fonction `nombre2lettre` qui prend en argument une note sur 20 et renvoie une note donnée par une lettre, selon la liste de correspondance suivante :

Note inférieure ou égale à 20 et supérieure ou égale à 15 → A
Note inférieure à 15 et supérieure ou égale à 10 → B
Note inférieure à 10 et supérieure ou égale à 5 → C
Note inférieure à 5 et supérieure ou égale à 0 → D
Autres notes → E (comme Erreur)

Tester votre fonction.

Exercice 5 : Compter

Écrire une fonction `compter` qui prend en argument un entier n et affiche les entiers de 1 à n.

Exercice 6 : Triangle

Écrire une fonction `triangle` qui prend en argument un entier n, puis qui affiche le triangle d'entiers de taille n. Ne pas faire d'appel à la fonction `compter` (la fonction `triangle` doit faire elle-même tous ses affichages). Par exemple, l'exécution de `triangle(4)` doit donner :

```
1
12
123
1234
```

Faire de même une fonction `triangle2` telle que le triangle obtenu pointe vers le bas :

```
1234
123
12
1
```

Exercice 7 : Continuer

Écrivez une fonction `continuer` qui demande a l'utilisateur « Continuer ? », et qui continue de lui poser la question tant que celui-ci lui répond « oui ».

Exercice 8 : Multiplie

Écrire une fonction `multiplie` qui demande à l'utilisateur un entier n (combien de nombres à multiplier), puis qui lit n nombres réels, puis affiche le produit de ces nombres. Tester votre fonction. Exemple d'exécution :

```
Combien de nombres ?  2
Nombre 1 :  4
Nombre 2 :  3
Le produit vaut 12.0
```

Exercice 9 : Produit

Écrire une fonction `produit` qui lit des entiers entrés par l'utilisateur jusqu'à ce que l'utilisateur entre 0, puis qui affiche le produit des entiers non nuls entrés par l'utilisateur. Tester votre fonction. Exemple d'exécution :

```
Entrez les nombres à multiplier en terminant par 0 :
4
3
0
Le produit vaut 12
```

Exercice 10 : Suite

Écrire une fonction `terme` qui prend en argument un entier n et qui renvoie le terme d'indice n de la suite définie par récurrence par $u_0 = 1$, $u_1 = 2$ et $u_n = 3 * u_{n-1} + u_{n-2}$ pour $n \geq 2$. Par exemple, `terme(4)` vaut 76.

Exercice 11 : Mult3pas9

Écrire une fonction `mult3pas9` qui prend en argument un entier n et renvoie `True` si n est multiple de 3 mais pas de 9, et `False` sinon.
Modifier votre fonction pour qu'elle ne comporte qu'une seule instruction (une seule ligne de code, pas de `if`).

Exercice 12 : Premiers

Écrire une fonction `premiers` qui prend en argument un entier n, puis qui affiche les nombres premiers inférieurs ou égaux à n. Par exemple, premiers(12) doit afficher 2, 3, 5, 7 et 11.

Exercice 13 : Premiers2

Écrire une fonction `premiers2` qui prend en argument un entier n, puis qui affiche les n premiers nombres premiers. Par exemple, premiers2(5) doit afficher 2, 3, 5, 7 et 11.