TEST PYTHON (BASE)

- Déposez votre devoir sur github, et quand vous avez fini, envoyez le lien en privée sur discord à YP-DS#7501.
- Nom du repository : ypds_test_python_base. (public)
- Utilisez un seul fichier jupyter notebook (.ipynb) pour tous les exercices.
- Essayez d'utiliser le type de cellule markdown pour les titres et explications.

Exercice 1 :

Créez un script qui permet aux utilisateur d'entrée une chaine de caractères correspond au jours de la semaine en utilisant une boucle, et créez une liste semaine contient les 7 jours de la semaine. Vérifiez chaque entrée de l'utilisateur :

√ Si l'utilisateur écrit 'quitter', interrompre la boucle et affichez 'Bey'

√ Si l'entrée fait partie du liste semaine. Vérifiez-la, et affichez le massage qui correspond :

  – Au travail s'il s'agit du lundi au jeudi
  – Chouette c'est vendredi s'il s'agit du vendredi
  – Repos ce week-end s'il s'agit du week-end.

√ Si l'entrée ne fait pas partie du liste semaine, affichez 'Entrée invalide'

Exercice 2 :

La liste ci-dessous représente une séquence d'ADN :
["A","C","G","T","T","A","G","C","T","A","A","C","G"]
Écrivez un script qui transforme cette séquence en sa séquence complémentaire. Rappel : la séquence complémentaire s'obtient en remplaçant A par T, T par A, C par G et G par C.

Exercice 3 :

Voici les notes d'un étudiant [14, 9, 13, 15, 12]. Écrivez un script qui affiche la note maximum (fonction max()), la note minimum (fonction min()) et qui calcule la moyenne. Affichez la valeur de la moyenne avec deux décimales. Affichez aussi la mention obtenue sachant que la mention est passable si la moyenne est entre 10 inclus et 12 exclus, assez-bien entre 12 inclus et 14 exclus et bien au-delà de 14.

Exercice 4 :

Initialisez deux entiers : a = 0 et b = 10.
Écrire une boucle affichant et incrémentant la valeur de a tant qu'elle reste inférieure à celle de b.
Écrire une autre boucle décrémentant la valeur de b et affichant sa valeur si elle est impaire. Boucler tant que b n'est pas 0.

Exercice 5 :

On désire sécuriser une enceinte pressurisée.
On se fixe une pression seuil et un volume seuil : pSeuil = 2.3, vSeuil = 7.41. On demande de saisir la pression et le volume courant de l'enceinte et d'écrire un script qui simule le comportement suivant :
– si le volume et la pression sont supérieurs aux seuils : arrêt immédiat ;
– si seule la pression est supérieure à la pression seuil : demander d'augmenter le volume de l'enceinte ;
– si seul le volume est supérieur au volume seuil : demander de diminuer le volume de l'enceinte ;
– sinon déclarer que « tout va bien ».
Ce comportement sera implémenté par une alternative multiple.

Exercice 6 :

Faites une boucle qui parcours les nombres de 0 à 100 et qui stocke les nombres pairs inférieurs ou égaux à 50 dans une liste appelée pairs_inf_50, et les nombres impairs strictement supérieur à 50 dans une autre liste appelée impairs_sup_50. Et puis, affichez les 2 listes.
Pour cet exercice, vous pourrez utiliser l'opérateur modulo % qui retourne le reste de la division entière entre deux nombres.

Exercice 7 :

Voici un extrait de l'article sur les nombres premiers tiré de l'encyclopédie en ligne wikipédia.
« Un nombre premier est un entier naturel qui admet exactement deux diviseurs distincts entiers et positifs (qui sont alors 1 et lui-même). Cette définition exclut 1, qui n'a qu'un seul diviseur entier positif. Par opposition, un nombre non nul produit de deux nombres entiers différents de 1 est dit composé. Par exemple 6 = 2 x 3 est composé, tout comme 21 = 3 x 7, mais 11 est premier car 1 et 11 sont les seuls diviseurs de 11.

Les nombres 0 et 1 ne sont ni premiers ni composés. »
Déterminez les nombres premiers inférieurs à 100. Combien y a-t-il de nombres premiers entre 0 et 100 ? Pour vous aider, nous vous proposons deux méthodes.
Méthode 1 (peu optimale mais assez intuitive)
Pour chaque nombre de 2 à 100, calculez le reste de la division entière (avec l'opérateur modulo %) depuis 1 jusqu'à lui-même. Si c'est un nombre premier, il aura exactement deux nombres pour lesquels le reste de la division entière est égal à 0 (1 et lui-même). Si ce n'est pas un nombre premier, il aura plus de deux nombres pour lesquels le reste de la division entière est égal à 0.
Méthode 2 (plus optimale et plus rapide, mais un peu plus compliquée)
Vous pouvez parcourir tous les nombres de 2 à 100 et vérifier si ceux-ci sont composés, c'est-à-dire qu'ils sont le produit de deux nombres premiers. Pratiquement, cela consiste à vérifier que le reste de la division entière (opérateur modulo %) entre le nombre considéré et n'importe quel nombre premier est nul. Le cas échéant, ce nombre n'est pas premier.

Exercice 8 :

Définir la liste : liste = [17, 38, 10, 25, 72], puis effectuez les actions suivantes :
– triez et affichez la liste ;
– ajoutez l'élément 12 à la liste et affichez la liste ;
– renversez et affichez la liste ;
– affichez l'indice de l'élément 17 ;
– enlevez l'élément 38 et affichez la liste ;
– affichez la sous-liste du 2e au 3e élément ;
– affichez la sous-liste du début au 2e élément ;
– affichez la sous-liste du 3e élément à la fin de la liste ;
– affichez la sous-liste complète de la liste ;

Exercice 9 :

Écrire une fonction compterMots ayant un argument (une chaîne de caractères) et qui renvoie un dictionnaire qui contient la fréquence de tous les mots de la chaîne entrée.
Ex : text = 'comment ça va !'
Le résultat de compterMots(text) devra {'c' : 1, 'o' : 1, 'm' : 2, 'e' : 1, 'n' : 1, 't' : 1, ' ' : 2, 'ç' : 1, 'a' : 2, 'v' : 1, ' !' : 1}

Exercice 10 :

Définir une classe Vecteur2D avec un constructeur fournissant les coordonnées par défaut d'un vecteur du plan (par exemple : x = 0 et y = 0).
Dans le programme principal, instanciez un Vecteur2D sans paramètre, un Vecteur2D

avec ses deux paramètres, et affichez-les.
3. Enrichissez la classe Vecteur2D précédente en lui ajoutant une méthode d’affichage et une méthode de surcharge d’addition de deux vecteurs du plan.
Dans le programme principal, instanciez deux Vecteur2D, affichez-les et affichez leur somme.