Exercices d'entrainement

(largement inspirés de http://www.france-ioi.org/)

1 - **Bases de la programmation**

Exercice 1 : Les quels sont faux ?

Parmi les sept programmes ci-dessous, lesquels sont justes (sans erreur de syntaxe) et lesquels sont faux (avec erreur de syntaxe) ?

1.

```
nbBillesRouges = 4
nbBillesBleues = 6
print(nbBillesRouges + nbBillesBleues)
```

2.

```
taille = 4
taille = 6
print(taille)
```

3.

```
nbBillesBleues = 3
print(nbBillesRouges)
```

4.

```
2 = 2
print(2)
```

5.

```
âge1 = 6
âge2 = 4
âge2 = âge1
print(âge2)
```

6.

```
prix = 10
prix - 2 = prix
print(prix)
```

7.

```
prix = âge - 7
âge = 12
print(prix)
```

2 - Sturcture IF et algèbre de Boole

Exercice 1 :

Ce que doit faire votre programme :

Écrivez un programme qui lit un numéro de mois d'un calendrier fictif, et affiche le nombre de jours de celui-ci. Voici les informations dont vous avez besoin :

Numéro du mois	Nombre de jours
1	30
2	30
3	30
4	31
5	31
6	31
7	30
8	30
9	30
10	31
11	29

Exercice 2 :

Vous devez écrire un programme qui détermine si deux soldats ont été de garde en même temps.

Votre programme doit lire quatre entiers (renseignés par l'utilisateur) : la date du début et la date de fin (incluse) du service du premier soldat puis celles du second soldat.

Si les deux soldats ont, à un moment (même une seule seconde), été de garde en même temps le programme devra écrire "Amis" et sinon "Pas amis".

Exercice 3 :

L'auberge dans laquelle vous vous arrêtez pour la nuit adapte ses prix en fonction de l'âge du client et du poids de ses bagages. Les règles ne vous étant pas très claires, vous décidez d'écrire un petit programme qui vous permettra facilement, à vous et à vos compagnons de voyage, de connaître le prix d'une nuit.

Ce que doit faire votre programme :

Une chambre ne coûte rien si on a 60 ans (l'âge de l'aubergiste !) et 5 écus si on a strictement moins de 10 ans. Pour les autres personnes c'est 30 écus plus un supplément de 10 écus si on a au moins 20 kilos de bagages.

Votre programme doit lire deux entiers, l'âge et le poids des bagages de la personne et doit afficher le prix, sous la forme d'un entier.

Exemple

entrée :

```
22
25
```

sortie :

```
40
```

Exercice 4 :

Alors que vous traversez une forêt vous ne pouvez vous empêcher d'admirer la végétation autour de vous et notamment les nombreuses espèces d'arbres. Malgré votre intérêt, vous êtes très mauvais botaniste et avez beaucoup de mal à identifier les différents arbres. Une personne que vous croisez vous donne quelques indications et vous décidez d'écrire un programme qui vous donnera le nom de l'arbre en fonction de ses caractéristiques.

Ce que doit faire votre programme :

Il existe 4 types d'arbres :

- le "Tinuviel" fait moins de 5 mètres de haut et ses feuilles sont composées de plus de 8 folioles
- le "Calaelen" fait plus de 10 mètres de haut et ses feuilles sont composées de plus 10 folioles
- le "Falarion" fait moins de 8 mètres de haut et ses feuilles sont composées de moins de 5 folioles
- le "Dorthonion" fait plus de 12 mètres de haut et ses feuilles sont composées de moins de 7 folioles

Votre programme lira deux entiers, la hauteur et le nombre de folioles de l'arbre, et affichera le nom de l'arbre correspondant.

Toutes les inégalités sont à prendre au sens large, c'est-à-dire que "moins" signifie "moins ou égal" ou et "plus" signifie "plus ou égal".

Exemples

Exemple 1

entrée :

```
12
12
```

sortie :

```
Calaelen
```

Exemple 2

entrée :

```
4
9
```

sortie :

```
Tinuviel
```

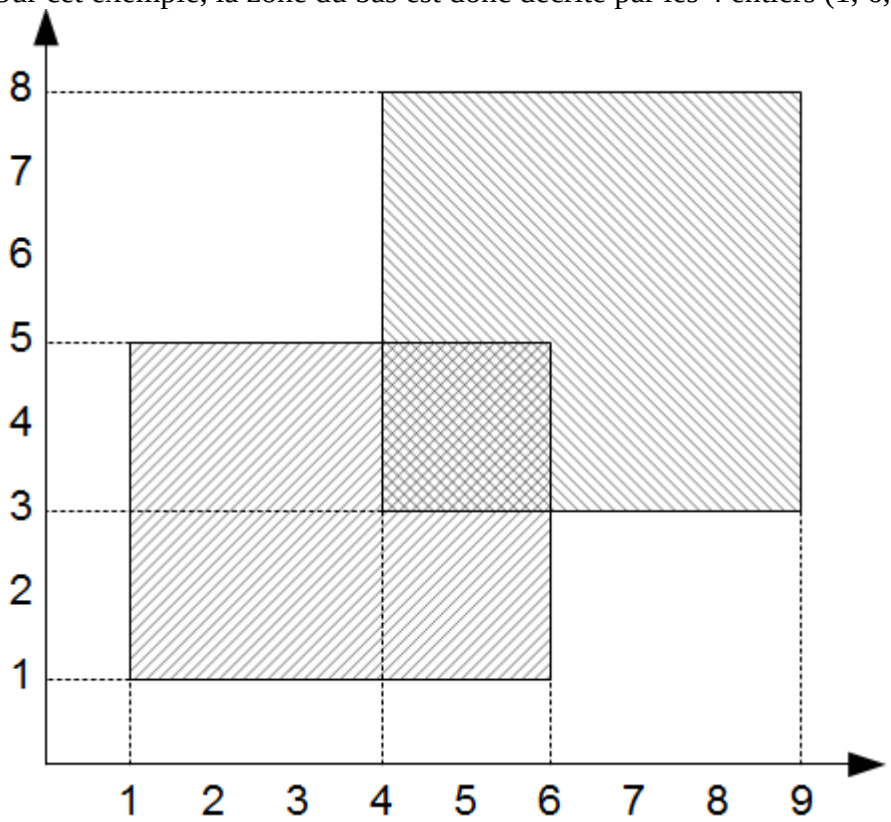
Exercice 5 :

Ce que doit faire votre programme :

Votre programme doit lire la description de plusieurs paires de zones rectangulaires, et pour chacune, déterminer si les deux rectangles s'intersectent.

Vous devez lire un premier entier, le nombre de paires de zones que votre programme devra tester. Ensuite, pour chaque paire possible, deux zones rectangulaires et parallèles aux axes vous sont données l'une après l'autre. Chaque zone est décrite par 4 entiers : son abscisse minimale et maximale puis son ordonnée minimale et maximale.

Sur cet exemple, la zone du bas est donc décrite par les 4 entiers (1, 6, 1, 5) et l'autre par (4, 9, 3, 8) :

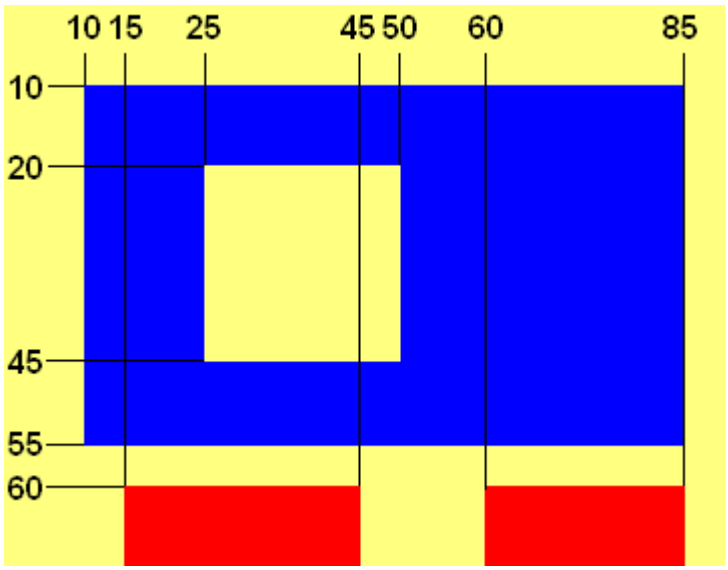


Pour chaque paire de zones, votre programme doit écrire "OUI" si les zones s'intersectent et "NON" sinon. Si elles ne font que se toucher sur les bords il doit écrire "NON".

Exercice 6 :

Ce que doit faire votre programme :

Sur une table est placée une feuille de papier rectangulaire de 90 cm de large et 70 cm de haut, composée de zones de différentes couleurs, comme le décrit la figure ci-dessous. Un certain nombre de personnes placent l'une après l'autre un jeton où elles le souhaitent sur la table, à l'exception des frontières entre les différentes zones.



On vous donne en entrée le nombre de jetons qui ont été déposés, puis, pour chaque jeton, ses coordonnées sur la feuille par rapport à l'origine en haut à gauche, sous la forme d'une abscisse et d'une ordonnée entre −1 000 et 1 000.

Votre programme devra qualifier chaque jeton avec l'un des textes suivants, en fonction de la couleur sur laquelle il se trouve :

- « En dehors de la feuille »
- « Dans une zone jaune »
- « Dans une zone bleue »
- « Dans une zone rouge »

Essayez d'écrire votre programme de sorte qu'il y ait au maximum une condition par possibilité de texte affiché.

Exemple

entrée :

```
4
16
12
30
22
64
62
-5
86
```

sortie :

```
Dans une zone bleue
Dans une zone jaune
Dans une zone rouge
En dehors de la feuille
```

Commentaires

Dans l'exemple, on a 4 jetons, de coordonnées (16 ; 12), (30 ; 22), (64 ; 62) et (-5 ; 86).

3 - **Boucle for**

Exercice 1 : Répétition

Votre programme doit écrire 135 fois la phrase : "Je dois aussi travailler les maths.", en plaçant cette phrase exactement une fois sur chaque ligne.

Important : votre programme ne doit pas faire plus de 5 lignes.

Exercice 2 : Invasion de batraciens

Ce que doit faire votre programme :

Sachant qu'il y a actuellement 1337 crapauds et que leur nombre double chaque semaine, votre programme devra afficher le nombre de crapauds qu'il y aura après la 12^{e} semaine.

Important : vous devez **utiliser une boucle** pour calculer le nombre de crapauds.

Exercice 3 : Compte à rebours

Votre programme devra lancer le décompte en partant de 100 puis annoncer le décollage, c'est-à-dire afficher une séquence d'annonces de la forme :

```
↳
100
99
...
2
1
0
Décollage !
```

en remplaçant les « … » par toutes les valeurs intermédiaires.

Exercice 4 : Calcul de distance 1

Votre programme doit d'abord lire un entier strictement positif, le nombre de jours de marche effectués jusqu'à présent. Il doit ensuite lire, pour chaque jour, la distance parcourue ce jour-là. Il doit alors afficher la distance totale parcourue sur tout le voyage.

Exemple

entrée :

```
6
36
14
38
28
29
31
```

sortie :

```
176
```

Exercice 5 : Calcul de distance 2

Votre programme doit d'abord lire un entier strictement positif, le nombre de jours de marche effectués jusqu'à présent. Il doit ensuite lire, pour chaque jour, la distance parcourue ce jour-là. Il doit alors afficher la distance maximale parcourue en une journée.

Exemple

entrée :

```
6
36
14
38
28
29
31
```

sortie :

```
38
```

Exercice 6 : Calcul de dénivelé

Votre programme lira d'abord un entier représentant le nombre de montées et descentes que vous avez réalisées. Pour chaque montée ou descente, il faut ensuite lire un entier représentant la variation d'altitude, cet entier étant strictement positif dans le cas d'une montée et strictement négatif dans le cas d'une descente (il n'y a rien à compter pour les tronçons qui sont bien à plat). Votre programme devra afficher l'altitude totale montée puis l'altitude totale descendue (ces deux nombres sont positifs).

Exemple

entrée :

```
5
4
7
-6
-3
2
```

sortie :

```
13
9
```

Exercice 7 : La muraille

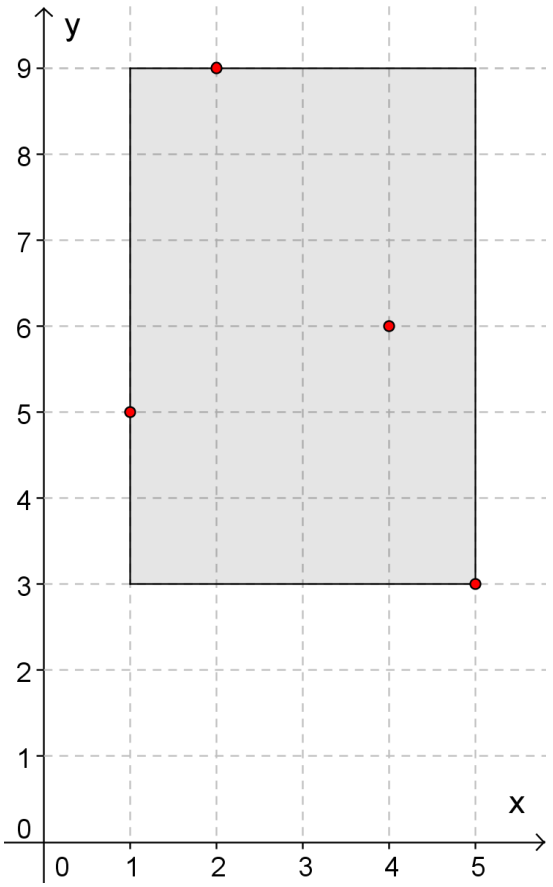
Le village dans lequel vous avez passé la nuit est en pleine effervescence au matin : encore une attaque de loups pendant la nuit !

C'est décidé, il va falloir construire une grande palissade tout autour du village. Les habitants insistent pour que cette clôture soit rectangulaire et ait une face au Nord, une au Sud, une à l'Est et une à l'Ouest, quitte à devoir travailler un peu plus que nécessaire. Ils ont maintenant besoin de votre aide pour savoir la quantité de bois dont ils vont avoir besoin pour construire cette palissade.

Ce que doit faire votre programme :

Le programme doit d'abord lire un entier strictement positif correspondant au nombre de maisons. Ensuite, pour chaque maison, il doit lire la position horizontale (l'abscisse, le "x") et sa position verticale (l'ordonnée, le "y") de cette maison. Toutes les abscisses et ordonnées sont des entiers compris entre zéro et 1 million.

Le programme doit alors afficher le périmètre de la plus petite clôture rectangulaire englobant toutes les maisons. Ce rectangle doit avoir ses côtés parallèles aux axes du repère, comme montré sur l'illustration.



Représentation graphique du premier exemple

Exemple

entrée :

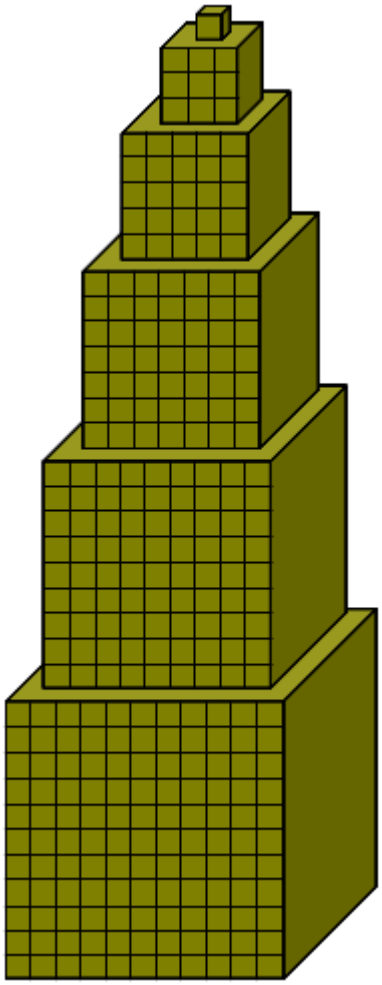
```
4
1
5
```

```
5
3
4
6
2
9
```

sortie :

```
20
```

4 - Transition : du for au while



Les enfants de la classe de maternelle décident de construire une très grande tour à l'aide de petits cubes en bois. Ils savent exactement la forme qu'ils souhaitent pour leur tour, mais ils n'arrivent pas à savoir s'ils auront suffisamment de cubes pour la construire. Ils vous demandent de les aider à calculer le nombre de cubes nécessaires.

Ce que doit faire votre programme :

L'objectif est de construire une tour à l'aide de petits cubes en bois, sachant que la forme de cette tour consiste en un ensemble de grands cubes placés les uns au-dessus des autres. La base de la tour est un cube de taille 17×17×17, c'est-à-dire composé de 17×17×17 = 4 913 petits cubes. Sur ce cube est posé un autre cube de taille 15×15×15. Au-dessus de ce dernier se trouve un cube de 13×13×13. La tour continue ainsi jusqu'à atteindre le sommet, qui consiste en un cube de taille 1×1×1.

Exemple d'une tour allant de 1×1×1 cubes à 11×11×11 cubes

Q1.

Votre programme doit calculer et afficher le nombre total de petits cubes nécessaires pour construire la pyramide. Effectuez les calculs dans le programme en y intégrant une boucle.

Q2.

Votre programme doit déterminer et afficher le nombre d'étage qu'il est possible de réaliser avec n petits cubes (n est renseigné par l'utilisateur). Effectuez les calculs dans le programme en y intégrant une boucle.

5 - Boucle While

Exercice 1 : Epidémie

Votre programme doit d'abord lire un entier renseigné par l'utilisateur, la population totale de la ville. Sachant qu'une personne était malade au jour 1 et que chaque malade contamine deux nouvelles personnes le jour suivant (et chacun des jours qui suivent), le programme doit afficher à partir de quel jour toute la population de la ville sera malade.

Commentaires

On a 1 malade le premier jour, donc 2 nouveaux malades le second jour, soit un total de 3 malades. On a donc 6 nouveaux malades au troisième jour, soit un total de 9 malades. On a donc 18 nouveaux malades au quatrième jour, soit…

Prouver la terminaison de ce programme

Exercice 2 : Total des dépenses

Une grande partie du travail de l'administration, en plus de gérer les enseignants, les étudiants, les cours… est de veiller au bon fonctionnement et en particulier à ce que les comptes soient bien tenus. En particulier il faut, une fois par an, faire un bilan annuel des dépenses.

Toutes les dépenses de l'année ont été enregistrées et classées dans une multitude de dossiers et il faut maintenant calculer la somme de toutes ces dépenses. Mais personne ne sait exactement combien de dépenses différentes ont été effectuées durant l'année écoulée !

Ce que doit faire votre programme :

Votre programme devra lire une suite d'entiers positifs saisie par l'utilisateur et afficher leur somme. On ne sait pas combien il y aura d'entiers, mais la suite se termine toujours par la valeur -1 (qui n'est pas une dépense, juste un marqueur de fin).

Prouver la terminaison de ce programme

Exemples

Exemple 1

entrée :

```
1000
2000
500
-1
```

sortie :

```
3500
```

Exemple 2

entrée :

```
-1
```

sortie :

```
0
```

6 - **Liste :**

Exercice 1 : Faire les courses

Pour la fabrication de gâteaux, 10 ingrédients sont nécessaires. Vous êtes missionné pour aller faire les courses.

Le comptable étant particulièrement pointilleux, il vous donnera exactement la quantité d'argent dont vous avez besoin, pas une pièce de plus. Heureusement vous savez à l'avance le prix de chaque ingrédient et la quantité dont vous avez besoin.

Ce que doit faire votre programme :

Il y a 10 ingrédients et ils ont tous un prix au kilo différent : 9, 5, 12, 15, 7, 42, 13, 10, 1 et 20.

La quantité nécessaire en kilo de chaque ingrédient est : 3, 5, 4, 6, 1, 7, 9, 5, 7, 23.

Votre programme devra calculer le coût total de ces achats.

Exercice 2 : Etude de la richesse

Un habitant est considéré comme riche si sa fortune est parmi la moitié de la population la plus riche.

Ce que doit faire votre programme :

La richesse (un entier) des n habitants est mémorisée dans une liste.

Le programme devra calculer puis afficher une valeur permettant de dire facilement si une personne est riche ou pas, simplement en regardant si la fortune de cette personne est plus grande ou plus petite que cette valeur.

Enfin, il créera une nouvelle liste où chaque habitant est qualifié : « riche » ou « pauvre »

Deux cas peuvent se présenter :

- Si le nombre d'habitants est impair, par exemple si leurs fortunes sont 10, 5, 12, 8, 3 alors la valeur recherchée est 8. En effet, il y aura alors 2 personnes "riches" (10 et 12), 2 "moins riches" (3 et 5) et 1 juste au milieu (8) qui ne donnera ni recevra de cadeau.
- Si le nombre d'habitants est pair, par exemple si leurs fortunes sont 10, 5, 12, 8, 3, 9 alors la valeur recherchée est entre 8 et 9. Il y a en effet 3 personnes "riches" (9, 10 et 12) et 3 "moins riches" (3, 5 et 8). Par convention on prendra la valeur 8.5, c'est-à-dire la moyenne de 8 et 9.

Rappel : méthode de tri d'une liste intégré dans python :

```
# Défini le tableau
poids = [45, 80, 2]
# Tri le tableau
poids.sort()
```

7 - **Tableau : association de terme**

Le programme aide l'organisation d'une compétition d'un jeu en duel.

Afin de constituer les matchs au hasard, un tirage au sort est organisé.
Votre programme doit donc :

- Créer une liste de nombres aléatoires de la taille n du nombre de joueur ;
- Afficher cette liste

Les matches sont constituées ainsi : la personne ayant tiré le plus petit entier affronte celle ayant tiré le plus grand, celle ayant tiré le deuxième plus petit affronte avec celle ayant tiré le deuxième plus grand, et ainsi de suite.

Le programme devra préparer un tableau permettant d'afficher la composition de chacune des rencontre, dans l'ordre : d'abord celle dont le plus petit numéro fait partie, puis celle dont le second plus petit numéro fait partie, et ainsi de suite. Au sein de chaque rencontre on affichera d'abord le plus petit numéro puis le plus grand.

(on pourra utiliser la fonction random, qui génère un nombre aléatoire entre 0 et 1) Exemple :

```
from random import random
print(random())
```

Exemple :
1° liste

```
10
80
1000
5
154
130
847
450
42
35
789
```

sortie :

```
5 1000
35 847
42 789
80 450
130 154
```

8 - **Chaine de caractère**

Exercice 1 : Création de chaine

En python, l'instruction chr permet de générer le caractère associé à l'entier correspondant dans la table ASCII.
Par exemple :

```
>>> print(chr(97))
a
```

Ecrire un programme créant une chaine de caractères contenant l'alphabet dans l'ordre et affichant cet alphabet.

Exercice 2 : Comparaison 1

En Python, il est possible de comparer deux chaînes de caractères selon l'ordre alphabétique (appelé également ordre lexicographique).
On peut donc comparer directement deux chaînes de caractères et tous les opérateurs de comparaisons sont disponibles, c'est-à-dire <, <=, ==, !=, => et >.

Votre programme doit demander à l'utilisateur d'indiquer 2 mots par 2 saisies successives.
Puis il affichera le mot le plus grand.

Exercice 3 : Comparaison 2

Votre programme doit demander à l'utilisateur d'indiquer 2 mots par 2 saisies succéssives.
Puis il affichera le mot le plus long .

Exercice 4 : Inversion de mots

Le programme consiste à lire des couples de prénoms et noms et à les afficher dans l'autre ordre.
Les couples de nom – prénom sont saisis par l'utilisateur.

Exemple

entrée :

```
Alan Turing
Ada Lovelace
Donald Knuth
Claude Shannon
```

sortie :

```
Turing Alan
Lovelace Ada
Knuth Donald
Shannon Claude
```

Exercice 5 : Liste d'entiers

Le programme consiste à lire une liste d'entier, à créer une liste et à trier cette liste. Enfin, il affichera la liste triée.
La liste d'entier est saisie par l'utilisateur en une seule ligne.

Exemple

entrée :

```
« 14 33 27 3 »
```

sortie :

```
[3,14,27,33]
```

Exercice 6 : Tourner le texte

Une bibliothèque municipale est remplie de milliers de livres. Afin d'éviter de passer leurs journées avec la tête tournée à 90 degrés pour pouvoir lire ce qui est écrit sur leur tranche, les bibliothécaires ont adopté un système d'étiquettes où les mots sont écrits de haut en bas avec une seule lettre par ligne.

Étant donné un texte saisi par l'utilisateur, écrit normalement sur une seule ligne, le programme doit afficher l'étiquette correspondante, avec un seul caractère par ligne.

Entrée

Une seule ligne de texte.

Sortie

Les caractères du texte, affichés verticalement.

Exemple

entrée :

```
Don Quichotte
```

sortie :

```
D
o
n

Q
u
i
c
h
o
t
t
e
```

Exercice 7 : Inverser l'ordre d'un texte : Écriture en miroir

Étant donné un texte saisi par l'utilisateur, écrit normalement sur une seule ligne, le programme doit inverser l'ordre des lettres et afficher le nouveau texte.

Exemple

entrée :

```
tniop a ritrap tuaf li riruoc ed tres en neiR
egangiomet nu tnos ne eutroT al te erveiL eL
```

sortie :

```
Rien ne sert de courir il faut partir a point
```

```
Le Lievre et la Tortue en sont un temoignage
```

Exercice 8 : ngms sns vlls

Étant donné un texte saisi par l'utilisateur, écrit normalement sur une seule ligne, le programme doit supprimer les voyelles et les espaces, et afficher le nouveau texte.

Exemple

entrée :

```
AUTANT EN EMPORTE LE VENT
```

sortie :

```
TNTNMPRTLVNT
```

Exercice 9 : Compter une lettre

Étant donné un texte saisi par l'utilisateur, écrit normalement sur une seule ligne, le programme doit le nombre d'apparition de la lettre 's' et afficher ce nombre.

Exemple

entrée :

```
JE VOUS REMECTZ A LA GRANDE CHRONICQUE PANTAGRUELINE RECONGNOISTRE LA GENEALOGIE ET
ANTIQUITE DONT NOUS EST VENU GARGANTUA
```

sortie :

```
16
```

9 - Chaines et boucles

Exercice 1 : Damier

Vous souhaitez jouer à une variante du jeu de dames que vous connaissez bien. Le jeu utilise un plateau similaire à un damier, mais de taille 40 par 40. Vous devez en imprimer un nouveau damier.

Ce que doit faire votre programme :

Un damier de dimension 4×4 peut se représenter sous la forme suivante :

```
↳
OXOX
XOXO
OXOX
XOXO
```

Votre programme doit afficher un damier de taille 40×40. Assurez-vous bien que la case tout en haut à gauche contienne un « O », comme c'est le cas dans le damier ci-dessus.

Exercice 2 : Table de multiplication

C'est l'heure du cours de mathématiques et aujourd'hui les enfants de CE1 vont travailler la multiplication. Malheureusement, l'institutrice ne retrouve que la petite table de multiplication, qui va jusqu'à 5 fois 5, mais pas la grande table, qui va jusqu'à 20 fois 20. Elle souhaiterait que vous lui imprimiez une nouvelle table allant jusqu'à 20 fois 20, pour qu'elle puisse l'afficher au mur.

Ce que doit faire votre programme :

Voici à quoi ressemble la table de multiplication allant jusqu'à 5 fois 5.

```
↳
1 2 3 4 5
2 4 6 8 10
3 6 9 12 15
4 8 12 16 20
5 10 15 20 25
```

Écrivez un programme qui affiche une table de multiplication allant jusqu'à 20 fois 20.

10 - Fonction

Exercice 1 : Fonction sans argument

Ecrire une fonction qui affiche à l'écran un texte formé de 40 étoiles.
Utiliser la fonction 5 fois

Exemple

```
****************************************
****************************************
****************************************
****************************************
****************************************
```

Exercice 2 : Fonction avec argument

Ecrire une fonction qui affiche à l'écran un texte formé de n étoiles – où n est un argument de la fonction.
Utiliser la fonction 3 fois : pour afficher 10 étoiles, puis 20, puis 40.

Exemple

```
**********
********************
****************************************
```

Exercice 3 : Fonction avec 2 argument

Ecrire une fonction qui affiche à l'écran un texte formé de n étoiles sur m lignes – où n et m sont des arguments de la fonction.
Utiliser la fonction pour afficher 4 lignes de 10 étoiles.

Exemple

```
**********
**********
**********
**********
```

Exercice 4 : Fonction avec retour (1° cas)

Écrivez une fonction nommée *moy* qui prend deux entiers en arguments et retourne la moyenne des deux nombres.

Exemple

Arguments :

```
4,3
```

sortie :

```
3.5
```

Exercice 5 : Fonction avec retour (2° cas)

Écrivez une fonction nommée *compare* qui prend deux entiers en arguments et retourne le plus grand des deux nombres.

Exemple

Arguments :

```
4,3
```

sortie :

```
4
```

Exercice 6 : Fonctions à 2 arguments de type liste

Écrivez une fonction qui prend en arguments les coordonnées (x_1, y_1) et (x_2, y_2)

de deux points et retourne la distance euclidienne entre ces deux points. On rappelle que la distance euclidienne entre deux points est égale à :

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Les coordonnées doivent être renseignées sous la forme de deux listes.

On pourra utiliser la fonction mathématique `sqrt(x)` qui retourne la racine carrée du paramètre *x* et qui s'importe avec cette ligne :

```
from math import sqrt
```

Exercice 7 : Exercice compliqué

Écrivez un programme qui affiche une ligne de « X », un rectangle de « # », et un triangle de « @ ». Les deux formes doivent être creuses (remplies avec des espaces).

L'entrée comporte quatre entiers, un par ligne :

- le nombre de « X » de la ligne à afficher ;
- le nombre de lignes du rectangle de « # » ;
- le nombre de colonnes du rectangle ;
- le nombre de lignes du triangle de « @ ».

Vous devez afficher les trois formes successivement, avec une ligne blanche entre chaque forme, comme le montre l'exemple.

Votre objectif doit être d'obtenir le code source le plus simple et clair possible, en le décomposant en fonctions.

Exemple

entrée :

```
15
5
12
6
```

sortie :

```
XXXXXXXXXXXXXXX

############
#          #
#          #
#          #
############

@
@@
@ @
@  @
@   @
@@@@@@
```