TP2 : exercices sur les boucles

A l'aide de la fiche de synthèse établir l'algorithme des différents exercices ci-dessous puis valider votre programme en langage python.

Exercice 1

Ecrire un programme qui affiche des nombres entiers de 0 à 15.

Algorithme pseudo-code	Python
pour n de 1 à 15 par pas de 1 faire afficher(n) finpour	

Exercice 2

Ecrire un programme qui affiche des nombres entiers de 15 à 0.

Algorithme pseudo-code	Python
pour n de 15 à 0 par pas de -1 faire afficher(n) finpour	

Exercice 3

Refaire l'exercice 1 en demandant à l'utilisateur deux valeurs entières debut et fin. Ecrire un programme qui affiche des nombres entiers entre les variables debut et fin (inclus).

Aide : Interaction avec l'utilisateur : la fonction input(), voir annexe en dernière page.

Algorithme pseudo-code	Python
afficher("votre valeur de debut") lire debut afficher("votre valeur de fin") lire debut pour n de debut à fin par pas de 1 faire afficher(n) finpour	

Exercice 4

Ecrire un programme qui demande un nombre de départ, et qui ensuite affiche les dix nombres suivants.

Algorithme pseudo-code	Python
afficher("votre valeur de debut") lire debut pour n de debut+1 à debut+10 par pas de 1 faire afficher(n) finpour	

Exercice 5

Ecrire un programme qui demande un nombre de départ (opérande), et qui ensuite écrit la table de multiplication de ce nombre.

Exemple pour la table de 4 :

4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
4 * 10 = 40

Algorithme pseudo-code	Python
afficher("votre table de multiplication") lire operande #a compléter finpour	

Exercice 6

Ecrire un programme qui demande successivement 5 nombres positifs à l'utilisateur, et qui lui dise ensuite quel était le plus grand (max) parmi ces 5 nombres :

Algorithme pseudo-code	Python
pour n de 1 à 5 par pas de 1 faire afficher("votre table de multiplication") lire operande #a compléter finpour afficher("le nombre le plus grand est",max)	

Exercice 7

Réécrire l'algorithme précédent, mais cette fois-ci on ne connaît pas d'avance combien l'utilisateur souhaite saisir de nombres. La saisie des nombres s'arrête lorsque l'utilisateur entre un chiffre négatif.

Algorithme pseudo-code	Python
valeur←0 max←0 Tant que valeur>=0 # a compléter fin tant que afficher("le nombre le plus grand est",max)	

Exercice 8

Ecrire un programme qui demande des nombres positifs à l'utilisateur, et effectue la somme de tous ces nombres. On ne connaît pas d'avance combien l'utilisateur souhaite saisir de nombres. La saisie des nombres s'arrête lorsque l'utilisateur entre un chiffre négatif.

Algorithme pseudo-code	Python
valeur←0 somme←0 tant que valeur>=0 fin tant que afficher("la somme des nombres est",somme)	

Exercice 9

Programmation d'un menu. Traduire le pseudo code en python.

Algorithme pseudo-code	Python
menu←'0' Tant que menu≠'q' afficher("1 : charger le fichier") afficher ("2 : sauvegarder le fichier") afficher ("3 : afficher les données") afficher ("4 : modifier les données") afficher ("q : quitter") afficher("votre choix ?") lire menu si menu='1' alors afficher("Chargement") sinon si menu='2' alors afficher("Sauvegarde") sinon si menu='3' alors afficher("Affichage") sinon si menu='4' alors afficher("modification") sinon si menu='q' alors afficher("Au revoir") sinon afficher("erreur") fin si fin si fin si fin si finsi fin tant que	

Exercice 10

Programmation du jeu « deviner un nombre ».

L'ordinateur choisit un nombre entier entre 1 et 20. Le programme demande à l'utilisateur un chiffre.
Si celui-ci est supérieur au nombre à deviner, on affiche ("chiffre trop grand") ;
Si celui-ci est inférieur au nombre à deviner, on affiche ("chiffre trop petit") ;
Dans les deux cas on redemande un autre chiffre jusqu'à ce que l'utilisateur ait trouvé le nombre à deviner.

Aide : Générer un nombre aléatoire, voir annexe en dernière page.

Algorithme pseudo-code	Python
nombre←tirage aléatoire entre (1 et 20) valeur←0 tant que valeur ≠ nombre afficher("votre chiffre") Lire valeur si valeur>nombre alors print("chiffre trop grand") sinon si valeur<nombre alors print("chiffre trop petit") sinon print("exact") fin si fin si fin tant que	

Exercice 11

Amélioration du jeu « deviner un nombre ». Reprendre le jeu et ajouter un compteur de coups.

Exemple d'affichage : « Exact, vous avez trouvé en 5 coups »

Algorithme pseudo-code	Python
nombre←tirage aléatoire entre (1 et 20) valeur←0 coups←0 tant que valeur ≠ nombre afficher("votre chiffre") Lire valeur fin tant que	

Exercice 12

Amélioration du jeu « deviner un nombre ».

Reprendre le jeu et demander à l'utilisateur à la fin de la partie s'il veut rejouer. Répondre par la lettre O pour oui et N pour non. Si l'utilisateur saisit une autre lettre que O ou N, il faudra alors lui reposer la question.

Aide	Python
Etude de la boucle tant que suivante : **while rep!='O' and rep!='N':** **instructions** 1) **rep='Z'** **rep!='O'** / **rep!='N'** / **rep!='O' and rep!='N'** **true** / **true** / **true** **On reste dans la boucle** 1) **rep='O'** **rep!='O'** / **rep!='N'** / **rep!='O' and rep!='N'** **false** / **true** / **false** **On sort de la boucle**	

Exercice 13

Rappel du programme de seconde :

Dans un repère orthonormé on considère les points A (xA ; yA) et B (xB ; yB). La distance entre les points A et B est :

$$AB = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$$

Le programme demande à l'utilisateur 4 valeurs décimales xA, yA, xB, yB, puis calcul la distance entre les points A et B.

Aide : utiliser une racine carrée : la fonction sqrt(), voir annexe en dernière page.

Algorithme pseudo-code	Python
afficher("xA ?") lire A afficher("yA ?") lire yA afficher("xB ?") lire xB afficher("yB ?") lire yB $AB \leftarrow \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$ afficher(AB)	

Annexe

Interaction avec l'utilisateur : la fonction input()

Instructions :	Résultat :
`chaine=input("saisir une chaine de caractère")` `print(type(chaine))`	`<class 'str'>`
`nombre=int(input("saisir un nombre entier"))` `print(type(nombre))`	`<class 'int'>`
`nombre=float(input("saisir un nombre décimal"))` `print(type(nombre))`	`<class 'float'>`

Notez la différence de type de variable :

- str : Chaine de caractère,
- int : entier,
- float : nombre décimal.

Générer un nombre aléatoire

Instructions :	Résultat :
`from random import *` `nombre = randint(1,20)`	Génère un nombre **entier** aléatoire entre 0 et 20. Exemple : 15
`x = uniform(12, 18)` `print(x)`	Génère un nombre **décimal** aléatoire entre 12 et 18. Exemple : **`14.271572580135519`**

Utiliser une racine carrée

Instructions :	Résultat :

`from math import sqrt`
`a=2`
`b=sqrt(a)`
`print(b)`
	1.4142135623730951