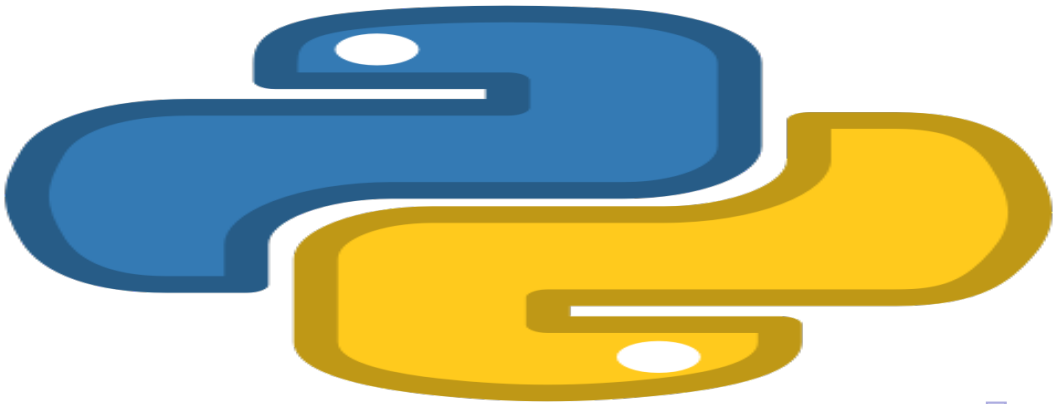
Python : introduction

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

elmouelhi.achref@gmail.com



Plan

Python

Python ?

- langage de programmation
  - orienté objet et procédural
  - fortement typé (les types ne sont pas convertis automatiquement, comme entre une chaîne et un entier sans conversion explicite)
  - interprété
  - à typage dynamique
  - sensible à la casse (case-sensitive) pour les mots-clés du langage, les variables, fonctions, classes...
- créé par **Guido van Rossum** en 1991, puis maintenu par **Python Software Foundation** depuis 2001 (version 2)

Python

Histoire de **Python**

- **1989 : Guido van Rossum** commence à travailler sur **Python** durant ses vacances de Noël, en s'inspirant du langage **ABC**.
- **1991 :** Première version publique de **Python** publiée.
- **Origine du nom :** Le langage est nommé **Python** en référence à la troupe comique britannique *Monty Python*, et non au serpent.
- **2001 :** Création de la **Python Software Foundation (PSF)** pour gérer le développement et la communauté **Python**.
- **2019 : Guido van Rossum** prend sa retraite.
- **Aujourd'hui :** Python est l'un des langages les plus populaires, utilisé dans divers domaines.

Python

Python : objectifs de **Guido van Rossum**

- **Facilité d'apprentissage** : souvent recommandé comme langage d'apprentissage pour les débutants
- **Lisibilité du code** : langage de programmation avec une syntaxe clairement simple et lisible
- **Productivité des développeurs** : avec des structures de données de haut niveau, une gestion automatique de la mémoire, une bibliothèque standard riche en fonctionnalités, un système de gestion d'exceptions...
- **Extensibilité** : conçu pour être extensible : facile d'intégrer du code écrit dans d'autres langages, comme **C** ou **C++**, dans des applications Python.
- ...

Python

Python, pourquoi ?

- Langage de haut niveau, doté de
  - gestion automatique et dynamique de mémoire : ramasse-miettes, pas de pointeur, pas d'allocation de mémoire...
  - système de gestion d'exceptions
- Disposant d'une bonne documentation, des supports vidéos, plusieurs exemples sur internet
- Énorme communauté : un des langages les plus utilisés dans le monde (`https://www.tiobe.com/tiobe-index/`)
- Permettant de développer des programmes :
  - extensibles (avec des bibliothèques C existantes)
  - portables : Windows, Mac OS, Linux
  - ...

Python

Python : multi-domaine

- **Applications Web** : disposant des frameworks très évolués et très populaires comme **Django** et **Flask**.
- **Applications de bureau** : avec **PyQt** et **Tkinter**.
- **Data** : très utilisé en Big Data, Machine Learning et Data Science avec **NumPy**, **pandas**, **Matplotlib**, **seaborn** et **scikit-learn**.
- **Script** : instructions simples à exécuter dans une console.
- **IoT (Internet des objets)** : avec **MicroPython** et **CircuitPython**.
- **Développement de jeux** : avec **Pygame**.
- ...

Python

`Hello world` **en Java**

```
public class Main {

  public static void main (String [] args) {

    System.out.print("Hello world");

  }

}
```

L'équivalent en Python

```
print("Hello world")
```

Python

Quelques extensions de fichier utilisées par **Python**

- `.py` : extension standard pour les fichiers de code source **Python**.
- `.pyc` : fichiers bytecode compilés.
- `.pyo` : fichiers bytecode Python compilés et optimisés.
- `.pyi` : fichiers de type **Python Interface** fournissant des informations sur les types et les signatures de fonctions pour les modules **Python**.
- `.pyx` : fichiers **Cython** (langage facilitant la création de modules **Python** compilées en code **C** ou **C++**).
- ...

Quelques versions de **Python**

- **Python** 1 (sortie en 1991)
- **Python** 1.6 (sortie en septembre 2000)
- **Python** 2.0 (sortie en octobre 2000)
- **Python** 2.1 (sortie en avril 2001)
- **Python** 2.7 (sortie en juillet 2010)
- **Python** 3.0 (sortie en décembre 2008)
- **Python** 3.8 (sortie en octobre 2019)
- **Python** 3.9 (sortie en octobre 2020)
- **Python** 3.10 (sortie en octobre 2021)
- **Python** 3.11 (sortie en octobre 2022)
- **Python** 3.12 (sortie en octobre 2023)
- **Python** 3.13 (sortie en octobre 2024)

Introduction

Python : téléchargement

`https://www.python.org/downloads/`

Introduction

Python : téléchargement

`https://www.python.org/downloads/`

Python : installation

Pensez à cocher la case dans la première fenêtre d'installation pour ajouter une variable de chemin pour **Python**, **PIP**...

Introduction

Python : téléchargement

`https://www.python.org/downloads/`

Python : installation

Pensez à cocher la case dans la première fenêtre d'installation pour ajouter une variable de chemin pour **Python**, **PIP**...

Pour vérifier la version de Python

```
python --version
```

Python

Quel IDE (Environnement de développement intégré) ?

- **Visual Studio Code** (À ne pas confondre avec Visual Studio)
- PyCharm
- Spider (inclus dans la distribution **Anaconda**)
- Eclipse
- ...

Python

Visual Studio Code : téléchargement

`code.visualstudio.com/download`

Python

Visual Studio Code : téléchargement

`code.visualstudio.com/download`

Visual **S**tudio **C**ode (ou **VSC**) , pourquoi ?

- Gratuit
- Multi-langage
- Multi-système d'exploitation
- Extensible via l'installation de quelques extensions

Python

Quelques raccourcis **VSC**

- Pour activer la sauvegarde automatique : aller dans `File > AutoSave`
- Pour indenter son code : `Alt` `Shift` `f`
- Pour commenter/décommenter : `Ctrl` `:`
- Pour sélectionner toutes les occurrences : `Ctrl` `f2`
- Pour sélectionner l'occurrence suivante : `Ctrl` `d`
- Pour placer le curseur dans plusieurs endroits différents : `Alt`

Python

Quelques extensions **VSC** pour **Python**

- **Python** : pour l'intelliSense
- **autopep8** : pour l'indentation et le formatage
- **autoDocstring - Python Docstring Generator** : pour la génération de la documentation (DocString)
- **Code Runner** : pour l'ajout d'un bouton d'exécution/interprétation

Python

Quelques distributions **Python**

- **Anaconda**
  - Distribution multi-plateforme simplifiant le processus d'installation, la gestion des packages et la gestion des environnements virtuel
  - Ayant son propre gestionnaire de packages appelé **Conda**
  - Incluant un ensemble de packages scientifiques tels que **NumPy**, **SciPy**, **pandas**, **Matplotlib**, **scikit-learn**...
- **Miniconda**
  - Version allégée d'**Anaconda**
  - Contenant uniquement **Python** et **Conda**
- **Python(x, y)** : distribution **Python** axée sur les applications scientifiques et techniques.
- **Enthought Canopy** : distribution **Python** fournissant une interface graphique pour les analystes de données ainsi qu'un éditeur de script intégré.
- ...

Python

PEP : **P**ython **E**nhancement **P**roposal

- Propositions d'amélioration de **Python**
- Ensemble de règles qui permet d'homogénéiser le code
- `https://www.python.org/dev/peps/`

Python

Les règles de nommage (**PEP** 8)

- Classes et exceptions : **Pascal case**
- Variables, fonctions, méthodes, fichiers (modules) et dossiers (packages) : **Snake case**
- Constantes : **Snake case** (**All Caps**, en français tout en majuscule en séparant les mots par des underscores)
- Modules et packages, on conseille d'utiliser des noms courts.
- Packages, on déconseille les underscores

Python

Les instructions (**PEP** 8)

- 80 caractères maximum par ligne.
- 4 espaces pour l'indentation (pas de tabulation).
- 2 lignes de séparations entre les différents éléments.
- 1 ligne vide à la fin de chaque fichier de code.
- 1 espace après `:` mais pas avant
- 1 espace avant et après chaque opérateur
- 1 instruction par ligne

Python

Consoles **Python**

- Allez dans la barre de recherche de **Windows**
- Cherchez
  - **Python 3.13** et ouvrez la console (invite de commandes), ou
  - **IDLE (Python 3.13)** et ouvrez la console (Power Shell)

Introduction

Vous pouvez interagir avec Python depuis n'importe quel console en saisissant

```
python
```

Introduction

Vous pouvez interagir avec Python depuis n'importe quel console en saisissant

```
python
```

Ou

```
py
```

Python

Remarques

- **Python** est sensible à la casse.
- Pas besoin d'un point-virgule pour marquer la fin d'une instruction.
- Les points-virgules peuvent être utilisés pour séparer les instructions écrites sur une même ligne bien que cela ne soit pas considéré comme une pratique courante et peut rendre le code moins lisible.
- Les blocs de code sont délimités par l'indentation (Pas de { ... }).

Python

Quelle que soit la console utilisée

- Une instruction **Python** commence par >>>
- Le résultat de l'instruction est affiché à la ligne sans >>>

Python

Quelle que soit la console utilisée

- Une instruction **Python** commence par >>>
- Le résultat de l'instruction est affiché à la ligne sans >>>

Afficher `Hello world` **en Python**

```
>>> print("Hello world")
Hello world
```

Python

Console simple vs IDLE (Integrated Development and Learning Environment)

- **IDLE** = console + coloration du code + auto-complétion (avec tab) + indentation automatique + ...
- Il est possible de configurer **IDLE** pour accéder à l'historique des commandes précédemment exécutées
- Il est possible de stocker l'historique des commandes lancées depuis un **IDLE** dans un fichier
- **IDLE** est évidemment moins rapide que la console

Python

Autres consoles **Python**

- **bpython** (`https://bpython-interpreter.org/`) : une console avec un éditeur de texte avancé offrant la possibilité de modifier les commandes précédentes, auto-indentation, système de suggestion pour l'auto-complétion....
- **ipython** (`https://ipython.org/`) : une console orientée data facilitant la visualisation de données et intégrant des outils de haute performance pour les calculs parallèles.

Python

En utilisant une console Python, on peut effectuer des opérations arithmétiques

```
>>> 3+2
5
```

Python

En utilisant une console Python, on peut effectuer des opérations arithmétiques

```
>>> 3+2
5
```

Ou des tests logiques

```
>>> 3+2>1+7
False
```

Python

Dans la console **IDLE** de **Python**, cliquez sur

- `Alt` + `P` pour récupérer les commandes précédentes
- `Alt` + `N` pour récupérer les commandes suivantes