**Exercices 02**

**Mise en situation**

Créez un nouveau fichier python(**.py**), dans votre environnement, pour chacun des exercices. Ensuite, assurez-vous de respecter les bonnes pratiques.

* Ajoutez un Docstring
* Avoir des commentaires
* Le code est bien indenté
* Vos variables sont bien nommées
* Le code est bien espacé
* La longueur des lignes sont adéquates
* Aucun mot réservé n'a été utilisé

Aussi, n’oubliez pas d’ajouter en commentaire votre : Nom, date et description du programme.

**01 – Boucles de base**

Soit la liste : vache, souris, levure, bactérie. Affichez l'ensemble des éléments de cette liste (un élément par ligne) de trois façons différentes (deux méthodes avec for et une avec while).

**02 – Boucle et jours de la semaine**

Constituez une liste semaine contenant les 7 jours de la semaine.

Écrivez une série d'instructions affichant les jours de la semaine (en utilisant une boucle for), ainsi qu'une autre série d'instructions affichant les jours du week-end (en utilisant une boucle while).

**03 – Nombres de 1 à 10 sur une ligne**

Avec une boucle, affichez les nombres de 1 à 10 sur une seule ligne.

**04 – Nombres pairs et impairs**

Soit impaire la liste de nombres [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21]. Écrivez un programme qui, à partir de la liste impaire, construit une liste paire dans laquelle tous les éléments d’impair sont incrémentés de 1.

**05 – Produit de nombres consécutifs**

Avez les fonctions list() et range(), créez la liste entiers contenant les nombres entiers pairs de 2 à 20 inclus. Calculez ensuite le produit des nombres consécutifs deux à deux d’entiers en utilisant une boucle.

**06 – Triangle**

Créez un script qui dessine un triangle comme celui-ci :

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*

**07 – Triangle inversé**

Créez un script qui dessine un triangle comme celui-ci :

\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

**08 – Triangle gauche**

Créez un script qui dessine un triangle comme celui-ci :

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*

**09 – Pyramide**

Créez un script qui dessine une pyramide dessinée à partir d'un nombre arbitraire de lignes N. Vous pourrez demander à l'utilisateur le nombre de lignes de la pyramide avec la fonction input().

**10 – Table de multiplication**

Lire un nombre entier positif et afficher la table de multiplication de ce nombre.

Exemple : la table de multiplication du nombre 5 est

5 x 1 = 5 5 x 6 = 30 5 x 11 = 55

5 x 2 = 10 5 x 7 = 35 5 x 12 = 60

5 x 3 = 15 5 x 8 = 40 5 x 13 = 65

5 x 4 = 20 5 x 9 = 45 5 x 14 = 70

5 x 5 = 25 5 x 10 = 50 5 x 15 = 75

**11 – Jours de la semaine**

Constituez une liste semaine contenant les 7 jours de la semaine.

1. À partir de cette liste, comment récupérez-vous seulement les 5 premiers jours de la semaine d'une part, et ceux du week-end d'autre part ? Utilisez pour cela l'indexage.
2. Cherchez un autre moyen pour arriver au même résultat (en utilisant un autre indexage).
3. Trouvez deux manières pour accéder au dernier jour de la semaine.
4. Inversez les jours de la semaine en une commande.

**12 – Saisons**

Créez 4 listes hiver, printemps, été et automne contenant les mois correspondants à ces saisons. Créez ensuite une liste saisons contenant les listes hiver, printemps, été et automne. Prévoyez ce que renvoient les instructions suivantes, puis vérifiez-le dans l'interpréteur :

1. saisons[2]
2. saisons[1][0]
3. saisons[1:2]
4. saisons[:][1]. Comment expliquez-vous ce dernier résultat ?

**13 – Table de multiplication par 8**

Affichez la table de multiplication par 8 en une seule commande avec les instructions range() et list().

**14 – Nombres pairs**

Répondez à la question suivante en une seule commande. Combien y a-t-il de nombres pairs dans l'intervalle [2, 10000] inclus ?

**15 – Gestionnaire de tâches**

Écrivez un programme qui simule un gestionnaire de tâches. L'utilisateur peut ajouter des tâches, les afficher et les marquer comme terminées.