

Semaine 1

Environnement de travail et introduction à Javascript

Intro. à la programmation - Aut. 2022



❖ Présentation

- ◆ Plan de cours et fonctionnement 

❖ Plateformes scolaires

- ◆ Léa
- ◆ Ordinateurs du cégep

❖ Environnement de travail

- ◆ Dossiers, fichiers, compression et infonuagique

❖ Introduction à Javascript

- ◆ Opérations mathématiques, expressions et variables



❖ Omnivox - Léa

◆ Petite démo pour ...

- Récupérer les notes de cours
- Récupérer les laboratoires
 - Remettre les laboratoires

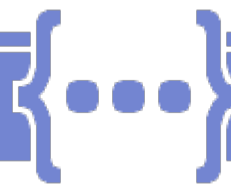
❖ Ordinateurs du cégep

◆ Tous les logiciels dont on a besoin sont installés dessus !

- Petite démo : Où ranger nos fichiers pour ne pas les perdre ?

◆ Installer le logiciel Visual Studio Code à la maison

- Document sera disponible sur Léa pour la procédure

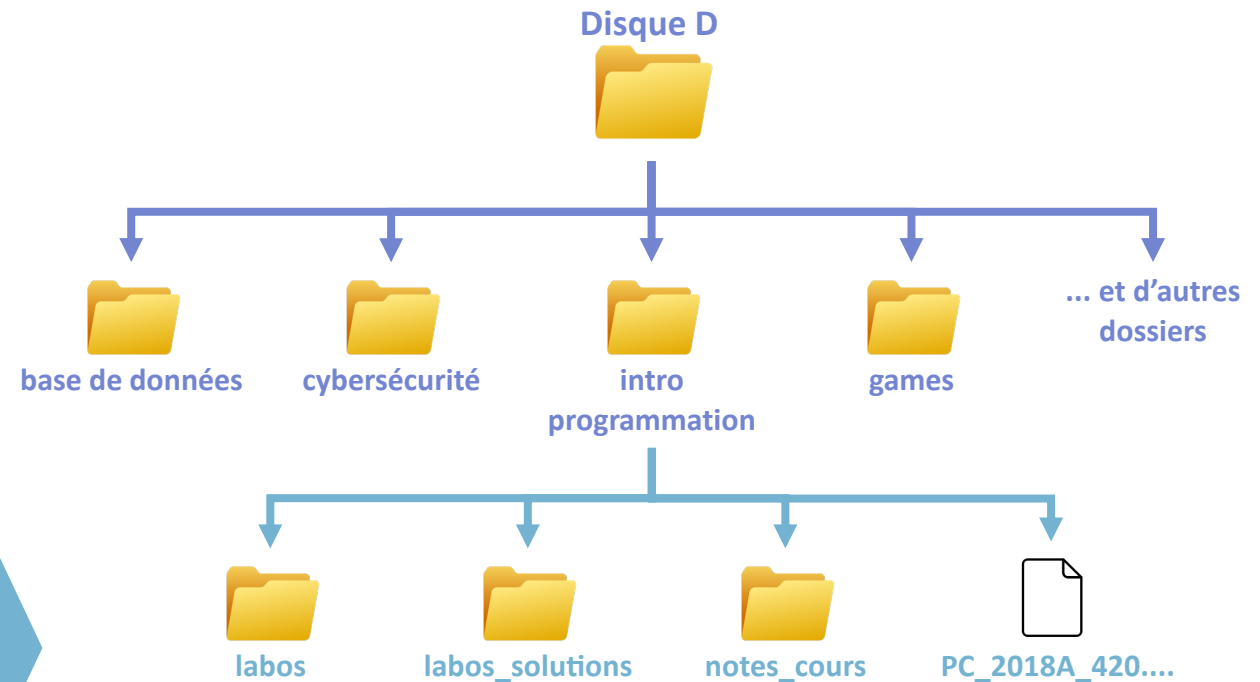


❖ Arborescence de dossiers

- ◆ Dans un système d'exploitation, les dossiers et fichiers sont organisés en arborescence (Ou en hiérarchie..)

Dossier « Disque D »

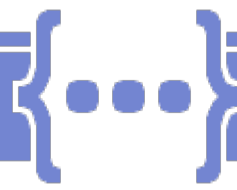
Ce PC > Disque D (D:) >		
Nom	Modifié le	Type
base de données	2020-12-19 13:00	Dossier de fichiers
cybersécurité	2021-04-23 19:13	Dossier de fichiers
games	2021-02-06 18:33	Dossier de fichiers
intro programmation	2021-07-30 15:58	Dossier de fichiers
Introduction aux bases de données	2021-04-09 00:51	Dossier de fichiers
logiciels de bureautique	2021-06-10 14:04	Dossier de fichiers
Programmation Web Serveur	2021-06-05 14:14	Dossier de fichiers



Dossier « intro programmation »

Ce PC > Disque D (D:) > intro programmation >		
Nom	Modifié le	Type
labos	2021-07-29 20:18	Dossier de fichiers
labos_solutions	2021-07-29 20:18	Dossier de fichiers
notes_cours	2021-07-30 16:04	Dossier de fichiers
PC_2018A_420-905-EM_Département-Inf...	2021-07-28 20:34	Document Micros...

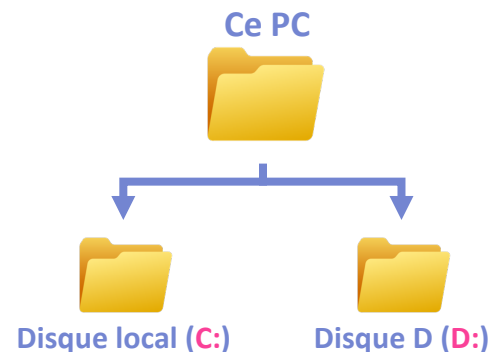
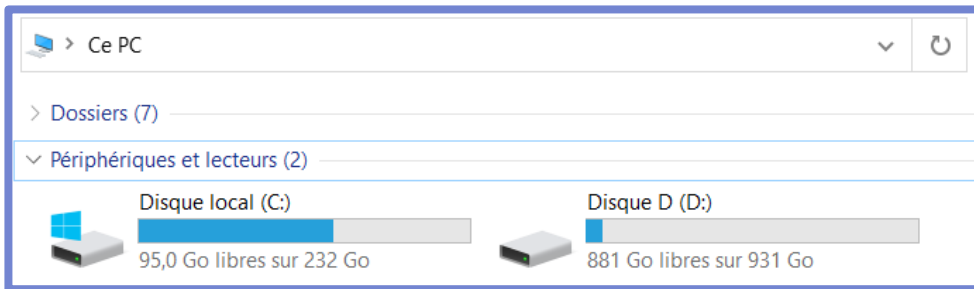




❖ Arborescence de dossiers

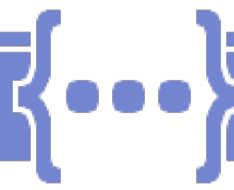
- ◆ La *racine* : C'est le tout début de l'arborescence, le « dossier qui contient tous les dossiers »
 - Sur **Windows 10**, il est nommé « **Ce PC** », par exemple.

Dossier racine : « Ce PC »



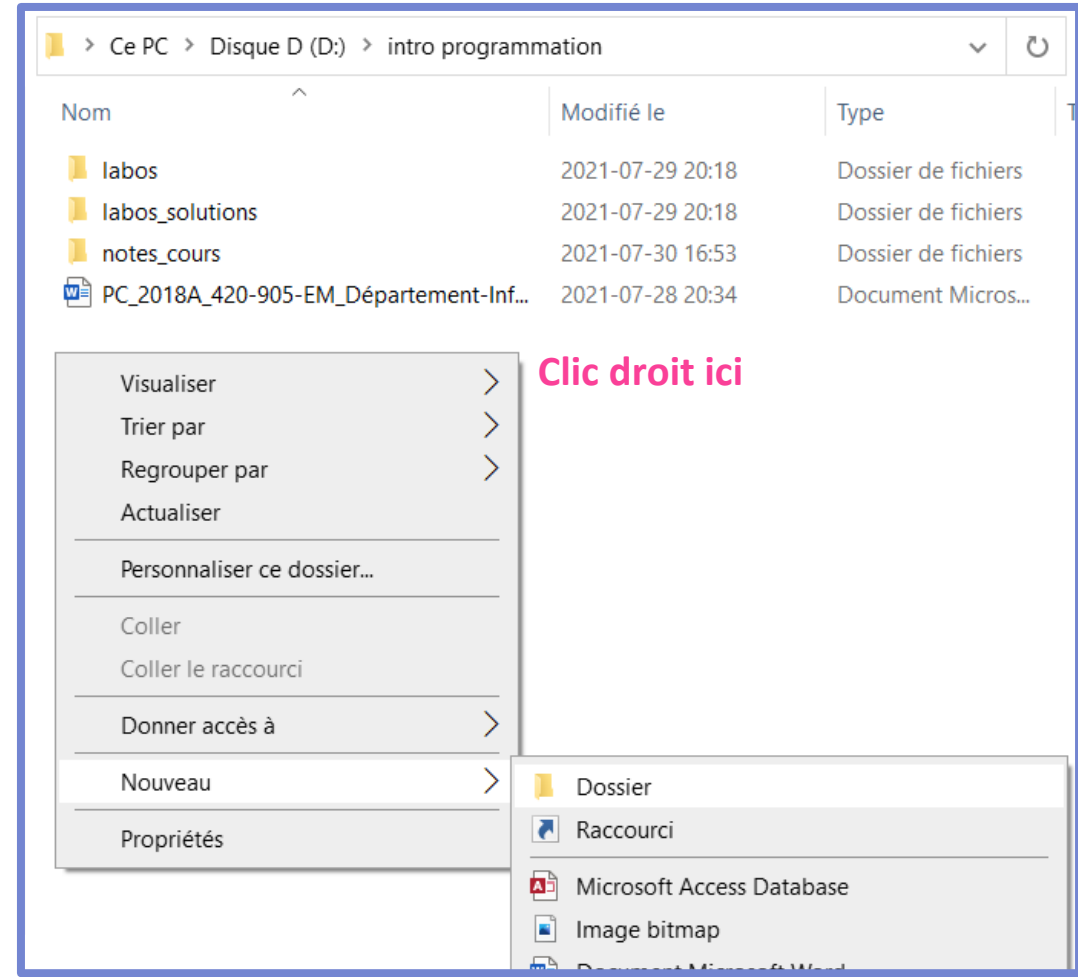
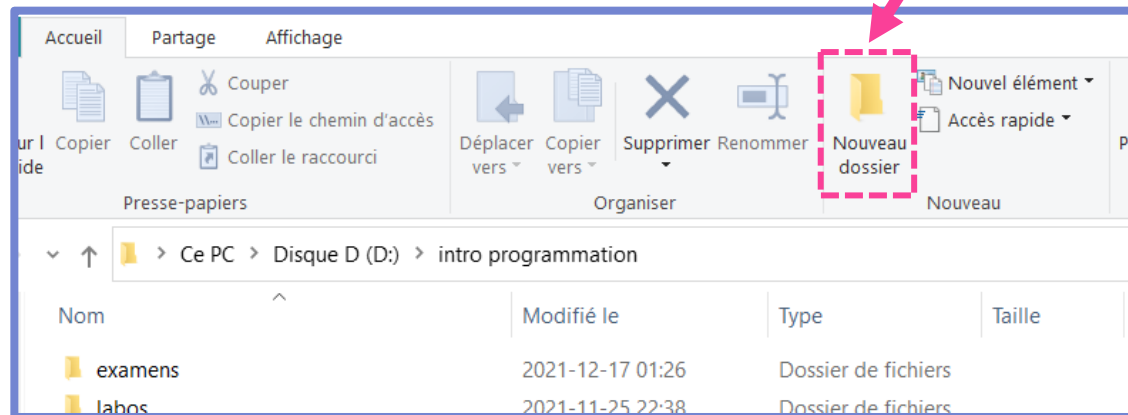
❖ Dans cet exemple, il y a deux « Disques » (**C:** et **D:**)

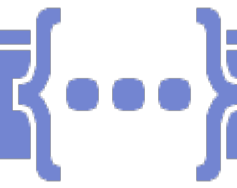
- ◆ Ce sont les deux disques qui stockent les données de l'ordinateur !
- ◆ Généralement, il n'y a qu'un seul disque (Le **C:**)
- ◆ Si on branchait une **clé USB** dans l'ordinateur, on verrait qu'un nouveau « **Disque** » apparaîtrait. (**E:**, **F:**, **G:**, ou autre ...)



❖ Créer un dossier

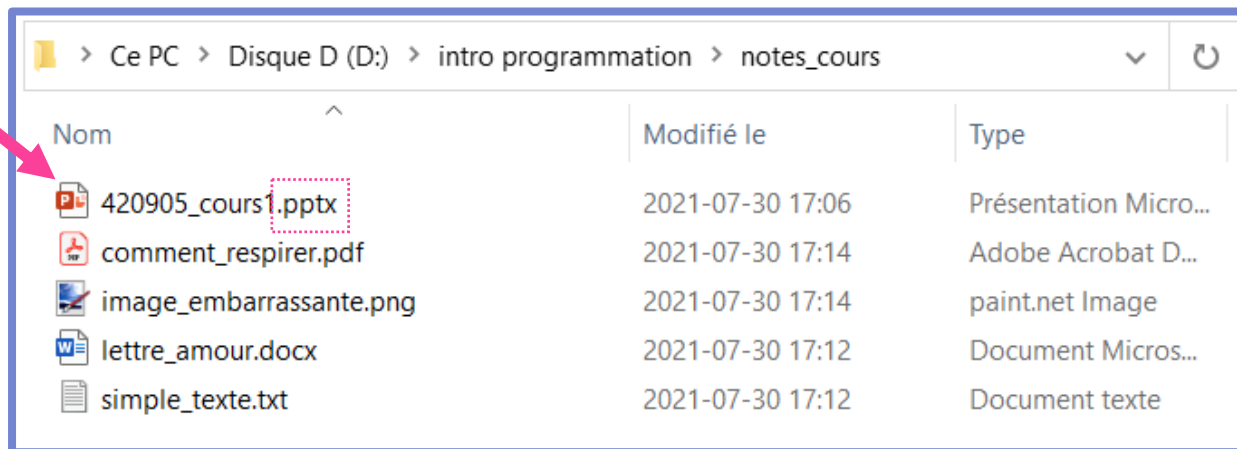
- ◆ Se rendre au dossier dans lequel on souhaite ajouter un dossier.
- ◆ Faire un **clic-droit** sur le fond blanc du dossier, puis...
 - Choisir « **Nouveau → Dossier** »
 - Nommez-le !

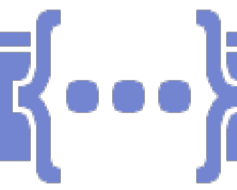




❖ Extensions de fichier

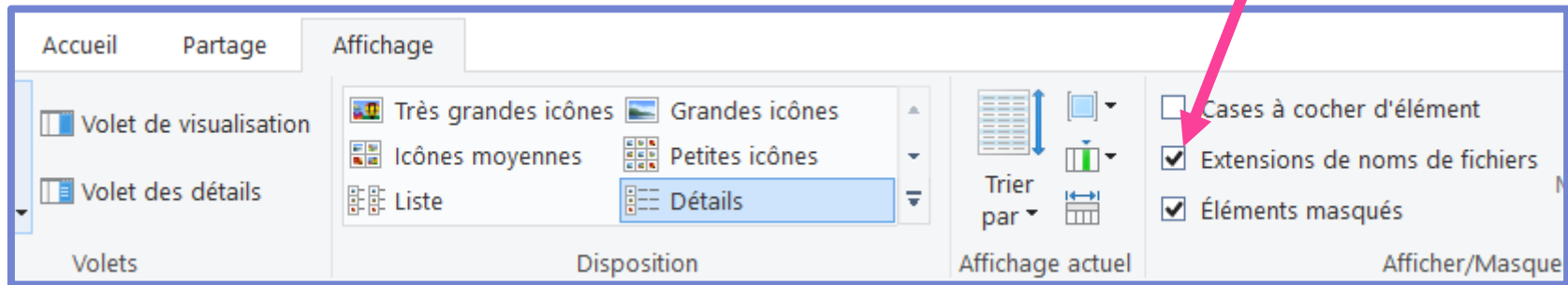
- ◆ Indiquent le **type de fichier** d'un document. Quelques exemples ...
 - **.pptx** : Présentation Microsoft Powerpoint
 - **.pdf** : Document texte / image non modifiable
 - **.png** (Ou encore **.jpeg**, **.bmp**, **.gif**, etc.) : Image
 - **.docx** : Document texte avec Microsoft Word
 - **.txt** : Simple fichier de texte
- ◆ Les icônes à gauche des fichiers peuvent également indiquer le **type**.





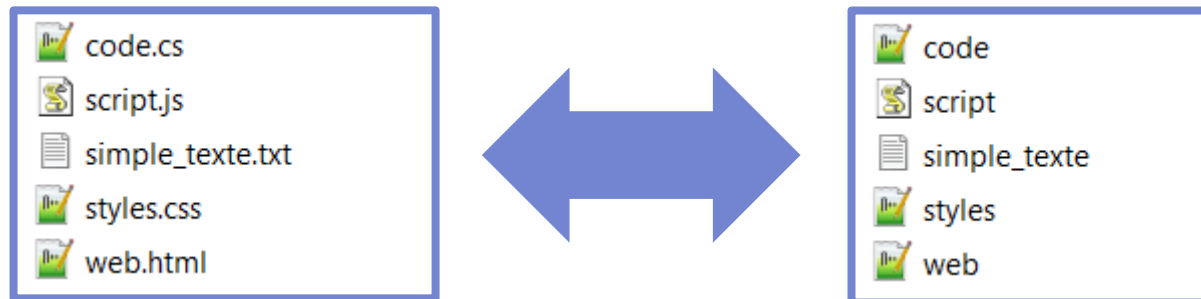
❖ Extensions de fichier

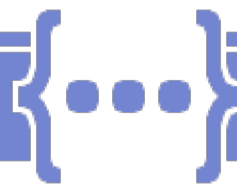
- ◆ Assurez-vous d'afficher les **extensions de fichier** s'ils sont cachés. 🙈



Le menu « **Affichage** » est disponible depuis n'importe quel dossier !

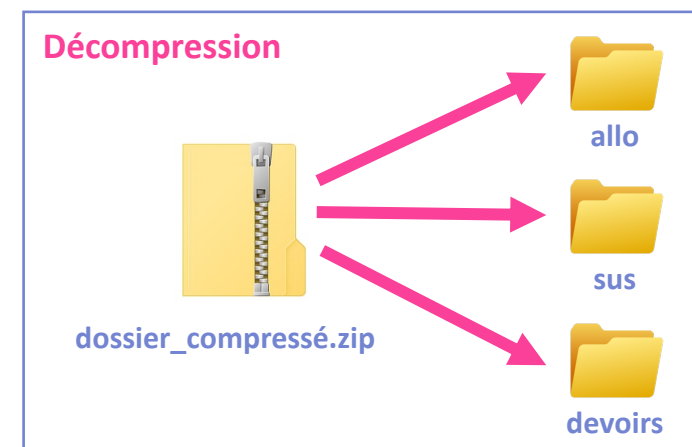
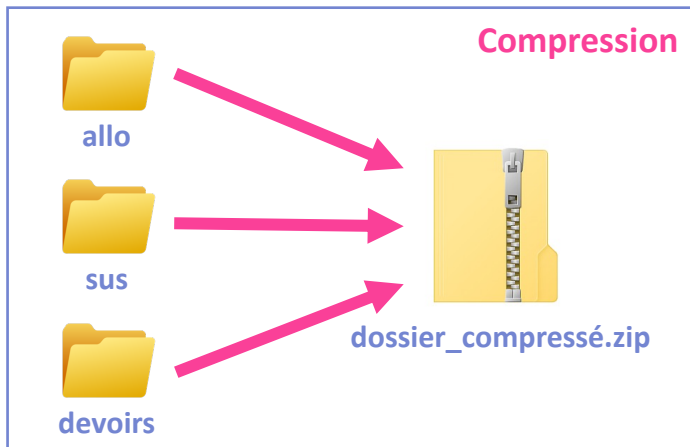
- ◆ Sinon il peut être difficile de différencier certains types ...

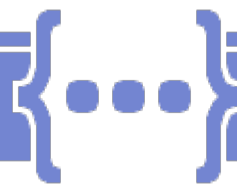




❖ Compression de fichier

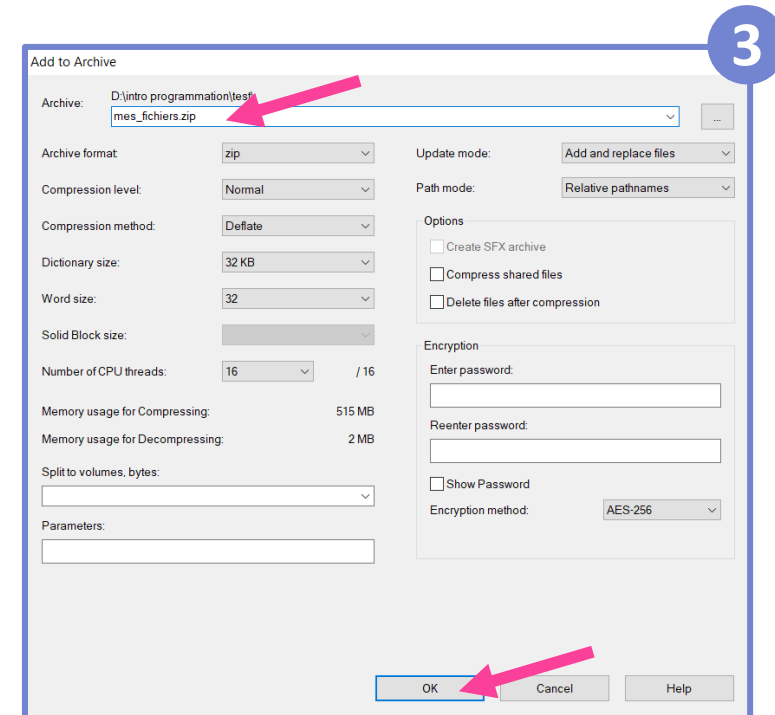
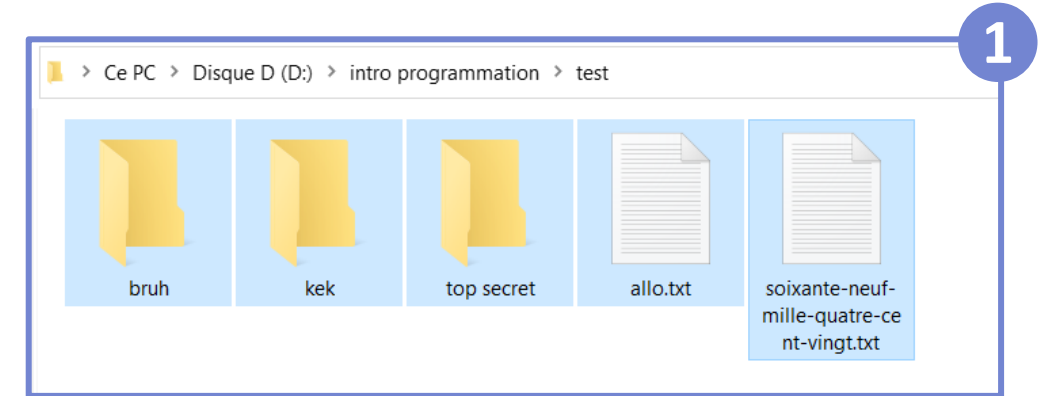
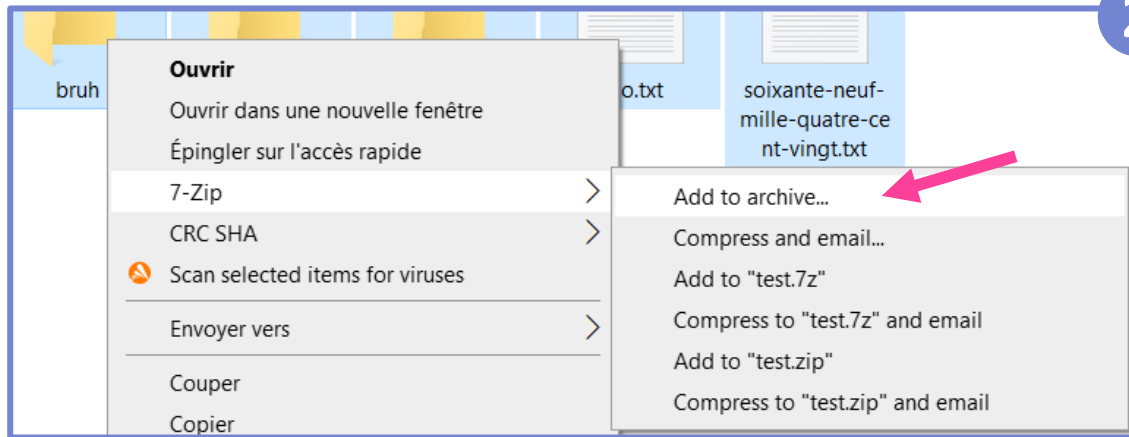
- ◆ C'est une action qui permet de « **regrouper des fichiers / dossiers** »
 - Réduit potentiellement **leur taille** (en données)
 - Permet de « **Partager** » / « **Envoyer** » un ou plusieurs dossiers et leur contenu. Par exemple...
 - L'upload / le téléverser sur Léa 🎓
 - L'envoyer par courriel ✉
 - Le stocker dans le Cloud (Dropbox, Google Drive, etc.) ☁
 - Lorsque **compressés**, les fichiers ne peuvent pas toujours être utilisés !
 - Il faut d'abord les « **décompresser** ».

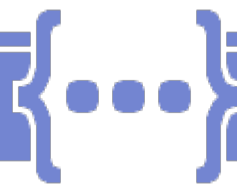




❖ Compresser des fichiers

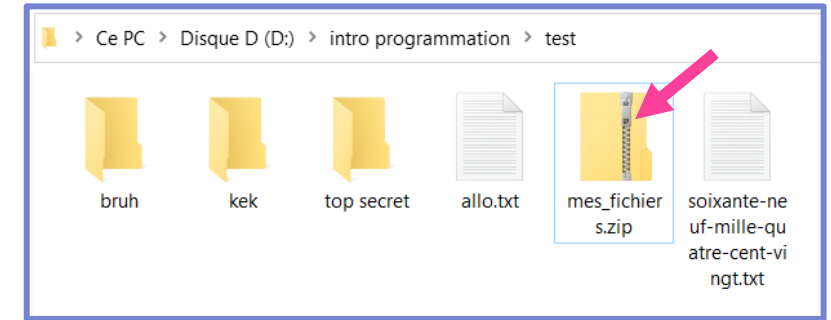
1. Sélectionner les fichiers et / ou dossiers
2. **Clic-droit** sur un de ces fichiers → **7-Zip** → « **Ajouter à l'archive** »
3. Choisir un nom et appuyer sur « **OK** ». (Ignorez le reste des options)





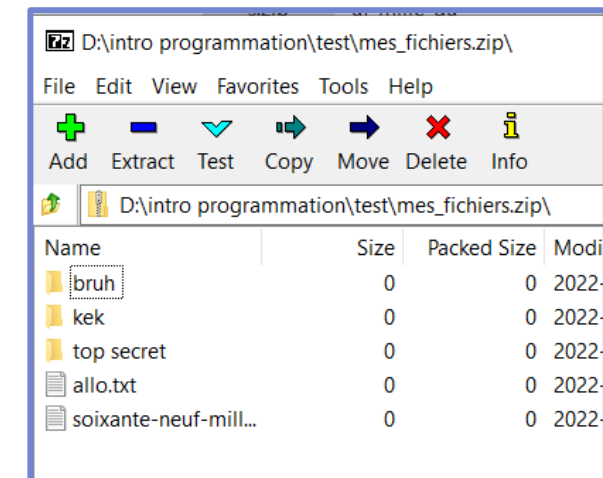
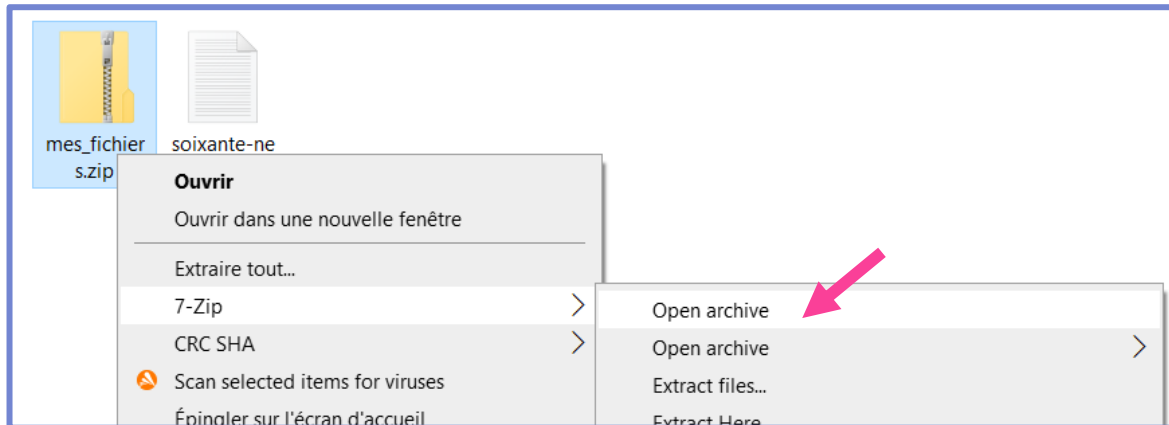
❖ Compresser des fichiers

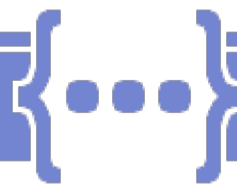
- ◆ On obtient un nouveau dossier, qui est **compressé**. (On remarque l'extension **.zip**)
- ◆ On peut le **renommer**, tant que son nom se termine par **.zip**
- ◆ Ce dossier compressé contient une **COPIE** des documents qu'on a sélectionnés.



❖ Vérifier le contenu d'un dossier compressé

- ◆ Il suffit de faire un **clic-droit** sur le dossier compressé -> **7-Zip** -> « **Ouvrir l'archive** »

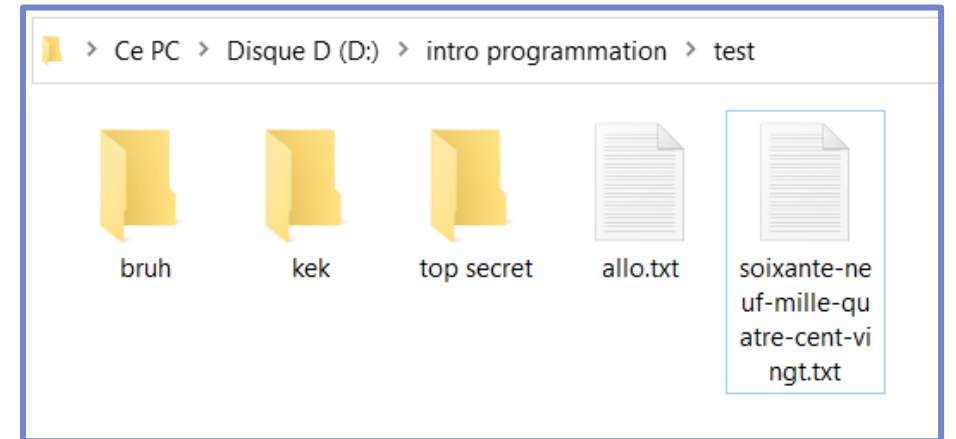
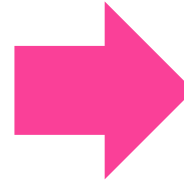
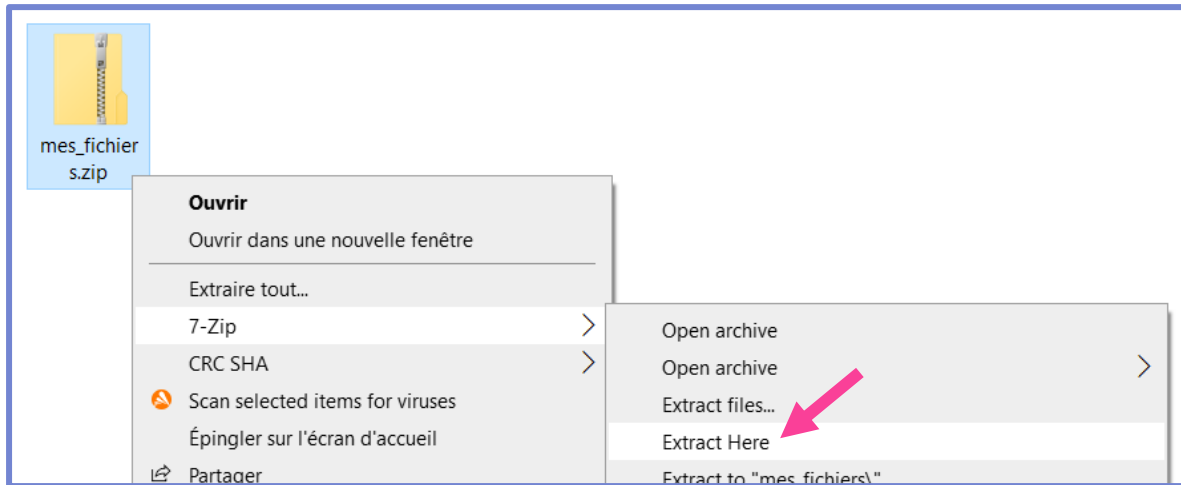


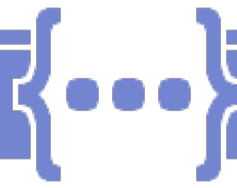


❖ Décompresser un fichier

◆ Permettra d'accéder et d'utiliser son contenu

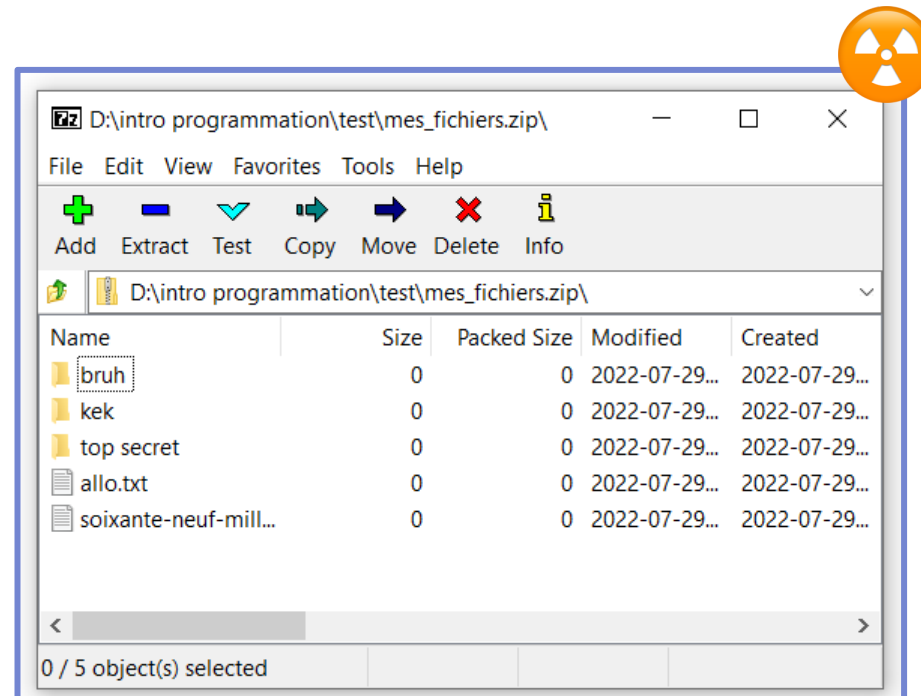
- **Clic-droit** sur le **fichier compressé** -> **7-Zip** -> « **Extraire ici** »
- On voit qu'on a retrouvé nos fichiers de départ !





❖ ATTENTION 🚫⚠️☣️😬😳😡

- ◆ Ne **JAMAIS** ouvrir / utiliser vos fichiers à partir de cette interface.
- ◆ Si vous tombez sur cette interface, cela signifie que vous êtes en train d'essayer de modifier des fichiers **actuellement compressés**.
 - Commencez par **décompresser** votre dossier avant d'utiliser son contenu !





❖ Qu'est-ce que Javascript ?

- ◆ Langage de programmation né en 1996 🧒
 - Les fichiers de code Javascript ont l'extension **.js**  script.js
- ◆ Très utilisé pour la programmation **Web** 🌐
 - La très très grande majorité des sites Web l'utilisent
 - Et c'est une des raisons qui en font un langage de choix pour apprendre à coder !
 - On peut utiliser ce langage dans les navigateurs **Web** ! (Firefox, Chrome, Edge, etc.)
 - **Nous allons le faire dans ce cours !**
- ◆ Exemples d'applications / projets qui utilisent **Javascript**

NETFLIX



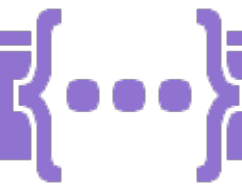
LinkedIn

Uber



2048





❖ Qu'est-ce que Javascript ?

◆ Exemple de morceau de code avec Javascript

- Pas très intuitif pour le moment ... ! 

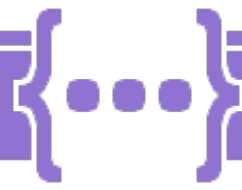
```
>
class Vehicle {
  constructor(make, model, color) {
    this.make = make;
    this.model = model;
    this.color = color;
  }

  getName() {
    return this.make + " " + this.model;
  }
}

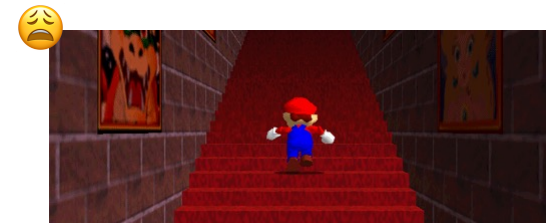
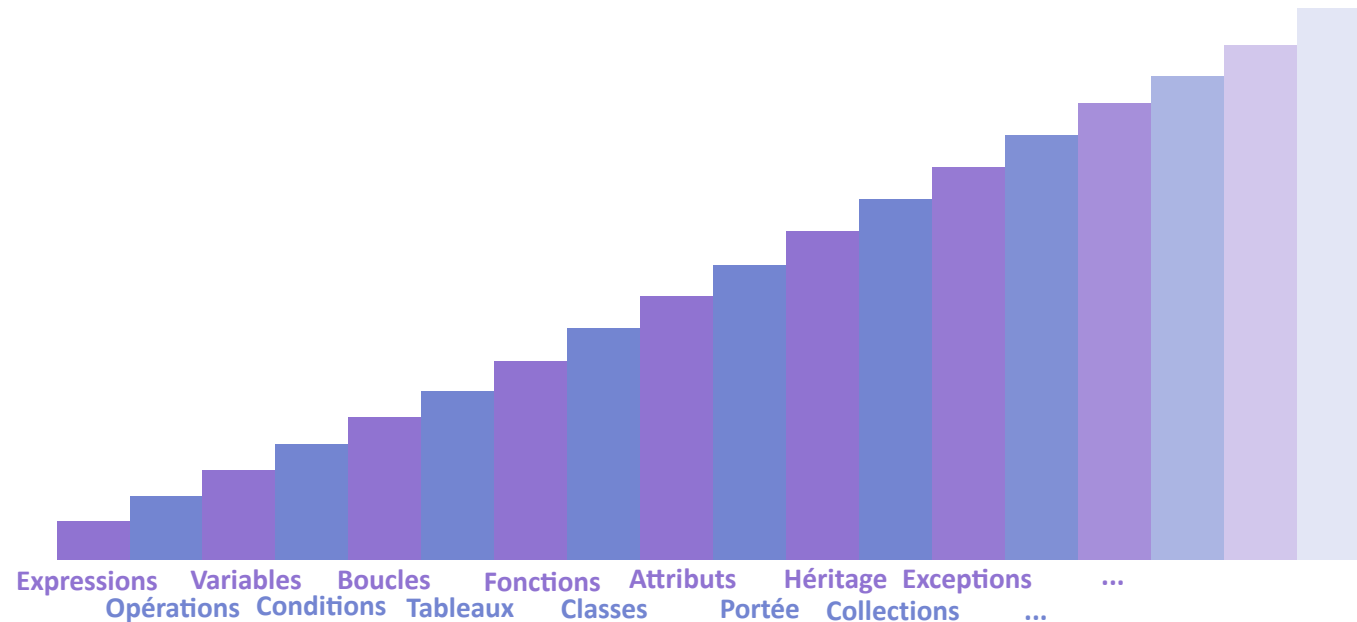
class Car extends Vehicle{
  getName(){
    return super.getName() + " - called base class function from child class.";
  }
}

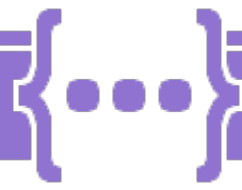
let car = new Car("Honda", "Accord", "Purple");

car.getName();
< "Honda Accord - called base class function from child class."
```



- ❖ Apprendre un langage de programmation
 - ◆ Il y a une **longue route** avant de pouvoir « coder des choses concrètes et complexes » comme des jeux, des sites Web et des applications.
 - Cette longue route est différente pour chaque type de projet, et nécessite parfois d'apprendre d'autres **langages de programmation** ou **technologies**.





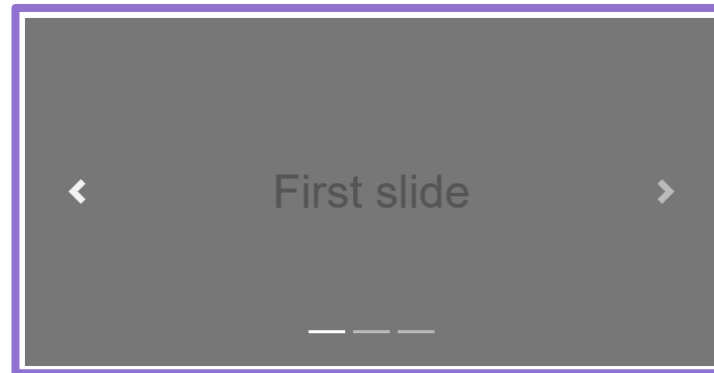
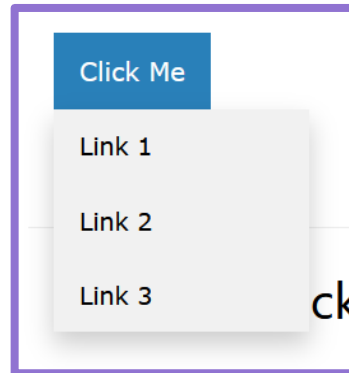
❖ Apprendre un langage de programmation

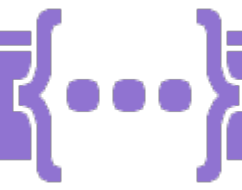
- ◆ Dans le cadre du cours, **Javascript** va nous permettre de modifier / interagir avec les éléments d'une **page Web** pour la rendre **interactive**.

- **Exemples**

- Un bouton change la couleur du texte
- Survoler un élément fait dérouler un menu avec plusieurs options
- Une galerie d'images qui alternent automatiquement

- ◆ Mais avant, nous avons beaucoup de notions à aborder pour pouvoir faire cela !





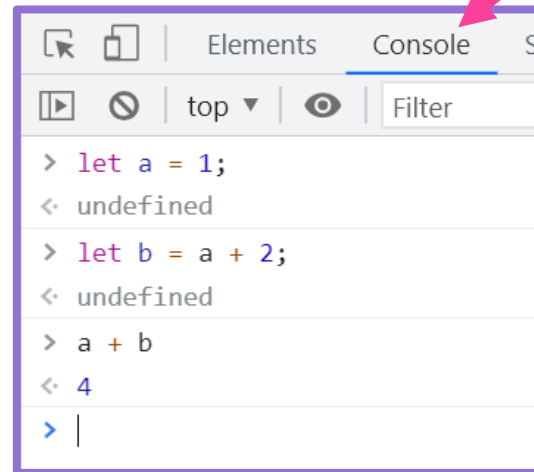
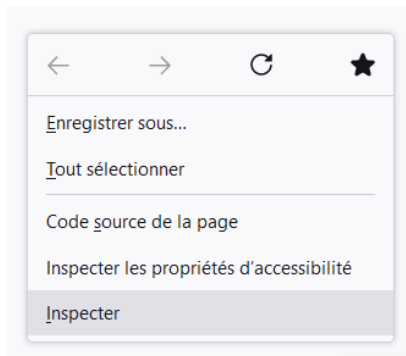
❖ Introduction à Javascript

- ◆ Javascript avec un navigateur Web
- ◆ Opérateurs arithmétiques de base
- ◆ Variables
 - Conventions de nommage, déclaration, affectation
- ◆ Types de données
- ◆ Autres opérateurs arithmétiques
 - Opérateurs d'affectation
 - Priorité des opérateurs
- ◆ Usage de variables
- ◆ Concaténation



❖ Javascript avec un navigateur Web

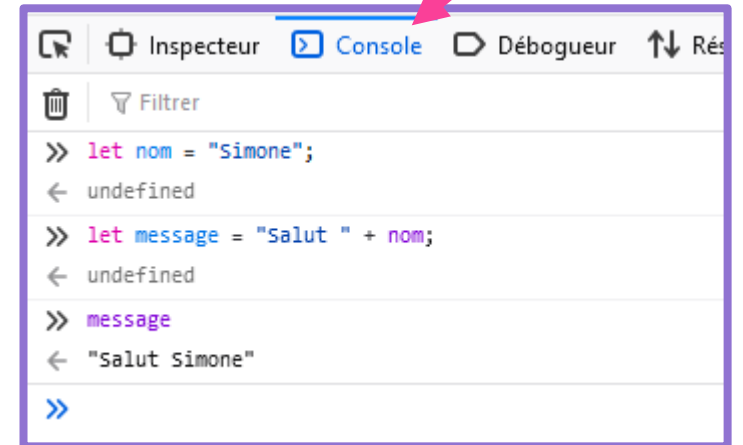
- ◆ Pour le moment, nous utiliserons la « Console du navigateur Web » de **Google Chrome** ou **Mozilla Firefox** pour pratiquer avec Javascript.
- ◆ Ouvrez un **navigateur Web** et appuyez sur **F12** (Ou faites clic-droit -> Inspecter -> Console)



Exemple avec **Google Chrome**



Vous pouvez
écrire du code
dans la **console** !



Exemple avec **Mozilla Firefox**





❖ Opérateurs arithmétiques

- ◆ Les programmes nécessitent souvent de faire des calculs mathématiques.
- ◆ Opérateurs simples :

- Addition +

» 1+3

← 4

» 5.5 + 3.5

← 9

» 7 + -3

← 4

- Soustraction -

» 3 - 4

← -1

» 5 - 1.5

← 3.5

» 2 - -2

← 4

- Multiplication *

» 2 * 3

← 6

» -2 * 1.5

← -3

» -1.25 * -4

← 5

- Division /

» 3 / 2

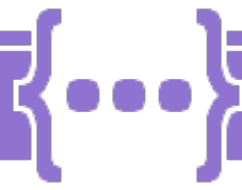
← 1.5

» 30 / 6

← 5

» -30 / 6

← -5



❖ Opérateurs arithmétiques

◆ Usage de **nombre**s décimaux

- Toujours utiliser un **point** (et non une virgule) pour séparer la partie entière de la partie décimale !

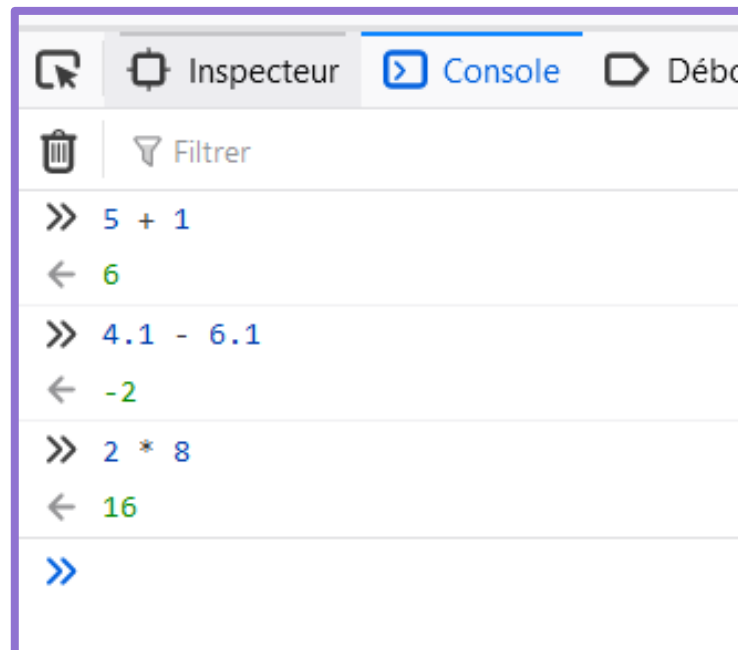
5.5

10.97

31.335

-52.5

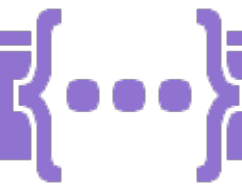
- On peut faire des calculs avec la console d'un navigateur ! 🥰🤔





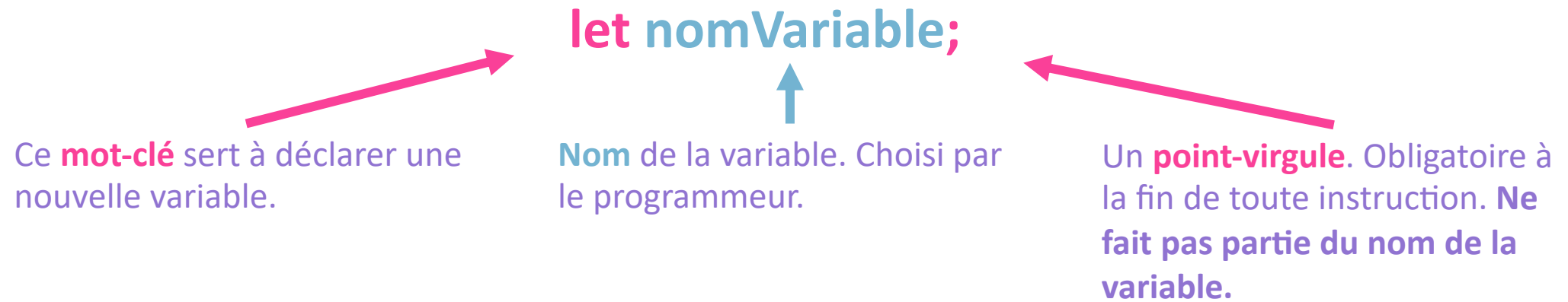
❖ Variables

- ◆ Espace dans la mémoire permettant de stocker une **donnée**
 - Dans un programme (ou logiciel, ou jeu, ou application, ...), on a parfois besoin de **stocker des informations** pour assurer son bon fonctionnement.
 - **Exemples**
 - Stocker les points de vie d'un personnage dans un RPG : « **78** » (points de vie) ❤️
 - Stocker la taille du pinceau utilisé dans Photostop : « **5.5** » (pixels) 🎨
 - Stocker le nom d'un item dans un jeu mobile « **Emerald** » 💎
 - Stocker votre niveau de concentration en classe « **10** » (%) 😬
 - Les **variables** servent exactement à stocker ce genre de données !



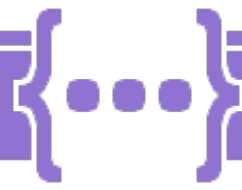
❖ Déclarer une variable

- ◆ Déclarer une variable permet de la créer et de pouvoir l'utiliser par la suite dans le code.
- ◆ Il faut utiliser la forme suivante pour déclarer une variable :



- ◆ Exemple

let prixRubis;



❖ Déclarer une variable

- ◆ Petite précision : Parfois, la **console** nous répond ← undefined
 - C'est normal ! 😊👏

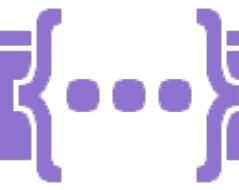
```
>> 5 + 4  
← 9
```

Quand on fait un calcul, la **console** **DOIT** nous répondre quelque chose : le résultat de l'opération !

```
>> let a;  
← undefined
```

Parfois, la **console** n'a rien de précis à nous répondre. (Ex. Quand on déclare une variable)
Dans ce cas, elle nous répond juste « undefined »





❖ Affecter une **valeur** à une variable

- ◆ Permet de stocker une information dans une variable
- ◆ L'affectation utilise l'opérateur =

```
>> prixRubis = 800;
```

Nom de la variable affectée
Placé à gauche du =

Valeur affectée
Placée à droite du =

- ◆ Une fois la **variable déclarée** et **affectée**, on peut demander à la console de nous dire ce qu'elle contient.

```
>> let prix = 5;
```

```
← undefined
```

```
>> prix
```

```
← 5
```

← On a **déclaré** et **affecté** une valeur à la variable **prix**.

← On écrit « **prix** » dans la **console** et elle nous rappelle ce que cette variable contient.



❖ Déclarer et affecter

- ◆ On peut **déclarer** plusieurs **variables** d'un coup (Séparées par des **virgules**)

```
>> let variable1, variable2, variable3;
```

- ◆ On peut **déclarer** et **affecter** immédiatement

```
>> let ageSylvain = 102;
```

- ◆ On peut **déclarer** et **affecter** plusieurs **variables** d'un coup

```
>> let triangle = 3, carre = 4, pentagone = 5;
```



❖ Déclarer et affecter

- ◆ On peut **affecter** une valeur pour **écraser** / **effacer** une précédente valeur

```
>> let prixSapphire = 700;  
    prixSapphire = 800;
```

← **prixSapphire** contient donc la valeur **800**
à partir de maintenant plutôt que **700** !

- ◆ On peut **affecter** une valeur à une variable en faisant un calcul

```
>> let prixDeTroisChiens = 50 * 3;
```



```
← undefined
```

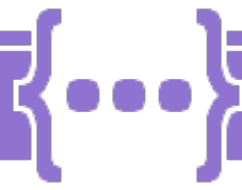
```
>> prixDeTroisChiens
```

```
← 150
```



❖ Noms de variables

- ◆ Chaque variable possède un nom unique qui permet de la distinguer
- ◆ **Règles** de nommage (obligatoires)
 - Doit commencer par une **lettre**.
 - Peut contenir des **lettres**, des **chiffres** et des **traits de soulignement** _
 - **Ne** peut **pas** contenir d'**espace** ni d'autres **caractères spéciaux** (?!#/%&*~\)
- ◆ **Conventions** de nommage (fortement suggérées)
 - Le nom d'une variable doit être **significatif**.  **nomDragon**, **prix**, **age**,  **abc**, **Imao**, **hm**, **p**
 - Si le nom est composé de plusieurs mots, **le premier commence par une minuscule** et les suivants par des **majuscules** (**birthDate**, **numberOfStudents**, **pointsDeVie**, ...)



❖ Noms de variables

- ◆ Chaque variable possède un nom unique qui permet de la distinguer
 - Voici ce qui arrive si deux variables ont le même nom ...

```
>> let a = 4;
```

```
← undefined
```

```
>> let a = 3;
```

❗ ▶ Uncaught SyntaxError: redeclaration of let a
 <anonymous> debugger eval code:1
 [\[En savoir plus\]](#)

Erreur ! Le programme n'est pas content : On essaye de créer une deuxième variable nommée **a**.



❖ Noms de variables

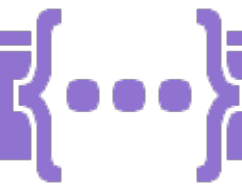
- ◆ Chaque variable possède un nom unique qui permet de la distinguer
 - **Attention !** Ceci fonctionne. Lorsqu'on met l'instruction `a = 3`, on ne crée pas une nouvelle variable. (Car on n'utilise pas « `let` ») On remplace seulement la valeur de `a`.

```
>> let a = 4;
```

```
← undefined
```

```
>> a = 3;
```

```
← 3
```



❖ Utiliser la **valeur** contenue dans une **variable**

◆ Une fois qu'une **variable** a été **déclarée** et **affectée**, on peut « l'appeler 📞 » pour utiliser la **valeur** qu'elle contient.

- Exemple, on déclare **a** et **b**. On essaye de les **additionner** 🤖

```
>> let a = 3;
```

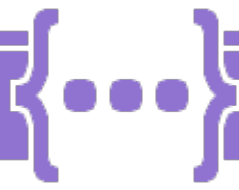
```
let b = 2;
```

```
← undefined
```

```
>> a + b
```

```
← 5
```

- Attention ! Les variables **a** et **b** n'ont pas été modifiées. Elles contiennent encore **3** et **2**. On a simplement demandé au programme quelle valeur est obtenue si on **additionne** les deux variables. 🧠😊



❖ Utiliser la **valeur** contenue dans une **variable**

◆ Voici d'autres exemples :

```
>> let a = 3;
```

```
← undefined
```

```
>> a * 4
```

```
← 12
```

On peut faire un calcul avec une **variable** et une valeur quelconque.

```
>> let a = 1;  
    let b = 2;  
    let c = 10;
```

```
← undefined
```

```
>> c - a - b;
```

```
← 7
```

On peut utiliser autant de **variables** que l'on souhaite dans un calcul.

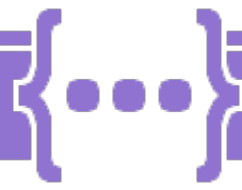
```
>> let a = 5.5;  
    let b = -3;  
    let c = 100 * 4;  
    let d = 10;
```

```
← undefined
```

```
>> a + b * c / d;
```

```
← -114.5
```





❖ Opérateurs d'affectation

- ◆ Il existe plusieurs « opérateurs » qui permettent de **modifier la valeur** affectée à une variable.

◆ Opérateur =

- Affectation simple (Nous connaissons déjà cet opérateur. C'est le plus simple.)
- Si la variable contenait déjà une valeur, on l'écrase.

```
>> let a = 3 * 4;  
← undefined  
  
>> a  
← 12  
  
>> a = 5;  
← 5
```

On écrase **12**. Maintenant **a** vaut **5**.



❖ Opérateurs d'affectation

◆ Opérateur ++

- Incrémentation : On ajoute 1 à la valeur actuelle.

```
>> let a = 2;
```

```
← undefined
```

```
>> ++a;
```

```
← 3
```

← On augmente la valeur de 1.

◆ Opérateur --

- Décrémentation : On enlève 1 à la valeur actuelle.

```
>> let a = 5;
```

```
← undefined
```

```
>> --a;
```

```
← 4
```

← On réduit la valeur de 1.

```
>> let a = 10;
```

```
++a;
```

```
--a;
```

```
++a;
```

```
++a;
```

```
--a;
```

```
--a;
```

Combient vaut a ? 🤔



❖ Opérateurs d'affectation

◆ Opérateur **+=**

- Affecte la valeur actuelle, **plus** une autre valeur

```
>> let a = 10;
```

```
← undefined
```

```
>> a += 5;
```

```
← 15
```

← On augmente la valeur de 5.

◆ Opérateur **-=**

- Affecte la valeur actuelle, **moins** une autre valeur

```
>> let a = 13;
```

```
← undefined
```

```
>> a -= 4;
```

```
← 9
```

← On réduit la valeur de 4.



❖ Priorité des opérateurs

◆ Ordre de priorité (Du premier au dernier)

1. Parenthèses ()
2. Multiplication et division * /
3. Addition et soustraction + -
4. Affectation =

◆ Les **parenthèses** permettent de modifier la **priorité des opérations**

```
>> let a = 4 + 6 / 2 + 3;
```

```
>> let a = (4 + 6) / 2 + 3;
```