

# Formation Python

École Secondaire Pierre-Dupuy

Olivier Chabot, 17 Novembre 2021

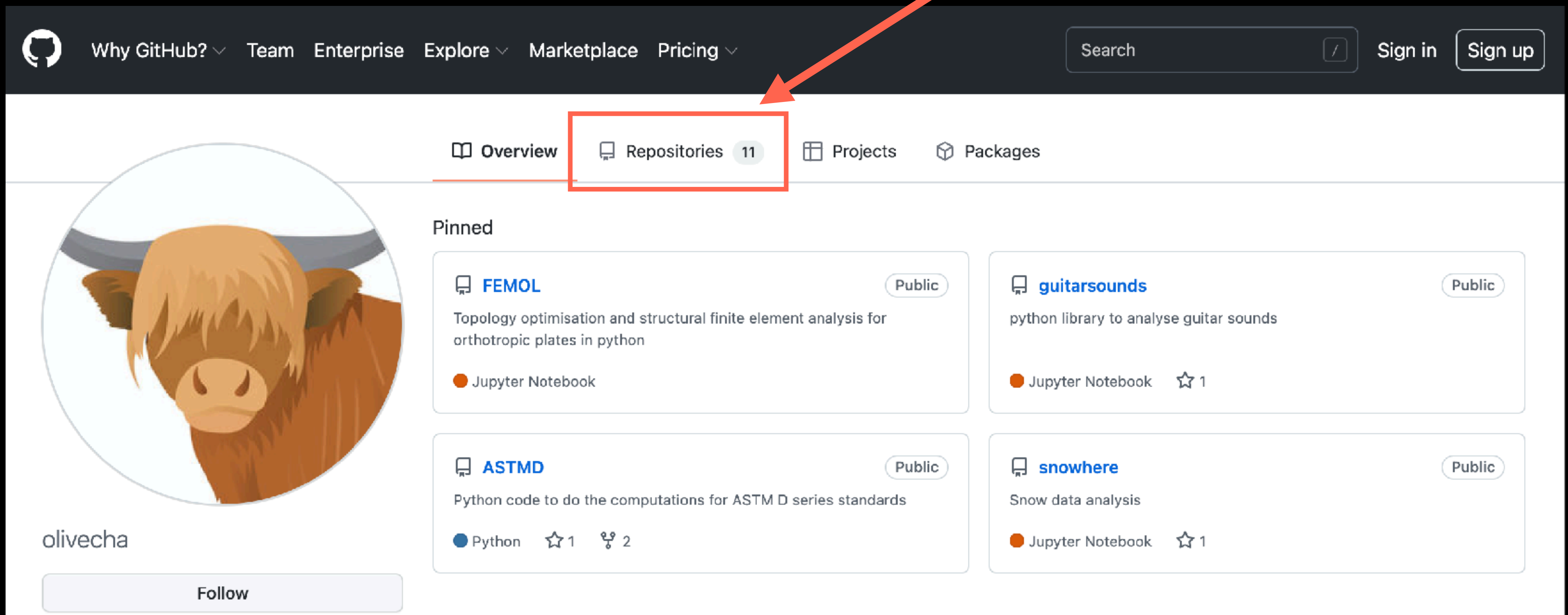
# Plan de la formation

- Aller chercher le contenu du cours d'aujourd'hui
- Récapitulatif séance 3
- Comprendre le jeu devine le nombre
- Les fonctions
- Comment faire votre propre jeu + exemple




# Télécharger des fichiers de **github**

- Se rendre sur <https://github.com/olivecha>





Why GitHub? ▾ Team Enterprise Explore ▾ Marketplace Pricing ▾ Search / Sign in Sign up


Overview Repositories 11 Projects Packages


  
olivecha  
Follow

Pinned

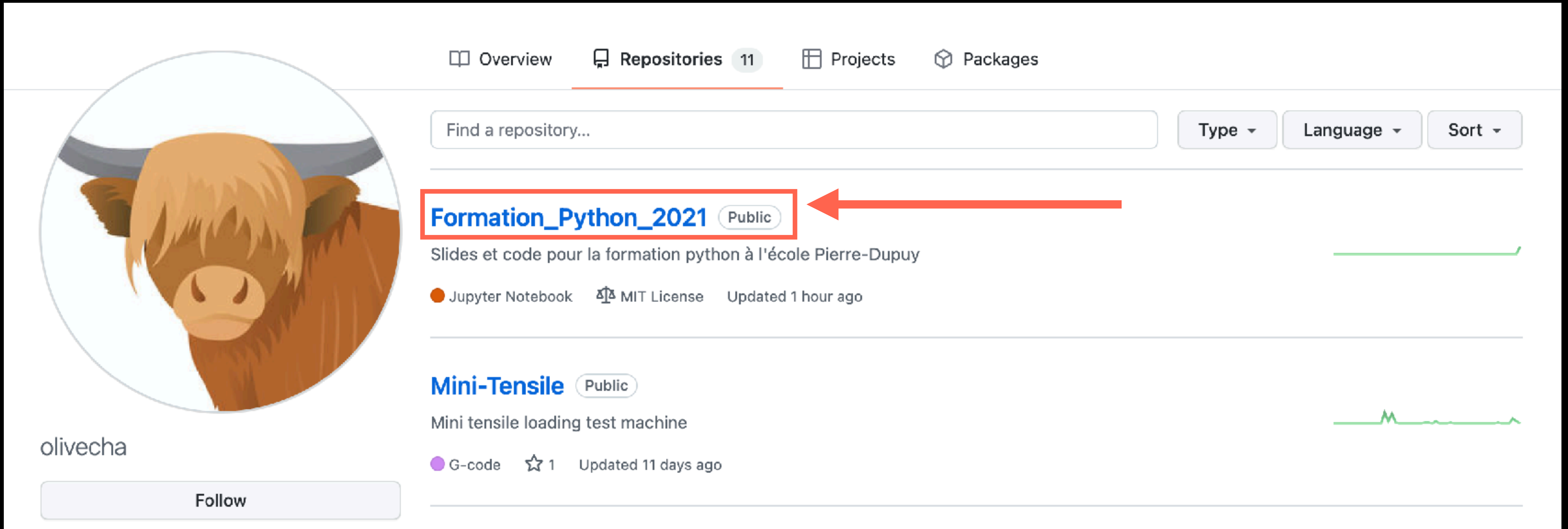
 **FEMOL** Public  
Topology optimisation and structural finite element analysis for orthotropic plates in python  
Jupyter Notebook

 **guitarsounds** Public  
python library to analyse guitar sounds  
Jupyter Notebook ☆ 1

 **ASTMD** Public  
Python code to do the computations for ASTM D series standards  
Python ☆ 1 🍴 2

 **snowhere** Public  
Snow data analysis  
Jupyter Notebook ☆ 1

# Répertoire **github** de la formation



The screenshot shows the GitHub profile of a user named 'olivecha'. The profile picture is a cartoon illustration of a Highland cow. The user's name 'olivecha' is displayed below the picture, along with a 'Follow' button. The 'Repositories' tab is selected, showing a list of repositories. The first repository, 'Formation\_Python\_2021', is highlighted with a red box and a red arrow pointing to it. This repository is public, contains Jupyter Notebooks, is licensed under MIT, and was updated 1 hour ago. The description for this repository is 'Slides et code pour la formation python à l'école Pierre-Dupuy'. The second repository, 'Mini-Tensile', is also public, contains G-code, has 1 star, and was updated 11 days ago. The description for this repository is 'Mini tensile loading test machine'.

Overview Repositories 11 Projects Packages

Find a repository... Type Language Sort

**Formation\_Python\_2021** Public

Slides et code pour la formation python à l'école Pierre-Dupuy

Jupyter Notebook MIT License Updated 1 hour ago

**Mini-Tensile** Public

Mini tensile loading test machine

G-code ☆ 1 Updated 11 days ago

olivecha

Follow

# Téléchargement du dossier .zip

olivecha / Formation\_Python\_2021 Public

Notifications

Star 0

Fork 0

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

main

1 branch

0 tags

Go to file

Code

About

Slides et code pour la formation python à l'école Pierre-Dupuy

Readme

MIT License

Releases

No releases published

Packages

No packages published

Clone

HTTPS GitHub CLI

https://github.com/olivecha/Formation\_

Use Git or checkout with SVN using the web URL.

Open with GitHub Desktop

Download ZIP

# Rappel sur les différents types

Les types sont la façon qu'utilise l'ordinateur pour représenter des variables

Nom	Signature	Exemples
Nombre entier	int	type(1) = int, liste[a], type(a) = int
Nombre à virgule	float	type(4/2) = float, type(3.1415) = float
Booléen	bool	type(1 < 2) = bool, if a: type(a) = bool
String	str	type('abc') = str, my_string='abc', my_string[0] = 'a'
Liste	list	type([1, 2, 3]) = list, a = [1, 2, 3], a[0] = 1



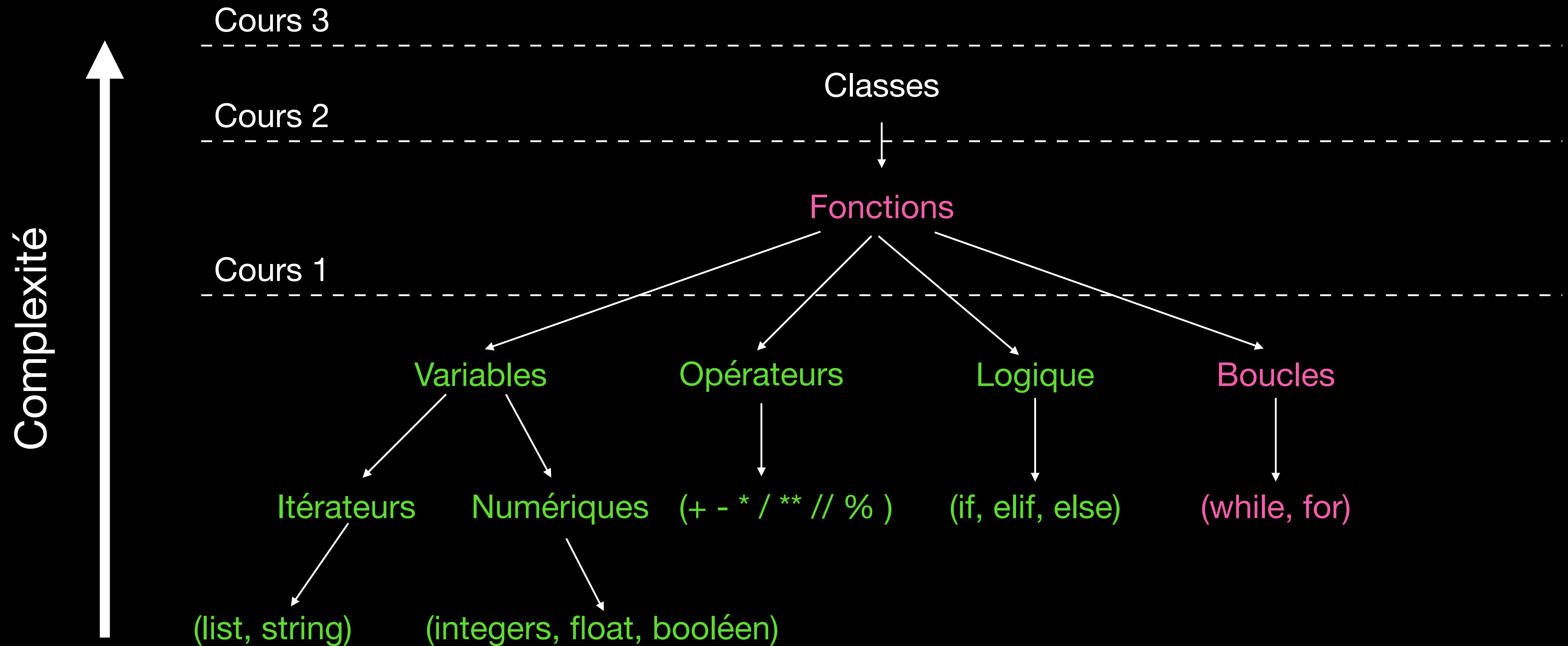
# La structure (if, elif, else)

```
if condition1:
    'Exécute si la condition 1 est vraie'

elif condition2:
    """
    Exécute si la condition 1 est fausse
    et la condition 2 est vraie
    """

else:
    """
    Exécute si toutes les conditions
    sont fausses
    """
```

# Récapitulatif de la séance 2





# La *méthode* `.append()` des listes

Interlude pour voir un essentiel de la boucle **for**

- Méthode :  
`liste.append(variable)`
- Ajoute une variable à la **fin** de la `liste`
- Très utile dans les boucles **for**

```
ma_liste = [1, 2, 3]
ma_liste.append(4)
print(ma_liste)
#>>> [1, 2, 3, 4]
```

**Exemple 2.9 :** La méthode `.append()`

# La boucle 'for' et la fonction .append()

```
# Exemple de boucle for avec .append()
mes_eleves = ['Max', 'Clo', 'Sam', 'Flo']
mes_eleves_avec_a = [] # liste vide

for chaque_eleve in mes_eleves:
    if 'a' in chaque_eleve:
        mes_eleves_avec_a.append(chaque_eleve)
```

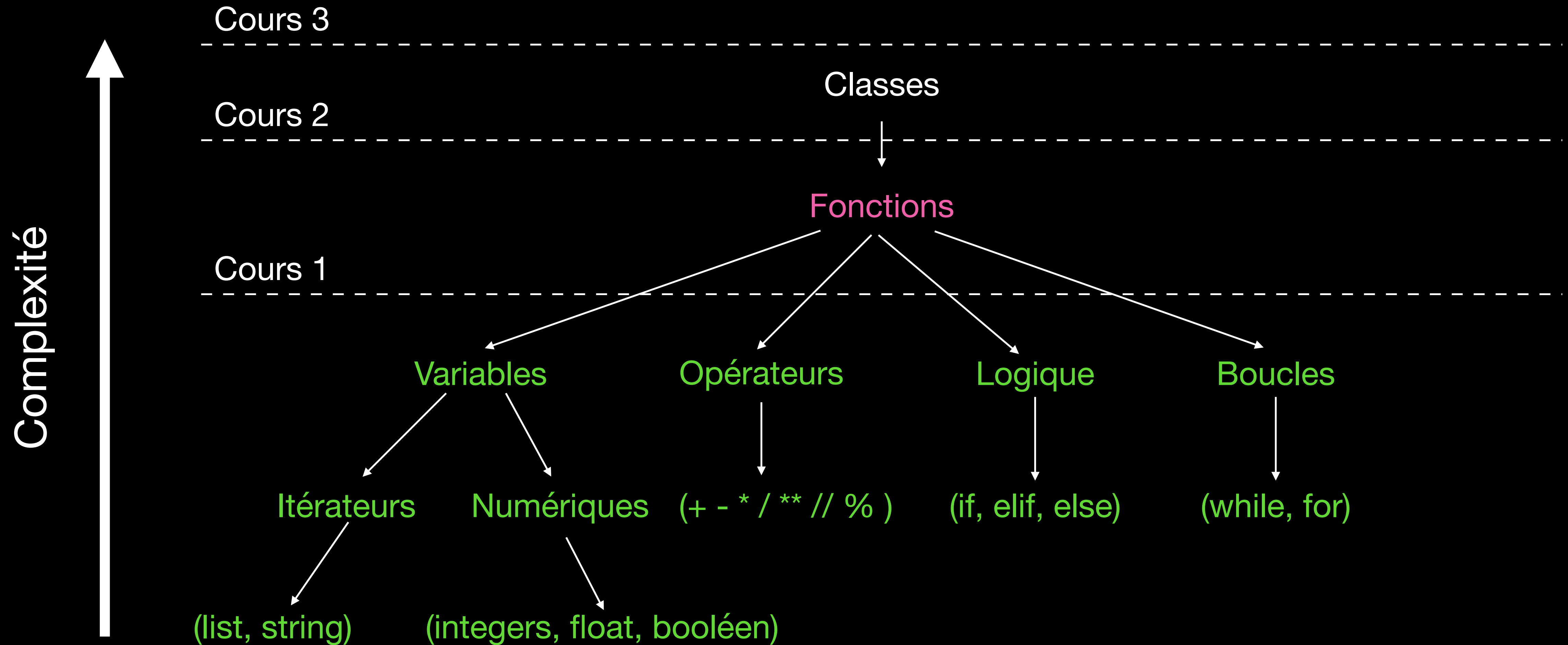
- La boucle **for** : faire quelque chose pour chaque chose dans un intégrateur
- La fonction **.append()** ajoute un item à la fin d'une liste

# Exemple de boucle 'while' avec la fonction randint du module random

```
# Exemple de boucle while :  
from random import randint  
nombre_dessais = 0  
nombre = randint(0, 100)  
  
while nombre != 100:  
    nombre = randint(0, 100)  
    nombre_dessais = nombre_dessais + 1  
  
print('L'ordinateur a fait', nombre_dessais, ' essais avant de tomber sur 100')
```

- La boucle 'while' permet de faire quelque chose pendant qu'une condition n'est pas vraie
- Elle peut s'exécuter un nombre arbitraire de fois

# Les fonctions : compartimenter le code



# Les fonctions existantes

Description	Exemple
Génère un nombre entier aléatoire dans un certain interval	<pre>print(random.randint(10, 20)) # &gt;&gt; 13</pre>
Affiche un message à l'utilisateur et enregistre l'input dans une variable	<pre>nom_utilisateur = input('Quel est votre nom ?') print('Votre nom est : ', nom_utilisateur)</pre> <p>Quel est votre nom ? Olivier Votre nom est :  Olivier</p>
Transforme un autre type en int, si les types sont compatibles	<pre>type(int('1'))</pre> <p>int</p>
Affiche l'aide d'une fonction	<pre># La fonction help() peut aider si jamais on oublie help(random.randint)</pre> <p>Help on method randint in module random:</p> <p>randint(a, b) method of random.Random instance Return random integer in range [a, b], including both end points.</p>

# Votre premier Jeu : devine mon nombre

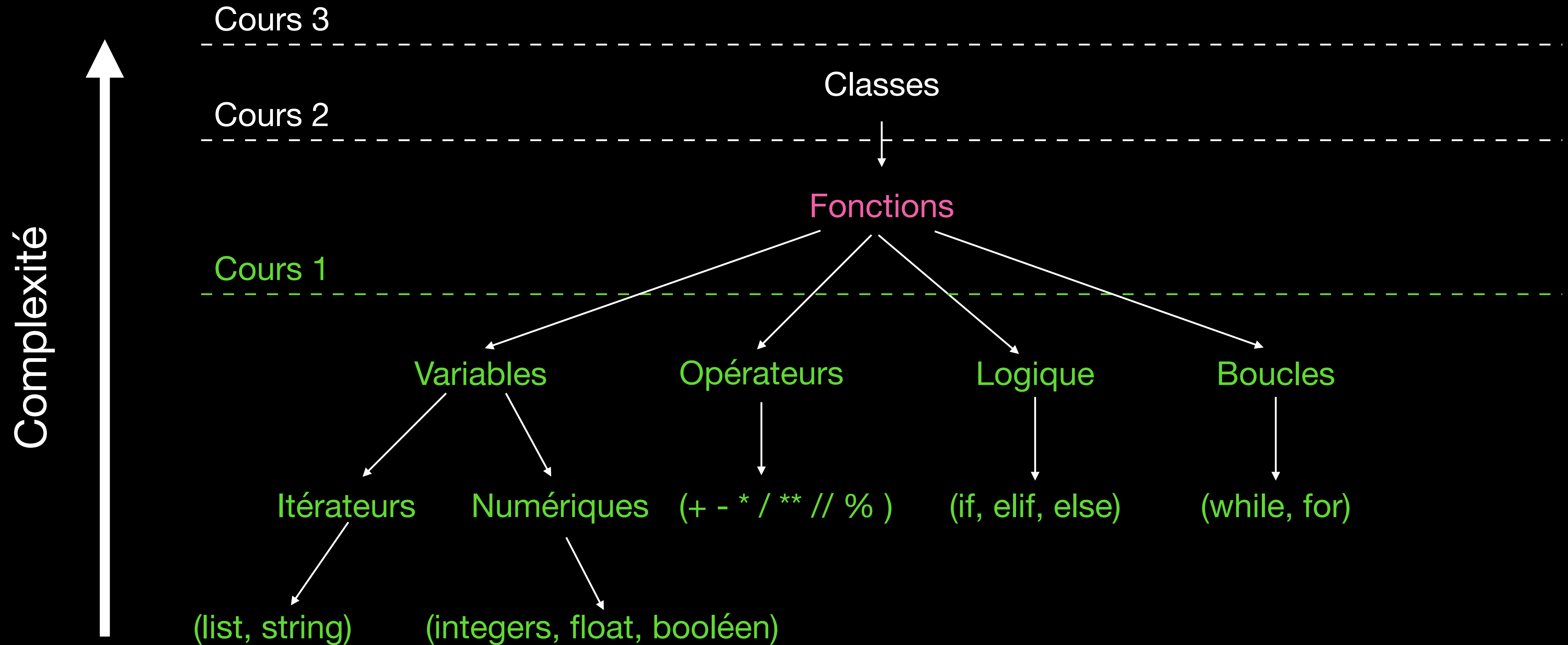
- L'ordinateur choisi un nombre entre 0 et 10
- L'utilisateur essaie des chiffres et l'ordinateur lui donne des indices tant qu'il ne l'a pas

## 4 Outils nécessaires pour construire le jeu :

- `string = input('input string')` Demande un input à l'utilisateur en imprimant l'input string
- `integer = int(variable)` Transforme un variable d'un certain type en integer (si possible)
- `random` Package permettant de générer des chiffres aléatoires
- `random int = random.randint(a, b)` Fonction du package random qui permet de générer un chiffre aléatoire entre a et b (inclus)



# Les blocs de construction d'un programme





# Syntaxe d'une fonction

je **DÉFINIS** une **FONCTION** prenant des **ARGUMENTS** :

```
def fonction(arguments):  
    """  
    Fonction qui change pas arguments  
    """  
    sortie = arguments  
    return sortie
```

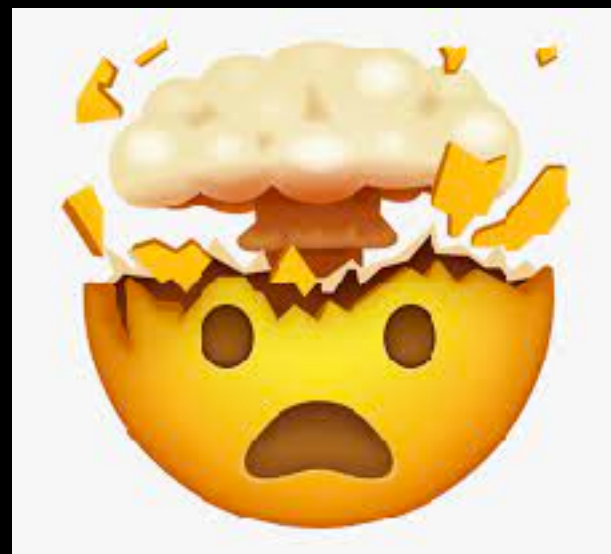
la **FONCTION** agit sur les **ARGUMENTS** et **RETOURNE** quelque chose

# Utilisation d'une fonction

j' **APPELLE** ma **FONCTION** avec **()**

```
fonction('a')  
>> 'a'
```

Les fonctions sont aussi des variables :  
(tout est une variable dans python)



```
f = fonction  
f('a')  
>> 'a'
```

# La portée d'une fonction (scope)

Les fonctions c'est comme Vegas :

**Ce qui se passe dans la fonction reste dans la fonction**

```
def fonction_1(a, b)
    # La variable 'c' est dans
    # le scope global
    return a + b + c
```

```
def fonction_2(a, b)
    # La variable 'c' appartient seulement
    # à la fonction
    c = 3
    return a + b + c
```

```
c = 4
ma_fonction_1(1, 1)
>> 6
ma_fonction_2(1, 1)
>> 5
print(c)
>> 4
```

- **Exemple 5 : Les fonctions**

# Les arguments mots clés

- Les arguments mot clefs permettent de d'avoir une valeur par défaut et d'éviter devoir se rappeler de l'ordre des arguments dans la fonction

```
def salutation(phrase = 'Bonjour', nom=''):
    """
    Fonction qui salue quelqu'un
    """
    print(phrase + ' ' + nom + ' !')
```

```
salutation()
>> 'Bonjour !'
salutation(nom='Maxime')
>> 'Bonjour Maxime !'
salutation(nom='Karine', phrase='Salut')
>> 'Salut Karine !'
```

# Les modules

## Une façon de garder l'ordre dans les fonctions

- Un module contient des fonctions auxquelles on peut accéder en *l'important*

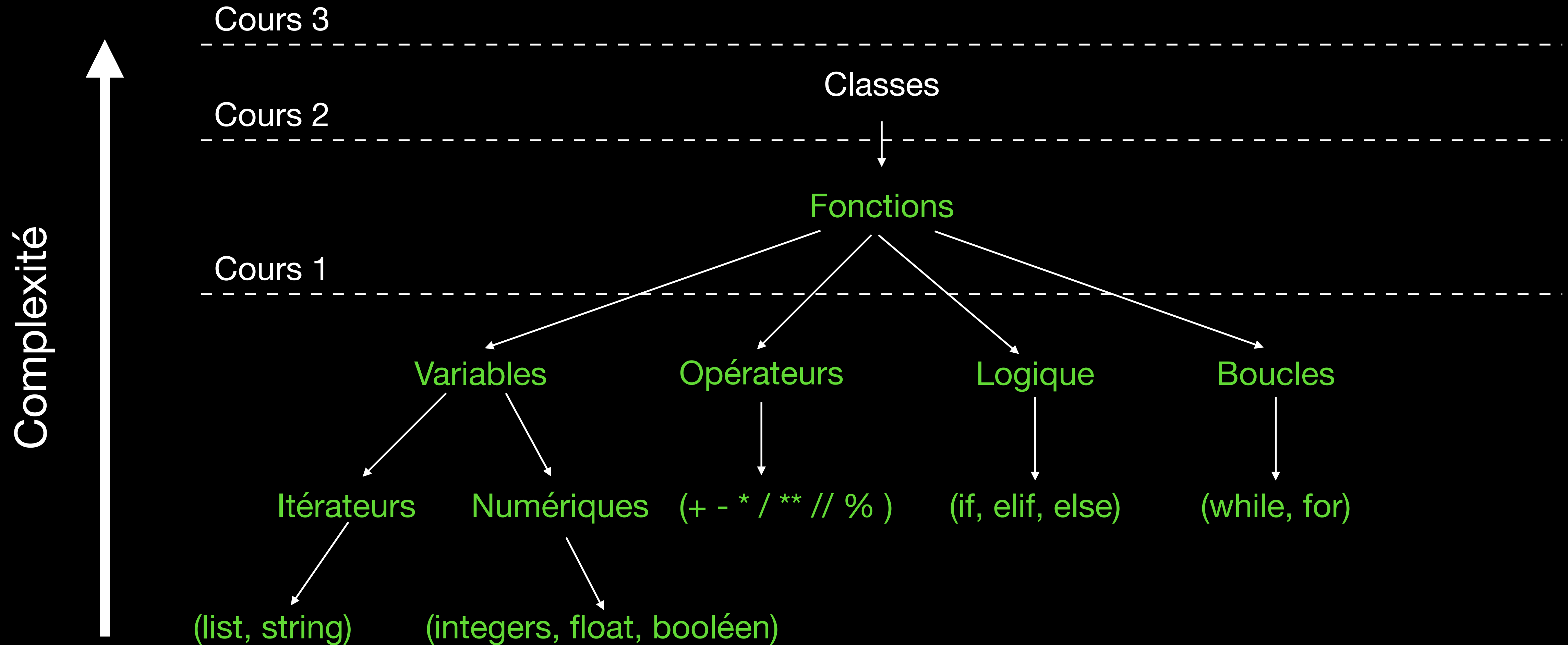
```
import random  
type(random)  
>> module
```

```
import random  
type(random.randint)  
>> method
```

```
from random import randint  
type(randint)  
>> method
```

Ici `method` est une fonction

# Les blocs de construction d'un programme



# **Jeu 2 : Le bonhomme pendu**

**Utiliser des fonctions pour mieux concevoir le jeu**