I. Comparaison des nombres.

Nous pouvons utiliser des comparaisons pour vérifier si un nombre est inférieur ou supérieur à un autre nombre.
Pour vérifier si un nombre est inférieur à un autre nombre, nous utilisons l'**opérateur inférieur à** `<` .

```
script.py

1 < 90
```

Si le nombre à gauche est inférieur au nombre à droite, comme dans *1 < 235*, le résultat est *True*

```
script.py

print(1 < 235)

Résultat

True
```

Si le nombre à gauche n'est pas inférieur au nombre à droite, comme dans *235 < 1*, le résultat est *False*.

```
script.py

print(235 < 1)

Résultat

False
```

Pour vérifier si un nombre est supérieur à un autre nombre, nous utilisons l'opérateur supérieur à, `>` .

```
script.py

print(101 > 90)

Résultat

True
```

Exercices

1. À quoi sert l'opérateur `>` ?
a) Pour vérifier si un nombre est égal à un autre nombre

b) Pour vérifier si un nombre est supérieur à un autre nombre

2. Comment appelons-nous le signe > ?
a) L'opérateur supérieur
b) L'opérateur inférieur

3. Qu'est-ce que ce code affiche dans la console ?
a) False
b) True

```
script.py

print(10 < 1)

Résultat

False
```

4. Qu'est-ce que ce code affiche dans la console?
a) True
b) False

```
script.py

print(1 > 1)

Résultat

False
```

5. Vérifiez si 5 est supérieur à 4.

```
script.py

print(5 > 4)

Résultat

True
```

6. Vérifier si 30 est inférieur à 40 avec l'opérateur inférieur, <.

```
script.py

print(30 < 40)

Résultat

True
```

7. Faire afficher ce code *True* dans la console.

script.py

```
print(10 > 9)
```

Résultat

```
True
```

8. Afficher *False* dans la console en codant l'opérateur > .

script.py

```
print(100 > 200)
```

Résultat

```
False
```

II. Comparer les strings.

Nous pouvons utiliser des comparaisons pour vérifier si une chaîne est égale ou non à une autre chaîne.

script.py

```
print("online" == "online")
print("online" != "offline")
```

Résultat

```
True
True
```

Pour vérifier si une chaîne est égale à une autre chaîne, nous utilisons également l'opérateur d'égalité, ==.

script.py

```
print("apple" == "apple")
```

Résultat

```
True
```

Si la chaîne de gauche est égale à la chaîne de droite, comme dans « *apple* » == « *apple* », le résultat est *True*.

script.py

```
print("apple" == "apple")
```

Résultat

```
True
```

Si la chaîne de gauche n'est pas égale à la chaîne de droite, comme dans *"apple" ="orange"*, le résultat est *False*.

script.py

```
print("apple" == "orange")
```

Résultat

```
False
```

On peut aussi comparer des variables qui stockent des chaînes entre elles, comme dans *fruit_1 == fruit_2*.

script.py

```
fruit_1 = "apple"
fruit_2 = "orange"

print(fruit_1 == fruit_2)
```

Résultat

```
False
```

III. Comparaison de types

Nous avons déjà vu quelques types de données comme des nombres et des chaînes. En termes de programmation, ces valeurs sont appelées **types**.

Les chaînes sont des caractères entre guillemets « », comme la valeur « *High* »

script.py

```
sugar_content = "High"
```

Entier est un autre type que nous avons déjà utilisé. Il représente des nombres entiers sans décimales comme *42*.
Float est un autre type de nombre que nous utilisons pour décrire des nombres avec un ou plusieurs chiffres après le point décimal, comme *3.14159*.

```
script.py

score = 42
```

Le type booléen ne contient que deux valeurs : True et False. Nous stockerons la valeur True dans cette variable.

```
script.py

pie = 3.14159
```

Lors du stockage de False dans *is_on*, nous disons que nous attribuons une valeur à une variable.

```
script.py

is_on = False
```

Exercices

1. Quelles sont les valeurs comme les booléens, les chaînes et les nombres appelés?
a) Variable
b) Types

2. Comment reconnaître les valeurs de type string ?
a) Par le fait qu'il s'agit toujours de mots ou de lettres
b) Par les guillemets « » qui les entourent

3. Qu'est-ce qu'on appelle des nombres entiers sans décimales ?
a) Nombres entiers
b) Nombres complexes

4. Quel type est stocké à l'intérieur du résultat?

```
script.py

result = 3.33
```

a) Un entier
b) Un float

5. Quelles sont les valeurs booléennes ?
a) « True » et « False »
b) True et False

6. Qu’est-ce qu’une affectation variable?
a) Stocker une valeur dans une variable
b) Donner un nom à une variable

7. Assigner une chaîne avec un nom de variable, un âge en nombre entier et un booléen à is_active

script.py

```
name = "Joey"
age = 28
is_active = True
```

8. Enregistrer une valeur de chaîne dans le nom de la variable.

script.py

```
name = "Jill"
```

9. Enregistrer une valeur entière dans la variable.

script.py

```
minutes_left = 10
```

10. Enregistrer une valeur float dans la variable.

script.py

```
average_score = 65.55
```

11. Donner positive_score la valeur booléenne True

script.py

```
positive_score = True
```

12. Attribuer la valeur 11 à la variable number_of_appearances

script.py

```
number_of_appearances = 11
```