EXERCICES

1. Répéter 2020

Afficher les uns en dessous des autres 10 fois de suite l'entier 2020.

2. Consécutifs

Afficher les uns en dessous des autres tous les entiers consécutifs entre 2020 et 2038.

3. Afficher deux listes l'une après l'autre

Écrire un code, utilisant deux boucles **`for`** *successives* qui produise l'affichage exact suivant (noter qu'il y a deux lignes dans le même affichage) :

```
42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61
65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84
```

4. Liste de trois éléments

On donne un entier x et on compte de 10 en 10 trois fois à partir de x. Construire la liste L qui contient les trois nombres obtenus. Par exemple, si x=42, la liste est formée des trois entiers 42, 52 et 62.

5. Liste : premier et dernier

On se donne une liste d'entiers `L`. Afficher sur une même ligne et séparés par une espace le premier élément de la liste et le dernier élément de la liste. Par exemple,

— si `L = [31, 12, 81, 9, 65]` alors l'affichage est : `31 65`

— si `L = [42]` alors l'affichage est : `42 42`

6. Liste : k-ème élément

On se donne une liste d'entiers `L`. On suppose que la liste `L` contient au moins deux éléments. On donne un entier $k \geq 1$. Afficher le k-ème élément de la liste si cet élément existe sinon, on affiche le message *rang invalide*.

Exemples d'exécution :

— `L = [31, 12, 81, 9, 65]`, `k = 3` -> `81`

— `L = [31, 12, 81, 9, 65]`, `k = 42` -> `rang invalide`

7. Changer certains éléments d'une liste

On donne une liste `L` d'entiers ayant au moins 3 valeurs. Modifier cette liste pour que les éléments x de la liste qui sont

— en première position

— en deuxième position

— en dernière position

soient remplacés par $10x$. Par exemple, si `L = [31, 12, 81, 9, 65]` alors, après modification, on aura `L = [310, 120, 81, 9, 650]`.

8. Test de présence

On donne

— une liste d'entiers `L`

— un entier `x`

On rappelle que `x in L` est un booléen qui teste la présence de `x` dans la liste `L`.

Construire un booléen `ok` qui vaut `True` si l'un des trois entiers `x`, `x - 1` et `x + 1` est dans la liste et `False` sinon. Exemple avec `x = 42` :

— si `L =[31, 12, 42, 9, 65]` alors `ok = True`

— si `L =[31, 12, 81, 43]` alors `ok = True`

— si `L =[41]` alors `ok = True`

— si `L =[31, 12, 81, 9, 65]` alors `ok = False`

9. Liste des solutions de l'équation du second degré

Soit l'équation du second degré $ax^2 + bx + c = 0$. On supposera que a, b, c sont des entiers, que a est non nul et que l'inconnue x est un nombre réel.

On rappelle le code Python de la résolution d'une équation du second degré :

```
a = 2
b = -3
c = 1
delta = b*b -4*a*c
if delta >0:
    print("2 solutions distinctes :")
    print("x1 =", (-b - delta**0.5)/(2.*a))
    print("x2 =", (-b + delta**0.5)/(2.*a))
else:
    if delta == 0:
        print("une seule solution")
        print("x =", (-b)/(2.*a))
    else:
        print("Aucune solution")
# output
2 solutions distinctes :
x1 = 0.5
x2 = 1.0
```

Adapter le code précédente pour construire la liste S contenant toutes les solutions de l'équation. Par exemple,

— si a = 2, b=-3 et c = 1 alors `S = [1, 0.5]`

— si a = 4, b=4 et c = 1 alors `S = [-0.5]`

— si a = 4, b=4 et c = 4 alors `S` est la liste vide

10. Mois de 31 jours et liste

On donne un numéro de mois entre 1 et 12. Créer une variable booléenne `est_mois_31` (prenant comme valeur True ou False) qui teste si `m` est le numéro d'un mois ayant 31 jours [1] comme janvier (numéro 1) ou juillet (numéro 7) mais pas février (numéro 2). On utilisera impérativement l'appartenance à une liste

11. Echanger les extrémités d'une liste

1. Les mois ayant 31 jours sont : janvier (1), mars (3), mai (5), juillet (7), août (8), octobre (10), décembre (12).

Cette question nécessite d'adapter aux éléments d'une liste le code d'échange des valeurs de deux variables, cf. un exercice du chapitre 1.

Soit une liste `L`. Modifier `L` pour que son premier élément et son dernier élément soient échangés.
Par exemple, si `L = [31, 12, 9, 65, 81]` alors après échange, on aura `L = [81, 12, 9, 65, 31]`.

12. Echanger côte à côte

Soit une liste `L`. Modifier `L` pour que

— le 1er élément et le 2e élément soient échangés de position

— le 3e élément et le 4e élément soient échangés de position

— ...

— et ainsi de suite jusqu'à ce que plus aucun échange ne soit possible.

Par exemple,

— si `L = [31, 12, 9, 65, 81, 42]` alors, après échanges, `L = [12, 31, 65, 9,42, 81]`

— si `L = [31, 12, 9, 65, 81]` alors, après échanges, `L = [12, 31, 65, 9, 81]`

— si `L = [31]` alors, après échange, `L = [31]`

13. Inverser une liste

Soit une liste `L`. Modifier `L` pour que

— le 1er élément et le dernier élément soient échangés de position

— le 2e élément et l'avant-dernier élément soient échangés de position

— ...

— et ainsi de suite jusqu'à ce que tous les échanges possibles soient effectués.

Par exemple,

— si `L = [31, 12, 9, 65]` alors, après échanges, `L = [65, 9, 12, 31]`

— si `L = [31, 12, 81, 9, 65]` alors, après échanges, `L = [65, 9, 81, 12, 31]`

— si `L = [31]` alors, après échange, `L = [31]`

14. Entiers qui se suivent

On vous donne deux variables `x` et `y` représentant des entiers. Définir une variable booléenne `ok` valant `True` si

— les entiers sont entre 1 et 4 ET

— ou bien sont consécutifs

— ou bien l'un vaut 1 et l'autre vaut 4.

Sinon, `ok` vaudra `False`.
Exemples

```
x=2, y=3 -> ok = True
x=3, y=2 -> ok = True
x=4, y=4 -> ok = False
x=7, y=8 -> ok = False
x=4, y=1 -> ok = True
x=3, y=1 -> ok = False
```

15. Parcours *indice -> élément*

Etant donné une liste, afficher ligne par ligne chaque indice de L, suivi d'une flèche `->` suivie de la valeur de `L` correspondant à l'indice. Par exemple, si `L = [31, 12, 81, 9, 31]` alors l'affichage est

```
0 -> 31
1 -> 12
2 -> 81
3 -> 9
4 -> 31
```

16. Entiers consécutifs en décroissant

On donne deux entiers a et b avec $a \geq b$, par exemple $a = 2020$ et $b = 2000$. Afficher les entiers consécutifs depuis a jusqu'à b en ordre décroissant.

On observera qu'il y a $a - b + 1$ nombres à afficher et on pourra, par exemple, introduire dans une boucle for une variable auxiliaire j initialisée à a. Mais d'autres codes sont envisageables.

17. Modifier une liste en multipliant par 10

On donne une liste d'entiers. Modifier cette liste pour que chaque élément x de la liste soit remplacé par $10x$. Par exemple, si `L = [31, 12, 81, 9, 65]` alors, après modification, on aura `L = [310, 120, 810, 90, 650]`.

18. Somme des n premiers entiers

On se donne un entier $n \geq 0$. Calculer la somme $1 + 2 + ... + n$. Vérifier que le résultat obtenu est bien $\frac{n(n+1)}{2}$.

19. Sommes de puissances consécutives

On se donne un entier $n \geq 0$. Calculer la somme $1 + 10 + ... + 10^{n-1}$. Vérifier que vous obtenez bien $\frac{10^n - 1}{9}$.

20. Compter de 0,5 en 0,5

On donne deux entiers a et b avec $a \leq b$ et on demande d'afficher tous les nombres entre a et b si on compte de 0,5 en 0,5. Par exemple si $a = 42$ et $b = 49$, il faut afficher les nombres suivants, de préférence sur une même ligne :

```
42 42.5 43 43.5 44 44.5 45 45.5 46 46.5 47 47.5 48 48.5 49
```

21. Non multiples

Combien y-a-t-il d'entiers entre 1 et 20000 qui ne sont ni pairs ni multiples de 5 ? Par exemple, 421 ou 2017 font partie des entiers concernés mais pas 42 ni 2015. Le nombre à trouver est 8000.

22. Produit d'entiers

Calculer le produit P des entiers entre 42 et 421, autrement dit $P = 42 \times 43 \times \cdots \times 420 \times 421$.

On trouvera un très grand entier qui commence par 1484.

23. Suite d'entiers

On considère la suite de nombres entiers dont les premiers termes sont :

```
10, 21, 43, 87, 175, 351, etc
```

Cette suite est construite de la manière suivante :

— son premier terme est 10

— si x est un terme de la suite, le terme suivant vaut $2x + 1$.

Par exemple, si $x = 87$ alors le terme suivant dans la suite est $2x + 1 = 2 \times 87 + 1 = 175$.

Ecrire un code qui à partir d'un entier $n > 0$ affiche le n^{e} terme de la suite. Exemple de comportements selon la valeur de `n` :

```
3 -> 43
1 -> 10
7 -> 703
```

24. Variations des éléments d'une liste de nombres

On donne une liste `L` d'entiers et on demande d'afficher les variations de chaque terme par rapport au précédent : si le terme augmente, le code affiche `augmente` sinon le code affiche `diminue`. Par exemple, si `L=[42, 81, 12, 9, 6, 34, 30, 35, 48, 82, 32]`, le code affichera :

```
augmente
diminue
diminue
diminue
augmente
diminue
augmente
augmente
augmente
diminue
```

25. Somme d'entiers impairs consécutifs

On donne un entier $N \geq 0$ et on demande de calculer la somme des N premiers entiers impairs, c'est-à-dire

$$1, 3, 5, 7, 9, 11, \text{etc}$$

Par exemple, si $N = 6$, on trouvera $1 + 3 + 5 + 7 + 9 + 11 = 36$.

26. Somme alternée plus/moins en décroissant

On cherche à écrire un code capable de calculer des expressions du genre

$$7 - 6 + 5 - 4 + 3 - 2 + 1.$$

Plus précisément, on se donne un entier $n > 0$ (ci-dessus, c'était $n = 7$) et on demande d'écrire un code qui renvoie la valeur de l'expression :

$$n - (n - 1) + (n - 2) - (n - 3) + \ldots$$

expression qui se poursuit jusqu'à ce que le terme courant vaille 1. Noter que l'expression commence par n et alterne soustraction et addition. Si $n = 2037$ ou $n = 2038$ on trouvera que la somme vaut 1019.

27. Afficher par paires verticales

On donne deux entiers a et b et on demande d'afficher, par ordre croissant tous les entiers de a à b sur deux lignes en plaçant alternativement les entiers sur la ligne du haut et sur la ligne du bas. Par exemple, si $a = 2016$ et $b = 2023$ le code doit afficher

```
2016 2018 2020 2022
2017 2019 2021 2023
```

et si $a = 2020$ et $b = 2038$ le code doit afficher

```
2020 2022 2024 2026 2028 2030 2032 2034 2036 2038
2021 2023 2025 2027 2029 2031 2033 2035 2037
```

28. Somme des multiples de 10

On donne une liste `L` d'entiers. Calculer la somme `s` des éléments de `L` qui sont des multiples de 10. Par exemple, si `L = [310, 12, 8100, 90, 31]` alors `s = 8500`.

29. Somme suivant les indices impairs

On donne une liste `L` d'entiers. Calculer la somme `s` des éléments de `L` dont l'indice est impair. Par exemple, si `L = [31, 12, 81, 9, 31]` alors `s = 21`.

30. Liste des N premiers multiples de d

Soient `N` et `d` deux entiers positifs. Créer la liste `L` contenant les `N` premiers multiples de `d` en commençant par `d`. Par exemple, si `N = 7` et `d = 10` alors `L = [10, 20, 30, 40, 50, 60, 70]`.

31. Calcul du minimum

On donne une liste L d'entiers. Calculer le plus petit élément mini de la liste. Par exemple, si `L = [31, 9, 81, 9, 31]` alors `mini = 9`.

32. Indice du minimum

On donne une liste d'entiers. Déterminer un indice `i_mini` correspondant au plus petit élément `mini` de la liste. Par exemple

— si `L = [31, 12, 81, 9, 31]` alors `mini = 9` et `i_mini = 3`

— si `L = [31, 9, 81, 9, 31]` alors `mini = 9` et `i_mini = 1` (ou encore `i_mini = 3`)

33. Minimum des entiers d'indices pairs

On donne une liste `L` d'entiers. Calcul `mini_pairs` le plus petit des éléments d'indices pairs de la liste. Par exemple, si `L = [81, 32, 12, 9, 10, 65, 46]` alors `mini_pairs = 10`.

34. Valeurs paires d'une liste

Soit une liste `L` d'entiers. Séparer cette liste en deux listes :

— la liste `P` des entiers pairs de `L`

— la liste `I` des entiers impairs de `L`

Par exemple, si `L = [31, 12, 9, 65, 81, 42]` alors `P = [12, 42]` et `I = [31, 9, 65, 81]`

35. Indices pairs

Soit une liste `L`. Séparer cette liste en deux listes :

— la liste `IP` des éléments de `L` d'indices pairs

— la liste `II` des éléments de `L` d'indices impairs

Par exemple, si `L = [31, 12, 9, 65, 81, 42]` alors `IP = [31, 9, 81]` et `II = [12, 65, 42]`.

36. Décompte pair/impair

On donne une liste `L` d'entiers et on demande de calculer

— le nombre `n_pair` d'indices `i` tel que l'élément `L[i]` est pair

— le nombre `n_impair` d'indices `i` tel que l'élément `L[i]` est impair

Par exemple, si `L = [31, 12, 9, 65, 81, 42]` alors `n_pair = 2` et `n_impair = 4`.

37. Minimum des entiers pairs (boucle `for`)

On donne une liste `L` d'entiers contenant au moins un entier pair. Construire une variable `miniPairs` qui renvoie le plus petit des éléments pairs de la liste. Exemples de comportements :

```
[81, 32, 12, 9, 12, 65, 46] -> 12
[81, 65, 46] -> 46
```

38. Minimum présent une seule fois

On donne une liste `L` d'entiers. Soit `mini` le plus petit élément de la liste. Ecrire une variable `miniUnique` qui vaut True ou False selon que `mini` apparaît une seule fois dans la liste ou au contraire, plusieurs fois. Par exemple

— si `L = [31, 12, 81, 9, 31]` alors `mini = 9` et `miniUnique` vaut `True` ;

— si `L = [31, 9, 81, 9, 31]` alors `mini = 9` et `miniUnique` renvoie `False`.

39. Tester les mois de 31 jours avec une liste

Construire une liste littérale `jours` telle que pour chaque numéro `i` entre 1 et 12 d'un mois d'une année non bissextile, `jours[i]` désigne le nombre de jours du mois de numéro `i`. Par exemple, `jours[10] = 31`.

En déduire une variable booléenne `mois31` qui, étant donné un numéro de mois `i`, vaut `True` si le mois de numéro `i` possède 31 jours et `False` sinon.

40. Pièces de monnaie

Dans un pays imaginaire, les pièces de monnaie sont de `a` unités ou de `b` unités. On demande d'écrire un code qui détermine si, oui ou non, il est possible de régler un certain montant donné de `m` unités. Par exemple si `a = 49` et `b = 10` alors on ne peut pas régler `m = 42` unités ni `m =105` mais on peut régler `m = 128` puisque `128 = 3*10 + 2*49`.

Vous ne devez pas utiliser de boucles imbriquées.

41. Somme de n entiers consécutifs à partir de d

On vous donne deux entiers positifs `d` et `n`. On vous demande de calculer la somme des `n` entiers consécutifs à partir de `d`. Par exemple, si $d = 10$ et $n = 4$, vous devez calculer $10+11+12+13 = 46$. Voici d'autres exemples :

```
(d, n) = (10, 1) -> 10
(d, n) = (10, 2) -> 21
(d, n) = (10, 100) -> 5950
```

42. Dix de plus ou de moins

On donne une liste `L` d'entiers et on demande de construire une variable booléenne OK valant True si chaque nombre de la liste vaut 10 de plus ou 10 de moins que l'élément suivant dans la liste. Sinon la variable OK vaudra `False`. Voici quelques exemples d'exécution attendue :

```
[42, 32, 22, 32, 42, 52, 62] -> True
[42, 52] -> True
[42] -> True
[42, 52, 32, 22, 32, 42, 52, 72] -> False
```

43. Le double du précédent

On donne une liste L d'entiers. Définir un booléen ok qui vaut True si chaque élément de la liste est le double du précédent et qui vaut False sinon. Exemple de comportements :

```
[10, 20, 40, 80] -> True
[12, 24, 48] -> True
[42] -> True
[0, 0, 0, 0, 0] -> True
[10, 40, 80] -> False
```

44. Liste constante

On donne une liste L et on demande de créer un booléen **tousEgaux** valant **True** si tous les éléments de la liste L sont égaux et **False** sinon. Par exemple si L = [42, 42, 42] alors **tousEgaux** vaudra **True** et si L = [42, 421, 42, 42] alors **tousEgaux** vaudra **False**.

45. Listes « opposées »

Ecrire un code qui partant deux listes d'entiers L et M crée un booléen **sontOpposees** valant `True` si les deux listes sont « opposées » et `False` sinon. Deux listes sont considérées comme « opposées » si elles ont le même nombre d'éléments et si, à des indices identiques, elles possèdent des éléments opposés (comme -81 et 81). Voici quelques exemples de comportements attendus :

```
[81, -12, 0, -81, -31] [-81, 12, 0, 81, 31] -> True
                                  [-81] [81] -> True
                               [0, 0] [0, 0] -> True
                                     [ ] [ ] -> True
                         [81, -12] [-81, -12] -> False
                       [-81, 12, 0] [81, -12] -> False
```

46. Suite croissante d'entiers consécutifs

Écrire un code qui à partir d'une liste L d'entiers définit une variable booléenne nommée **consecutifs** qui vaut **True** si la liste est constituée d'entiers CONSÉCUTIFS croissants et **False** sinon. Ci-dessous, voici quelques exemples de comportements attendus

```
[81, 82, 83]       -> True
[82, 81, 83]       -> False
[2013, 2038, 3000] -> False
[81]               -> True
```

47. Verbes du premier groupe

On donne une liste de verbes à l'infinitif, écrits en minuscule. Écrire un programme qui, affiche les verbes de la liste qui sont du premier groupe, c'est-à-dire se terminant en **er**, comme *chanter, manger*, etc. Par exemple, si la liste est formée des verbes

boire, chanter, fuir, voir, avancer, lire

le programme doit afficher juste :

```
chanter
avancer
```

48. Chaînes formées seulement de lettres parmi a ou b

On appellera *babachaîne* toute chaîne constituée de caractères uniquement parmi les deux lettres minuscules **a** ou **b**. Par exemple, la chaîne **bbabaaab** ou encore la chaîne **bb** sont des *babachaîne*, mais pas **dada** (qui contient le caractère **d**).

(1) Soit une *babachaîne* C. Écrire un code Python qui renvoie le nombre d'occurrences de la lettre *a* dans la chaîne C. Par exemple,

- si C est la chaîne **ababa**, le programme doit renvoyer 3 ;
- si C est la chaîne **bbb**, le programme doit renvoyer 0.

(2) Soit une chaîne *babachaîne* C de longueur impaire. Écrire un code Python qui renvoie la lettre la plus présente dans la chaîne C. Par exemple,

- si C est la chaîne **ababa**, le programme doit renvoyer **a** ;
- si C est la chaîne **bbb**, le programme doit renvoyer **b**.

49. Palindrome

(1) À partir d'une chaîne de caractères, comme **citoyen**, construire la chaîne inversée, dans l'exemple, cela donne **neyotic**.

(2) Le mot **RADAR** est un *palindrome* : quand on lit ses lettres de la droite vers la gauche, le mot est inchangé. Dire si un mot est oui ou non un palindrome.

50. Doubler les lettres

À partir d'une chaîne telle que **bali**, on veut construire la chaîne **bbaallii**.

Plus précisément, étant donné une chaîne de caractères, sans accent, écrire une nouvelle chaîne telle que chaque caractère de la chaîne initiale soit doublé.

51. Suite de Prouhet

La suite de Prouhet est la suivante :

```
0
01
0110
01101001
```

Cette suite est une suite de chaînes composées de 0 et de 1 de la manière suivante : chaque chaîne T s'obtient à partir de la précédente U en adjoignant à U la suite V obtenue à partir de U en échangeant tout 0 en 1 et tout 1 en 0.

Quelle est la 20^{e} chaîne de la suite de Prouhet.

52. Créer un mot de passe

Un mot de passe `m` est considéré comme sûr si les conditions suivantes sont réalisées :

- `m` est formé d'au moins huit caractères ;
- `m` contient au moins :
  - une lettre minuscule
  - une lettre majuscule
  - un chiffre
  - un signe de ponctuation parmi le tiret, le point d'exclamation ou d'interrogation, la virgule, le point, le point-virgule, le deux-points, l'astérisque et le tilde.

Créer une variable `mot_de_passe_valide` qui vaut `True` si le mot de passe `m` est sûr et `False` sinon.

53. Conjuguer un verbe du premier groupe

On donne un verbe du premier groupe, par exemple, *danser*. Ecrire un code qui affiche la conjugaison de ce verbe au présent de l'indicatif. Par exemple, avec *danser*, le code affichera :

```
je danse
tu danses
il/elle danse
nous dansons
vous dansez
ils/elles dansent
```