
Cours IA

Python pour l'IA

3^{ème} Année Lic.

Dr. A. LAKHFIF
2022/2021

Python : notions de base

- Python est un langage interprété, à l'inverse des autres langages compilés comme Java, C++.,,,
- On peut exécuter le code en utilisant la ligne de commande:

```
>>> print("Hello, World!")  
Hello, World!
```

- Ou exécuter un fichier python (extension .py):

```
C:\Users\Your Name>python myfile.py
```

- un tutoriel python complet:

<https://www.w3schools.com/python/default.asp>

Puissance du langage Python

- Supposons que nous avons deux matrices A et B, et nous voulons faire leur produit **AB**

Example C++

```
C = new matrix[A.row_count, B.col_count]

for(int i=0; i < A.row_count, i++)
{
    for(int j=0; j<B.col_count; j++)
    {
        float s = 0;
        for(int k=0; k<A.col_count; k++)
        {
            s += A[i,k] *B[k, j];
        }
        C[i, j] = s;
    }
}
```

Example Python

```
C = A * B;
```

Python : Bibliothèques



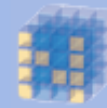
Numpy

- **Numpy** est une bibliothèque Python très importante qui fournit une manipulation des tableaux à **N** dimensions pratique et rapide.
 - Pour le traitement de données de dimension large, Elle fourni une large collection de fonctions mathématiques de haut niveau opérant sur les tableaux .
 - en utilisant **NumPy**, nous pouvons effectuer les opérations suivantes :
 - Opérations mathématiques et logiques sur les tableaux (vecteurs).
 - Opérations de l'algèbre linéaire.

pour installer le module **numpy** : Exécutez la commande suivante:

```
pip install numpy
```

Python : Bibliothèques



NumPy

Exemple :

```
import numpy as np
print("I like ", np.pi)
```

```
C:\Users\AzStation>python
Python 3.8.8rc1 (tags/v3.8.8rc1:dfd7d68, Feb 17 2021, 11:01:21) [MSC v.1928 64 bit (AMD64)] o
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> print("I like ", np.pi)
I like  3.141592653589793
>>>
```

Python : Bibliothèques



- **Scipy** est une bibliothèque Python très importante
- SciPy**, une **bibliothèque** mathématiques **Python** a usage scientifique en ingénierie, Elle est open source sous licence BSD. **SciPy** dépend de **NumPy**,
- **Scipy Contient Les Modules Commun Utilisés En Science & Engineering,**
 - **Optimisation,** **Scipy**
 - **Algèbre Linéaire,**
 - **Intégration**
 - **Traitement du Signal & Image**
 - **Équations Différentielles Ordinaires (ODE Solver)**

pour installer le module **Scipy** : Exécutez la commande suivante:

```
python- m pip install-- user numpy scipy matplotlib ipython jupyter pandas sympy nose
```

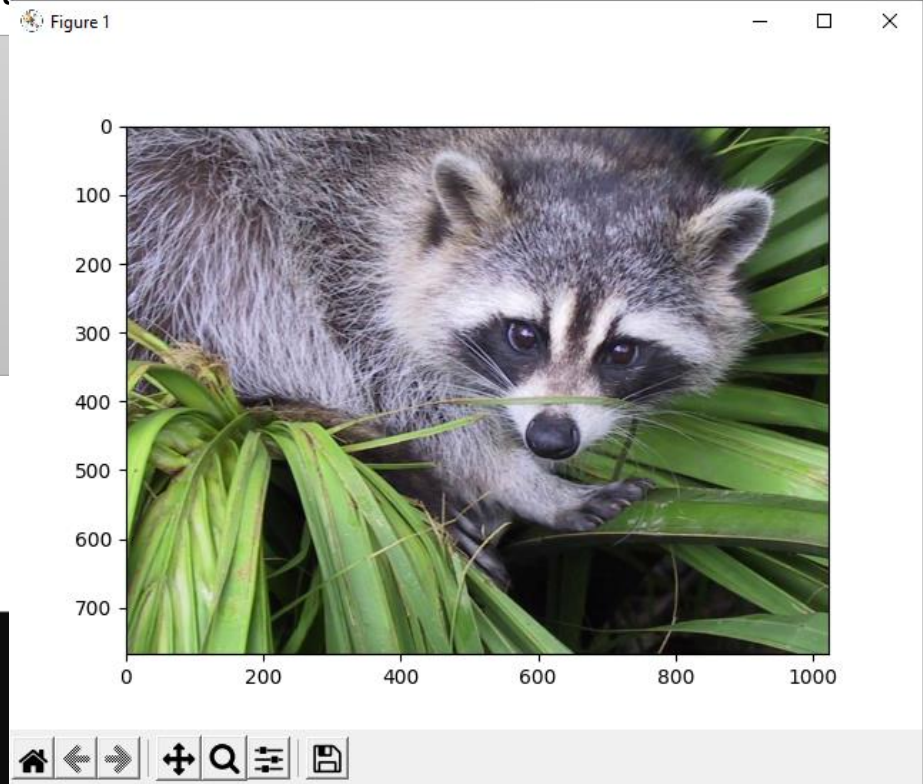
Python : Bibliothèques



Pour tester la bibliothèque **SciPy** library & Matplotlib .

```
from scipy import misc
import matplotlib.pyplot as plt
face = misc.face ()
plt.imshow(face (
plt.show ()
```

```
>>> from scipy import misc
>>> import matplotlib.pyplot as plt
>>>
>>> face = misc.face()
>>> plt.imshow(face)
<matplotlib.image.AxesImage object at 0x0000010193CF0A60>
>>> plt.show()
```



Python : Bibliothèques

$$\begin{array}{rcl} 1x + 5y & = & 6 \\ 3x + 7y & = & 9 \end{array}$$



```
In [20]: from scipy import linalg

equation = np.array([[1, 5], [3, 7]])
solution = np.array([[6], [9]])

roots = linalg.solve(equation, solution)

print("Found the roots:")
print(roots)

print("\n Dot product should be zero if the solutions are correct:")
print(equation.dot(roots) - solution)
```

Found the roots:

```
[[0.375]
 [1.125]]
```

Dot product should be zero if the solutions are correct:

```
[[0.]
 [0.]]
```


Python : Bibliothèques

scikit_learn



Scikit-learn est une librairie pour Python spécialisée dans l'apprentissage automatique . (machine learning)

- Un outil simple et puissant pour l'analyse prédictive des données
- Accessible à tout le monde, réutilisable dans différents contextes
- Construit sur **NumPy, SciPy, & matplotlib**
- Open source, - BSD license

Matplotlib

Matplotlib

[Matplotlib](#) est une bibliothèque de traçage de graphes 2D. Vous pouvez visualiser les données en utilisant **Matplotlib**.

Vous pouvez générer des images des figures dans différents formats. Vous pouvez tracer différents types de diagrammes tels que

- des graphiques à barres,
- des graphiques d'erreur,
- des histogrammes,
- des nuages de points, etc.,

vous pouvez installer le **matplotlib** en utilisant la commande suivante.

```
pip install matplotlib
```

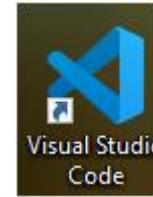
Python : Bibliothèques

TensorFlow & PyTorch

- **TensorFlow** : library deep learning développé by Google
- **PyTorch** library deep learning développé by Facebook.
-

Votre premier programme Python

• **IDE**: Visual Studio Code, Spyder, PyCharm, etc.



- ✓ Créer un fichier **.py**
- ✓ Ecrire un programme simple
- ✓ Exécuter le programme
- ✓ Vérifier la sortie (résultat)

Python : Installation

Python Releases for Windows

- **Windows**

- Download la dernière version 3. x

- **Linux**

```
$python3 --version  
$sudo apt-get update  
$sudo apt-get install python3.6
```

- **MacOS**

Mac OS X 64-bit/32-bit installer

```
python--version
```

- [Latest Python 3 Release - Python 3.6.4](#)
- [Latest Python 2 Release - Python 2.7.14](#)
- [Python 3.6.4 - 2017-12-19](#)
 - Download [Windows x86 web-based installer](#)
 - Download [Windows x86 executable installer](#)
 - Download [Windows x86 embeddable zip file](#)
 - Download [Windows x86-64 web-based installer](#)
 - Download [Windows x86-64 executable installer](#)
 - Download [Windows x86-64 embeddable zip file](#)
 - Download [Windows help file](#)

Python : Package Managers

Gestionnaires de Package

Afin d'installer les packages, nous pouvons utiliser le gestionnaire de Package

- Pip
- Anaconda
- Miniconda

Vous pouvez installer [Anaconda](#) pour obtenir toutes les bibliothèques et modules nécessaires à la DataScience.

• Installation des packages Python avec Conda

```
$conda install tensorflow-gpu
```

- Spécifier les versions des packages

```
$conda install tensorflow-gpu=1.15.0
```

- certains packages sont disponibles sous des chaînes spéciales :

```
$conda install -c vpython vpython
```

Types communs dans Python

- **Numeric:** integers, float, complex
- **Sequence:** list, tuple, range
- **Binary:** byte, bytearray
- **True/False:** bool
- **Text:** string

Python : Concepts de bases Structure de données

Types communs dans Python

C

```
#include "stdio.h"

int main() {
    int x = 3;
    x = 4.5;
}
```

python

```
x = 3
x = 4.5
```

Que se passe-t-il lors de l'exécution

Python : Concepts de bases Structure de données

Types communs

C

```
#include "stdio.h"

int main() {
    int x = 3;
    x = 4.5;
}
```

statiquement typés

python

```
x = 3
x = 4.5
```

Typage Dynamique

Python : Concepts de bases Structure de données

C

Boucles

```
#include "stdio.h"
int main() {
    int i = 0;
    for(i=0; i < 10; i++) {
        printf("%d\n",i);
    }
}
```

`range(start, stop[, step])`

python

```
for i in range(0,10):
    print(i)
```

Python utilise les indentations au lieu des parenthèse

Boucles

`range(start, stop[, step])`

python

```
for i in range(0,10,2):  
    print(i)
```

Que donne l'exécution de ce code,,,?

Boucles

`range(start, stop[, step])`

python

```
for i in range(0,10,2):  
    print(i)
```

0
2
4
6
8

Boucles

python

```
i = 2  
while i < 12:  
    print(i)  
    i+=3
```

2

5

8

11

Conditions

python

```
for i in range(0,10):  
    if i % 2 == 0:  
        print(i)
```

0

2

4

6

8

% : modulo donne le reste de
la division : ex: 3 % 2 = 1

Python : Concepts de bases Structure de données

Fonctions

C

```
int my_abs( int val) {  
    if(val < 0) {  
        return 0-val;  
    }  
    return val;  
}
```

python

```
def my_abs(val):  
    if val < 0:  
        return 0-val  
    return val
```

```
def my_abs(val):  
    if val < 0:  
        return 0-val  
    return val
```

```
print(my_abs(-7))
```

7

Fonctions sur les chaines

- Case

```
>>> word = 'Hello'  
>>> word.lower()
```

hello



```
>>> word.upper()
```

HELLO



- Concaténation

```
>>> '1' + '2'
```

'12'



```
>>> 'Hi' + ' there.'
```

'Hi there.'



Fonctions sur les chaines

- Replication

```
>>> '12'*2
```

```
'1212'
```

```
>>> '1'*2 + '2'*3
```

```
'11222'
```

- Enlever des caractères

```
>>> s= '   Extras \n'
```

```
>>> s.strip()
```

```
'Extras'
```

```
>>> s = '***10***'
```

```
>>> s.strip('*')
```

```
'10'
```

Fonctions sur les chaines

- Conversion

```
>>> word = '1234'  
>>> int(word)
```

```
1234
```

```
>>> float(word)
```

```
1234.0
```

```
>>> word = 'Hi'  
>>> int(word)
```

< Error >

- Copie une partie d'une chaine

```
>>> word = 'Hello'  
>>> word[1:3]
```

```
'el'
```

```
>>> word[4:7]
```

```
'o'
```

Python : Concepts de bases Structure de données

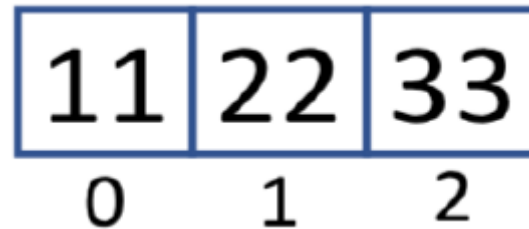
Listes

- Listes

```
>>> list = [11,22,33]
```

```
>>> list
```

```
[11, 22, 33]
```



- Parcours d'une liste

```
>>> list[1]
```

```
22
```

```
>>> list[3]
```

Error – index out of range

```
>>> list = [11,22,33]
```

```
>>> for i in list:
```

```
...     print(i)
```

```
11
```

```
22
```

```
33
```

Python : Concepts de bases Structure de données

Listes

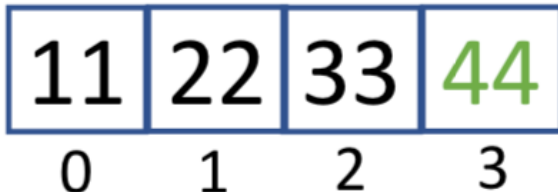
- Ajout

```
>>> list = [11,22,33]
```

```
>>> list.append(44)
```

```
>>> list
```

```
[11, 22, 33, 44]
```



- Suppression

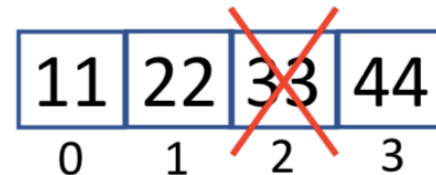
```
>>> list = [11,22,33,44]
```

```
>>> list.pop(2)
```

```
33
```

```
>>> list = [11,22,33,44]
```

```
>>> list.remove(33)
```



Python : Concepts de bases Structure de données

Dictionnaires

- Couple de clé/valeur

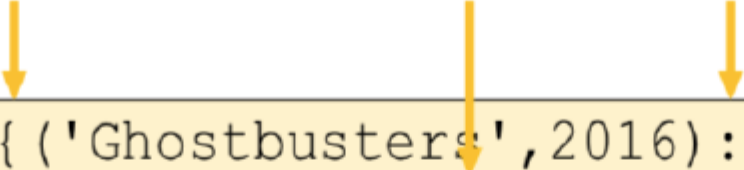
Key	Value
'A12367'	'David Wu'
'A27691'	'Maria Sanchez'
'A16947'	'Tim Williams'
'A21934'	'Sarah Jones'

Key	Value
'CSE8A'	['Christine Alvarado', 'Beth Simon', 'Paul Cao']
'CSE141'	['Dean Tullsen', 'Steve Swanson', 'Leo Porter']
...	...

Key	Value
('Ghostbusters', 2016)	5.4
('Ghostbusters', 1984)	7.8
('Cars', 2006)	7.1
...	...

Python : Concepts de bases Structure de données

Dictionnaires



```
>>> dict = { ('Ghostbusters', 2016): 5.4,  
             ('Ghostbusters', 1984): 7.8 }
```

```
>>> tuple1
```

```
{ ('Ghostbusters', 2016): 5.4,  
  ('Ghostbusters', 1984): 7.8 }
```

```
>>> dict[('Ghostbusters', 2016)]
```

```
5.4
```

```
>>> len(dict)
```

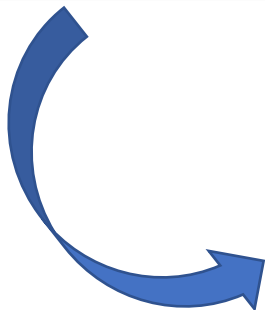
```
2
```

Les ensemble : Set()

Set() est une collection non ordonnée d'items.

```
my_set = {1, 2, 3}
print(my_set)

# set of mixed datatypes
my_set = {1.0, "Hello", (1, 2, 3)}
print(my_set)
```



```
{1, 2, 3}
{1.0, (1, 2, 3), 'Hello'}
```