tps

May 3, 2025

TP : Prise en main du langage Python

Exercice 1

Écrire un programme pour calculer la moyenne de trois notes données par l'utilisateur.

Exercice 2

Écrire un programme tel que à partir de la distance en ***kilomètre*** et la durée en ***heure*** calcule la vitesse en ***km/h***.

Exercice 3

Écrire un programme pour calculer dans un espace de deux dimensions la distance entre deux points et leur point milieu.

(**Extra : 3 dimensions**)

Exercice 4

Écrire un programme pour calculer l'aire et le périmètre d'un cercle.

Exercice 5

Écrire un programme pour permuter les valeurs de deux variables.

Exercice 6

Écrire un programme pour convertir un nombre entier de secondes en un nombre entier d'heures, de minutes et de secondes.

TP : Instructions conditionnelles et répétitives

Exercice 1

Écrire un programme qui renvoie la parité d'un nombre donné.

Exercice 2

1) Pour un nombre entier positif x, calculer la somme des entier consécutifs de 1 à x.
2) Pour un nombre entier positif x, calculer sa factorielle.

Exercice 3

Écrire un programme pour :

1) Demander à l'utilisateur d'entrer des notes tel que tant qu'il y a des notes supplémentaires, ajouter-les au total.
2) Arrêter lorsque l'utilisateur saisit la valeur *-1*.
3) Calculer leur moyenne et afficher *admis avec la mention* si la moyenne >= 10 et *non admis* sinon.

Exercice 4

L'ordinateur tire un nombre aléatoire entre 0 et 100, l'utilisateur doit le trouver et pour cela propose des valeurs. L'ordinateur indique pour chaque valeur proposée si la valeur est trop petite, trop grande ou s'il a trouvé.

1) Écrire un programme pour jouer à ce jeu. En combien de coups est-on sûr de trouver?
2) Modifier le programme pour qu'il s'arrête si l'utilisateur n'a pas trouvé au bout du nombre de coups de la question 2.

Exercice 5

1) Écrire un programme pour tester si un nombre est premier.
2) Écrire un programme pour retourner tous les nombres premiers dans un interval donné.

Exercice 6

Écrire un programme qui, à partir d'un nombre entier positif, crée un triangle former des nombres tel que par exemple dans le cas de 10 :

0
0 1
0 1 2
0 1 2 3
0 1 2 3 4
0 1 2 3 4 5
0 1 2 3 4 5 6
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9 10

TP : Les fonctions

Exercice 1

Écrire une fonction ***celsius_vers_fahrenheit*** qui prend une température en Celsius et la convertit en Fahrenheit en utilisant la formule :

$$f = 9/5 \times c + 32$$

.

Exercice 2

Écrire une fonction ***est_pair*** qui prend un nombre et répond par *oui* si le nombre est pair et *non* sinon.

Exercice 3

Écrire une fonction ***appartient(bi, bs, n)*** qui teste si n appartient à l'intervalle $[bi, bs]$.

Exercice 4

Écrire une fonction qui prend un nombre x et calcule son image $f(x) = 1 + 1/x$. La fonction doit vérifier que $x \neq 0$.

Exercice 5

Compléter la fonction ***imc*** par l'affichage de la classe de la personne :

- en dessous de $18,5 kg/m^2$, on considère que la personne est maigre,
- entre 18,5 et $24,9 kg/m^2$, on considère que la personne a un poids normal,
- entre 25 et $29,9 kg/m^2$, on considère que la personne est en surpoids,
- au-dessus de $30 kg/m^2$, on considère que la personne est en obésité.

Exercice 6

Écrire une fonction ***comparer*** qui prend deux nombres et compare les deux.

Exercice 7

Créer une fonction ***factorielle*** qui calcule la factorielle d'un nombre.

Exercice 8

Créer une fonction ***premier*** qui teste la primalité d'un nombre.

TP : Les listes

Exercice 1

Soit liste = [0,1,2,3,5,8,13,21,34,55,89]

Donner les résultats des instructions suivantes :

- liste[1:7]
- liste[:]
- liste[-1:-5]
- liste[-5:-1]
- liste[:-2]
- liste[6:]
- liste[: : 3]
- liste[: : -3]

Exercice 2

Créer une liste des nombres impairs à partir de 1 et inférieurs à une borne donnée.

Exercice 3

Créer une fonction $cube(liste)$ qui renvoie une liste contenant le cube de chaque élément de la liste originale.

Exercice 4

Créer une fonction $somme(liste)$ qui renvoie la somme des éléments de la liste.

Exercice 5

Créer une fonction qui prend deux arguments : une lise et un élément, et compte les occurrences de cet élément dans la liste.

Exercice 6

- Créer une fonction qui renvoie le maximum (ou minimum) d'une liste de nombres donnée.
- Créer une fonction qui, en utilisant la fonction précédente, trie la liste dans l'ordre croissant (ou décroissant) itérativement (ou récursivement).

Exercice 7

Créer une fonction qui fusionne deux listes triées dans l'ordre croissant.

Exercice 8

Créer une fonction $ProduitScalaire(V1, V2)$ qui renvoie le produit scalaire de deux vecteurs de même taille $V1$ et $V2$.

Exercice 9

Créer une fonction $ProduitMatriciel(M1, M2)$ qui renvoie le produit de deux matrices $M1$ et $M2$.

Exercice 10

Un polynôme peut être représenté par une liste dont les éléments sont les coefficients et les indices sont les exposants.

Exemple : Le polynôme $P(x) = x^3 + 2x + 7$ est représenté par P=[7, 2, 0, 1].

- Créer une fonction $valeur(P, x)$ qui calcule $P(x)$.
- Créer une fonction $addition(P1, P2)$ qui retourne la représentation de l'addition des deux polynômes représentés respectivement par $P1$ et $P2$.

Exercice 11

Gestion des élèves et leurs notes :

On dispose d'une liste d'élèves, où chaque élément est une sous-liste contenant l'identifiant d'élève et ses notes de trois matières.

Exemple de données : students = [['student1', 20, 18, 19], ['student2', 12, 14, 16], ['student3', 9, 10, 11]]

- Créer une liste d'élèves avec plusieurs et différentes valeurs.
- Calculer et ajouter la moyenne de chaque élève dans la liste.
- Parcourir la liste et afficher pour chaque élève son identifiant et sa note moyenne.
- Calculer la moyenne de la classe.
- Afficher les élèves dont la moyenne >= la moyenne de la classe.
- Trouver le meilleur élève.

TP : Les chaînes de caractères

Exercice 1

Écrire un programme qui compte le nombre de mots dans une phrase (les mots sont séparés par des espaces).

Exercice 2

Écrire une fonction qui prend un texte et le renvoie inversé.

Exercice 3

Un palindrome est un texte qui se lit de la même manière de gauche à droite et de droite à gauche comme "*radar*" et "*ressasser*". Écrire une fonction qui prend un texte et vérifie s'il s'agit d'un palindrome.

Exercice 4

Le chiffrement de César consiste à décaler chaque lettre d'un texte d'un certain nombre de positions dans l'alphabet.

- Écrire une fonction qui chiffre un texte en décalant chaque lettre d'un nombre donné de positions. Il faut faire correspondre, pour chaque lettre, la valeur de son rang dans l'alphabet, à partir de 0 : a=0, b=1, …, z=25.
- Ajouter une fonction pour déchiffrer le texte en inversant le décalage.
- Brute force sur le chiffrement de César : Un texte chiffré avec César peut être facilement cassé en testant tous les décalages possibles. Écrire un programme qui prend un message chiffré par César et affiche toutes les versions possibles.

Exercice 5

Le chiffrement de Vigenère consiste à additionner un texte clé au texte clair. Après avoir fait correspondre la longueur du texte à la clé, on additionne le rang de chaque lettre du texte au rang de la lettre correspondante dans la clé. Le résultat est le rang de la lettre chiffrée.

- Écrire une fonction qui renvoie le chiffrement de Vigenère d'un texte donné.
- Ajouter une fonction pour déchiffrer le texte.

Exercice 6

Compresser une chaîne en notation RLE (Run-Length Encoding) : L'encodage RLE consiste à remplacer une séquence répétée de caractères par un seul caractère suivi du nombre d'occurrences.

Exemple : "*haaalloo*" devient "$h1a3l2o2$".

Écrire une fonction qui prend un texte et le compresse en utilisant cette technique.

TP : Les dictionnaires

Exercice 1

Écrire une fonction *compteur*(*texte*) qui prend une chaîne de caractères et renvoie un dictionnaire avec le nombre d'occurrences de chaque caractères dans le texte.

Exemple : compteur('bla bla') renvoie {'b' : 2, 'l' : 2, 'a' : 2, ' ' : 1}.

Exercice 2

Écrire une fonction *maximum*(*dictionnaire*) qui renvoie la clé associée à la valeur maximale d'un dictionnaire.

Exemple : maximum({'x1':20, 'x2':15, 'x3':17}) renvoie x1.

Exercice 3

Écrire une fonction *tri*(*dictionnaire*) qui renvoie une liste des clés triées en fonction des valeurs de manière croissante.

Exemple : tri({'x1':20, 'x2':15, 'x3':17}) renvoie ['x2', 'x3, 'x1'].

Exercice 4

Afin de traiter l'évolution des populations de différents pays entre 1977 et 2007 on utilise un dictionnaire contenant pour chaque pays une liste des populations en différentes années.

Extrait du dictionnaire :

population = {

'Morocco' : [18396941, 20198730, 22987397, 25798239, 28529501, 31167783, 33757175],

'Palestine (Gaza)' : [1261091, 1425876, 1691210, 2104779, 2826046, 3389578, 4018332],

'Indonesia' : [136725000, 153343000, 169276000, 184816000, 199278000, 211060000, 223547000],

'Iraq' : [11882916, 14173318, 16543189, 17861905, 20775703, 24001816, 27499638],

'Spain' : [36439000, 37983310, 38880702, 39549438, 39855442, 40152517, 40448191],

'United States' : [220239000, 232187835, 242803533, 256894189, 272911760, 287675526, 301139947]

}

On donne la liste des années de recensement : années = [1977, 1982, 1987, 1992, 1997, 2002, 2007].

- Écrire une fonction *recensement*($pays, an$) qui prend un pays et une année et affiche la population de ce pays en cette année.
- Écrire une fonction *recensement_maximal*(*pays*) qui prend un pays et renvoie l'année associée à la population maximale dans ce pays.
- Écrire une fonction *moyenne*(*pays*) qui prend un pays et calcule et renvoie la population moyenne de ce pays.

- Écrire une fonction $total(an)$ qui prend une année et renvoie la population totale de tous les pays en cette année.

TP : Recherche séquentielle et dichotomique

- Créer une fonction *recherche_séquentielle* qui prend une liste et une valeur, et renvoie $True$ si cette valeur se trouve dans la liste, et $False$ sinon.
- Créer une fonction, itérative puis récursive, *recherche_dichotomique* qui prend une liste triée et une valeur, et renvoie $True$ si cette valeur se trouve dans la liste, et $False$ sinon.
- Comparer le nombre d'itérations dans les deux fonctions (séquentielle et dichotomique itérative) sur une liste et ce dans le pire des cas (l'élément recherché se trouve dans la fin de la liste)

TP : Pile et File

Dans cette suite d'exercices, on n'utilisera que les primitives de pile et de file vues précédemment.

Exercice 1

1. Écrire une fonction $hauteur(pile)$ qui renvoie la hauteur d'une pile donnée.
2. Écrire une fonction $copie(pile)$ qui permet de créer et renvoyer une copie d'une pile donnée.
3. Écrire une fonction $application(pile, fonction)$ qui prend une pile et une fonction et renvoie la pile formée des images des éléments de la pile par la fonction.

Exercice 2

1. Écrire une fonction $hauteur(file)$ qui renvoie la hauteur d'une file donnée.
2. Écrire une fonction $renverser(file)$ qui renverse une file donnée.
3. Écrire une fonction $rotation(file, pas)$ qui effectue une rotation de la file d'un nombre de pas.

Exercice 3

Écrire une fonction $valuer(expression)$ qui évalue une expression postfixée (notation polonaise inversée) en utilisant une pile.

Exemple : '3 4 + 2 -' renvoie 5

Exercice 4

Écrire une fonction $bien\_forme(expression)$ qui utilise une pile pour vérifier si les parenthèses dans une expression sont bien ouvertes et fermées.

Une expression est bien parenthésée, si à tout moment le nombre de parenthèses ouvrantes est supérieur ou égal au nombre de parenthèses fermantes avec égalité en fin d'expression.

1. On parcourt l'expression de gauche à droite;
2. On empile pour chaque '(' rencontrée;
3. En rencontrant une ')' on essaye de dépiler; dans ce cas, si la pile est vide, alors l'expression n'est pas bien parenthésée (la première condition n'est pas vérifiée), sinon on continue.
4. Quand le parcours termine, si la pile est vide l'expression est bien parenthésée, sinon elle ne l'est pas (la deuxième condition n'est pas vérifiée).

TP : Prise en main du langage Python

Exercice 1

Écrire un programme pour calculer la moyenne de trois notes données par l'utilisateur.

Exercice 2

Écrire un programme tel que à partir de la distance en ***kilomètre*** et la durée en ***heure*** calcule la vitesse en ***km/h***.

Exercice 3

Écrire un programme pour calculer dans un espace de deux dimensions la distance entre deux points et leur point milieu.

(**Extra : 3 dimensions**)

Exercice 4

Écrire un programme pour calculer l'aire et le périmètre d'un cercle.

Exercice 5

Écrire un programme pour permuter les valeurs de deux variables.

Exercice 6

Écrire un programme pour convertir un nombre entier de secondes en un nombre entier d'heures, de minutes et de secondes.

TP : Instructions conditionnelles et répétitives

Exercice 1

Écrire un programme qui renvoie la parité d'un nombre donné.

Exercice 2

Écrire un programme qui demande deux nombres et compare les deux.

Exercice 3

Écrire un programme qui demande les deux bornes d'un intervalle puis un nombre et teste si ce dernier appartient à cet intervalle.

Exercice 4

Écrire un programme qui affiche les nombres impairs de 1 à une borne donnée.

Exercice 5

Écrire un programme tel que pour un nombre entier positif x,

1) calculer la somme des entier consécutifs de 1 à x.
2) calculer sa factorielle.

Exercice 6

Écrire un programme pour :

1) Demander à l'utilisateur d'entrer des notes tel que tant qu'il y a des notes supplémentaires, ajouter-les au total.
2) Arrêter lorsque l'utilisateur saisit la valeur *-1*.
3) Calculer leur moyenne et afficher *admis* si la moyenne >= 10 et *non admis* sinon.

Exercice 7

L'ordinateur tire un nombre aléatoire entre 1 et 8, l'utilisateur doit le trouver et pour cela propose des valeurs. L'ordinateur indique pour chaque valeur proposée si la valeur est trop petite, trop grande ou s'il l'a trouvé.

1) Écrire un programme pour jouer à ce jeu.
2) En combien de coups est-on sûr de trouver le nombre?
3) Modifier le programme pour qu'il s'arrête si l'utilisateur n'a pas trouvé au bout du nombre de coups de la question 2.

TP : Les fonctions

Exercice 1

Écrire une fonction ***celsius_vers_fahrenheit*** qui prend une température en Celsius et la convertit en Fahrenheit en utilisant la formule :

$$f = 9/5 \times c + 32$$

.

Exercice 2

Écrire une fonction ***est_pair*** qui prend un nombre et répond par *oui* si le nombre est pair et *non* sinon.

Exercice 3

Écrire une fonction ***appartient(bi, bs, n)*** qui teste si n appartient à l'intervalle $[bi, bs]$.

Exercice 4

Écrire une fonction qui prend un nombre x et calcule son image $f(x) = 1 + 1/x$ en vérifiant d'abord que $x \neq 0$.

Exercice 5

Compléter la fonction ***imc*** par l'affichage de la classe de la personne :

- en dessous de $18,5 kg/m^2$, on considère que la personne est maigre,
- entre 18,5 et $24,9 kg/m^2$, on considère que la personne a un poids normal,
- entre 25 et $29,9 kg/m^2$, on considère que la personne est en surpoids,
- au-dessus de $30 kg/m^2$, on considère que la personne est en obésité.

Exercice 6

Écrire une fonction ***comparer*** qui prend deux nombres et compare les deux.

Exercice 7

Créer une fonction ***factorielle*** qui calcule la factorielle d'un nombre.

TP : Les listes

Exercice 1

Soit liste = [0,1,2,3,5,8,13,21,34,55,89]

Donner les résultats des instructions suivantes :

- liste[1:7]
- liste[:]
- liste[3:1]
- liste[-3:-1]
- liste[:-2]
- liste[6:]
- liste[::3]
- liste[::-3]

Exercice 2

Créer une liste des nombres impairs à partir de 1 et inférieurs à une borne donnée.

Exercice 3

Créer une fonction $cube(liste)$ qui renvoie une liste contenant le cube de chaque élément de la liste originelle.

Exercice 4

Créer une fonction $somme(liste)$ qui renvoie la somme des éléments de la liste.

Exercice 5

Créer une fonction $recherche\_séquentielle(liste, valeur)$ qui renvoie l'indice de la première occurrence de la valeur dans la liste si elle existe, et -1 sinon.

Exercice 6

Créer une fonction $recherche\_dichotomique(liste, valeur)$ qui prend une liste triée et une valeur et renvoie $True$ si la valeur se trouve dans la liste, et $False$ sinon.

Principe :

On divise la liste en deux sous-listes et on compare l'élément du milieu de la liste avec la valeur recherchée :

- Si c'est la bonne valeur, on renvoie $True$.
- Si la valeur est plus petite, on recherche dans la sous-liste gauche.
- Si la valeur est plus grande, on recherche dans la sous-liste droite.

On recommence jusqu'à trouver l'élément ou jusqu'à ce que la sous-liste soit vide et on renvoie $False$.

Exercice 7

Créer une fonction $nombre\_occurences(valeur, liste)$ qui compte les occurrences d'une valeur dans une liste.

Exercice 8

Créer une fonction $maximum(liste)$ qui renvoie le maximum d'une liste de nombres.

TP : Graphisme en deux dimensions

Exercice 1

Soit la fonction définie par : $\forall x \in \mathbb{R}, f(x) = \frac{e^x - e^{-x}}{2}$.

1. Créer un vecteur X qui contient 100 abscisses uniformément distribuées sur l'intervalle $[-3, 3]$.
2. Créer un autre vecteur qui contient l'image de X par f.
3. Tracer la courbe de f.
4. Tracer sans faire de calcul f^{-1} la fonction réciproque de f.

Exercice 2

Soit la fonction définie par : $\forall x \in \mathbb{R}, f(x) = x - ln(1 + |x|)$

1. Tracer le graphe de f sur l'intervalle $[-1, 1]$ dans un repère muni d'une grille.
2. Déduire graphiquement la dérivabilité de f en 0.

TP : Les suites

Exercice 1

Soit $(U_n)_{n\in\mathbb{N}}$ une suite explicite définie par : $U_n = \frac{1}{2^n}$.

1. Créer une fonction qui calcule le $n^{ème}$ terme de la suite.
2. Créer une liste contenant les termes d'indices 0 à une borne choisie.
3. Tracer graphiquement la suite.
4. Calculer la limite de la suite et vérifier le résultat graphiquement.

Exercice 2

Soit $(U_n)_{n\in\mathbb{N}}$ une suite définie par : $U_n = (n-3)^2$.

1. Créer une fonction qui calcule le $n^{ème}$ terme de la suite.
2. Créer une liste contenant les termes d'indices 0 à une borne choisie.
3. Tracer graphiquement la suite.
4. Étudier la monotonie de la suite et vérifier le résultat graphiquement.

Exercice 3

Une usine assure, en 2025, une production de 100 000 articles. Elle s'engage à augmenter sa production de 3% chaque année.

1. Quelle est la nature de cette suite production?
2. Créer une fonction qui renvoie la production en une année donnée.
3. Créer une liste de nombres d'articles produits entre 2025 et 2035.
4. Représenter graphiquement la production entre 2025 et 2035.
5. Combien d'articles auront été fabriqués dans cette période?

Exercice 4

Soit la suite de Fibonacci $(F_n)_{n\in\mathbb{N}}$ définie par :

$$\begin{cases} F_0 = 0 \\ F_1 = 1 \\ F_{n+2} = F_{n+1} + F_n \end{cases}$$

1. Créer une fonction pour calculer le $n^{ème}$ terme de la suite.
2. Stocker dans une liste les $\frac{F_n}{F_{n-1}}$ pour $n \in [1, 20]$.
3. Tracer graphiquement cette suites de rapports.
4. Tracer la ligne d'équation $f(x) = \frac{1+\sqrt{5}}{2}, \forall x \in \mathbb{R}$ (Le nombre d'or).
5. Que peut-on en déduire?