

# Initiation à la programmation avec Python

—

Par Benjamin Tardy

# Tour de table & attentes

# Présentation de la formation



# Public concerné et prérequis

Toute personne devant apprendre à programme

Pas de prérequis / aucune connaissance particulière

# Objectifs de la formation

Structure un programme

Maîtriser les éléments de lexique et de syntaxe pour écrire un programme

Exécuter un programme

Savoir le tester et le debugger

# Pédagogie et attentes

L'apprentissage par la pratique est le plus efficace :

- Live coding
- Exercices courts et TP long

Un programme

# Qu'est-ce qu'un programme

- Un ensemble d'opérations (instructions) destinées à être exécutées sur un ordinateur
- Un programme est conçu pour accomplir une tâche spécifique, telle que trier une liste de nombres, calculer la moyenne d'une série de nombres ou afficher un message à l'utilisateur
- Un ordinateur sans programme ne fait strictement rien (différence entre un calculateur et un ordinateur)
- Le programme est écrit par un programmeur dans un langage de programmation compréhensible par la machine
- Le programme est exécuté par le processeur de la machine



# Les langages de programmations

Une machine ne peut comprendre qu'une suite d'instructions binaire du type : 0001100010001111010101.

Cette façon d'écrire n'est pas du tout naturelle pour un humain. Le langage de programmation nous permet de pouvoir communiquer avec la machine

Il existe de nombreux langages de programmation différents, chacun avec ses propres avantages et inconvénients. Voici une liste de certains des langages de programmation les plus connus et les plus utilisés :

- **Java** : Java est un langage de programmation populaire, utilisé pour développer des applications pour des ordinateurs, des serveurs, des smartphones et des appareils embarqués. Il est souvent utilisé pour développer des applications Web et mobiles.
- **Python** : Python est un langage de programmation populaire pour l'apprentissage automatique, la science des données et l'analyse de données. Il est également utilisé pour le développement Web, la création d'interfaces graphiques utilisateur, et pour la création de scripts et d'outils de système.
- **JavaScript** : JavaScript est un langage de programmation utilisé pour le développement Web, la création d'interfaces utilisateur interactives, et pour la création de jeux et d'applications mobiles. Il est souvent utilisé en conjonction avec HTML et CSS pour créer des sites Web dynamiques.
- **C++** : C++ est un langage de programmation utilisé pour développer des applications système, des jeux, des logiciels de bureau, des logiciels de gestion de réseau et d'autres applications nécessitant des performances élevées.

# Les langages de programmations

- **C#** : C# est un langage de programmation orienté objet utilisé pour développer des applications Windows et des jeux, ainsi que pour le développement Web et la création d'applications mobiles.
- **PHP** : PHP est un langage de programmation utilisé pour développer des sites Web dynamiques et des applications Web.
- **Ruby** : Ruby est un langage de programmation utilisé pour le développement Web et la création d'applications mobiles.
- **Swift** : Swift est un langage de programmation utilisé pour développer des applications iOS et macOS.
- **Objective-C** : Objective-C est un langage de programmation utilisé pour développer des applications iOS et macOS, ainsi que pour le développement de logiciels système.
- **SQL** : SQL est un langage de programmation utilisé pour gérer les bases de données et les données.

Il existe de nombreux autres langages de programmation utilisés pour une variété de tâches, tels que Perl, Go, Kotlin, Rust, etc. Le choix du langage de programmation dépendra de la tâche à accomplir, des préférences personnelles et de la communauté de développeurs.

-

# Les différents paradigmes

Un paradigme est un modèle de programmation, une approche pour résoudre des problèmes de programmation. Chaque paradigme a sa propre manière de décrire le processus de résolution de problèmes. Il existe plusieurs paradigmes de programmation, les plus courants sont :

- la programmation *impérative* : ce paradigme décrit comment résoudre un problème en donnant des **instructions précises** à l'ordinateur pour exécuter. Le code écrit dans ce paradigme est basé sur des **instructions séquentielles** et des structures de contrôle de flux. L'exemple le plus courant de ce paradigme est le langage C.

Exemple pour rejoindre un point B en partant d'un point A :

- Prenez la ligne D du RER direction Paris jusqu'à Gare de Lyon
- Traversez la Seine à pied par le pont Charles de Gaulle jusqu'à la gare d'Austerlitz
- Prenez la ligne C du RER direction Versailles jusqu'à Champs de Mars / Tour Eiffel

# Les différents paradigmes

Un paradigme est un modèle de programmation, une approche pour résoudre des problèmes de programmation. Chaque paradigme a sa propre manière de décrire le processus de résolution de problèmes. Il existe plusieurs paradigmes de programmation, les plus courants sont :

- la programmation *déclarative* : ce paradigme décrit comment résoudre un problème en déclarant les contraintes et les règles que doit suivre la solution. L'ordinateur utilise ensuite ces déclarations pour déterminer la solution. L'exemple le plus courant de ce paradigme est le langage SQL pour la gestion de bases de données.

Exemple pour rejoindre un point B en partant d'un point A :

- Le rendez-vous est sous la tour eiffel

**SQL** : `SELECT * from nomdelatable`

# Les différents paradigmes

Un paradigme est un modèle de programmation, une approche pour résoudre des problèmes de programmation. Chaque paradigme a sa propre manière de décrire le processus de résolution de problèmes. Il existe plusieurs paradigmes de programmation, les plus courants sont :

- la programmation *orientée objet* : ce paradigme décrit comment résoudre un problème en créant des objets qui interagissent les uns avec les autres pour accomplir une tâche. Chaque objet possède des propriétés et des méthodes qui lui permettent d'interagir avec les autres objets. Les exemples les plus courants de ce paradigme sont Java et Python.

Chaque paradigme a ses propres avantages et inconvénients. Le choix du paradigme dépendra du problème à résoudre et des préférences personnelles du programmeur.

Les langages de programmation les plus récents sont **multi-paradigmes** par exemple Python prend en charge les paradigmes impératif, orienté objet, fonctionnel et déclaratif.

# Le moteur du programme : l'algorithme

L'algorithme est une étape importante dans la conception et la programmation de tout programme informatique. Il sert de guide pour écrire les instructions spécifiques du programme et garantit que le programme fonctionne de manière fiable et efficace.

Il s'agit d'une séquence d'instructions clairement définies et non ambiguës qui permettent de résoudre un problème ou d'accomplir une tâche spécifique. En d'autres termes, c'est un ensemble d'étapes logiques et ordonnées qui permettent de résoudre un problème ou d'atteindre un objectif.

Un exemple simple d'algorithme pourrait être une recette de cuisine. La recette fournit une série d'étapes précises qui doivent être suivies dans un ordre spécifique pour produire un plat particulier. Si les étapes sont suivies correctement, le plat sera réussi. De même, dans la programmation informatique, un algorithme peut être utilisé pour trier une liste d'éléments ou pour trouver le chemin le plus court entre deux points.

# Le pseudo langage

Le pseudo-langage est une façon d'écrire un algorithme ou un programme en utilisant une syntaxe qui ressemble à celle d'un langage de programmation, mais qui est plus facile à comprendre pour les humains.

En général, un pseudo-code utilise des instructions simples en langage naturel, plutôt que des instructions de programmation complexes et détaillées. Il permet de décrire de manière claire et concise les étapes nécessaires pour résoudre un problème ou accomplir une tâche.

<https://fr.wikipedia.org/wiki/Pseudo-code>

```
Début
  Lire nombre1
  Lire nombre2
  somme = nombre1 + nombre2
  moyenne = somme / 2
  Afficher moyenne
Fin
```

```
Début
  Lire une liste de nombres
  maximum = premier nombre de la
liste
  Pour chaque nombre dans la
liste
    Si nombre > maximum alors
      maximum = nombre
    Fin Si
  Fin Pour
  Afficher maximum
Fin
```

# Le pseudo langage

Exercice : Écrire un algorithme permettant de vérifier et d'afficher si une valeur est pair ou impair.





# Langage compilé vs interprété

- Certains langages sont **compilés** avant exécution : ils sont traduits avant d'être exécutés.

## Exemple

Le langage C est compilé en langage machine. Une fois compilé, le programme est directement compris par la machine et est donc plus rapide.

- D'autres langages sont dits **interprétés** : ils sont lus et convertis au fur et à mesure de l'exécution, il n'y a donc pas à passer par un compilateur.

## Exemple

Python est un langage interprété ligne par ligne, il est donc plus lent que le langage C.

# Python

---

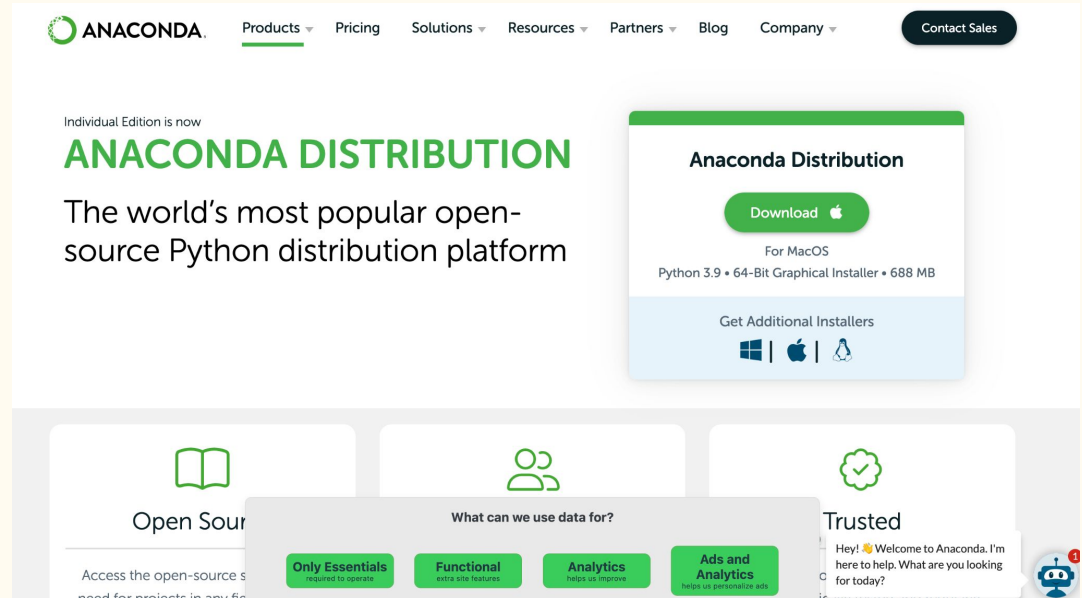
# Python

Python est un langage de programmation interprété et de haut niveau, conçu pour être facile à lire et à écrire. Il a été créé par Guido van Rossum et publié pour la première fois en 1991.

- **Lisibilité:** Python utilise une syntaxe claire et simple qui le rend facile à lire et à comprendre, même pour les débutants.
- **Portabilité :** Vous pouvez exécuter le code sur plusieurs plateformes, comme Windows, Mac et Linux.
- **Grande communauté :** Python a une grande communauté de développeurs qui contribuent régulièrement à l'amélioration du langage, ainsi qu'à la création de bibliothèques et de frameworks. Vous pouvez facilement trouver des réponses à vos questions et des ressources pour vous aider.
- **Bibliothèques et Frameworks :** Il existe de nombreuses bibliothèques et frameworks, qui peuvent vous aider à accomplir rapidement des tâches complexes.
- **Versatile :** Python est utilisé dans une grande variété de domaines, comme l'analyse de données, l'apprentissage automatique, la robotique, le développement web, la science des données et bien d'autres.
- **Gratuit**

# Python : installation

- Vous pouvez installer python 3.X en téléchargeant une version récente ici : <https://www.python.org/>
- MAIS il est souvent plus simple d'installer la distribution Anaconda : <https://www.anaconda.com/products/distribution>
  - **Installation facilitée**
  - **Bibliothèques pré-installées**
  - **Gestion d'environnements**
  - **Facilité de mise à jour**
  - **Support multiplateforme**



# Qu'est-ce qu'une librairie ?

Une bibliothèque Python (ou librairie) est un ensemble de modules, de fonctions et de classes prédéfinies, conçues pour résoudre des problèmes spécifiques. Voici quelques points clés à retenir sur les bibliothèques Python :

- **Réutilisabilité** : permettent aux développeurs de réutiliser du code existant pour résoudre des problèmes courants, plutôt que de devoir écrire tout le code à partir de zéro.
- **Gain de temps** : elles peuvent aider les développeurs à gagner du temps en offrant des solutions prédéfinies pour des tâches courantes.
- **Documentation** : elles sont accompagnées d'une documentation détaillée, qui permet aux développeurs de comprendre comment utiliser les fonctions et les classes de la bibliothèque.
- **Support communautaire** : elles sont souvent créées et maintenues par des communautés de développeurs, ce qui signifie que les développeurs peuvent accéder à des forums et à des canaux de communication pour obtenir de l'aide et des conseils sur l'utilisation de la bibliothèque.

En résumé, cela vous permet d'accéder à des fonctions spécifiques qui ne sont pas natives dans les fonctions python de base !

# Quelques exemples de librairies

Voici trois exemples de bibliothèques Python largement utilisées dans des domaines différents :

**Numpy** : largement utilisée pour la manipulation de tableaux multidimensionnels et les calculs scientifiques. Elle offre des fonctions pour le calcul mathématique de base, la transformation de Fourier, l'algèbre linéaire, la génération de nombres aléatoires et bien plus encore. Numpy est souvent utilisé dans les domaines scientifiques et d'analyse de données.

**Scikit-learn** : populaire pour l'apprentissage automatique. Elle offre des algorithmes pour la classification, la régression, le regroupement, la réduction de dimensionnalité et la préparation des données. Scikit-learn est souvent utilisé dans les domaines de l'analyse de données, de la science des données et de l'ingénierie des systèmes.

**Flask** : pour le développement d'applications web. Elle offre des fonctionnalités pour la gestion des routes, la gestion des requêtes et des réponses, les sessions et les cookies. Flask est souvent utilisé dans les domaines du développement web et de la création d'API.

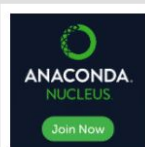
Il existe des centaines de librairies différentes ...

# Les librairies : qualité et pérennité

- Pour les choses les traitements les plus simples privilégier les librairies appartenant au standard python.
  - <https://docs.python.org/3/py-modindex.html>
- Utiliser une liste de librairies vérifiée / organisée / utilisée par des milliers d'utilisateurs
  - <https://awesome-python.com/>
  - <https://python.libhunt.com/>
- Google est votre ami, les forums de discussions aussi.
  - <https://stackoverflow.com/>
  - <https://medium.com/>
  - <https://towardsdatascience.com/>

# Notebook vs IDE





Discover premium data science content

Documentation

Anaconda Blog



Applications on base (root)

Channels

Refresh



CMD.exe Prompt

0.1.1

Run a cmd.exe terminal with your current environment from Navigator activated

Launch



Datalore

Online Data Analysis Tool with smart coding assistance by JetBrains. Edit and run your Python notebooks in the cloud and share them with your team.

Launch



IBM Watson Studio Cloud

IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.

Launch



JupyterLab

2.2.6

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Launch



Jupyter Notebook

6.1.4

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Launch



Powershell Prompt

0.0.1

Run a Powershell terminal with your current environment from Navigator activated

Launch



PyCharm Community

2020.3.3

An IDE by JetBrains for pure Python development. Supports code completion, listing, and debugging.

Launch



Qt Console

4.7.7

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Launch



Spyder

4.1.5

Scientific Python Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Launch



VS Code

1.55.1

Streamlined code editor with support for development operations like debugging, task running and version control.

Launch



Glueviz

1.0.0

Multidimensional data visualization across files. Explore relationships within and among related datasets.

Install



Orange 3

3.26.0

Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

Install



PyCharm Professional

A full-fledged IDE by JetBrains for both Scientific and Web Python development. Supports HTML, JS, and SQL.



RStudio

1.1.456

A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

# Jupyter notebook vs IDE

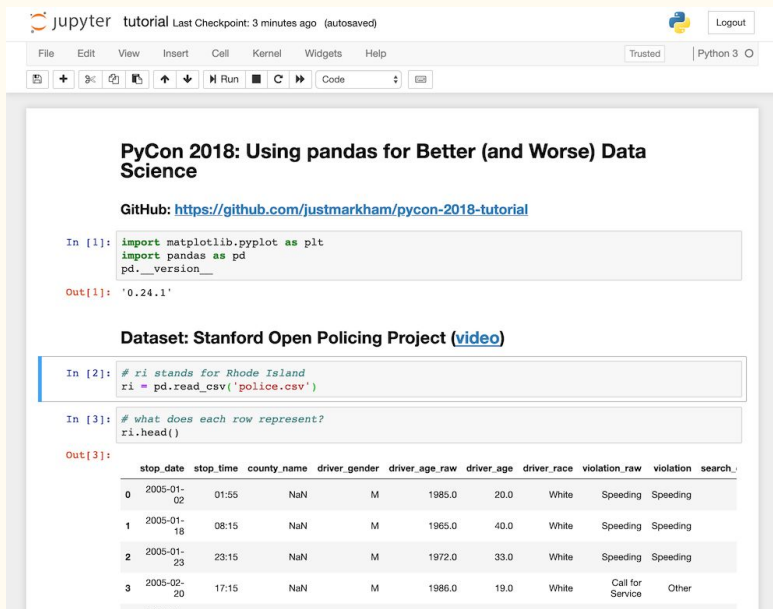
## **Avantages :**

- Une application web
- Insérer du markdown pour décrire des étapes ou commenter des graphiques
- Très utile pour expérimenter, tester et afficher
- Partager vos notebooks avec vos collègues

## **Inconvénients :**

- Ne s'industrialise pas
- Pas utilisable dans github
- Susceptible de faire des erreurs si des cellules précédentes sont lancées

# Jupyter notebook / jupyter lab



**jupyter tutorial** Last Checkpoint: 3 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

**PyCon 2018: Using pandas for Better (and Worse) Data Science**

GitHub: <https://github.com/justmarkham/pycon-2018-tutorial>

```
In [1]: import matplotlib.pyplot as plt
import pandas as pd
pd.__version__
```

```
Out[1]: '0.24.1'
```

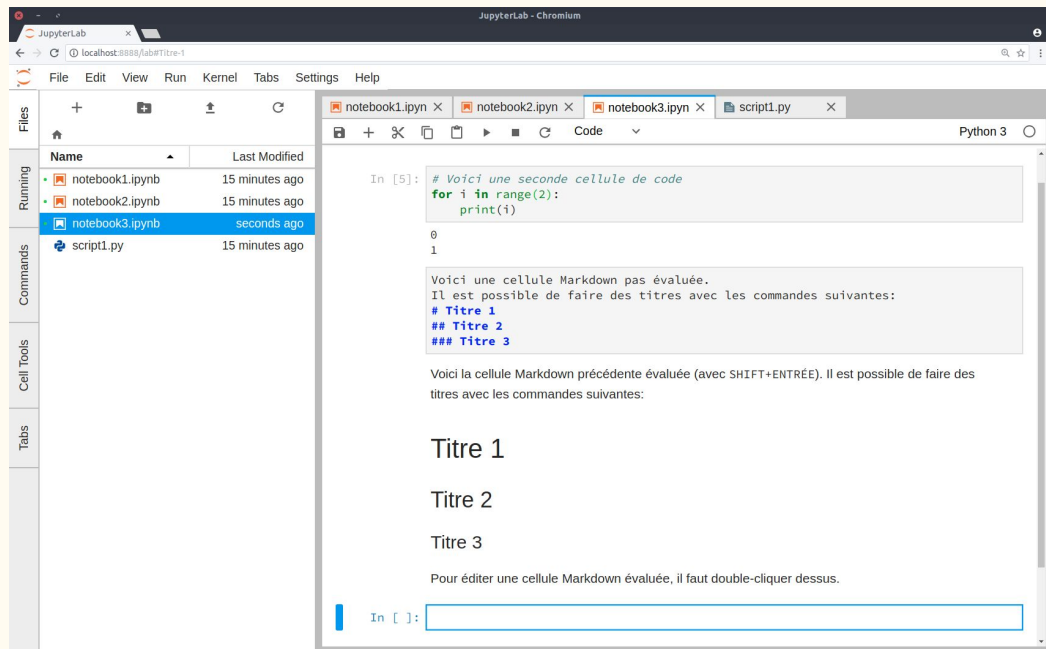
**Dataset: Stanford Open Policing Project (video)**

```
In [2]: # ri stands for Rhode Island
ri = pd.read_csv('police.csv')
```

```
In [3]: # what does each row represent?
ri.head()
```

```
Out[3]:
```

	stop_date	stop_time	county_name	driver_gender	driver_age_raw	driver_age	driver_race	violation_raw	violation	search...
0	2005-01-02	01:55	NaN	M	1985.0	20.0	White	Speeding	Speeding	
1	2005-01-18	08:15	NaN	M	1965.0	40.0	White	Speeding	Speeding	
2	2005-01-23	23:15	NaN	M	1972.0	33.0	White	Speeding	Speeding	
3	2005-02-20	17:15	NaN	M	1986.0	19.0	White	Call for Service	Other	



**JupyterLab - Chromium**

localhost:8888/lab?token=1

File Edit View Run Kernel Tabs Settings Help

notebook1.ipynb x notebook2.ipynb x notebook3.ipynb x script1.py x Python 3

Files

Name	Last Modified
notebook1.ipynb	15 minutes ago
notebook2.ipynb	15 minutes ago
notebook3.ipynb	seconds ago
script1.py	15 minutes ago

Running

Commands

Cell Tools

Tabs

```
In [5]: # Voici une seconde cellule de code
for i in range(2):
    print(i)

0
1
```

Voici une cellule Markdown pas évaluée.  
Il est possible de faire des titres avec les commandes suivantes:

```
# Titre 1
## Titre 2
### Titre 3
```

Voici la cellule Markdown précédente évaluée (avec SHIFT+ENTRÉE). Il est possible de faire des titres avec les commandes suivantes:

Titre 1

Titre 2

Titre 3

Pour éditer une cellule Markdown évaluée, il faut double-cliquer dessus.

```
In [ ]:
```

# Les bonnes pratiques de code en Python

Python est un langage très lisible, il est possible de comprendre ce que fait un programme sans l'exécuter s'il suit un certain nombre de bonnes pratiques.

La communauté a abouti à un certain nombre de principes que l'on peut retrouver dans la PEP 8 (Python Enhancement Proposal).

# Les bonnes pratiques de code en Python

1. Encodage : utf-8
2. L'indentation : elle doit être de 4 espaces
3. Code Layout : 79 caractères maximum par lignes, utiliser le caractère “\” en fin de ligne et écrire la suite de vos instructions après le retour à la ligne
4. L'import des librairies se fait au début du programme
5. Mettre les espaces aux bons endroits (ni trop, ni trop peu)
6. Commenter son code en anglais

Maintenant, place à  
la pratique !

# Les erreurs en Python

—

# Les messages d'erreur en Python

```
liste_de_films = ['Mission Impossible', 'Interstellar', 'Prisoners']
```

```
liste_de_films[4]
```

```
-----  
IndexError                                Traceback (most recent call last)  
<ipython-input-36-ae1dac51d07> in <module>  
----> 1 liste_de_films[4]  
  
IndexError: list index out of range
```



# Les messages d'erreur en Python

**Les messages d'erreur se lisent de bas en haut**

En effet la dernière ligne indique le type d'erreur et l'avant dernière donne l'instruction qui a généré l'erreur. Le reste du message décrit la « pile » des appels ayant généré l'erreur.

# Les messages d'erreur en Python

## Connaître les principaux messages d'erreurs

- NameError

```
>>> a=8
>>> print(A)
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    print(A)
NameError: name 'A' is not defined
```

# Les messages d'erreur en Python

## Connaître les principaux messages d'erreurs

- Opération mathématique impossible

```
>>> x=0
>>> 1/x
Traceback (most recent call last):
  File "<pyshell#23>", line 1, in <module>
    1/x
ZeroDivisionError: division by zero
```

# Les messages d'erreur en Python

## Connaître les principaux messages d'erreurs

- Erreur de syntaxe

```
>>> if a=3:
```

```
SyntaxError: invalid syntax
```

# Les messages d'erreur en Python

## Connaître les principaux messages d'erreurs

- Erreur de type

```
>>> a=6
>>> print("a="+6)
a=6
>>> print("a="+a)
Traceback (most recent call last):
  File "<pyshell#34>", line 1, in <module>
    print("a="+a)
TypeError: must be str, not int
```

# Les messages d'erreur en Python

## Connaître les principaux messages d'erreurs

- Erreur d'indentation

```
: def films(titres):  
    resultats = []  
    return resultats
```

```
File "<ipython-input-39-ff37ff9715cb>", line 2  
    resultats = []  
    ^  
IndentationError: expected an indented block
```

# Les messages d'erreur en Python

## Connaître les principaux messages d'erreurs

- Erreur de parenthèses

```
>>> a=2*(3+4) )  
SyntaxError: invalid syntax
```

# Les messages d'erreur en Python

Ce n'est pas forcément la ligne qui est marquée en erreur qu'il faut corriger...

- Erreur de parenthèses

```
def films(titres):  
    return resultats
```

```
films('Titanic')
```

```
-----  
NameError                                Traceback (most recent call last)  
<ipython-input-42-2f745d2a948e> in <module>  
----> 1 films('Titanic')  
  
<ipython-input-40-aa68d39af552> in films(titres)  
      1 def films(titres):  
----> 2     return resultats  
  
NameError: name 'resultats' is not defined
```



# Utilisation d'un debugger

# Tests unitaires