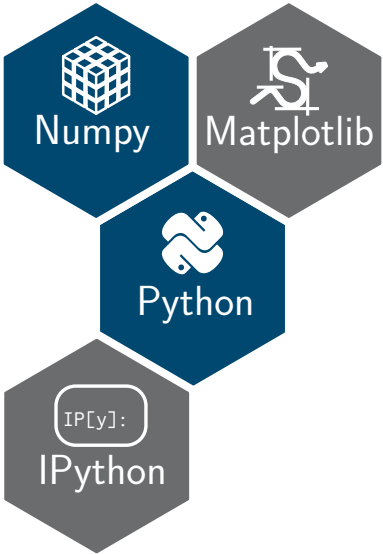


Option « Programmation en Python »

L'environnement Python

Environnements virtuels

- Un environnement virtuel permet d'avoir des installations de Python décorrélées les unes des autres et **isolées du système**
  - permet de travailler avec différentes versions de Python (version 2.X, 3.X)
  - permet de travailler avec différentes versions de modules (`matplotlib`, `numpy`,...)
  - **permet de nous affranchir des droits administrateurs**

Environnements virtuels

- Déclaration d'un environnement virtuel

```
>_ python3.5 -m venv ~/python.d/my_python_env

>_ ls ~/python.d/my_python_env
bin  include  lib  lib64  pip-selfcheck.json  pyvenv.cfg  share
```

- Chargement de l'environnement virtuel

```
>_ source ~/python.d/my_python_env/bin/activate

>_ which python
~/python.d/my_python_env/bin/python
```

⚠ La commande `source` doit être exécutée à chaque nouvelle session !

- Déclaration d'un environnement virtuel

```
>_ python3.5 -m venv ~/python.d/my_python_env

>_ ls ~/python.d/my_python_env
bin  include  lib  lib64  pip-selfcheck.json  pyvenv.cfg  share
```

- Chargement de l'environnement virtuel

```
>_ source ~/python.d/my_python_env/bin/activate

>_ which python
~/python.d/my_python_env/bin/python
```

⚠ La commande `source` doit être exécutée à chaque nouvelle session !

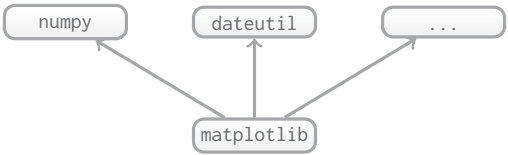
Gestionnaire de modules `pip`

- `pip` est **un système de gestion de paquets** utilisé pour installer et gérer des librairies écrites en Python
  - gestion des versions de modules/librairies (`matplotlib`, `numpy`, `django`, ...)
  - gestion & installation des dépendances

- Liste des libraries consultables sur le site *Python Package Index* (96 537 paquets)

Gestionnaire de modules `pip`

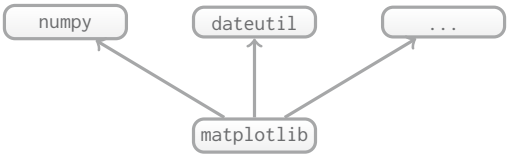
- `pip` est **un système de gestion de paquets** utilisé pour installer et gérer des librairies écrites en Python
  - gestion des versions de modules/librairies (`matplotlib`, `numpy`, `django`, ...)
  - gestion & installation des dépendances



- Liste des libraries consultables sur le site *Python Package Index* (96 537 paquets)

Gestionnaire de modules `pip`

- `pip` est **un système de gestion de paquets** utilisé pour installer et gérer des librairies écrites en Python
  - gestion des versions de modules/librairies (`matplotlib`, `numpy`, `django`, ...)
  - gestion & installation des dépendances



- Liste des libraries consultables sur le site *Python Package Index* (96 537 paquets)

Gestionnaire de modules `pip`

- Installation d'un module

```
>_ pip install matplotlib
```

- Installation d'une version donnée d'un module

```
>_ pip install matplotlib==1.5.0
```

- Suppression d'un module

```
>_ pip uninstall matplotlib
```

- Mise à jour d'un module

```
>_ pip install matplotlib --upgrade
```

Gestionnaire de modules `pip`

- Installation d'un module

```
>_ pip install matplotlib
```

- Installation d'une version donnée d'un module

```
>_ pip install matplotlib==1.5.0
```

- Suppression d'un module

```
>_ pip uninstall matplotlib
```

- Mise à jour d'un module

```
>_ pip install matplotlib --upgrade
```

Gestionnaire de modules `pip`

- Installation d'un module

```
>_ pip install matplotlib
```

- Installation d'une version donnée d'un module

```
>_ pip install matplotlib==1.5.0
```

- Suppression d'un module

```
>_ pip uninstall matplotlib
```

- Mise à jour d'un module

```
>_ pip install matplotlib --upgrade
```

Gestionnaire de modules `pip`

- Installation d'un module

```
>_ pip install matplotlib
```

- Installation d'une version donnée d'un module

```
>_ pip install matplotlib==1.5.0
```

- Suppression d'un module

```
>_ pip uninstall matplotlib
```

- Mise à jour d'un module

```
>_ pip install matplotlib --upgrade
```

Gestionnaire de modules `pip`

- Liste des modules installés

```
>_ pip freeze
```

- Liste des modules pouvant être mis à jour

```
>_ pip list --outdated
```

- Documentation de `pip` et de ses commandes https://pip.pypa.io

Gestionnaire de modules `pip`

- Liste des modules installés

```
>_ pip freeze
```

- Liste des modules pouvant être mis à jour

```
>_ pip list --outdated
```

- Documentation de `pip` et de ses commandes https://pip.pypa.io

Interpréteurs python

- Interpréteur standard → utilisation limitée

```
>_ python
Python 3.5.2 (default, Oct 14 2016, 12:54:53)
[GCC 6.2.1 20160916 (Red Hat 6.2.1-2)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 2*2
4
>>> exit()
```

- Interpréteur intéractif ipython
  - **historique des commandes** → touches ↑ et ↓
  - **auto-complétion** → touche <TAB>
  - édition en ligne du code
  - extraction automatique de **la documentation des fonctions** et objets python
  - interaction avec le shell du système d'exploitation

- Interpréteur standard → utilisation limitée

```
>_ python
Python 3.5.2 (default, Oct 14 2016, 12:54:53)
[GCC 6.2.1 20160916 (Red Hat 6.2.1-2)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 2*2
4
>>> exit()
```

- Interpréteur intéractif ipython
  - **historique des commandes** → touches ↑ et ↓
  - **auto-complétion** → touche <TAB>
  - édition en ligne du code
  - extraction automatique de **la documentation des fonctions** et objets python
  - interaction avec le shell du système d'exploitation

- Interpréteur standard → utilisation limitée

```
>_ python
Python 3.5.2 (default, Oct 14 2016, 12:54:53)
[GCC 6.2.1 20160916 (Red Hat 6.2.1-2)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 2*2
4
>>> exit()
```

- Interpréteur intéractif `ipython`
  - **historique des commandes** → touches ↑ et ↓
  - **auto-complétion** → touche <TAB>
  - édition en ligne du code
  - extraction automatique de **la documentation des fonctions** et objets `python`
  - interaction avec le shell du système d'exploitation

- Installation *via* pip

```
>_ pip install ipython
```

- Utilisation de ipython

```
>_ ipython
Python 3.5.2 (default, Oct 14 2016, 12:54:53)
Type "copyright", "credits" or "license" for more information.

IPython 5.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: 2*2
Out[1]: 4
```

- Premier programme python : *"Hello world"*

```
In [1]: print("Hello world")
Hello world
```

- Obtenir de l'aide *via* l'opérateur ?

```
In [2]: print?
Docstring:
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

Prints the values to a stream, or to sys.stdout by default.
Optional keyword arguments:
file:  a file-like object (stream); defaults to the current sys.stdout.
sep:   string inserted between values, default a space.
end:   string appended after the last value, default a newline.
flush: whether to forcibly flush the stream.
Type:      builtin_function_or_method
```

- Premier programme python : *"Hello world"*

```
In [1]: print("Hello world")
Hello world
```

- Obtenir de l'aide *via* l'opérateur ?

```
In [2]: print?
Docstring:
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

Prints the values to a stream, or to sys.stdout by default.
Optional keyword arguments:
file:  a file-like object (stream); defaults to the current sys.stdout.
sep:   string inserted between values, default a space.
end:   string appended after the last value, default a newline.
flush: whether to forcibly flush the stream.
Type:      builtin_function_or_method
```

- Historique des commandes

```
In [1]: x = 10

In [2]: <UP>

In [2]: x = 10
```

- Auto-complétion

```
In [1]: x = 10

In [2]: x.<TAB>
x.bit_length   x.denominator  x.imag         x.real
x.conjugate    x.from_bytes   x.numerator    x.to_bytes
```

- Historique des commandes

```
In [1]: x = 10

In [2]: <UP>

In [2]: x = 10
```

- Auto-complétion

```
In [1]: x = 10

In [2]: x.<TAB>
x.bit_length   x.denominator  x.imag         x.real
x.conjugate    x.from_bytes   x.numerator    x.to_bytes
```

Fonctions internes à ipython : ces fonctions sont préfixées du caractère %

- %whos : afficher un résumé des variables déclarées

```
In [1]: x = 10

In [2]: %whos
Variable   Type    Data/Info
----------------------------
x          int     10
```

- %timeit : évalue le temps moyen d'exécution d'un code

```
In [1]: %timeit x = 10
10000000 loops, best of 3: 13.7 ns per loop
```

Fonctions internes à ipython : ces fonctions sont préfixées du caractère %

- %history : affiche l'historique des commandes tapées depuis la session courante de ipython

```
In [1]: x = 10

In [2]: %history
x = 10
%history
```

La commande %history -g vous retournera l'ensemble des commandes saisies depuis la toute première session ipython.

ipython fournit enfin des alias vers les commandes unix standards

```
In [1]: alias
Total number of aliases: 16
Out[1]:
[('cat', 'cat'),
 ('clear', 'clear'),
 ('cp', 'cp'),
 ('ldir', 'ls -F -o --color %l | grep /$'),
 ('less', 'less'),
 ('lf', 'ls -F -o --color %l | grep ^-'),
 ('lk', 'ls -F -o --color %l | grep ^l'),
 ('ll', 'ls -F -o --color'),
 ('ls', 'ls -F --color'),
 ('lx', 'ls -F -o --color %l | grep ^-..x'),
 ('man', 'man'),
 ('mkdir', 'mkdir'),
 ('more', 'more'),
 ('mv', 'mv'),
 ('rm', 'rm'),
 ('rmdir', 'rmdir')]
```