Exercice 1 :

Écrire les réponses sur papier, puis utiliser l'interpréteur python pour vérifier et surtout comprendre les différences.
Quelle est la valeur de chacune des expressions suivantes une fois entrées dans l'interpréteur Python ?

code.py

```
>>> 4 // 5 * 3 + 2 ** 3
...
>>> 4 // 5 * (3 + 2) ** 3
...
>>> 2 == 1 + 1
...
>>> 2 == 1 + 1 + 1
...
>>> (2 == 1 + 1 + 1) and (2 == 1 + 1)
...
>>>2 == 1 + 1 + 1 or 2 == 1 + 1
...
>>>1 == 0 // 0
...
>>>(not (0 == 0)) and 1 == 0 // 0
...
>>> "2" == '1' + "1"
...
```

Exercice 2 :

Écrire les réponses puis utiliser l'interpréteur python pour vérifier et surtout comprendre les différences.
Écrire sur papier tout ce qui est affiché par ce code python, puis le vérifier en éxécutant le script.

code.py

```
hauteur = 50
print("Est-il vrai que hauteur = 50 ?")
print(hauteur == 50)
b=56
print(b != 5 and b != 6)
print(b <= 5 or b >= 6)
print(b == 50+6 or b >= 6)
print(b >= 6 and b < 50+6)
```

Exercice 3 : Opérations sur les entiers.

Un groupe d'élèves souhaite former des équipes pour un tournoi sportif. Le nombre total d'élèves et la taille de chaque équipe sont donnés. Écrire un programme qui :

1. Demande les entrées nécessaires.
2. Calcule combien d'équipes complètes peuvent être formées.
3. Détermine combien d'élèves resteront sans équipe.
4. Demande à l'utilisateur si il y a un autre calcul à faire ou pas.

Exemple d'affichage possible :

Entrez le nombre total d'élèves : 23
Entrez la taille d'une équipe : 5
Nombre d'équipes complètes : 4
Nombre d'élèves sans équipe : 3

Exercice 4 : Écrire un programme qui calcule le quotient et le reste de la division euclidienne de 2 entiers. Le programme ne doit pas utiliser les fonctions python % ou // . Remarque : vous pouvez utiliser la fonction **int()** si besoin et judicieusement.

Exercice 5 : Puzzle de Parson.

Remettre les commandes dans un ordre permettant à cette fonction de renvoyer le maximum parmi 4 entiers en Python.

code.py

```
return m
m = b
m = c
m = d
m : int = a
def max4(a : int ,b : int ,c :int ,d : int ) -> int :
if b > m :
if c > m :
if d > m :
```

Exercice 6 : Examiner la série de commandes ci-dessous et prédire le résultat de l'éxécution du script. Vérifier vos résultats avec l'interpréteur python.

code.py

```
x : int =1
print(x)
y: int = 2
x=x+y
y=x**y
print(y)
print(x,y)
```

Exercice 7 : Rédigez un programme qui calcule le prix d'une commande de soda pour un festival. Les trois valeurs suivantes sont représentées par les variables :

- ***nbr*** : entier désignant le nombre de fûts commandés
- ***prix*** : prix unitaire d'un fût.
- ***reduc*** : coefficient (entre 0 et 1) représentant la réduction dont bénéficie le client.

Le programme affiche le montant de la facture : $\boxed{m = nbr \times prix \times reduc}$.

1. Affecter les variables comme indiqué ci-dessus dans le cas dune commande de 27 fûts dont le prix unitaire est de 22,95 euros pour un client bénéficiant de 5 % de réduction.
2. Modifier le programme pour demander à l'utilisateur les valeurs des 3 variables.

Exercice 8 : Écriture d'un script qui :

1. Initialise deux entiers : a = 0 et b = 10 .
2. Écrit une première boucle affichant et incrémentant la valeur de a tant quelle reste inférieure à celle de b.
3. Écrit une deuxième boucle décrémentant la valeur de b et affichant sa valeur si elle est impaire. Boucler tant que b nest pas nul.

Exercice 9 :

Écrire un script qui demande un entier **n** et affiche ensuite à l'aide d'une boucle **while**, tous les entiers impairs inférieurs à **n**.

Exercice 10 :

Écrivez une fonction **entrelacement(s1,s2)** qui prend en paramètres deux chaînes de caractères **s1** et **s2** de même longueur et qui renvoie la chaîne qui contient en alternance un caractère de **s1** suivi d'un caractère de **s2**. Par exemple, **entrelacement('abc' , '123')** renvoie **a1b2c3**.

Exercice 11 :

1. Écrivez une fonction **nombre_apparitions(c , s)** qui prend en paramètres un caractère **c** et une chaîne de caractères **s** et qui renvoie le nombre de fois où **c** apparait dans **s**. Par exemple :

code.py

```
>>> nombre_apparitions('e','les merveilleuses')
5
```

2. En utilisant la fonction précédente, écrivez une fonction **absence_de_e(s)** qui prend en paramètre une chaîne de caractères **s** et renvoie **True** si **s** ne contient ni le caractère **e** ni **E**.
3. Écrire des tests de la fonction **absence_de_e(s)**.

Exercice 12 :

1. Écrire une fonction **affiche_miroir(s)** qui prend une chaîne de caractères **s** et qui affiche la chaîne **s** et son image miroir (**s** à l'envers).

SHELL

```
>>> affiche_miroir('abc')
abc cba
```

2. Écrire une fonction **miroir(s)** qui cette fois-ci retourne la chaîne de caractères **s** à l'envers, de sorte que l'on puisse répondre à la question 1 par :

code.py

```
def affiche_miroir(s):
        print(s,miroir(s))
```

3. Écrire une fonction **est_un_palindrome(s)** qui retourne **True** si **s** est un palindrome, autrement dit un mot qui est égal à son image miroir. Proposez une solution qui n'utilise pas la fonction **miroir** et qui ne crée aucune nouvelle chaîne de caractères. Écrire quatre tests avec **assert**.

Exercice 13 :

Créer dans votre répertoire de travail le fichier texte (**message.txt**)de 3 lignes comme ci-dessous.

Écrire un programme Python qui lit le fichier texte contenant plusieurs lignes et affiche le contenu ligne par ligne avec la précision du numéro de ligne (Ligne 1 : etc) comme dans l'exemple ci-après.

Contenu de message.txt :

```
Bonjour
Comment ça va ?
Bonne journée.
```

SHELL

```
>>> lit_fichier("message.txt")
Ligne 1 : Bonjour
Ligne 2 : Comment ça va?
Ligne 3 : Bonne journée
```

Exercice 14 :
Écrire un programme qui lit un fichier texte et crée un nouveau fichier où toutes les lignes vides sont supprimées.

Exercice 15 :
Écrire un programme qui demande à l'utilisateur de saisir son prénom, son âge et sa ville, puis affiche ces informations sous forme de phrase formatée avec une **f-string**.

Exercice 16 :
Écrire un programme qui demande à l'utilisateur de saisir la longueur et la largeur d'un rectangle, calcule sa surface, et affiche un message contenant le résultat formaté avec une **f-string**.

Exercice 17 :
Écrire un programme qui demande à l'utilisateur un entier **"n"**, puis affiche la table de multiplication correspondante jusqu'à 10, avec un affichage formaté en utilisant les **f-string**.

Exercice 18 :
Écrire un programme qui demande à l'utilisateur de saisir un nombre entier et indique s'il est pair ou impair.

Exercice 19 :
Écrire un programme qui demande à l'utilisateur de saisir une note (entre 0 et 20) et affiche un message basé sur cette note :

- Si la note est supérieure ou égale à 16, le programme affiche "Très bien".
- Si elle est entre 12 (inclus) et 16 (exclus), le programme affiche "Bien".
- Si elle est entre 8 (inclus) et 12 (exclus), le programme affiche "Passable".
- Sinon, le programme affiche "Insuffisant".

Exercice 20 : Écrire un programme qui demande à l'utilisateur de saisir uniquement un entier. Le programme vérifie si cet entier satisfait une ou plusieurs des conditions suivantes et affiche un message approprié.
Les conditions à tester sont :

- L'entier est divisible par 3.
- L'entier est pair.
- L'entier est strictement positif.
- L'entier est compris entre 10 et 50 inclus.

Exercice 21 : Écrire un programme qui demande à l'utilisateur de saisir uniquement un entier. Le programme vérifie si cet entier satisfait la condition suivante et affiche un message approprié.

- Le nombre est un multiple de 5 mais pas de 10.

Le programme doit demander à l'utilisateur s'il souhaite tester un autre nombre après chaque exécution.

Exercice 22 :
Écrire un programme qui lit un fichier texte et compte le nombre total de mots qu'il contient.