UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE
ÉCOLE DE MANAGEMENT
DE LA SORBONNE

Python
Usage des Bibliothèques

Enseignant Responsable

Manuele Kirsch Pinheiro

Manuele.Kirsch-Pinheiro@univ-paris1.fr

Exemples

https://replit.com/@ManueleKirsch/SystemeInformationInformatique

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE
ÉCOLE DE MANAGEMENT
DE LA SORBONNE

Les bibliothèques en Python

- Une bibliothèque est un **catalogue de fonctions**
  - Ensemble des codes prêts et disponibles pour usage
  - **Solutions préexistantes** pour des problèmes connus
  - On y trouve de **fonctions**, de **classes** et de **constantes**
  - **Réutilisation d'un savoir-faire**

On ne réinvente pas la roue !

- Les bibliothèques ont largement contribué à la renommée de Python
  - Des nombreuses ressources supplémentaires sont disponibles
  - Sujets très variés :
    - **Data analyse, Machine Learning**, mais aussi biologie, Big Data…

Bibliothèques

```
from math import sqrt
r = sqrt(2.0)
```

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE
ÉCOLE DE MANAGEMENT
DE LA SORBONNE

Python : Fonctions & Bibliothèques

- **Bibliothèque standard** disponible dans l'installation Python offre des nombreuses possibilités
  - Exemples : math, randon, decimal, datetime, calendar…

- Pour utiliser une bibliothèque, on doit importer la bibliothèque, puis indiquer la(es) fonction(s) ou classe(s) qu'on veut utiliser :

À partir de la bibliothèque ***math*** *on veut la fonction* ***sqrt***

```
from math import sqrt

racine = sqrt(2.0)
```

qu'on utilise ici

À partir de la bibliothèque ***randon*** *on veut la fonction* ***randint***

```
from random import randint

tirageAuSort = randint(1,6)
```

qu'on utilise après dans le code

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE
ÉCOLE DE MANAGEMENT
DE LA SORBONNE

Python : Fonctions & Bibliothèques

- D'autres bibliothèques sont disponibles
  - Exemple : **numpy**, **pandas**, **scikit-learn**, **matplotlib**, tensorflow…

- On doit **installer** les nouvelles bibliothèques
  - Ajouter la bibliothèque à l'installation Python de base
  - Ce n'est qu'après l'avoir installé, qu'on peut l'utiliser

- Installation d'une nouvelle bibliothèque
  - Outil « **pip** » disponible dans l'installation de base Python
  - Il va falloir ouvrir un **terminal** et entrer la commande
    - **python3 -m pip install numpy**
    - **python3 -m pip install scikit-learn**

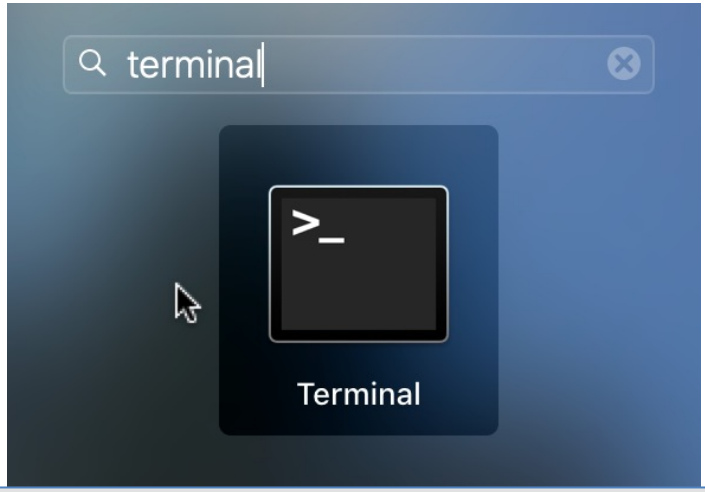
UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE
ÉCOLE DE MANAGEMENT
DE LA SORBONNE

Python : Fonctions & Bibliothèques

- **Exemple installation NumPy (sur Mac)**

1. Ouvrir un terminal
   - Applications → Utilitaires → Terminal

2. Vérifier si « pip » est bien disponible
   - **python3** -m pip --version



kirsch — -zsh — 80×24

```
Last login: Sat Feb 27 19:14:40 on ttys000
kirsch@lilith ~ % python3 -m pip --version
pip 20.2.3 from /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/
site-packages/pip (python 3.9)
kirsch@lilith ~ %
```

C'est bon ! On peut continuer !

kirsch — -zsh — 80×24

```
Last login: Thu Mar 18 15:02:39 on ttys000
kirsch@lilith ~ % python -m pip --version
/System/Library/Frameworks/Python.framework/Versions/2.7/Resources/Python.app/Co
ntents/MacOS/Python: No module named pip
kirsch@lilith ~ %
```

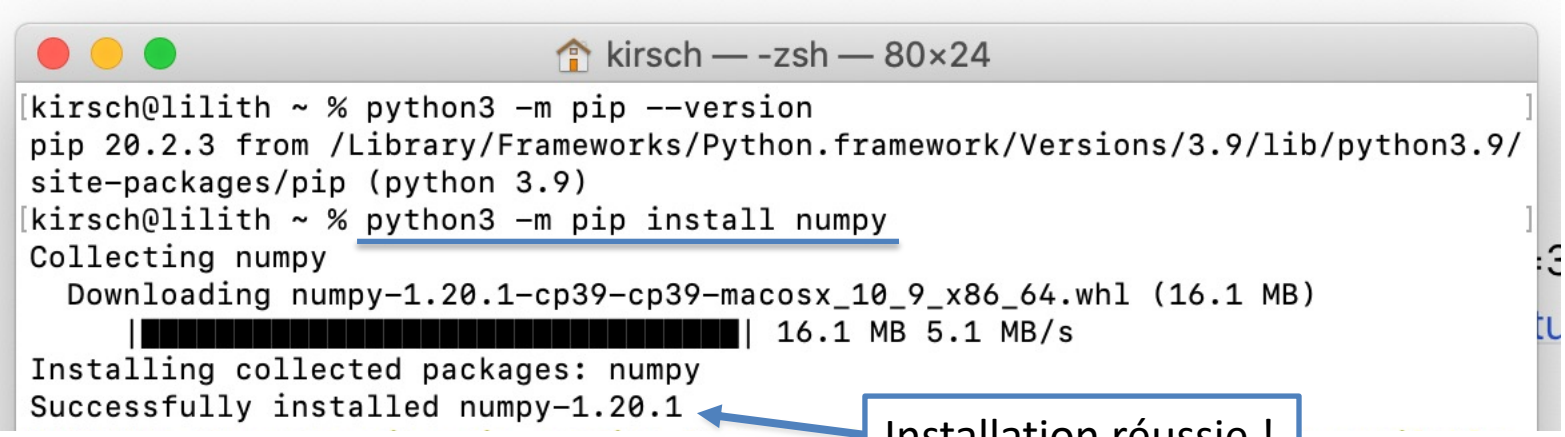
Problème : La version de Python est trop ancienne, il faut installer une version plus à jour.

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE
ÉCOLE DE MANAGEMENT
DE LA SORBONNE

Python :
Fonctions & Bibliothèques

- **Exemple installation NumPy (sur Mac)**

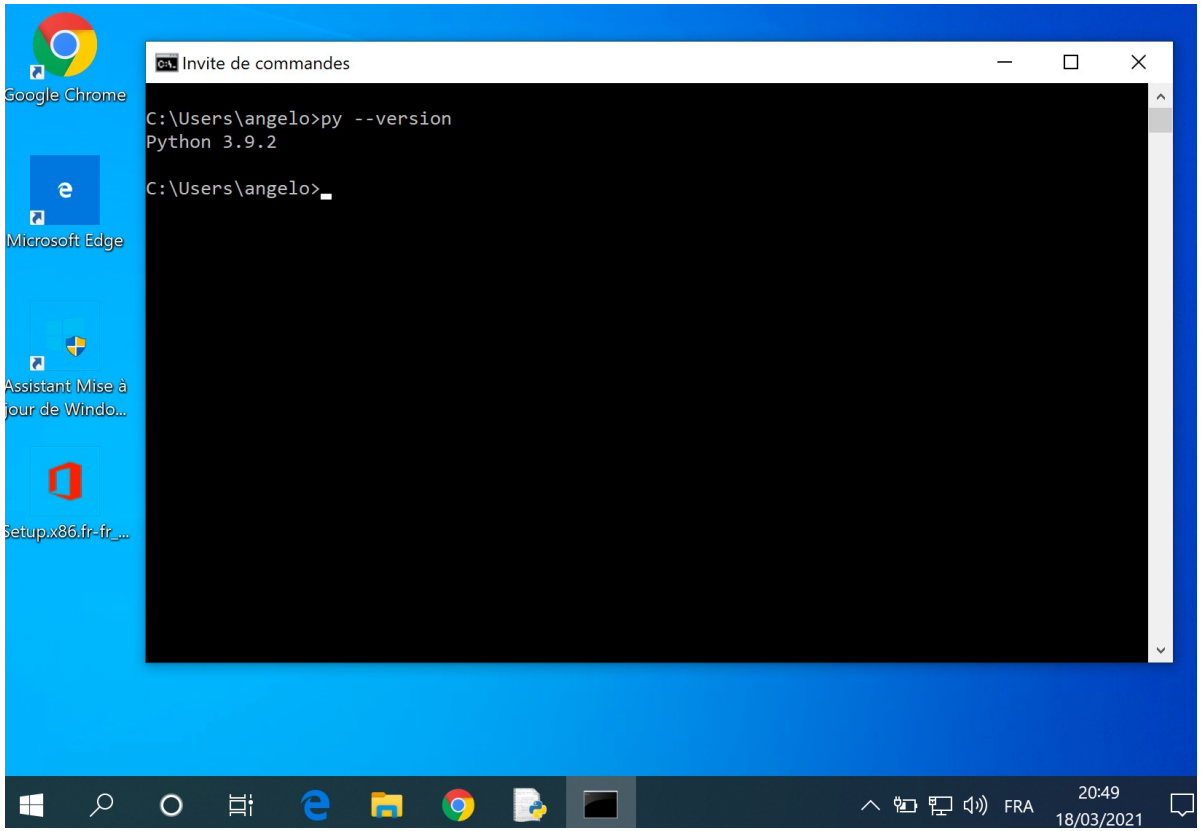
3. installer la bibliothèque NumPy
   - **python3** -m pip install numpy

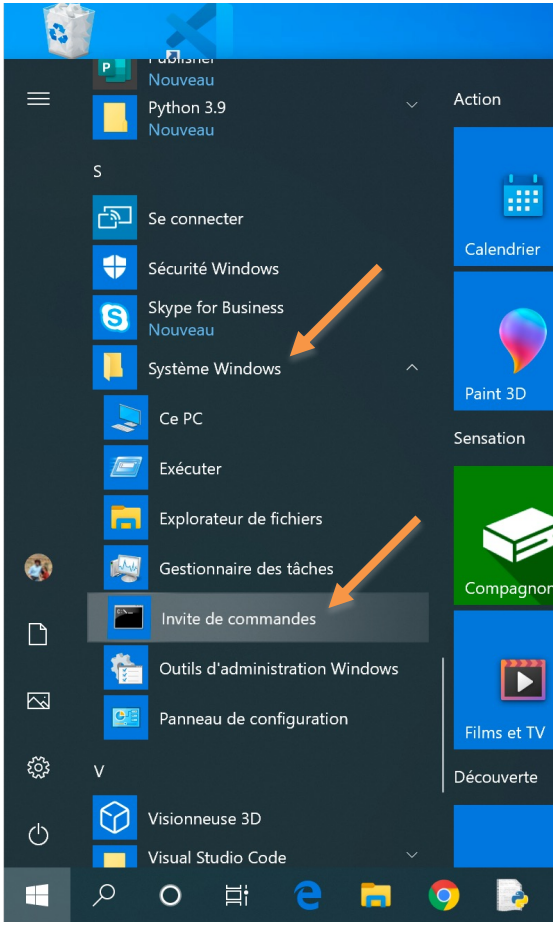


UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE
ÉCOLE DE MANAGEMENT
DE LA SORBONNE

Python :
Fonctions & Bibliothèques

- **Exemple installation NumPy (sur Windows)**
  1. Ouvrir un terminal
     - Système Windows → Invité de commandes

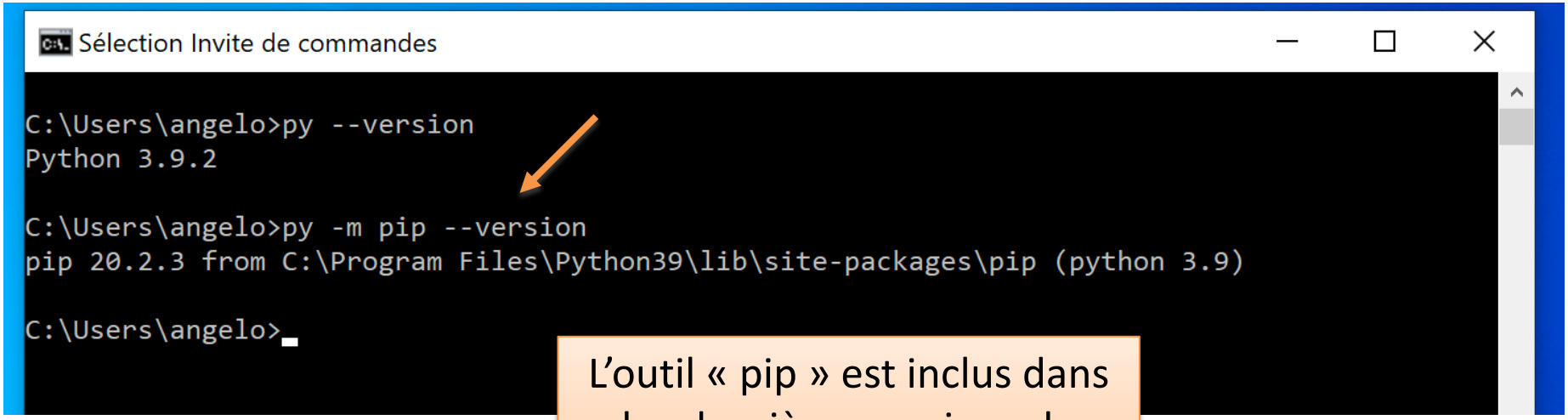




Python :
Fonctions & Bibliothèques

- **Exemple installation NumPy (sur Windows)**

2. Vérifier si « pip » est bien disponible
   - **py** -m pip --version

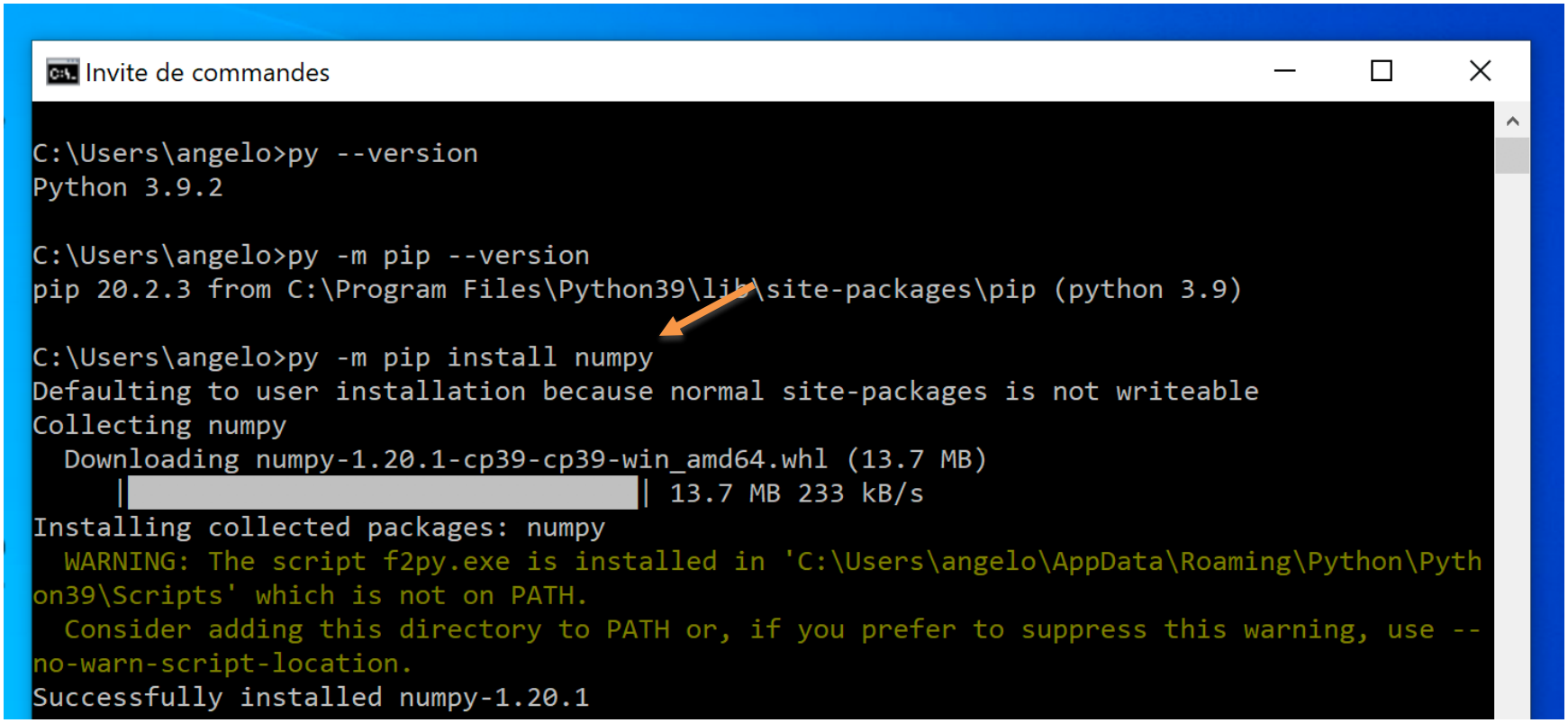


L'outil « pip » est inclus dans les dernières versions de Python
(à partir de la version 3.4)

Python :
Fonctions & Bibliothèques

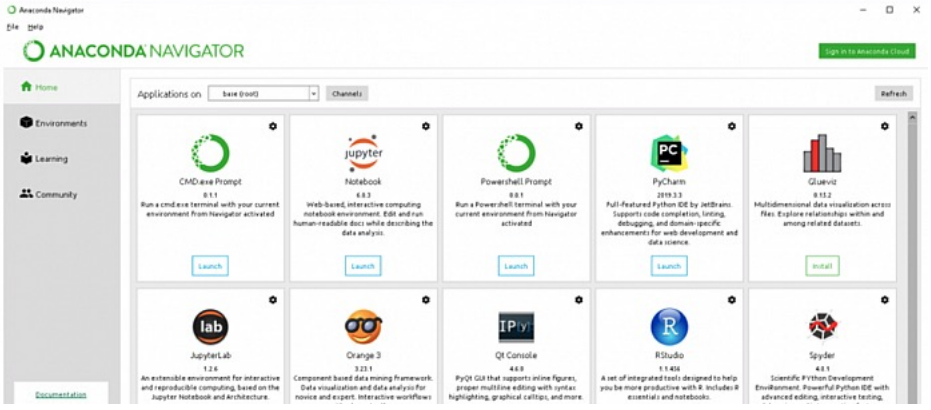
- **Exemple installation NumPy (sur Windows)**

3. installer la bibliothèque NumPy
   - **py** -m pip install numpy



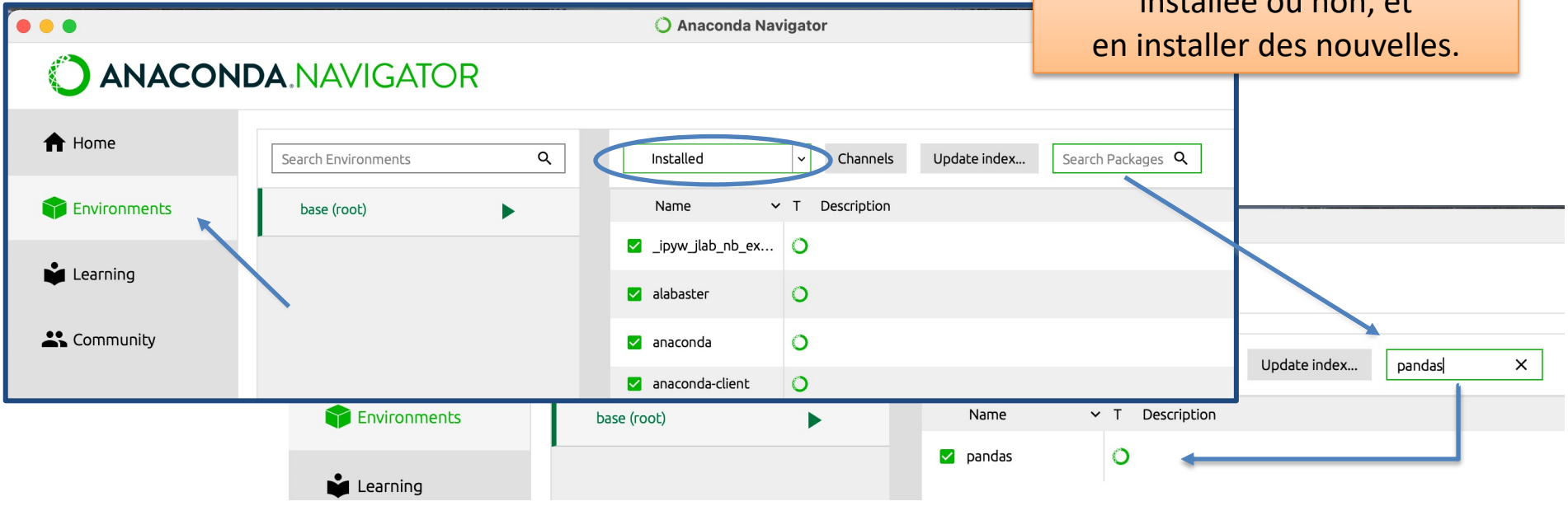
UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE
ÉCOLE DE MANAGEMENT
DE LA SORBONNE

Python : Fonctions & Bibliothèques



- **Installation avec Anaconda**
  - Anaconda est un « **gestionnaire de paquets** »
  - Il permet de **gérer son installation** Python et ses **bibliothèques**
  - Interface graphique : **Anaconda.Navigator**

On peut savoir ce qui est installé, savoir si une bibliothèque est installée ou non, et en installer des nouvelles.



UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE
ÉCOLE DE MANAGEMENT
DE LA SORBONNE

Python : Bibliothèque NumPy

Quelques liens :

- https://numpy.org/doc/stable/user/absolute_beginners.html
- https://numpy.org/doc/stable/user/basics.html

- **Exemple bibliothèque NumPy**
  - Bibliothèque dédié au **calcul scientifique**
  - Simple et **performante**
  - Propose la structure **array** (classe **ndarray**) pour la manipulation des **matrices multidimensionnelles**
    - Matrice contenant des éléments d'un même type (float, int, objet)
    - Nombreuses opérations de calcul scientifique (algèbre linéaire, vecteurs…)
  - Exemple : ExempleNumpy.py

Importer la bibliothèque
as → alias « **np** » (pour faire court)

Création d'une matrice
2 x 3 remplie de zéros

Position [1][2]

0 0 0
0 0 **3**

Création d'une matrice 2x2 capable de contenir n'importe quel type d'objet

```
import numpy as np

matrice = np.zeros( (3, 2) )
matrice[1][2] = 3

matriceObjets = np.empty( (2, 2) , dtype='object')
matriceObjets[1][0] = 'Toto'
```

Position [1][0]

None None
'Toto' None

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE
ÉCOLE DE MANAGEMENT
DE LA SORBONNE

Python : Bibliothèque NumPy

- Quelques éléments intéressants : **quelques opérations**
  - Exemple : ExempleNumpy.py

```
matrice = np.zeros( (2, 3) )
```

```
1.  0.
2.  0.
3.  3.
```

```
mI = matrice.transpose()
```

mI

```
1. 2. 3.
0. 0. 3.
```

Matrice transposée

```
diagonal = matrice.diagonal()
```

→ `[ 1.  0. ]` Diagonal

```
sommeM = matrice.sum()
```

9.0

Somme tous éléments

```
sommeLigne = matrice.sum(axis=1)
```

→ `[ 1.  2.  6. ]` (**axe = 1**)

Somme de chaque **ligne**

```
sommeColonne = matrice.sum(axis=0)
```

→ `[ 6.  3. ]` Somme de chaque **colonne** (**axe x=0**)

```
sup2 = matrice[matrice >= 2]
```

`[ 2.  3.  3. ]`

Éléments **>= 2**

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE
ÉCOLE DE MANAGEMENT
DE LA SORBONNE

Python : Bibliothèque Pandas

- **Pandas**
  - Importante bibliothèque dédiée à l'analyse de données
  - Deux structures de données majeures : **Series** et **DataFrame**

Series
séries de valeurs indexées
(c.a.d. séquence *<key, value>*)
Très utiles pour les **séries temporelles**

DataFrame : structure tabulaire, où les données sont organisées en **colonnes** et **indexées**

index	value
'Suisse'	8
'France'	70
'USA'	320
'Chine'	1200

Structure la plus utilisée pour la data analyse

	colonnes	
index	'Habitants'	'Capital'
'Suisse'	8	'Geneve'
'France'	70	'Paris'
'USA'	320	'Washington'
'Chine'	1200	'Pequin'

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE
ÉCOLE DE MANAGEMENT
DE LA SORBONNE

Python : Bibliothèque Pandas
DataFrame

```
import pandas as pnd
```

Dictionnaire avec les données

```
monDF = pnd.DataFrame ( { 'Habitants' : [ 8, 70, 320, 1200 ],
                          'Capital': ['Geneve', 'Paris', 'Washington', 'Pequin'] },
                          index=['Suisse', 'France', 'USA', 'Chine'])
```

listes avec les valeurs d'index

colonnes

index	'Habitants'	'Capital'
'Suisse'	8	'Geneve'
'France'	70	'Paris'
'USA'	320	'Washington'
'Chine'	1200	'Pequin'

```
monDF['Capital']
```

'Capital'
'Geneve'
'Paris'
'Washington'
'Pequin'

*On peut récupérer les données d'une **colonne***
`monDF ['nom colonne']`

```
monDF.loc['France']
```

'Habitants'	'Capital'
70	'Paris'

*Ou les données d'une **ligne** avec l'opération **loc***
`monDF.loc ['val index']`

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE
ÉCOLE DE MANAGEMENT
DE LA SORBONNE

Python : Bibliothèque Pandas
DataFrame

- On peut lire le **contenu d'un fichier** vers un **DataFrame**
- Différents **formats** : **CSV**, Excel, JSON, SQL…
- Opérations ***read_xxx : read_csv, read_excel, read_sql***…

Fichier CSV

```
CODE FACTURE;DATE FACTURE;CODE CLIENT; MONTANT
FA-2008-0010;2008-03-12;CLI009; 752.98
FA-2008-0006;2008-02-04;CLI038; 374.84
```

En-tête (nom des colonnes)

Données séparées par ;
*(**séparateur**)*

Fichier à lire

```
ventes = pnd.read_csv('files/VentesAgenceU.csv',
                      delimiter=';',
                      header=[0],
                      index_col=[0] )
```

Séparateur

Ligne(s) d'entête

Index

Autres options possibles
Par ex. si index est une date :

```
parse_dates=True
dayfirst=True
```

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE
ÉCOLE DE MANAGEMENT

Python : Bibliothèque Pandas DataFrame

```
ventes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 50 entries, FA-2008-0010 to nan
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   DATE FACTURE   47 non-null     object
 1   CODE CLIENT    47 non-null     object
 2   SECTEUR        47 non-null     object
 3   VENDEUR        47 non-null     object
 4    MONTANT       47 non-null     object
 5   Unnamed: 6     0 non-null      float64
dtypes: float64(1), object(5)
memory usage: 2.7+ KB
```

Une fois le fichier lu, on peut vérifier les informations :

- monDF.**info** ()
- monDF.**describe** ()
- monDF.**head** (*n*)
- monDF.**tail** (*n*)

info *: informations sur le DF*

head *: premières lignes du DF*

```
[2]: ventes.head()
```

[2]:

CODE FACTURE	DATE FACTURE	CODE CLIENT	SECTEUR	VENDEUR	MONTANT	Unnamed: 6
FA-2008-0010	12/03/2008	CLI009	AUTOMATISME	BOUVET	752,98	NaN
FA-2008-0006	04/02/2008	CLI038	AUTOMATISME	ENGUENT	374,84	NaN
FA-2008-0009	03/03/2008	CLI098	ELECTRICITE	BOUVET	935,47	NaN
FA-2008-0012	31/03/2008	CLI114	ELECTRICITE	DEVEAUX	752,98	NaN
FA-2008-0008	22/02/2008	CLI115	AUTOMATISME	BOUVET	677,36	NaN

describe *: analyse rapide des valeurs numériques du DF*

```
[3]: ventes.describe()
```

[3]:

	Unnamed: 6
count	0.0
mean	NaN
std	NaN
min	NaN
25%	NaN
50%	NaN
75%	NaN
max	NaN

Valeurs vides