1 Identifier les types des valeurs suivantes :

`"Exercice"`, `123.`, `True`, `1e10`, `"False"`, `'3,14'`.

2 Insérer le minimum de parenthèses dans les expressions suivantes pour que les égalités soient correctes.

```
2 + 3 * 5 + 4 == 21
5 + 2 * 3 + 4 == 25
4 + 5 * 2 + 3 == 29
```

3 Écrire l'expression qui calcule la moyenne de deux nombres `a` et `b`.

4 Calculer les valeurs des expressions suivantes pour `x = 0`, `x = 10` et `x = -20` :

```
x < 10 and x > -10
x < -10 or x > 10
x <= 10 and x * x >= 100
x > -25 and x < -5 or x > 5 and x < 25
```

5 Qu'est-ce qui est affiché par chacun des programmes suivants ?

```
if ( 12 * 2 == 24 ):
    print("Logique.")
```

```
if ( 12 * 2 == 24 ) == False:
    print("Logique.")
```

```
if ( 12 * 2 == 23 ) == False:
    print("Logique.")
print("Ou pas.")
```

```
if ( 12 * 2 == 23 ):
    print("Logique.")
else:
    print("Ou pas.")
```

6 Combien de fois s'exécute le corps de chacune des boucles ci-dessous, et quelle est la valeur de `s` à la fin de l'exécution ?

```
s = 0
for i in range(10):
    s = s + i
```

```
s = 1
for i in range(1,6):
    s = s * i
```

```
s = 0
while s < 20:
    s = s + 5
```

```
s = 1
while s != 100:
    s = s * 2
```

7 Écrire une boucle qui affiche la table de multiplication par 7 sous la forme :

```
7 x 1 = 7
7 x 2 = 14
...
```

8 Écrire une boucle qui calcule la factorielle du nombre 10 (produit des nombres 1 à 10).

9 a. Quelle est la valeur de `n` à la fin du programme suivant ?

```
n = 0
a = 27
b = 5
while a >= b:
    n = n + 1
    a = a - b
```

b. Montrer que `n` est le quotient de la division euclidienne de `a` par `b`.

c. Ajouter une instruction conditionnelle (`if`) au programme pour tester que c'est bien le cas.

10 Transformer le programme de l'exercice 9 en une fonction `quotient` qui renvoie le résultat de la division euclidienne (entière) entre deux nombres.

11 Écrire une fonction `pair` qui retourne `True` si le nombre passé en argument est pair.

12 a. Quelle est la valeur de `a` à la fin de l'exécution du programme suivant ?

```
a = 21

def double(x)
    print(x * 2)

a = double(a)
```

b. Qu'affiche ce programme ?

13 Identifier les types d'erreurs dans le code suivant et les corriger.

```
pi = 3,14
def aire-cercle(rayon)
    return pi * r * 2
  print(aire_cercle(10)
```


14 a. Trouver les valeurs suivantes, par exemple dans Wikipedia, et écrire les valeurs Python correspondantes en unités SI.

- Distance `d` de la Terre au Soleil
- Vitesse `c` de la lumière
- Circonférence `t` de la Terre
- Population `p` de la Terre

b. À partir des valeurs précédentes, écrire les expressions qui calculent les valeurs suivantes (attention aux unités !).

- Temps mis par la lumière du Soleil pour atteindre la Terre
- Vitesse moyenne de la Terre autour du Soleil, en supposant que la trajectoire est circulaire
- Nombre de fois qu'un signal électrique fait le tour de la Terre en 1 seconde
- Surface de la Terre
- Densité moyenne de population de la Terre, sachant que 70,7 % de la surface de la Terre est immergée

15 a. En utilisant une fois et une seule chacun des nombres de 1 à 4, écrire 11 expressions arithmétiques qui utilisent les quatre opérations et les parenthèses dont les résultats sont les nombres de 0 à 10.

b. *(Niveau Titan)* Essayer de trouver des expressions qui utilisent chaque opérateur au plus une seule fois.

16 Dans le Bourgeois Gentilhomme (Acte II, scène 4), le Maître de Philosophie dit :

> On les peut mettre premièrement comme vous avez dit : « Belle Marquise, vos beaux yeux me font mourir d'amour ». Ou bien : « D'amour mourir me font, belle Marquise, vos beaux yeux ». Ou bien : « Vos yeux beaux d'amour me font, belle Marquise, mourir ». Ou bien : « Mourir vos beaux yeux, belle Marquise, d'amour me font ». Ou bien : « Me font vos yeux beaux mourir, belle Marquise, d'amour ».
>
> *Molière*

Définir plusieurs variables contenant les morceaux de la phrase *"Belle Marquise, ..."*, de telle sorte que l'on puisse afficher, avec la fonction `print` et l'opérateur `+` des chaînes de caractères, les différentes variantes énoncées. (*On ignore la distinction entre majuscules et minuscules*).

17 Montrer les identités suivantes, appelées identités de l'algèbre de Boole en écrivant et en exécutant un programme qui calcule les valeurs des 4 expressions ci-dessus pour toutes les valeurs possibles des variables booléennes `a` et `b`.

```
not (a and b) == not a or not b
not (a or b) == not a and not b
```

On rappelle que `a and b` est vrai seulement si `a` et `b` sont vrais, `a or b` est vrai si `a` est vrai ou `b` est vrai, et `not a` est vrai si `a` est faux.

18 L'opérateur booléen `or` ne correspond pas toujours à ce que l'on entend par "ou" dans la vie quotidienne : si un restaurant affiche dans son menu "fromage ou dessert", cela signifie que l'on peut choisir l'un ou l'autre, mais pas les deux. Mais si l'on a deux variables booléennes `fromage` et `dessert`, l'expression `fromage or dessert` est vraie si l'une, l'autre ou les deux variables sont vraies.

L'équivalent du "ou" du restaurant s'appelle un *ou exclusif*, ou `xor` ("exclusive-or").

Écrire une fonction `xor(a, b)` qui retourne le ou exclusif de `a` et `b`.

19 On considère l'affectation `res = a and b`.

a. Remplacer cette affectation par une instruction conditionnelle imbriquée équivalente, sans utiliser `and`.

b. Même question pour l'affectation `res = a or b`.

20 Écrire une instruction conditionnelle qui affiche la phrase "Il va faire très froid / froid / frais / bon / un peu chaud / chaud / très chaud" selon la valeur de la variable `t` représentant la température maximale prévue. Vous choisirez les seuils entre chaque niveau de température.

21 a. Écrire un programme qui indique si une année `a` est bissextile. Une année est bissextile si elle est divisible par 4. Cependant, les siècles ne sont pas bissextiles, sauf les multiples de 400.

b. Compléter le programme pour calculer le nombre de jours du mois `m`, `m` étant un entier compris entre 1 et 12. De janvier à juillet, les mois impairs ont 31 jours, les autres 30 ; de août à décembre c'est l'inverse. De plus, le mois de février a 29 ou 28 jours selon que l'année est bissextile ou non.

22 a. Écrire le code de l'exercice 21 sous forme des deux fonctions suivantes.

```
def est_bissextile(annee):
""" retourne True si l'annee est bissextile """
...

def nb_jours_mois(mois, annee):
""" retourne le nombre de jours du mois
    de l'annee donnee """
...
```

b. Écrire une fonction qui calcule le nombre de jours depuis le début de l'année jusqu'à une date donnée.

```
def nb_jours(jour, mois, annee):
""" retourne le nombre de jours
    depuis le debut de l'annee """
...
```

c. On représente les jours de la semaine par les nombres 0 = Lundi à 6 = Dimanche. Étant donné le code du premier jour de l'année (5 = Samedi pour l'année 2022), écrire une fonction qui retourne le jour de la semaine d'une date donnée.

```
def jour_semaine(jour, mois, annee, jour0):
    """ retourne le jour de la semaine
        de la date donnee. `jour0` est le
        code du premier jour de l'annee """
    ...
```

d. Vérifier que le 14 juillet 2022 tombe un jeudi.

23 a. Écrire une boucle bornée (`for`) qui affiche les nombres pairs entre 2 et 20.

b. Écrire une boucle non bornée (`while`) qui fasse la même chose.

c. Mêmes questions pour afficher les nombres pairs entre 20 et 2 par ordre décroissant.

24 La fonction `random()` de la bibliothèque `random` produit un nombre aléatoire entre 0 et 1.

À l'aide de cette fonction et de la fonction `round()` de la bibliothèque `math`, écrire un programme qui affiche de manière aléatoire la chaîne "Ha", "Haha", "Hahaha", "Hahahaha" ou "Hahahahaha" un nombre aléatoire de fois entre 1 et 10.

25 a. Quelle est la valeur de `a` à la fin de l'exécution du programme ci-dessous

- lorsque `a = 10` et `b = 3`
- lorsque `a = 10` et `b = 15`

```
r = 0
while b != 0:
    r = a % b
    a = b
    b = r
```

b. Montrer que `a` contient, à la fin du programme, le PGCD (plus grand commun diviseur) de `a` et `b`.

26 a. Écrire une boucle qui affiche les puissances de 2 (2**1, 2**2, 2**3, etc.) qui sont inférieures à un million.

b. Modifier le programme pour *compter* le nombre de puissances de 2 inférieures à un million.

27 Écrire un programme qui affiche les chiffres d'un nombre à partir du dernier, un par ligne.

Par exemple, pour le nombre 1 234, le programme affiche :

```
4
3
2
1
```

28 a. Écrire un programme qui utilise une boucle pour afficher les valeurs de la fonction `sin(x)` pour `x` entre 0 et π.

On découpera l'intervalle [0, π] de façon à afficher 15 valeurs de `x`.

Aide : Pour utiliser la fonction `sin` il faut l'importer :

```
from math import sin
```

b. Pour afficher le graphe de la fonction précédente de manière visuelle, on veut remplacer chaque valeur affichée par une barre horizontale de longueur proportionnelle à la valeur.

À l'aide de l'opérateur `*` des chaînes de caractères et de la fonction `round(x)` qui retourne le nombre entier le plus proche de `x`, écrire l'instruction qui affiche une chaîne constituée de caractères "=" dont la longueur est proportionnelle à un nombre `x` compris entre 0 et 1 et de longueur 30 lorsque `x` vaut 1.

c. Utiliser l'instruction de la question précédente dans le code de la première question pour afficher le graphe de la fonction `sin`.

Le résultat a l'aspect suivant :

```
======
============
==================
====================
========================
==========================
==========================
==========================
========================
======================
==================
============
======
```


d. Pour pouvoir afficher des valeurs négatives, il faut décaler l'axe vertical correspondant à la valeur 0 vers la droite. Pour cela :

- si la valeur à afficher est positive, afficher 30 espaces suivis de la barre ;
- si la valeur à afficher est négative, afficher un nombre d'espaces égal à 30 moins la longueur de la barre, suivis de la barre.

Écrire le programme correspondant, pour obtenir le résultat suivant sur l'intervalle [0, 2π].

```
                              ========
                              ==============
                              ==================
                              ====================
                              ================
                              ============
                              ====
                        ====
                   ============
              ==================
         ====================
    ====================
     ==================
          ==============
               ========
```

e. Modifier l'affichage pour remplacer chaque barre par un caractère '*' à son extrémité, afin d'obtenir le résultat suivant :

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

29 À l'aide de la bibliothèque `turtle` et de la fonction `random()` de la bibliothèque `random`, écrire un programme qui fait avancer et tourner la tortue de manière aléatoire.

Pour éviter que la tortue se perde en dehors de la fenêtre, arrêter le programme si elle sort d'un cadre de 400 pixels autour du centre de la fenêtre. Les fonctions `xcor()` et `ycor()` de `turtle` retournent la position *x*, *y* de la tortue.

30 La suite de Fibonacci est une suite de nombres qui commence par 0 et 1 et dont chaque nombre suivant est la somme des deux nombres précédents.

Cette suite de nombres intervient dans de nombreux phénomènes naturels, notamment la croissance des plantes.

a. Écrire un programme qui affiche les 100 premiers nombres de Fibonacci.

b. Modifier le programme pour afficher également le rapport de deux nombres consécutifs de la série. Ce rapport tend vers le *nombre d'or*, qui est également présent dans la nature et très utilisé, par exemple en architecture. Le rapport de la hauteur et de la largeur d'une feuille A4 est égal au nombre d'or.

31 La *suite de Syracuse* est définie comme suit :

- on part d'un nombre entier positif ;
- s'il est pair, on le divise par deux, sinon on le multiplie par trois et on ajoute 1 ;
- si le nombre obtenu est 1, on s'arrête, sinon on applique à nouveau la règle ci-dessus.

a. Écrire une fonction `Syracuse(n)` qui affiche la suite des nombres à partir de `n`.

b. Modifier la fonction pour qu'elle n'affiche pas la suite, mais retourne le nombre d'étapes avant d'arriver à 1, appelé *durée de vol*. Afficher ce nombre pour `n` entre 2 et 100.

c. Modifier la fonction pour qu'elle retourne le plus grand nombre atteint par la suite, appelé *altitude*. Afficher ce nombre pour *n* entre 2 et 100.

Remarque : La question de savoir si la suite s'arrête quel que soit le nombre de départ est une question mathématique ouverte. On ne connait pas de contre-exemple, mais on n'a pas de preuve qu'elle s'arrête toujours.

32 a. Qu'affiche le programme suivant ?

```
for i in range(10):
    s = ''
    for j in range(i):
        s = s + '*'
    print(s)
```

b. Écrire une version plus simple en utilisant l'opérateur '*' de répétition des chaînes de caractères.

c. Modifier le programme pour qu'il affiche la même figure à l'envers.

33 a. Écrire un programme qui, étant donné un nombre entre 2 et 12, affiche toutes les combinaisons possibles permettant d'obtenir ce nombre avec deux dés.

b. Étendre le programme ci-dessus pour afficher, pour chaque nombre entre 2 et 12, toutes les combinaisons possibles permettant d'obtenir ce nombre avec deux dés.

c. Modifier le programme pour faire la même chose avec trois dés.

d. Modifier les programmes b. et c. pour afficher seulement le nombre de combinaisons possibles pour chaque total, au lieu des combinaisons elles-mêmes.

34 On veut calculer la monnaie à rendre sur un paiement en euros (sans les cents).

La monnaie est rendue avec des pièces de 1 € et des billets de 5 €, 10 €, 20 € et 50 €.

a. Étant donné un prix à payer et un montant reçu du client, écrire un programme qui calcule et affiche le nombre de pièces et de billets de chaque valeur à rendre.

b. Le contenu de la caisse est représenté par un ensemble de variables correspondant au nombre de pièces et de billets de chaque valeur :

```
nb_1euro = 15   # nombre de pièces de 1€
nb_5euros = 4   # nombre de billets de 5€
nb_10euros = 12 # nombre de billets de 10€
nb_20euros = 5  # nombre de billets de 20€
nb_50euros = 4  # nombre de billets de 50€

prix = 12       # prix à payer
paiement = 50   # paiement du client
```

Modifier le programme précédent pour afficher la monnaie à rendre en fonction du contenu de la caisse et mettre à jour le contenu de la caisse.

c. Que se passe-t-il si la caisse n'a pas assez de monnaie ? Quelle solution proposez-vous ?

d. (*Niveau Titan*) Modifier le programme en créant la fonction suivante pour simplifier le code.

```
def rendre(a_rendre, valeur, nombre):
    """ Retourne le nombre de pieces ou billets
de la valeur donnée à rendre en fonction du
nombre de pièces ou de billets de cette valeur
dans la caisse"""
```

35 On veut calculer les intérêts d'un placement pendant 10 ans d'une somme de 10 000 € qui rapporte 1,5 % par an.

a. Écrire une fonction qui étant donné un montant et un taux annuel, retourne les intérêts cumulés au bout d'un an.

b. Écrire une seconde fonction qui prend en paramètres le montant initial, le taux annuel et la durée de l'investissement en années, et retourne le montant épargné à la fin de la période.

c. Appliquer cette fonction aux données de l'exercice.

36 Les programmes suivants sont-ils corrects ? Si oui qu'affichent-ils ? Sinon de quel type d'erreur s'agit-il ?

```
def f(a):
    a = a * 2
    return a

print(f(5))
print(a)
```

```
a = 100

def f(a):
    a = a * 2
    return a

print(f(5), a, f(a))
```

37 On veut calculer la somme des inverses des carrés des n premiers entiers.

a. Le programme suivant est-il correct ? Sinon indiquer les types d'erreurs et les corriger.

```
def somme_inv_carres(n)
    s = 0
    for i in range(n)
        s = s + 1/i*i
    return
```

b. Vérifier, une fois le programme corrigé, que pour de grandes valeurs de n, le résultat s'approche de $\frac{\pi^2}{6}$.

38 On veut compter le nombre de fois qu'une fonction est appelée.

a. Compléter le programme suivant avec une variable globale `compteur` pour afficher le nombre de fois que la fonction `f` est appelée.

```
def f(n):
    print(n)

for i in range(100):
    if random() < 0.5:
        f(i)
```

b. Compléter le programme pour pouvoir calculer et afficher la valeur moyenne du paramètre `n` passé à la fonction `f`.

c. Ajouter une fonction qui remet à zéro le compteur et la moyenne.

39 Les nombres à virgule flottante de Python peuvent révéler des surprises car certains nombres réels ne peuvent pas être représentés de manière exacte sur un ordinateur.

a. Exécuter le code suivant : que se passe-t-il ? (Taper Control-C pour interrompre le programme.)

```
x = 0.0
while x != 1.0:
    print(x)
    x = x + 0.1
```

b. Pour s'assurer que le programme s'arrête, remplacer le test de non égalité par un test d'infériorité et vérifier qu'il fonctionne.

c. Remplacer 1.0 par 3.0 dans le test de la boucle non bornée (`while`). Quelle est la dernière valeur de `x` ? Est-ce que le programme fonctionne correctement ?

d. Proposer un autre test d'arrêt qui fonctionne dans les deux cas.

40 Un nombre premier est un entier positif qui admet exactement deux diviseurs distincts entiers et positifs : lui-même et 1.

Pour déterminer qu'un nombre est premier, il faut vérifier qu'il n'est *pas* divisible par un entier autre que 1.

On rappelle qu'en Python, l'opérateur `%` calcule le reste de la division entière : si `a` est un diviseur de `N`, alors `N % a` vaut 0.

On rappelle également que `range(min, max)` permet de spécifier un intervalle de valeurs entières allant de `min` (inclus) à `max` (exclus).

a. Écrire une fonction qui prend un nombre entier `n` en paramètre et retourne un booléen indiquant si le nombre est premier.

On utilisera une boucle bornée (`for`) pour tester tous les diviseurs potentiels de `n`.

b. Transformer le programme précédent en utilisant une boucle non bornée (`while`) afin que la fonction retourne `False` dès qu'elle détecte que le nombre n'est pas premier.

c. Pour améliorer la performance du programme, modifier la fonction pour qu'elle ne teste que les diviseurs compris entre 2 et $\sqrt{n}$: en effet, si $n = a \times b$, alors soit a, soit b est inférieur ou égal à $\sqrt{n}$.

Note : La fonction Python `sqrt(n)` retourne la racine carrée de n. Elle doit être importée grâce à l'instruction :

```
from math import sqrt
```

d. Écrire un programme qui affiche les M premiers nombres premiers.

41 Le but de cet exercice est d'écrire un programme qui joue à pierre-papier-ciseaux.

a. Écrire une fonction qui prend les choix des deux joueurs en paramètres et retourne 1 si le joueur A gagne, –1 si c'est le joueur B, et 0 s'il y a égalité.

b. Écrire une fonction qui tire les choix des deux joueurs au hasard et détermine celui qui gagne, sachant qu'en cas d'égalité un nouveau tirage a lieu et le nombre de points en jeu augmente de 1. La fonction retourne un nombre positif si c'est A qui a gagné ce tour, un nombre négatif si c'est B. La valeur absolue de ce nombre est le nombre de points gagnés.

c. Écrire un programme qui initialise les scores et joue 50 tours en mettant à jour les scores et en les affichant.

42 On veut calculer la valeur `x` pour laquelle une fonction `f(x)` vaut une valeur donnée `v`.

a. Écrire une boucle qui fait varier `x` d'une valeur donnée `min` à une valeur donnée `max` par pas de 0.1. La boucle s'arrête pour une valeur `x` telle que `f(x) < v` et `f(x+0.1) > v` (on suppose que cette valeur existe).

b. Pour affiner l'estimation de `x`, on applique le même procédé avec `min = x`, `max = x + 0.1` et un pas de 0.01. Écrire la boucle correspondante.

c. On peut ainsi continuer à affiner l'estimation en réduisant l'intervalle de recherche et le pas.

Écrire le programme sous forme de deux boucles imbriquées :

- la boucle interne parcourt l'intervalle pour trouver l'estimation de `x` ;
- la boucle externe met à jour `min` et `max` et le pas, jusqu'à ce que l'estimation soit de l'ordre de 1E-6.

d. Appliquer l'algorithme à la fonction `sin(x)` et la valeur `v = 0.5`. Vérifier que la solution est proche de 30° $\left(\frac{\pi}{6}\right)$.

43 a. Les deux programmes suivants sont-ils équivalents ?

```
for i in range(10):
    print(i)
```

```
i = 0
while i < 10:
    print(i)
    i = i + 1
```

b. Est-ce toujours le cas si l'on insère l'instruction `i = i * 2` avant `print(i)` ?