Ce qu'il faut savoir sur les opérations.

Concernant les opérations arithmétiques.

Les opérations d'addition (+), de soustraction (-), de multiplication (*) et de division (/) sont **les opérations les plus élémentaires que vous avez apprises durant votre cursus scolaire.**

Je peux en utiliser autant que je le souhaite dans une instruction qu'ils s'agissent directement de int ou de int stocké dans une variable.

Ainsi, dans le code suivant :

var myNumber = 4;var result = 1 + myNumber * myNumber / 2;

`result` vaudra donc ... ?

9.

Si vous ne le saviez pas déjà, il y a **une priorité dans l'ordre dans lequel va se dérouler l'opération.**

Ce sont les **multiplications** et les **divisions** qui vont s'effectuer en premier (le premier signe rencontré en partant de la gauche)

Puis les soustractions et les additions.

Donc :

1. D'abord c'est la multiplication qui va s'effectuer. (4 * 4) ce qui donne 16.
2. Ensuite c'est la division (16/2) ce qui donne 8.
3. Pour ensuite ajouter 1. (8 + 1) donne 9.

Si vous voulez vous même prioriser les opérations vous devez utiliser les parenthèses !

Une astuce à connaitre :

Supposons que :

```
var myNumber = 1;
```

Faire :

```
myNumber = myNumber + 3;
```

Peut être synthétisé en

```
myNumber += 3;
```

Ce qui est également le cas pour

myNumber = myNumber * 4;myNymber *= 4; myNumber = myNumber - 10;myNumber -= 10; myNumber = myNumber / 2;myNumber /= 2;

🚨Une information très importante 🚨

Supposons le code suivant :

```
var result = 1 / 2;
```

Combien vaut `result` ?

1) Facile : 0,52) Je sais pas mais je sens qu'il y a un piège.

Il y a un piège effectivement. `result` vaudra 0.

Si nous calculons comme des humains, nous savons que 1/2 vaut 0,5. Mais alors pourquoi le code calcule-t-il ça à 0 ?Parce que nous travaillons avec des `int` (pour integer) qui sont **des nombres ENTIERS** comme leur nom l'indique. Donc, la variable `result` sera aussi un `int`. Il ne prendra que la fraction entière de 0,5 à savoir 0 !

Si je veux correctement travailler avec des nombres décimaux je vais devoir utiliser des `float` (par exemple) en écrivant :

```
var result = 1.0f/2.0f;
```

👉 "Je n'ai pas très bien compris ce qu'est le Modulo ?"

Pas d'inquiétude, c'est normal. Le modulo est le reste de la division euclidienne de deux nombres. ça c'était pour la définition mathématique. Mais concrètement ?

Soit l'opération suivante.

14 divisé par 3.

Si je prends ma calculatrice et que je réalise l'opération, je vais avoir 4,6666666666667. Mais dans une division euclidienne, on travaille avec des entiers (**donc pas de chiffres à virgule**). Le résultat 4,6666666666667 n'est pas convenable.

En réalité, si je veux savoir combien de fois j'ai le chiffre 3 à l'intérieur de 14, Je peux raisonner à rebours et faire :

3 * 4 qui me vaudra 12 auquel j'ajoute 2. Donc pour avoir 14 je peux faire (3*4) +2.

4 est ce qu'on appelle le quotient et **2 le reste.**

Vous m'argumenterez que l'on peut faire (3*5) -1 ou encore (3*2) + 8 mais quand on raisonne **en division euclidienne il faut que le quotient soit le plus grand possible et le reste soit supérieur à 0.**

Sachant tout ça, si je fais 14 % 3 (et non plus une division). Mon résultat sera ... 2. Parce que 2 est le reste de (3 * 4) +2.

D'autres exemples pour que vous compreniez :

18 % **5** donnera 3 car on peut décomposer 18 en (3***5**)**+3**.

17 % **2** donnera 1 car on peut décomposer 17 en (8***2**)**+1**.

28 % **4** donnera 0 car on peut le décomposer 28 en (7***4**)**+0**.

Ok j'ai compris, mais, franchement, à quoi ça sert ?

Je prends de l'avance sur les cours d'après mais si j'écris l'instruction 👇 :

```
myVariable % 2 == 0
```

Je peux savoir si `myVariable` est divisible par 2 (en d'autres termes si `myVariable` est pair). **Donc le modulo sert à savoir si un nombre est divisible**

par un autre. Et si ce nombre est deux, à savoir si la variable est pair ou impair.