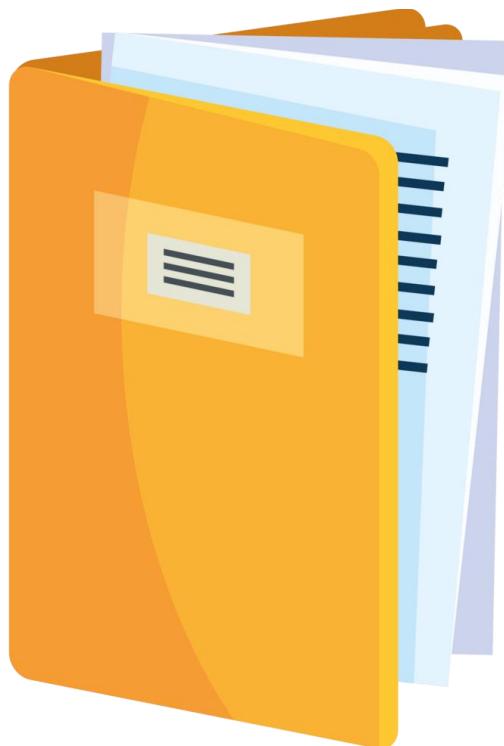


😊 EMOJI CODE 😊

DOCUMENTATION



Create by Yanis Llorens

Table des matières



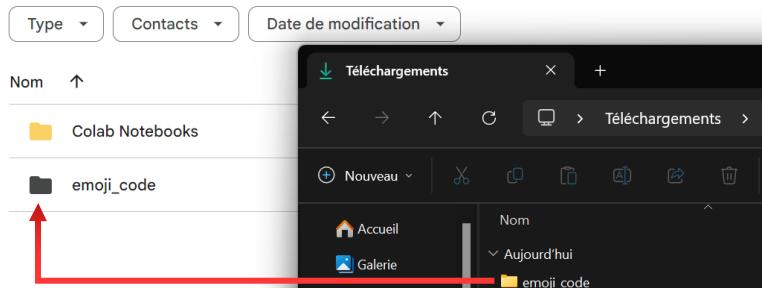
<u>SETUP</u>	3
<u>TABLEAU DES REFERENCES</u>	4
<u>VARIABLES</u>	5
<u>RANDOM / ALEATOIRE</u>	6
<u>MESSAGE CONSOLE</u>	7
<u>INPUT CONSOLE</u>	8
<u>BRACKETS</u>	9
<u>SIGNE ET COMPARAISON</u>	10
<u>IF / ELSE</u>	11
<u>FOR LOOP</u>	12
<u>WHILE LOOP</u>	13

Setup

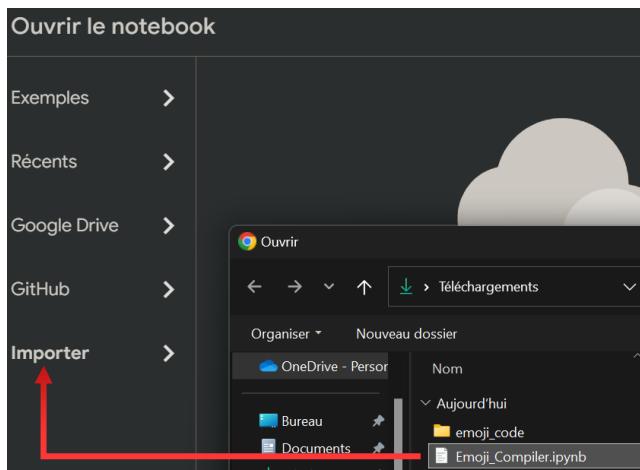
Après avoir download emoji_compiler.zip, extraire les fichiers puis :

- ① Accéder à **Google Drive** : <https://drive.google.com/drive/u/1/my-drive> puis **glisser** le fichier emoji_code dans le drive

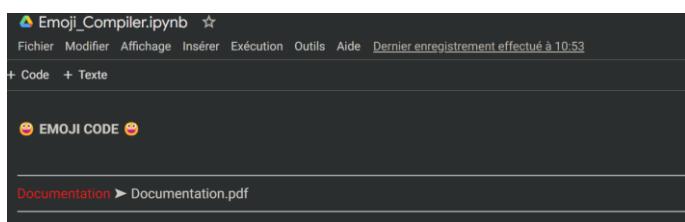
Mon Drive ▾



- ② Se rendre sur **Google Colab** : <https://colab.research.google.com/> et **importer** et ouvrir la feuille



- ③ Suivre les instructions



Après avoir installé le projet sur votre drive, il est important de suivre les instructions pour autoriser l'accès à Google Drive.

① Executer (Installe les packages nécessaires)

```
[1] !apt-get update  
!apt-get install -y flex bison
```

② Executer (Permet d'accéder au drive)

Autoriser ce notebook à accéder à vos fichiers Google Drive ?

Ce notebook demande l'accès à vos fichiers Google Drive. L'accès à Google Drive autorisera le code exécuté dans ce notebook à modifier les fichiers contenus dans votre Google Drive. Assurez-vous d'examiner le code du notebook avant de lui accorder l'accès.

Non, merci Se connecter à Google Drive

Tableau des références

Emoji	Signification
⌚	while
⌚⌚	for
👀	if
👀	else
✉️	print
⬇️🔢	input float
⬇️🔤	input string
🎲	random
🔒	bracket enter
🔓	bracket exit
➕	= variable+1
➖	= variable-1
EQUALS	==
👉	>
👉👉	<
🔗	link

Variables

Une « **variable** » peut représenter différents types de données, comme des nombres décimaux, des entiers ou des chaînes de caractères. En programmation, les « **variables** » servent de conteneurs pour stocker ces valeurs.

```
nom_variable = entier  
nom_variable = decimal  
nom_variable = chaîne
```

Voilà un exemple de comment créer une variable dans le code.

⚠️ Attention les variables ne prendront pas de caractère spécial or « _ » et les noms suivants sont interdit : « float », « int », « bool », « string », « double », ...

```
var_entier = 5  
var_decimal = 5.55  
var_chaine = "chaine"
```

Random / Aléatoire

Le principe de « **random** » consiste à générer des valeurs imprévisibles à chaque exécution du programme, simulant ainsi des événements aléatoires.

```
nom_variable = aléatoire entre 1 et 10
```

Voilà un exemple de comment créer une variable aléatoire dans le code.

```
var_random = 🎲(1,10)
```

Message Console

Afficher un message dans la console.

```
Afficher("message")
```

Exemple d'utilisation de `=>` pour afficher un message.

```
=> : "message"  
var = 1  
=> : "affiche var : {var}"  
  
var_1 = 0  
var_2 = 1  
=> : "affiche var 1 : {var_1} et var 2 : {var_2}"
```

Input Console

Demander la valeur d'une variable à l'utilisateur.

```
nom_variable = Demander nombre à l'utilisateur  
nom_variable = Demander texte à l'utilisateur
```

Exemple d'utilisation de  et  pour demander la valeur de la variable à l'utilisateur. Le  sert à demander un nombre et le  sert à demander un texte

```
number =  ("Enter number : ")  
    : "return {number}"  
  
text =  ("Enter text : ")  
    : "return {text}"
```

Brackets

Les « brackets » sont utilisées pour délimiter des blocs de code, comme les corps des fonctions, des boucles et des structures de contrôle.

```
if/for/while/...
↳ Enter bloc
↳ faire code
↲ Sortie bloc
```

Exemple d'utilisation de  et  pour signaler l'entrée et la sortie d'un bloc.

```
var = 5
⌚ (var == 1)
⌚ ↳ faire code
🔒
```

Signe et Comparaison

Différents signes et comparaisons vont permettre une meilleure compréhension du code. Définissons 5 signes : « == », « ++ », « -- », « > », « < ».

```
Calcul
var = var + 1 revient à var++
var = var - 1 revient à var-- 

Comparaison
var = 1
var vaut 1 ? revient à var == 1 (oui)
var sup à 1 ? revient à var > 1 (non)
var inf à 1 ? revient à var < 1 (non)
var sup ou égal à 1 ? revient à var >= 1 (oui)
var inf ou égal à 1 ? revient à var <= 1 (oui)
```

Exemple d'utilisation de  ,  ,  ,  ,  afin de créer des comparaisons avec :

- « == » signifie 
- « ++ » signifie 
- « -- » signifie 
- « > » signifie 
- « < » signifie 

```
var = 1
var+
var-
(var == 1)

↳ faire code

```

```
(var > 1)

↳ faire code

```

```
(var < 1)

↳ faire code

```

If / Else

En C, « **if** » est utilisé pour exécuter un bloc de code si une condition est vraie, tandis que « **else** » exécute un autre bloc si la condition est fausse.

```
si (condition)
↳ faire code
```

```
si (condition)
↳ faire code
sinon
↳ faire code
```

Exemple d'utilisation de 🧐 pour une condition if et 🧐 pour une condition else.

```
var = 5
ধারণা (var > 1)
↳ faire code
ধারণা
↳ faire code
ধারণা
```

For Loop

La boucle « **for** » permet d'exécuter une série d'instructions un nombre déterminé de fois en initialisant, testant et incrémentant une variable de contrôle.

```
tant que (variable ; condition ; incrément)
↳ faire code
```

```
tant que (var = 0 ; var > 1 ; var = var + 1)
↳ faire code
```

Exemple d'utilisation de  et  pour créer une boucle for.

 Attention la condition ne peut pas être « == » :  , mettre cette condition revient à ne rien faire.

```
loop (var = 0 ⚡ var 🔜 5 ⚡ var + )
↳ faire code
end
```

While Loop

Une boucle « **while** » permet de répéter une série d'instructions tant qu'une condition est vraie.

⚠️ Attention la condition d'arrêt d'une boucle « **while** » doit être mise dans la boucle (voir exemple).

```
tant que (condition)
↳ faire code
```

```
var = 5
tant que (var > 1)
↳ faire code
↳ var = var - 1
```

Exemple d'utilisation de  pour créer une boucle while.

```
var = 5
 (var > 1)

↳ faire code
↳ var = var - 1

```