Apprentissage de la programmation

avec Python 3

Christian VINCENT

LETTRES
SORBONNE
UNIVERSITÉ

L1 - Semestre 1 - année 2023-2024

Pourquoi Python et comment ?

- Chapitre 0 -

L'apprentissage de la programmation (et donc de l'algorithmique) permet de développer :

L'apprentissage de la programmation (et donc de l'algorithmique) permet de développer :

- Un esprit plus analytique, plus conceptuel, bref plus mathématique

L'apprentissage de la programmation (et donc de l'algorithmique) permet de développer :

- Un esprit plus analytique, plus conceptuel, bref plus mathématique
- Le passage de la pensée globale à une forme plus séquentielle de celle-ci

L'apprentissage de la programmation (et donc de l'algorithmique) permet de développer :

- Un esprit plus analytique, plus conceptuel, bref plus mathématique
- Le passage de la pensée globale à une forme plus séquentielle de celle-ci
- La démystification du coté «magique» de l'informatique

L'apprentissage de la programmation (et donc de l'algorithmique) permet de développer :

- Un esprit plus analytique, plus conceptuel, bref plus mathématique
- Le passage de la pensée globale à une forme plus séquentielle de celle-ci
- La démystification du coté «magique» de l'informatique
- La compréhension d'algorithme, leur mise en œuvre, et les prémices de la notion «d'intelligence artificielle»

Les bienfaits de la programmation...

Les bienfaits de la programmation...

- Savoir programmer fait partie d'une attitude responsable et citoyenne, et permet de devenir "acteur" et non plus "simple consommateur".

Les bienfaits de la programmation...

- Savoir programmer fait partie d'une attitude responsable et citoyenne, et permet de devenir "acteur" et non plus "simple consommateur".

- Il est important pour comprendre les "manipulations" de notre société, de savoir comment elles opèrent.

Les bienfaits de la programmation...

- Savoir programmer fait partie d'une attitude responsable et citoyenne, et permet de devenir "acteur" et non plus "simple consommateur".

- Il est important pour comprendre les "manipulations" de notre société, de savoir comment elles opèrent.

Grandes dates de l'histoire de *Python*

Grandes dates de l'histoire de *Python*

- *Python* est un langage de programmation.

Grandes dates de l'histoire de *Python*

- *Python* est un langage de programmation.
- Guido van Rossum (Pays-Bas) est à l'origine de ce langage (développé à partir du langage ABC).

Grandes dates de l'histoire de *Python*

- *Python* est un langage de programmation.
- Guido van Rossum (Pays-Bas) est à l'origine de ce langage (développé à partir du langage ABC).
- Publication de la version 0.9 en 1 991.

Grandes dates de l'histoire de *Python*

- *Python* est un langage de programmation.
- Guido van Rossum (Pays-Bas) est à l'origine de ce langage (développé à partir du langage ABC).
- Publication de la version 0.9 en 1 991.
- 2 008 sortie conjointe de la version 2.6 et première version de la version 3, la 3.0.

Grandes dates de l'histoire de *Python*

- *Python* est un langage de programmation.
- Guido van Rossum (Pays-Bas) est à l'origine de ce langage (développé à partir du langage ABC).
- Publication de la version 0.9 en 1 991.
- 2 008 sortie conjointe de la version 2.6 et première version de la version 3, la 3.0.
- Actuellement, en février 2 021, fin du développement de la branche 2.x et en septembre 2 023 version 3.11.5

Il existe des centaines de langages de programmation

Il existe des centaines de langages de programmation

- Langages de *bas niveau* :

Il existe des centaines de langages de programmation

- Langages de *bas niveau* : c'est-à-dire assez proche du système binaire de la machine.

Il existe des centaines de langages de programmation

- Langages de *bas niveau* : c'est-à-dire assez proche du système binaire de la machine. On y trouve des langages comme :

Il existe des centaines de langages de programmation

- Langages de *bas niveau* : c'est-à-dire assez proche du système binaire de la machine. On y trouve des langages comme : l'assembleur

Il existe des centaines de langages de programmation

- Langages de *bas niveau* : c'est-à-dire assez proche du système binaire de la machine. On y trouve des langages comme : l'assembleur, le C,

Il existe des centaines de langages de programmation

- Langages de *bas niveau* : c'est-à-dire assez proche du système binaire de la machine. On y trouve des langages comme : l'assembleur, le C,le C++, et d'autres...

Il existe des centaines de langages de programmation

- Langages de *bas niveau* : c'est-à-dire assez proche du système binaire de la machine. On y trouve des langages comme : l'assembleur, le C,le C++, et d'autres...
- Langages de *haut niveau* :

Il existe des centaines de langages de programmation

- Langages de *bas niveau* : c'est-à-dire assez proche du système binaire de la machine. On y trouve des langages comme : l'assembleur, le C,le C++, et d'autres...
- Langages de *haut niveau* : plus éloigné du système, mais plus souple, plus facile à appréhender.

Il existe des centaines de langages de programmation

- Langages de *bas niveau* : c'est-à-dire assez proche du système binaire de la machine. On y trouve des langages comme : l'assembleur, le C,le C++, et d'autres...
- Langages de *haut niveau* : plus éloigné du système, mais plus souple, plus facile à appréhender. On y trouve notamment **Python** !

Il existe des centaines de langages de programmation

- Langages de *bas niveau* : c'est-à-dire assez proche du système binaire de la machine. On y trouve des langages comme : l'assembleur, le C,le C++, et d'autres...
- Langages de *haut niveau* : plus éloigné du système, mais plus souple, plus facile à appréhender. On y trouve notamment **Python** !
- Il y a les langages interprétés et les langages compilés.

Il existe des centaines de langages de programmation

- Langages de *bas niveau* : c'est-à-dire assez proche du système binaire de la machine. On y trouve des langages comme : l'assembleur, le C,le C++, et d'autres...
- Langages de *haut niveau* : plus éloigné du système, mais plus souple, plus facile à appréhender. On y trouve notamment **Python** !
- Il y a les langages interprétés et les langages compilés.
  - Dans le 1^e cas, la suite des instructions est exécutée directement (**Python**)

Il existe des centaines de langages de programmation

- Langages de *bas niveau* : c'est-à-dire assez proche du système binaire de la machine. On y trouve des langages comme : l'assembleur, le C,le C++, et d'autres...
- Langages de *haut niveau* : plus éloigné du système, mais plus souple, plus facile à appréhender. On y trouve notamment **Python** !
- Il y a les langages interprétés et les langages compilés.
  - Dans le 1^e cas, la suite des instructions est exécutée directement (**Python**) : plus lent, mais détection d'erreurs plus simple.

Il existe des centaines de langages de programmation

- Langages de *bas niveau* : c'est-à-dire assez proche du système binaire de la machine. On y trouve des langages comme : l'assembleur, le C,le C++, et d'autres...
- Langages de *haut niveau* : plus éloigné du système, mais plus souple, plus facile à appréhender. On y trouve notamment **Python** !
- Il y a les langages interprétés et les langages compilés.
  - Dans le 1^e cas, la suite des instructions est exécutée directement (**Python**) : plus lent, mais détection d'erreurs plus simple.
  - Dans le 2^e cas, la suite des instructions est d'abord traduite en "langage machine" : plus rapide à exécuter, mais plus exigeant à la conception.

Meilleurs langages en 2021 selon l'IEEE

(l'IEEE est la plus grande association mondiale de professionnels techniques) :

Source :

https ://www.developpez.com/

Rank	Language	Type	Score
1	Python		100.0
2	Java		95.4
3	C		94.7
4	C++		92.4
5	JavaScript		88.1
6	C#		82.4
7	R		81.7
8	Go		77.7
9	HTML		75.4
10	Swift		70.4

Top 10 des langages les plus demandés par les employeurs en 2018 selon l'IEEE :

Language Ranking: Jobs

Rank	Language	Type	Score
1	Python		100.0
2	C		96.0
3	Java		95.9
4	JavaScript		89.6
5	C++		88.3
6	Go		87.3
7	R		85.7
8	HTML		81.3
9	C#		79.8
10	SQL		71.9

Top 10 des meilleurs langages pour le développement de sites et applications web selon l'IEEE :

Rank	Language	Type	Score
1	Python		100.0
2	Java		95.4
3	JavaScript		88.1
4	C#		82.4
5	Go		77.7
6	HTML		75.4
7	PHP		68.0
8	Dart		67.7
9	Ruby		63.6
10	Rust		63.1

Top 10 des langages les plus populaires pour le développement d'applications d'entreprise, de bureau et d'applications scientifiques selon l'IEEE :

Rank	Language	Type	Score
1	Python		100.0
2	Java		95.4
3	C		94.7
4	C++		92.4
5	C#		82.4
6	R		81.7
7	Go		77.7
8	Swift		70.4
9	Matlab		68.3
10	SQL		65.0

- *Python* sous une licence libre proche de la licence BSD

- *Python* sous une licence libre proche de la licence BSD
- *Python* est un langage

- *Python* sous une licence libre proche de la licence BSD
- *Python* est un langage
  - de programmation objet

- *Python* sous une licence libre proche de la licence BSD
- *Python* est un langage
  - de programmation objet
  - multiparadigme

- *Python* sous une licence libre proche de la licence BSD
- *Python* est un langage
  - de programmation objet
  - multiparadigme
  - multiplateforme

- *Python* sous une licence libre proche de la licence BSD
- *Python* est un langage
  - de programmation objet
  - multiparadigme
  - multiplateforme
- Il permet

- *Python* sous une licence libre proche de la licence BSD
- *Python* est un langage
  - de programmation objet
  - multiparadigme
  - multiplateforme
- Il permet
  - la programmation impérative structurée

- *Python* sous une licence libre proche de la licence BSD
- *Python* est un langage
  - de programmation objet
  - multiparadigme
  - multiplateforme
- Il permet
  - la programmation impérative structurée
  - la programmation fonctionnelle

- *Python* sous une licence libre proche de la licence BSD
- *Python* est un langage
  - de programmation objet
  - multiparadigme
  - multiplateforme
- Il permet
  - la programmation impérative structurée
  - la programmation fonctionnelle
  - la programmation orientée objet

- *Python* sous une licence libre proche de la licence BSD
- *Python* est un langage
  - de programmation objet
  - multiparadigme
  - multiplateforme
- Il permet
  - la programmation impérative structurée
  - la programmation fonctionnelle
  - la programmation orientée objet
- Il contient

- *Python* sous une licence libre proche de la licence BSD
- *Python* est un langage
  - de programmation objet
  - multiparadigme
  - multiplateforme
- Il permet
  - la programmation impérative structurée
  - la programmation fonctionnelle
  - la programmation orientée objet
- Il contient
  - un typage dynamique fort

- *Python* sous une licence libre proche de la licence BSD
- *Python* est un langage
  - de programmation objet
  - multiparadigme
  - multiplateforme
- Il permet
  - la programmation impérative structurée
  - la programmation fonctionnelle
  - la programmation orientée objet
- Il contient
  - un typage dynamique fort
  - une gestion automatique de la mémoire par ramasse-miettes

- *Python* sous une licence libre proche de la licence BSD
- *Python* est un langage
  - de programmation objet
  - multiparadigme
  - multiplateforme
- Il permet
  - la programmation impérative structurée
  - la programmation fonctionnelle
  - la programmation orientée objet
- Il contient
  - un typage dynamique fort
  - une gestion automatique de la mémoire par ramasse-miettes
  - un système de gestion d'exceptions

- *Python* fonctionne sur la plupart des plateformes informatiques (en autres Linux, MacOs, Windows, ...)

- *Python* fonctionne sur la plupart des plateformes informatiques (en autres Linux, MacOs, Windows, ...)
- *Python* est conçu pour optimiser la productivité des programmeurs

- *Python* fonctionne sur la plupart des plateformes informatiques (en autres Linux, MacOs, Windows, ...)
- *Python* est conçu pour optimiser la productivité des programmeurs
- *Python* offre des outils de haut niveau et des dizaines de milliers de bibliothèques dédiées

- *Python* fonctionne sur la plupart des plateformes informatiques (en autres Linux, MacOs, Windows, ...)
- *Python* est conçu pour optimiser la productivité des programmeurs
- *Python* offre des outils de haut niveau et des dizaines de milliers de bibliothèques dédiées
- *Python* possède une syntaxe simple à utiliser,

- *Python* fonctionne sur la plupart des plateformes informatiques (en autres Linux, MacOs, Windows, ...)
- *Python* est conçu pour optimiser la productivité des programmeurs
- *Python* offre des outils de haut niveau et des dizaines de milliers de bibliothèques dédiées
- *Python* possède une syntaxe simple à utiliser, possède plusieurs IDE de qualité pour le développement

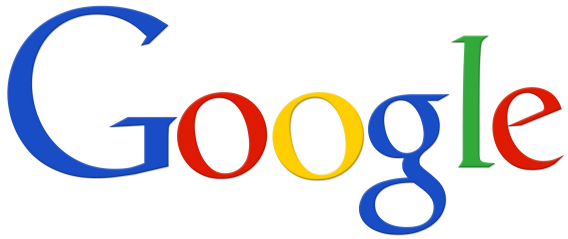
- *Python* fonctionne sur la plupart des plateformes informatiques (en autres Linux, MacOs, Windows, ...)
- *Python* est conçu pour optimiser la productivité des programmeurs
- *Python* offre des outils de haut niveau et des dizaines de milliers de bibliothèques dédiées
- *Python* possède une syntaxe simple à utiliser, possède plusieurs IDE de qualité pour le développement
- *Python* est très utilisé, notamment pour le prototypage, dans le domaine scientifique (notamment chez les data scientists)

Python est utilisé par de grandes sociétés :

Python est utilisé par de grandes sociétés :

Dysney Animation Studio

Python est utilisé par de grandes sociétés :



Dysney Animation Studio
Google, Youtube

Python est utilisé par de grandes sociétés :

Dysney Animation Studio
Google, Youtube
Mozilla

Python est utilisé par de grandes sociétés :

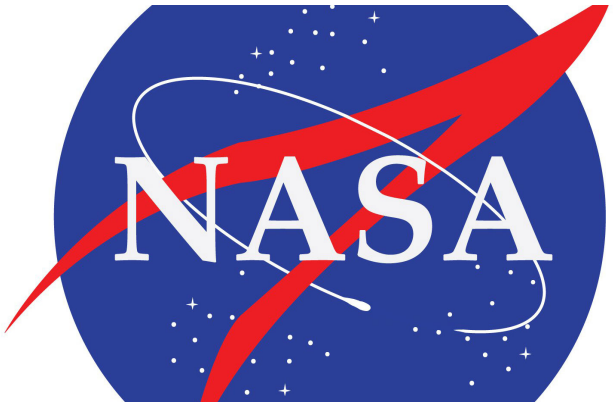
CANONICAL

Dysney Animation Studio
Google, Youtube
Mozilla
Canonical

Python est utilisé par de grandes sociétés :

Dysney Animation Studio
Google, Youtube
Mozilla
Canonical
Microsoft

Python est utilisé par de grandes sociétés :



Dysney Animation Studio
Google, Youtube
Mozilla
Canonical
Microsoft
Nasa

Python est utilisé par de grandes sociétés :

Dysney Animation Studio
Google, Youtube
Mozilla
Canonical
Microsoft
Nasa
Spotify

Python est utilisé par de grandes sociétés :

Dysney Animation Studio
Google, Youtube
Mozilla
Canonical
Microsoft
Nasa
Spotify
Box ADSL, réseaux sociaux

Python est utilisé par de grandes sociétés :

Dysney Animation Studio
Google, Youtube
Mozilla
Canonical
Microsoft
Nasa
Spotify
Box ADSL, réseaux sociaux
Cisco

- Deux branches de développement de *Python* coexistent

- Deux branches de développement de *Python* coexistent
  - La branche 2.x qui est l'ancienne "version" de *Python*,

- Deux branches de développement de *Python* coexistent
  - La branche 2.x qui est l'ancienne "version" de *Python*, la dernière version du 4 mars 2019 est la 2.7.16

- Deux branches de développement de *Python* coexistent
  - La branche 2.x qui est l'ancienne "version" de *Python*, la dernière version du 4 mars 2019 est la 2.7.16
    Elle subsiste pour des raisons de conformité avec d'anciens programmes.

- Deux branches de développement de *Python* coexistent
  - La branche 2.x qui est l'ancienne "version" de *Python*, la dernière version du 4 mars 2019 est la 2.7.16
    Elle subsiste pour des raisons de conformité avec d'anciens programmes.Cette branche est désormais abandonnée.
  - La branche 3.x dont la version au 1 septembre 2023 est la 3.11.5

Miniconda

- Deux branches de développement de *Python* coexistent
  - La branche 2.x qui est l'ancienne "version" de *Python*, la dernière version du 4 mars 2019 est la 2.7.16
    Elle subsiste pour des raisons de conformité avec d'anciens programmes.Cette branche est désormais abandonnée.
  - La branche 3.x dont la version au 1 septembre 2023 est la 3.11.5
- Nous utiliserons la version 3.x !

Miniconda

- Deux branches de développement de *Python* coexistent
  - La branche 2.x qui est l'ancienne "version" de *Python*, la dernière version du 4 mars 2019 est la 2.7.16
    Elle subsiste pour des raisons de conformité avec d'anciens programmes.Cette branche est désormais abandonnée.
  - La branche 3.x dont la version au 1 septembre 2023 est la 3.11.5
- Nous utiliserons la version 3.x !
- Pour l'installation le mieux est d'installer *Miniconda*

Miniconda

- Deux branches de développement de *Python* coexistent
  - La branche 2.x qui est l'ancienne "version" de *Python*, la dernière version du 4 mars 2019 est la 2.7.16
    Elle subsiste pour des raisons de conformité avec d'anciens programmes.Cette branche est désormais abandonnée.
  - La branche 3.x dont la version au 1 septembre 2023 est la 3.11.5
- Nous utiliserons la version 3.x !
- Pour l'installation le mieux est d'installer *Miniconda* ou *Anaconda*

Miniconda

- Deux branches de développement de *Python* coexistent
  - La branche 2.x qui est l'ancienne "version" de *Python*, la dernière version du 4 mars 2019 est la 2.7.16
    Elle subsiste pour des raisons de conformité avec d'anciens programmes.Cette branche est désormais abandonnée.
  - La branche 3.x dont la version au 1 septembre 2023 est la 3.11.5
- Nous utiliserons la version 3.x !
- Pour l'installation le mieux est d'installer *Miniconda* ou *Anaconda* disponibles à cette adresse :
  https://www.anaconda.com/

L'installation d'*Anaconda* est à ces adresses selon votre système :

L'installation d'*Anaconda* est à ces adresses selon votre système :

- https ://docs.anaconda.com/anaconda/install/windows/
- https ://docs.anaconda.com/anaconda/install/mac-os/
- https ://docs.anaconda.com/anaconda/install/linux/

L'installation d'*Anaconda* est à ces adresses selon votre système :

- https ://docs.anaconda.com/anaconda/install/windows/
- https ://docs.anaconda.com/anaconda/install/mac-os/
- https ://docs.anaconda.com/anaconda/install/linux/

L'installation de *Miniconda* est à cette adresse :

- https ://conda.io/en/latest/miniconda.html

- L'installation d'*Anaconda* procède automatiquement à l'installation de *Python* et de plusieurs IDE [Environnement de développement (Integrated Development Environment)] : IDLE, Spyder, Ipython, Jupyter notebook

- L'installation d'*Anaconda* procède automatiquement à l'installation de *Python* et de plusieurs IDE [Environnement de développement (Integrated Development Environment)] : IDLE, Spyder, Ipython, Jupyter notebook
- Il est bien sûr possible d'utiliser d'autres IDE en les paramétrant pour accéder à *Anaconda* : Eclipse, Sublime Text, PyCharm, Pyzo, etc.

- L'installation d'*Anaconda* procède automatiquement à l'installation de *Python* et de plusieurs IDE [Environnement de développement (Integrated Development Environment)] : IDLE, Spyder, Ipython, Jupyter notebook
- Il est bien sûr possible d'utiliser d'autres IDE en les paramétrant pour accéder à *Anaconda* : Eclipse, Sublime Text, PyCharm, Pyzo, etc.
- Attention, l'installation de *Miniconda* n'entraîne pas tout cela (installation de *Python* et de bibliothèques de base). À vous d'installer ensuite ce que vous voulez...

L'installation d' *Anaconda* / *Miniconda* terminée, vous pouvez accéder à *Python* de différentes façons.

L'installation d' *Anaconda* / *Miniconda* terminée, vous pouvez accéder à *Python* de différentes façons.

Windows : utilisez le menu "démarrer" pour accéder aux différentes options : vous serez amené à utiliser "Anaconda Prompt" et surtout "Jupyter Notebook".

L'installation d' *Anaconda* / *Miniconda* terminée, vous pouvez accéder à *Python* de différentes façons.

Windows : utilisez le menu "démarrer" pour accéder aux différentes options : vous serez amené à utiliser "Anaconda Prompt" et surtout "Jupyter Notebook".

Linux ou MacOS : Ouvrez un terminal et entrez le nom de l'application qui vous intéresse :

L'installation d' *Anaconda* / *Miniconda* terminée, vous pouvez accéder à *Python* de différentes façons.

Windows : utilisez le menu "démarrer" pour accéder aux différentes options : vous serez amené à utiliser "Anaconda Prompt" et surtout "Jupyter Notebook".

Linux ou MacOS : Ouvrez un terminal et entrez le nom de l'application qui vous intéresse :"ipython", "jupyter notebook", ...

L'installation d' *Anaconda* / *Miniconda* terminée, vous pouvez accéder à *Python* de différentes façons.

Windows : utilisez le menu "démarrer" pour accéder aux différentes options : vous serez amené à utiliser "Anaconda Prompt" et surtout "Jupyter Notebook".

Linux ou MacOS : Ouvrez un terminal et entrez le nom de l'application qui vous intéresse :"ipython", "jupyter notebook", ...

Notre IDE de développement sera "Jupyter Notebook" pour la suite de l'année.

Utilisation d'*Anaconda* / *Miniconda* :

Utilisation d'*Anaconda* / *Miniconda* :

- Environ 250 packages scientifiques sont installés avec *Anaconda*, mais vous pouvez installer un nouveau package avec, dans un terminal (MacOS ou Linux) ou une fenêtre Anaconda Prompt (Windows)

Utilisation d'*Anaconda* / *Miniconda* :

- Environ 250 packages scientifiques sont installés avec *Anaconda*, mais vous pouvez installer un nouveau package avec, dans un terminal (MacOS ou Linux) ou une fenêtre Anaconda Prompt (Windows)

à taper dans un terminal

```
conda install nom-du-package
```

Utilisation d'*Anaconda* / *Miniconda* :

- Environ 250 packages scientifiques sont installés avec *Anaconda*, mais vous pouvez installer un nouveau package avec, dans un terminal (MacOS ou Linux) ou une fenêtre Anaconda Prompt (Windows)

à taper dans un terminal

```
conda install nom-du-package
```

- Par exemple, vous voulez, avec *Miniconda* installer le package "numpy" (pour le calcul matriciel entre autre), vous tapez

Utilisation d'*Anaconda* / *Miniconda* :

- Environ 250 packages scientifiques sont installés avec *Anaconda*, mais vous pouvez installer un nouveau package avec, dans un terminal (MacOS ou Linux) ou une fenêtre Anaconda Prompt (Windows)

à taper dans un terminal

```
conda install nom-du-package
```

- Par exemple, vous voulez, avec *Miniconda* installer le package "numpy" (pour le calcul matriciel entre autre), vous tapez

à taper dans un terminal

```
conda install numpy
```

Pour voir la liste de tous les packages installés...

Pour voir la liste de tous les packages installés...

à taper dans un terminal

```
conda list
```

Pour voir la liste de tous les packages installés...

à taper dans un terminal

```
conda list
```

Pour "upgrader" un package quelconque, numpy par ex :...

Pour voir la liste de tous les packages installés...

à taper dans un terminal

```
conda list
```

Pour "upgrader" un package quelconque, numpy par ex :...

à taper dans un terminal

```
conda update numpy
```

Pour voir la liste de tous les packages installés...

à taper dans un terminal

```
conda list
```

Pour "upgrader" un package quelconque, numpy par ex :...

à taper dans un terminal

```
conda update numpy
```

Pour chercher un package dont on connaît le nom (ou en partie)...

Pour voir la liste de tous les packages installés...

à taper dans un terminal

```
conda list
```

Pour "upgrader" un package quelconque, numpy par ex :...

à taper dans un terminal

```
conda update numpy
```

Pour chercher un package dont on connaît le nom (ou en partie)...

à taper dans un terminal

```
conda search scipy
```

Pour supprimer un package précis... (scipy ici)

Pour supprimer un package précis... (scipy ici)

à taper dans un terminal

```
conda remove scipy
```

Pour supprimer un package précis... (scipy ici)

à taper dans un terminal

```
conda remove scipy
```

Pour upgrader *Anaconda* / *Miniconda*...

Pour supprimer un package précis... (scipy ici)

à taper dans un terminal

```
conda remove scipy
```

Pour upgrader *Anaconda* / *Miniconda*...

à taper dans un terminal

```
conda update conda
```

Pour supprimer un package précis... (scipy ici)

à taper dans un terminal

```
conda remove scipy
```

Pour upgrader *Anaconda* / *Miniconda*...

à taper dans un terminal

```
conda update conda
```

Pour updater et upgrader Python...

Pour supprimer un package précis... (scipy ici)

à taper dans un terminal

```
conda remove scipy
```

Pour upgrader *Anaconda* / *Miniconda*...

à taper dans un terminal

```
conda update conda
```

Pour updater et upgrader Python...

à taper dans un terminal

```
conda update python
```

Pour tout mettre à jour !

Pour tout mettre à jour !

à taper dans un terminal

```
conda update --all
```

Pour tout mettre à jour !

à taper dans un terminal

```
conda update --all
```

L'installation d'un package ou sa mise à niveau entraîne souvent l'installation ou la mise à niveau d'autres packages.

Linux ou MacOS : ouvrez un terminal et entrez

terminal

```
jupyter-notebook
```

Linux ou MacOS : ouvrez un terminal et entrez

terminal

```
jupyter-notebook
```

ou mieux

Linux ou MacOS : ouvrez un terminal et entrez

terminal

```
jupyter-notebook
```

ou mieux

terminal

```
jupyter-lab
```

Linux ou MacOS : ouvrez un terminal et entrez

terminal

```
jupyter-notebook
```

ou mieux

terminal

```
jupyter-lab
```

et validez !

Linux ou MacOS : ouvrez un terminal et entrez

terminal

```
jupyter-notebook
```

ou mieux

terminal

```
jupyter-lab
```

et validez !

Windows : dans le menu "démarrer", cliquez sur "Jupyter Notebook" ou "Jupyter Lab"

Linux ou MacOS : ouvrez un terminal et entrez

```
terminal
jupyter-notebook
```

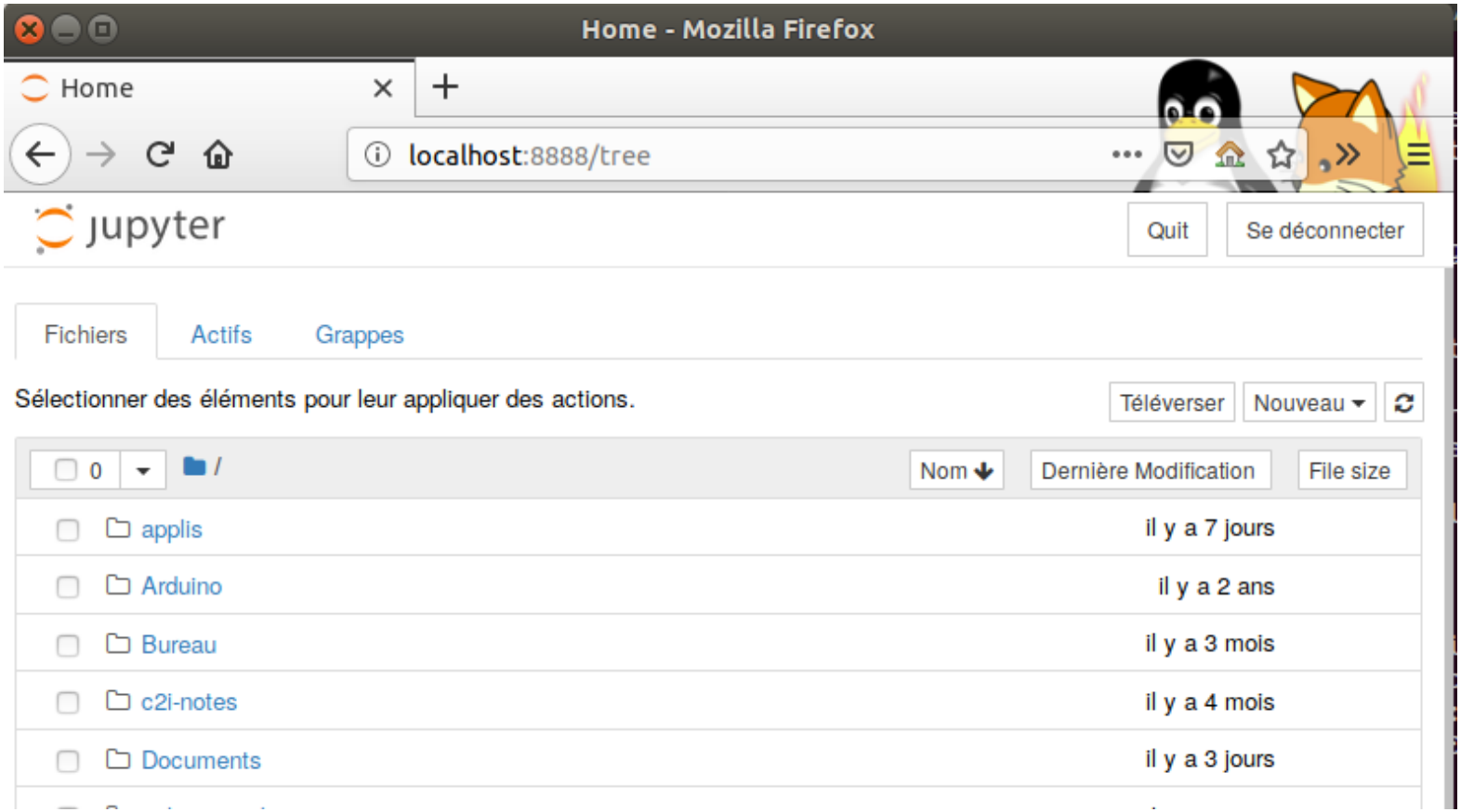
ou mieux

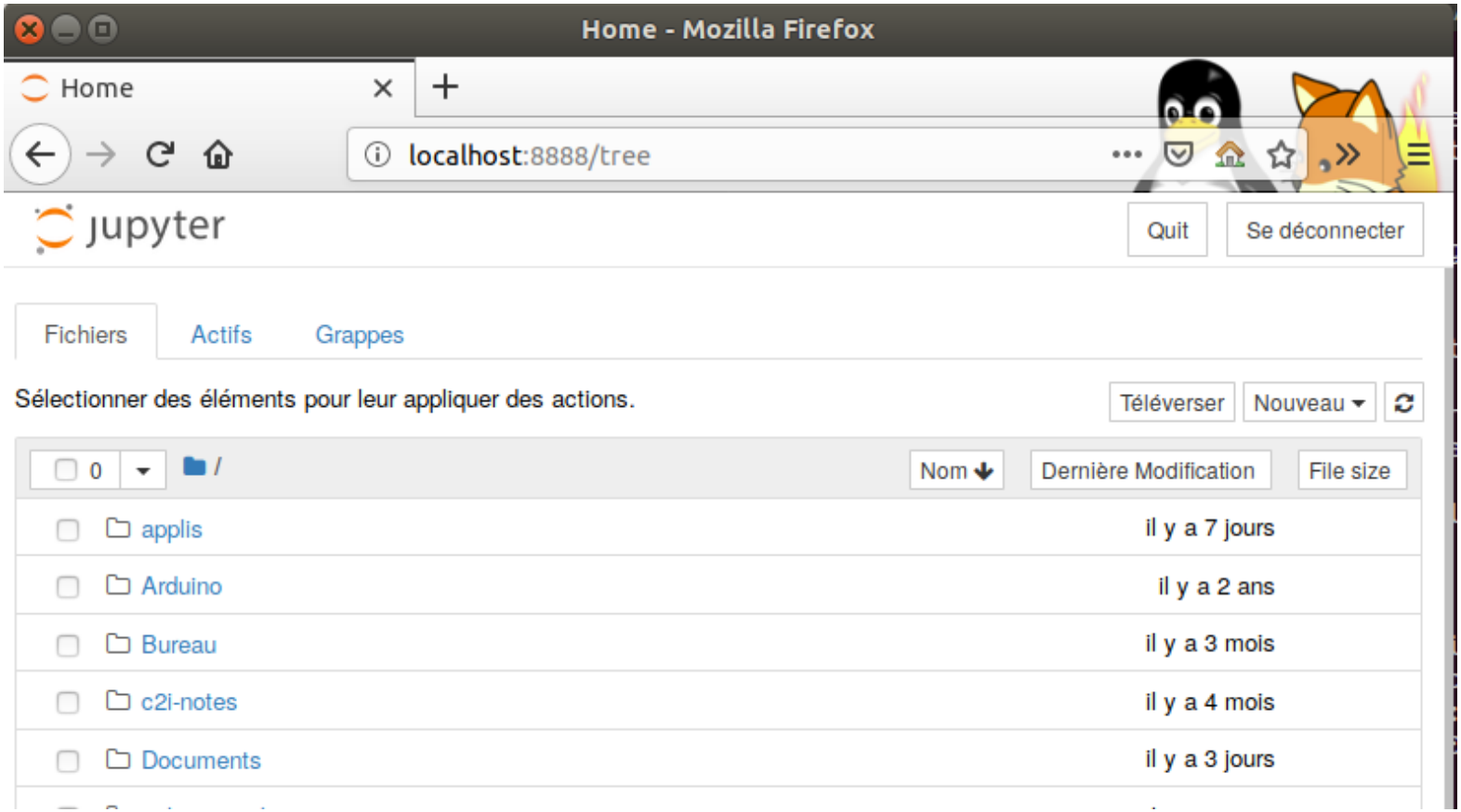
```
terminal
jupyter-lab
```

et validez !

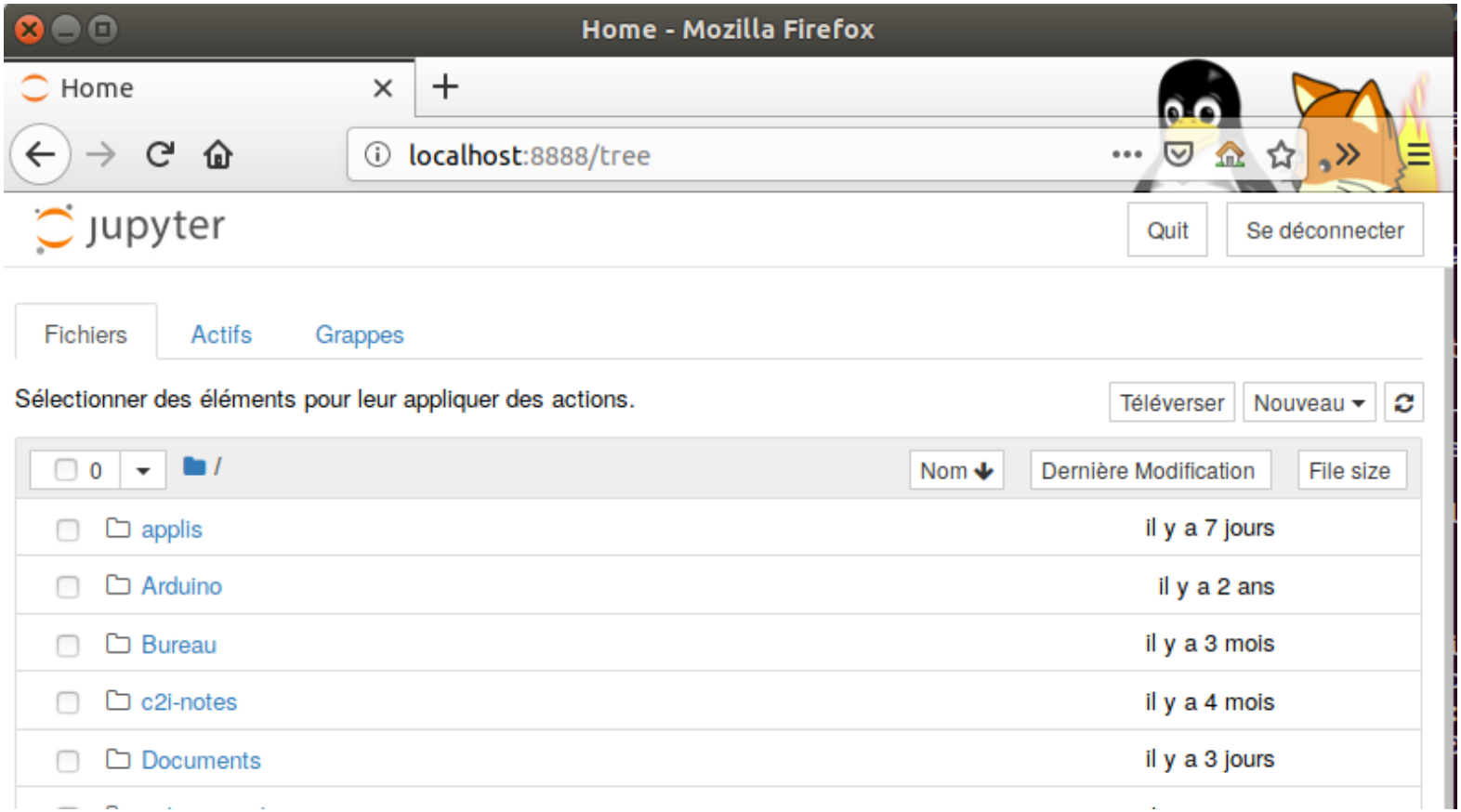
Windows : dans le menu "démarrer", cliquez sur "Jupyter Notebook" ou "Jupyter Lab"

Votre navigateur (utilisez *Firefox* de préférence) va s'ouvrir, il vous servira d'éditeur...

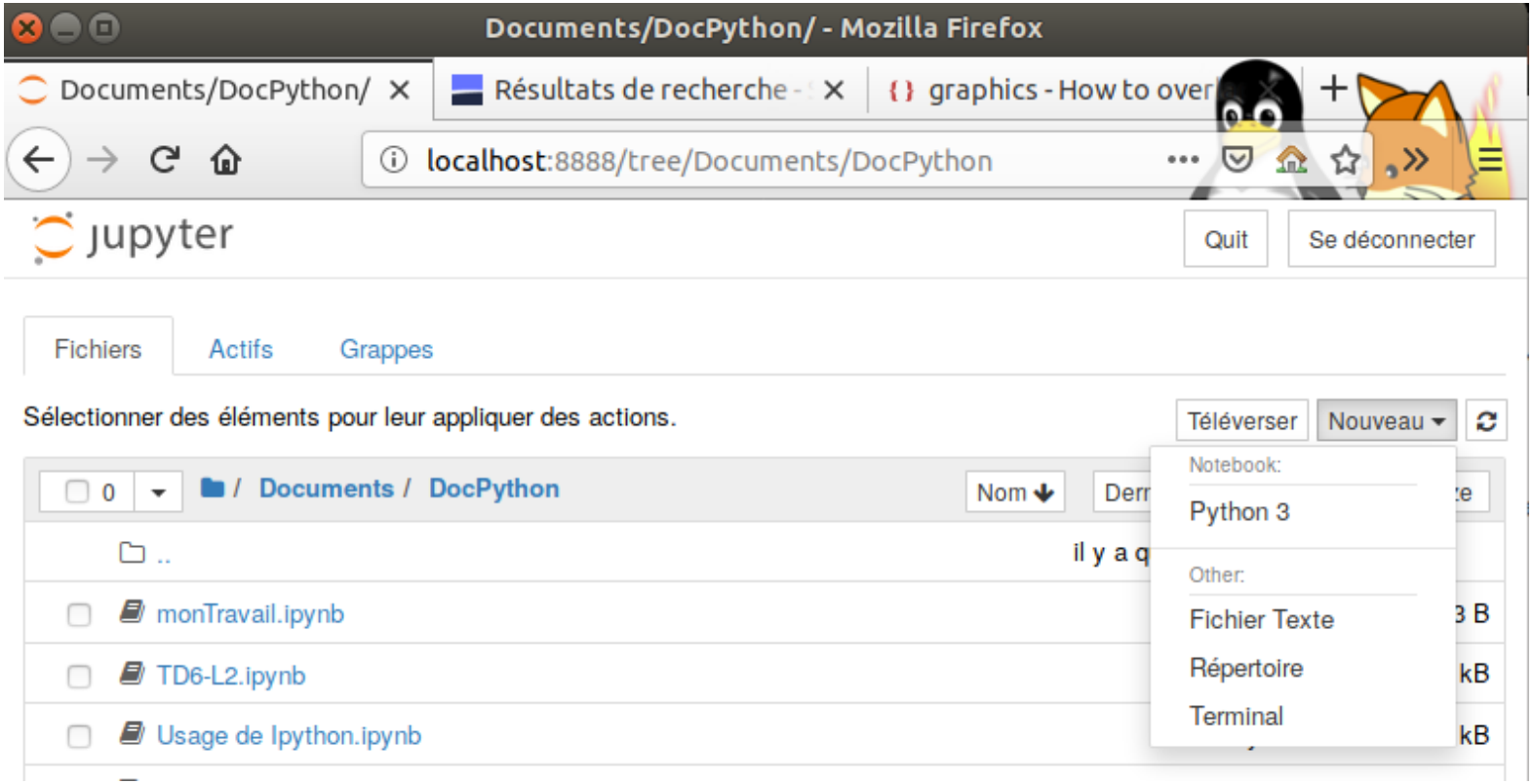

Home - Mozilla Firefox
Home
localhost:8888/tree
jupyter
Quit
Se déconnecter
Fichiers
Actifs
Grappes
Sélectionner des éléments pour leur appliquer des actions.
Téléverser
Nouveau
0
/
Nom
Dernière Modification
File size
applis
il y a 7 jours
Arduino
il y a 2 ans
Bureau
il y a 3 mois
c2i-notes
il y a 4 mois
Documents
il y a 3 jours

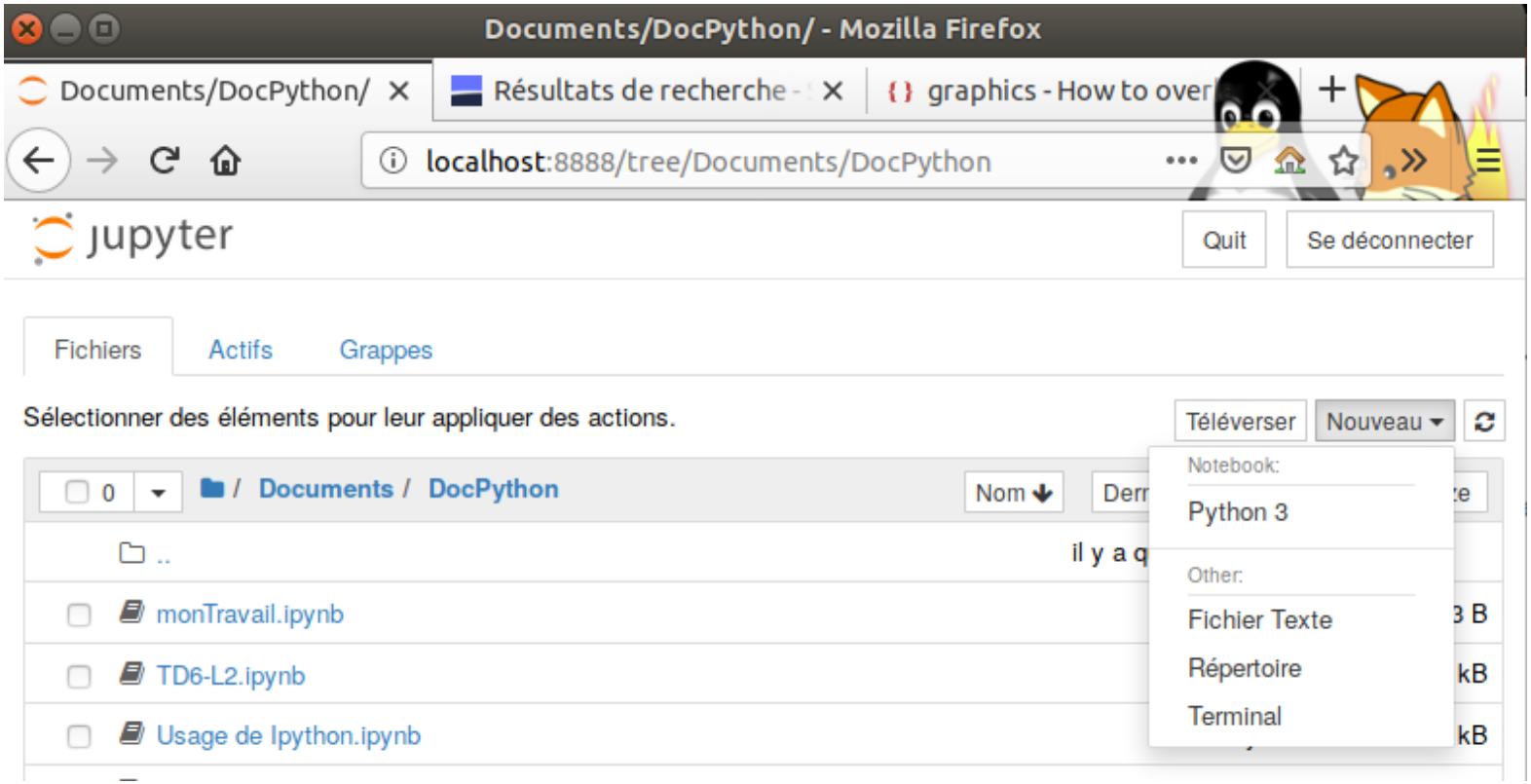


Le navigateur par défaut s'ouvre et visualise le dossier personnel de l'utilisateur.

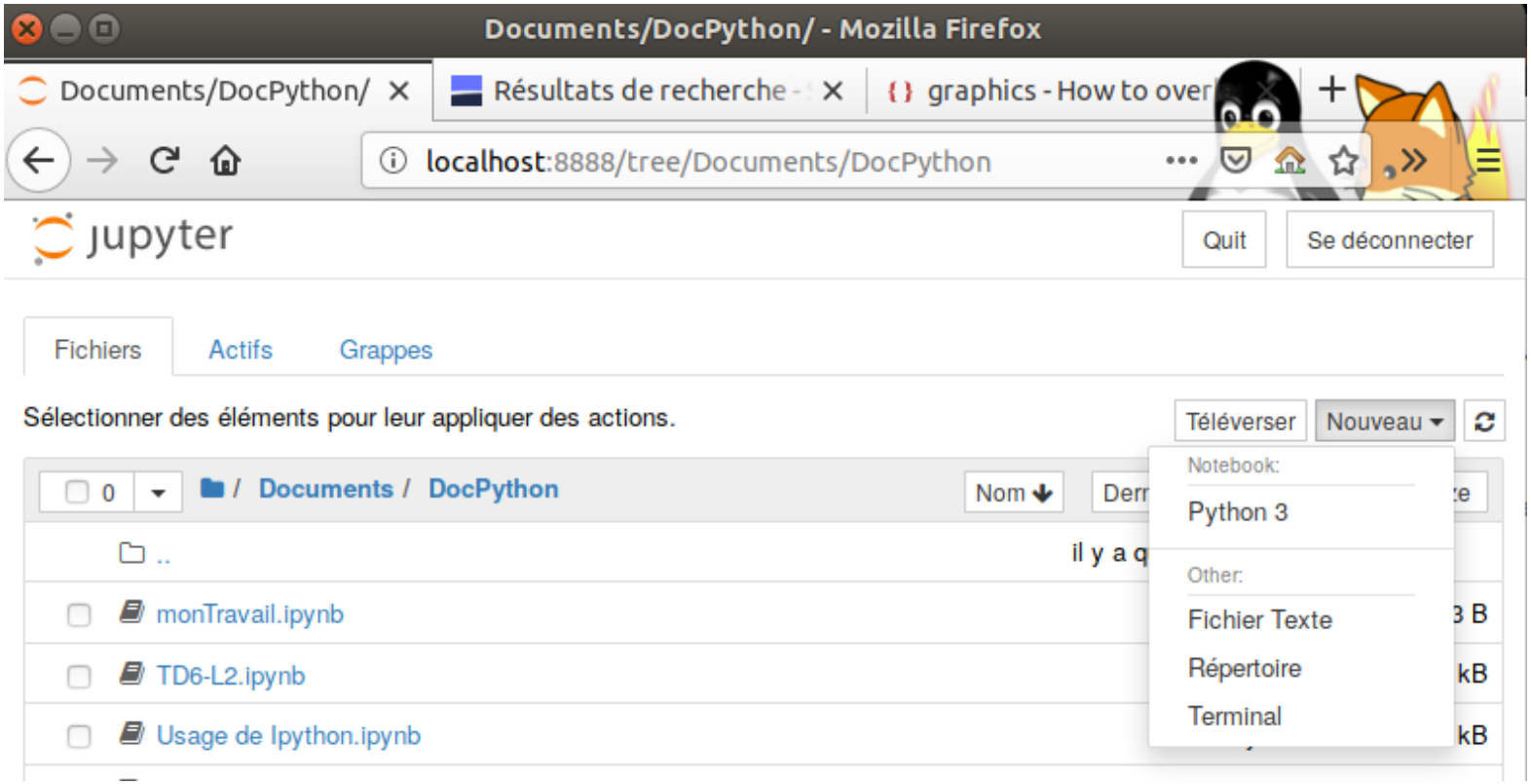


Le navigateur par défaut s'ouvre et visualise le dossier personnel de l'utilisateur. On veut placer (par exemple) son 1^e travail dans le dossier "Documents/DocPython"...


Documents/DocPython/ - Mozilla Firefox
Documents/DocPython/
Résultats de recherche -
graphics - How to over
localhost:8888/tree/Documents/DocPython
jupyter
Quit
Se déconnecter
Fichiers
Actifs
Grappes
Sélectionner des éléments pour leur appliquer des actions.
Téléverser
Nouveau
Notebook:
Python 3
Other:
Fichier Texte
Répertoire
Terminal
/ Documents / DocPython
Nom
..
monTravail.ipynb
TD6-L2.ipynb
Usage de Ipython.ipynb



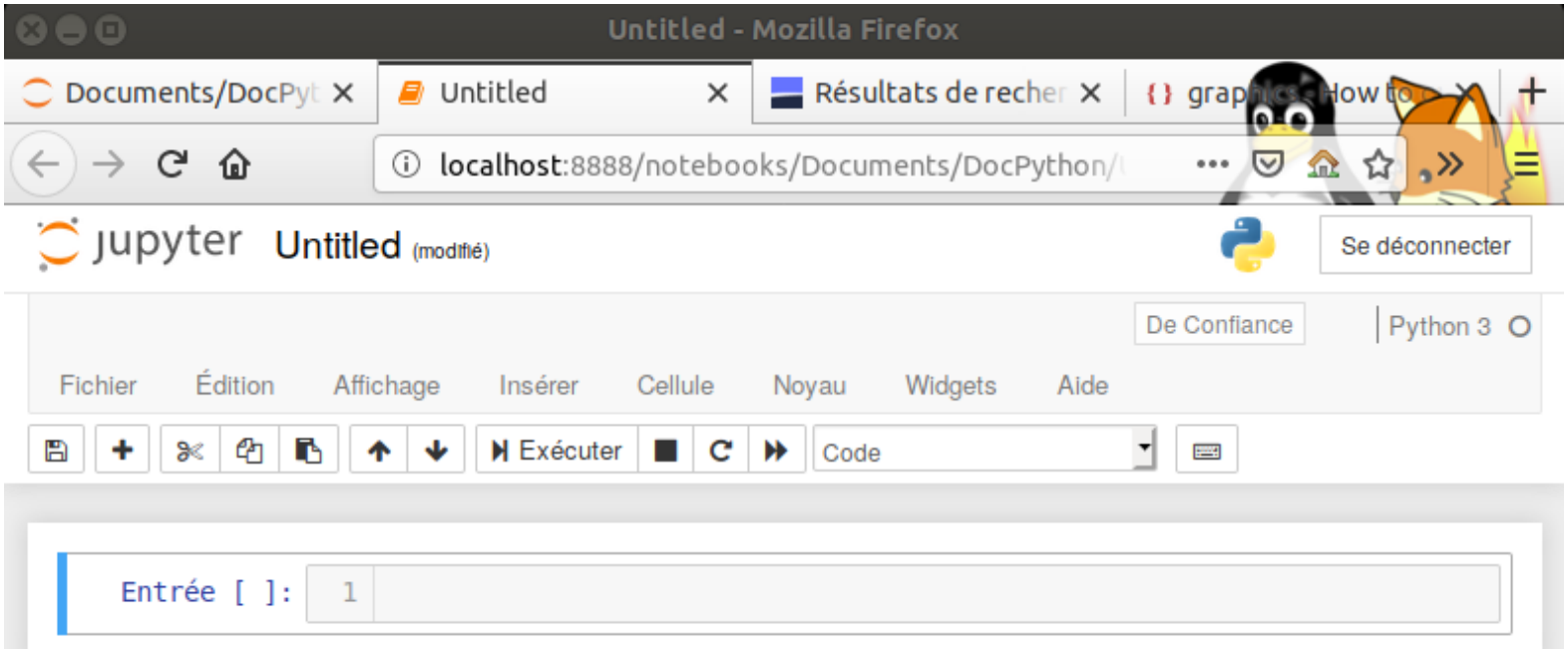
Ici, le travail va être placé dans "Documents > DocPython".



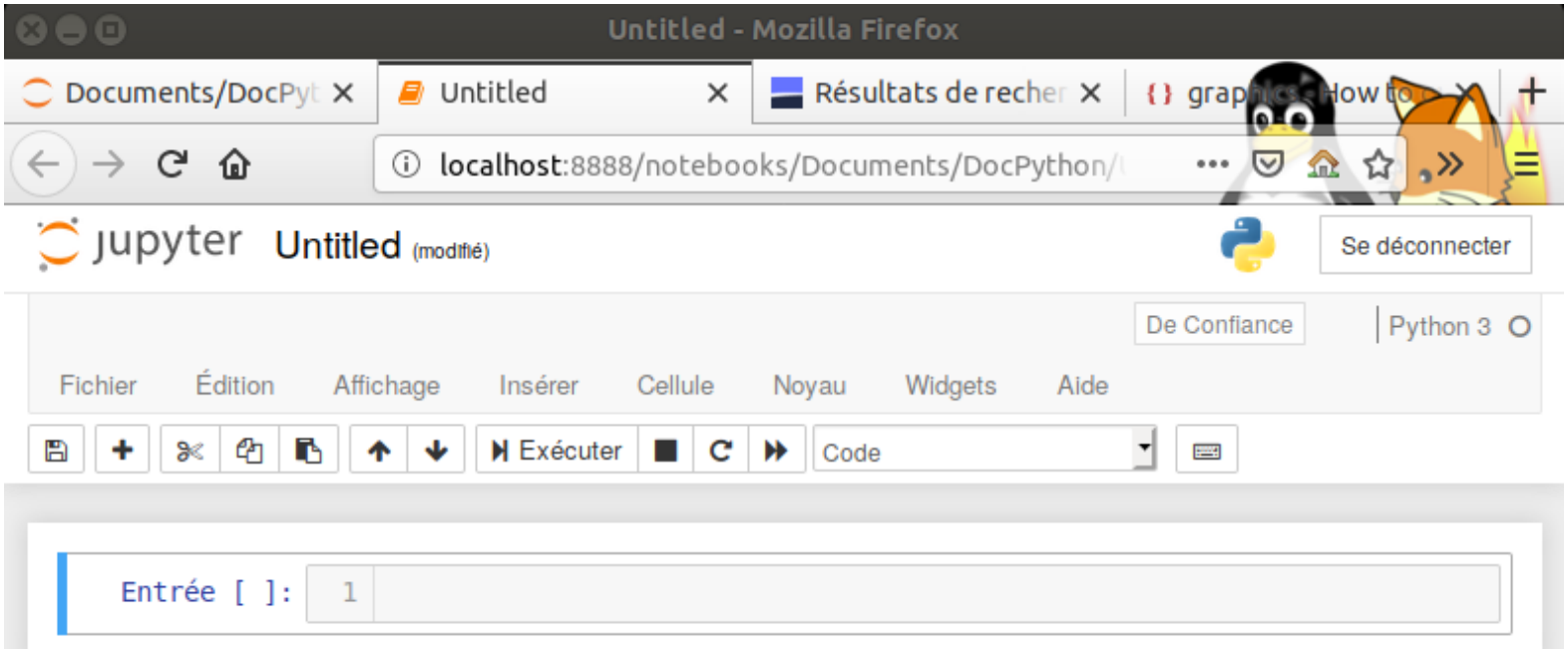
Ici, le travail va être placé dans "Documents > DocPython". On clique sur "Nouveau" et on choisit "Python 3"

On arrive sur cet écran, qui constitue l'interface d'édition...

On arrive sur cet écran, qui constitue l'interface d'édition...

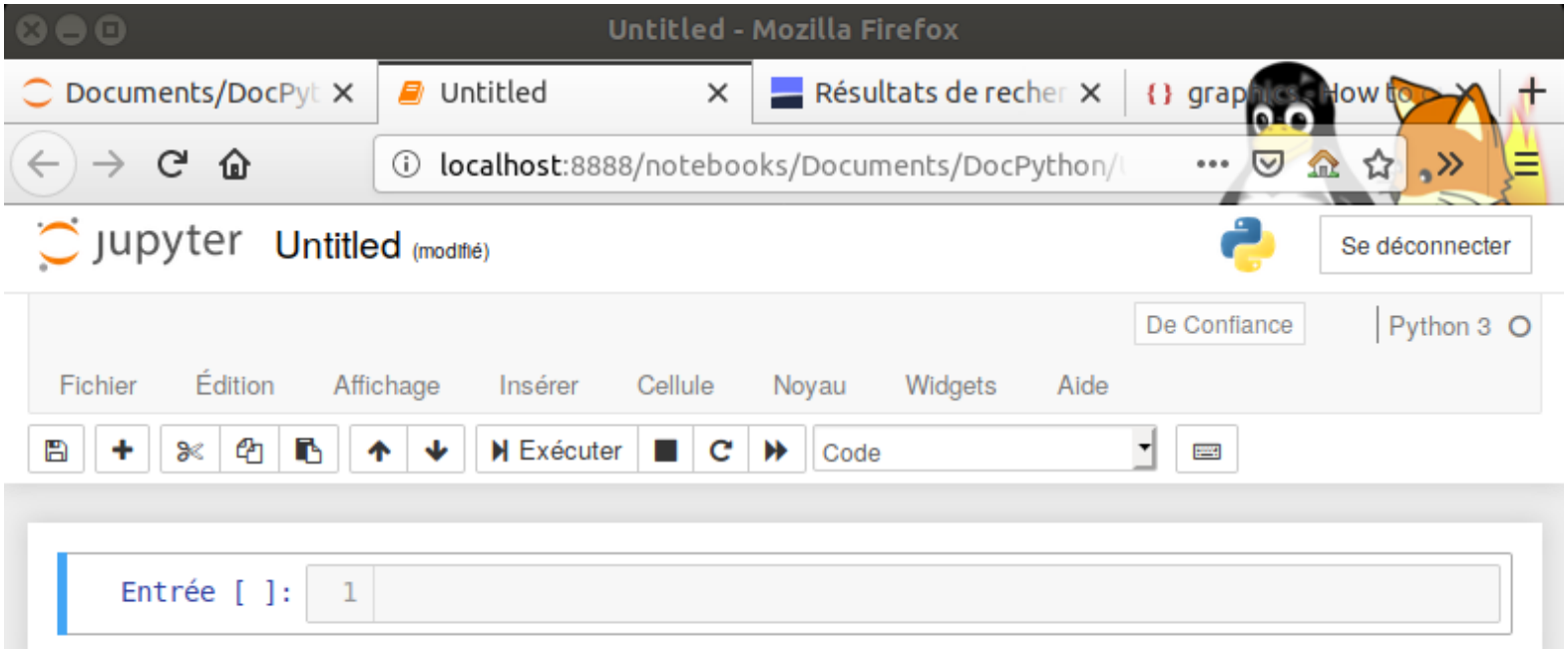


On arrive sur cet écran, qui constitue l'interface d'édition...



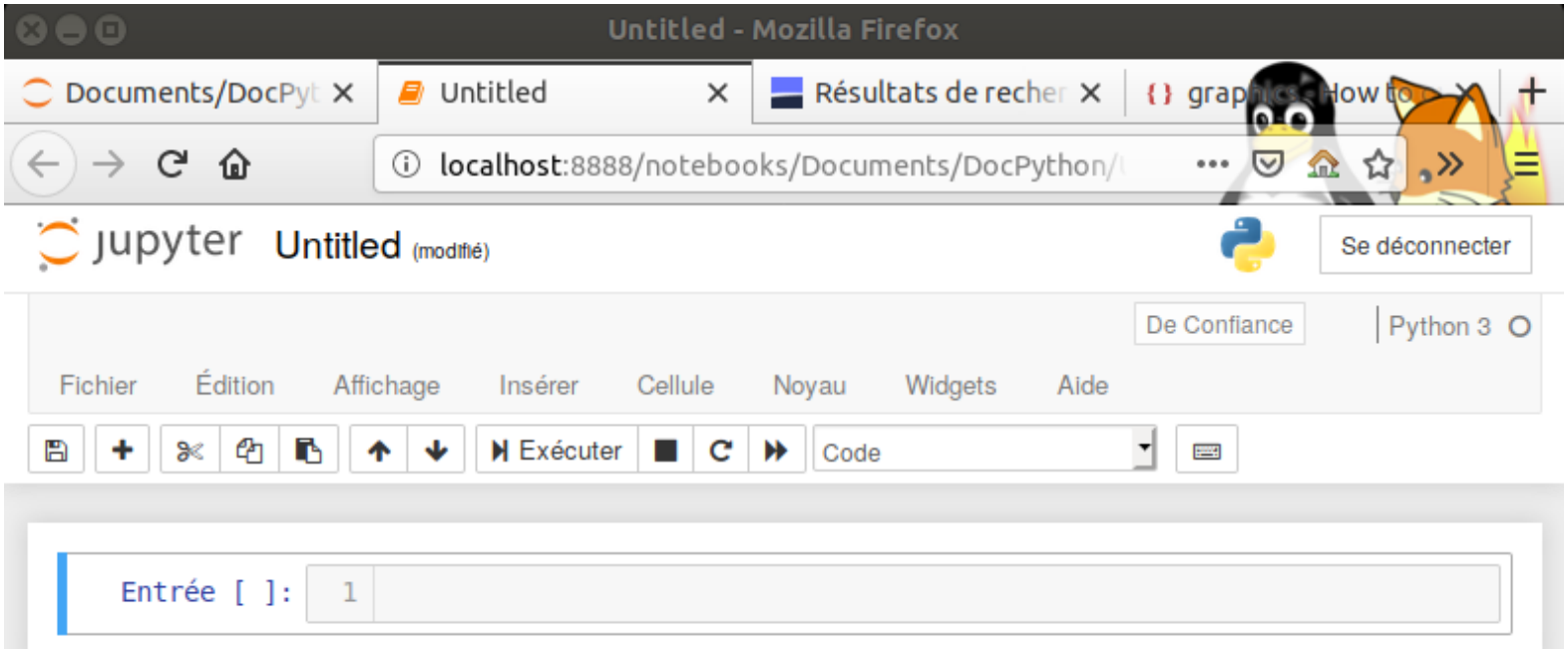
On commence par donner un titre à notre nouveau "fichier programme" en cliquant sur "Untitled",

On arrive sur cet écran, qui constitue l'interface d'édition...

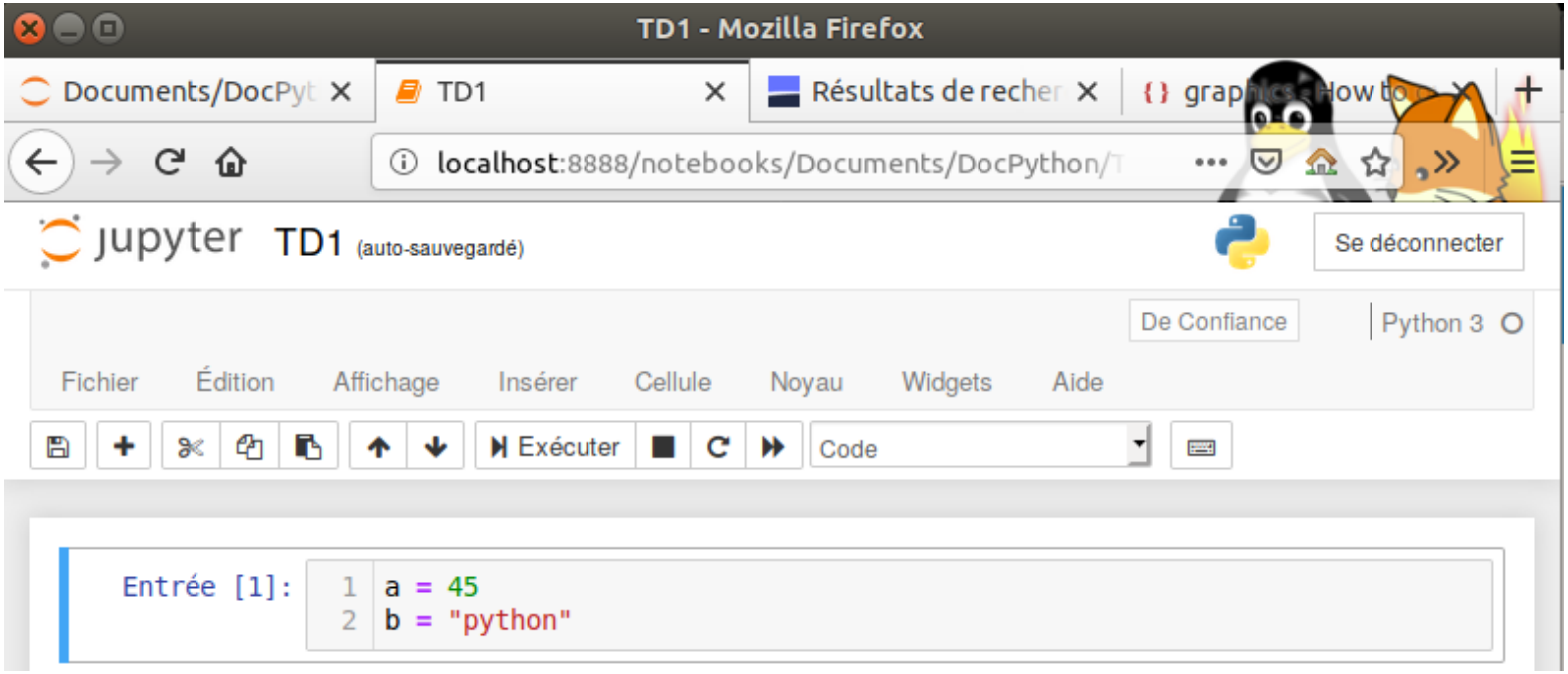


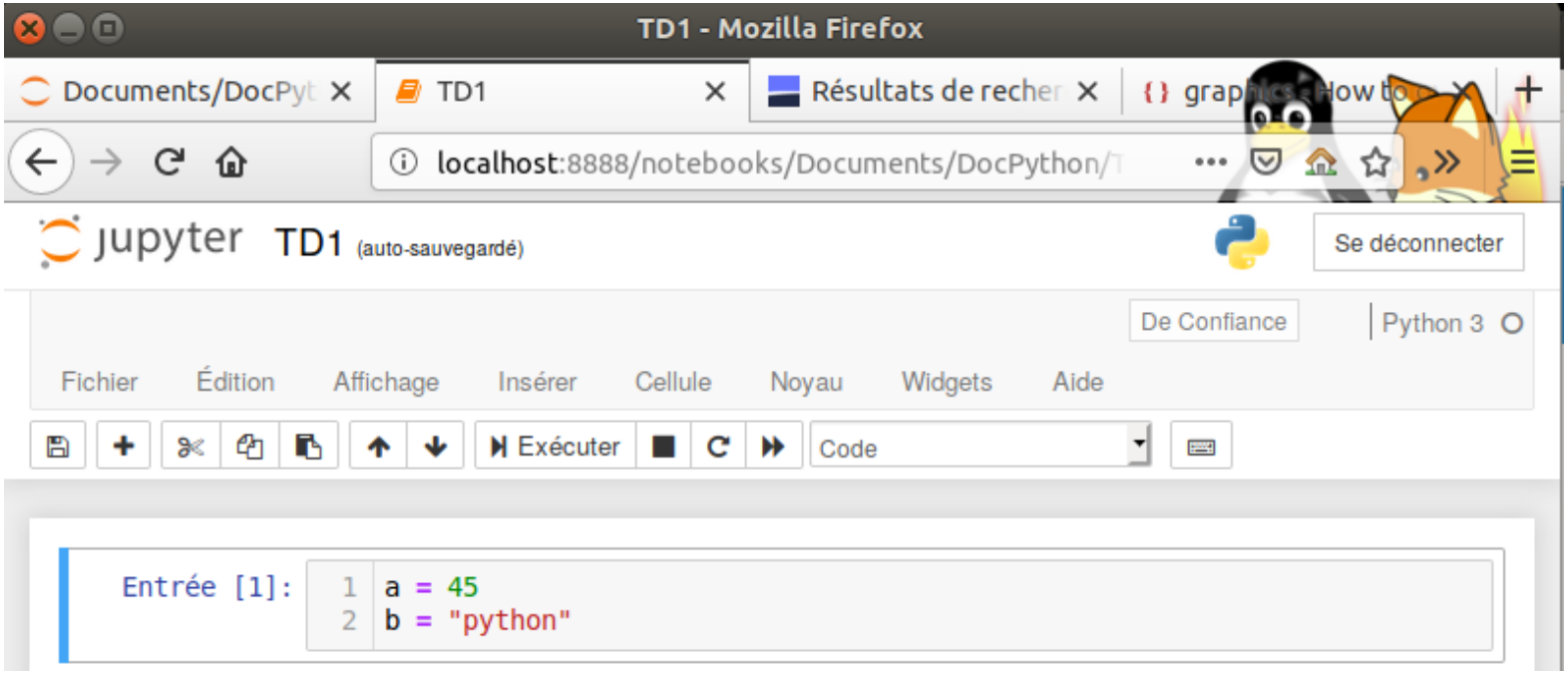
On commence par donner un titre à notre nouveau "fichier programme" en cliquant sur "Untitled", et on entre le nom (ex : **TD1**).

On arrive sur cet écran, qui constitue l'interface d'édition...

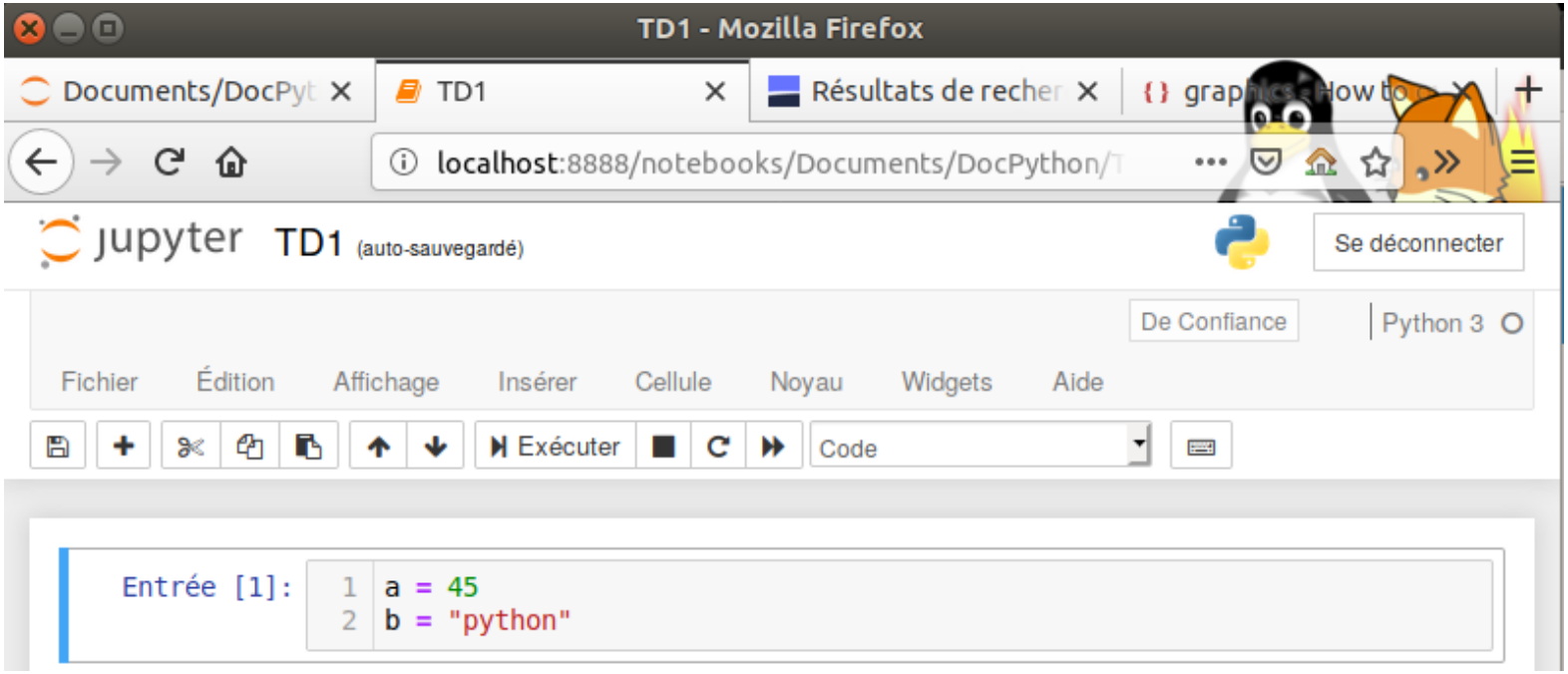


On commence par donner un titre à notre nouveau "fichier programme" en cliquant sur "Untitled", et on entre le nom (ex : **TD1**). Le nom apparaît à droite de "Jupyter".

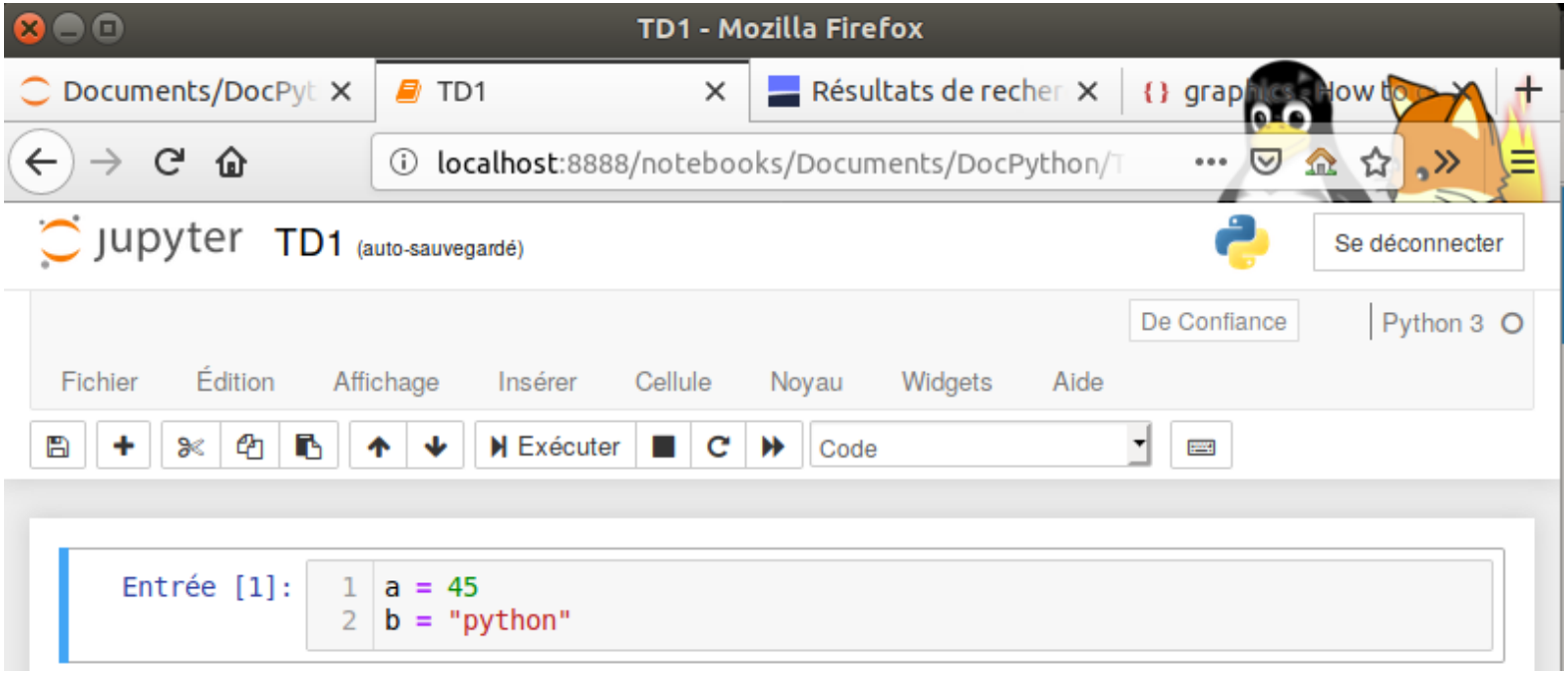

TD1 - Mozilla Firefox
Documents/DocPyt
TD1
Résultats de recher
localhost:8888/notebooks/Documents/DocPython/
jupyter
TD1
(auto-sauvegardé)
Se déconnecter
De Confiance
Python 3
Fichier
Édition
Affichage
Insérer
Cellule
Noyau
Widgets
Aide
Exécuter
Code
Entrée [1]:
1 a = 45
2 b = "python"



- Chaque zone de saisie est appelé une *cellule*.



- Chaque zone de saisie est appelé une *cellule*.
- Dans une cellule, on appuie sur "Entrée" à la fin de chaque ligne, pour passer à la suivante



- Chaque zone de saisie est appelé une *cellule*.
- Dans une cellule, on appuie sur "Entrée" à la fin de chaque ligne, pour passer à la suivante
- Pour exécuter une cellule (= l'ensemble des lignes qu'elle contient), on clique sur 

Chaque cellule possède deux comportements.

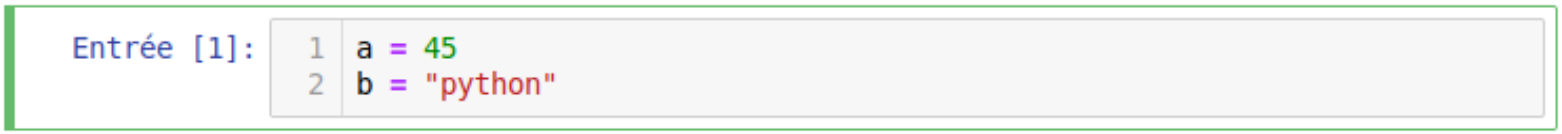
Chaque cellule possède deux comportements.

- Ici, on a cliqué sous le mot "Entrée", un bord bleu apparaît à gauche de la cellule, on est en mode commande.

Chaque cellule possède deux comportements.

- Ici, on a cliqué sous le mot "Entrée", un bord bleu apparaît à gauche de la cellule, on est en mode commande.

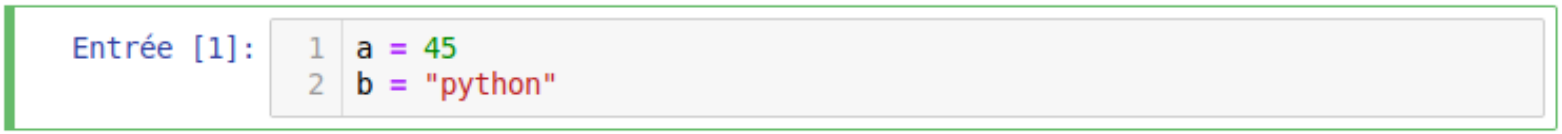
- Ici, on a cliqué sur la zone de saisie (où a = 45 et b = "python" sont écrits), un bord vert apparaît à gauche de la cellule, on est en mode édition.



Chaque cellule possède deux comportements.

- Ici, on a cliqué sous le mot "Entrée", un bord bleu apparaît à gauche de la cellule, on est en mode commande.

- Ici, on a cliqué sur la zone de saisie (où a = 45 et b = "python" sont écrits), un bord vert apparaît à gauche de la cellule, on est en mode édition.



- Certaines actions liées au *Notebook* se font en mode commande, d'autres en mode édition.

Il existe d'autres façons d'exécuter une ou des cellules.

Il existe d'autres façons d'exécuter une ou des cellules.
Il est recommandé d'utiliser pour cela les raccourcis claviers (Menu Aide)

Il existe d'autres façons d'exécuter une ou des cellules.
Il est recommandé d'utiliser pour cela les raccourcis claviers (Menu Aide)
Voici les indispensables :

Il existe d'autres façons d'exécuter une ou des cellules.
Il est recommandé d'utiliser pour cela les raccourcis claviers (Menu Aide)
Voici les indispensables :

- Maj+Entrée : exécute la cellule, sélectionne la cellule suivante ou crée une suivante.

Il existe d'autres façons d'exécuter une ou des cellules.
Il est recommandé d'utiliser pour cela les raccourcis claviers (Menu Aide)
Voici les indispensables :

- Maj+Entrée : exécute la cellule, sélectionne la cellule suivante ou crée une suivante.
- Ctrl+Entrée : exécute la ou les cellules sélectionnées.

Il existe d'autres façons d'exécuter une ou des cellules.
Il est recommandé d'utiliser pour cela les raccourcis claviers (Menu Aide)
Voici les indispensables :

- Maj+Entrée : exécute la cellule, sélectionne la cellule suivante ou crée une suivante.
- Ctrl+Entrée : exécute la ou les cellules sélectionnées.
- Alt+Entrée : exécute la cellule et insére une cellule à la suite.

Il existe d'autres façons d'exécuter une ou des cellules.
Il est recommandé d'utiliser pour cela les raccourcis claviers (Menu Aide)
Voici les indispensables :

- Maj+Entrée : exécute la cellule, sélectionne la cellule suivante ou crée une suivante.
- Ctrl+Entrée : exécute la ou les cellules sélectionnées.
- Alt+Entrée : exécute la cellule et insére une cellule à la suite.
- D+, : en mode commande, supprime la(les) cellule(s) sélectionnée(s) auparavant.

Lien avec le système de fichiers.

Lien avec le système de fichiers.

Dans le *gestionnaire de fichiers* de votre système (explorateur, finder ou nautilus), le fichier sur lequel on travaille ici, se trouve dans :

Lien avec le système de fichiers.

Dans le *gestionnaire de fichiers* de votre système (explorateur, finder ou nautilus), le fichier sur lequel on travaille ici, se trouve dans :

RépertoirePersonnel / Documents / DocPython

Lien avec le système de fichiers.

Dans le *gestionnaire de fichiers* de votre système (explorateur, finder ou nautilus), le fichier sur lequel on travaille ici, se trouve dans :

RépertoirePersonnel / Documents / DocPython

et porte le nom complet : **TD1.ipynb**

- On peut exporter ce fichier sous différents formats (.py, .ipynb, .html, .tex, etc.).

- On peut exporter ce fichier sous différents formats (.py, .ipynb, .html, .tex, etc.).
- Cela se fait par le menu "Fichier".

- On peut exporter ce fichier sous différents formats (.py, .ipynb, .html, .tex, etc.).
- Cela se fait par le menu "Fichier".
- En général, nous utiliserons les cellules pour y mettre du code *Python*, mais il est possible d'y mettre d'autres entrées

- On peut exporter ce fichier sous différents formats (.py, .ipynb, .html, .tex, etc.).
- Cela se fait par le menu "Fichier".
- En général, nous utiliserons les cellules pour y mettre du code *Python*, mais il est possible d'y mettre d'autres entrées
  - Markdown

- On peut exporter ce fichier sous différents formats (.py, .ipynb, .html, .tex, etc.).
- Cela se fait par le menu "Fichier".
- En général, nous utiliserons les cellules pour y mettre du code *Python*, mais il est possible d'y mettre d'autres entrées
  - Markdown
  - Texte brut

- On peut exporter ce fichier sous différents formats (.py, .ipynb, .html, .tex, etc.).
- Cela se fait par le menu "Fichier".
- En général, nous utiliserons les cellules pour y mettre du code *Python*, mais il est possible d'y mettre d'autres entrées
  - Markdown
  - Texte brut
  - Image, ...

- On peut exporter ce fichier sous différents formats (.py, .ipynb, .html, .tex, etc.).
- Cela se fait par le menu "Fichier".
- En général, nous utiliserons les cellules pour y mettre du code *Python*, mais il est possible d'y mettre d'autres entrées
  - Markdown
  - Texte brut
  - Image, ...
- Il est ainsi possible d'exporter l'ensemble sous forme de slide pour exposer son travail.

 Tant qu'une cellule n'a pas été exécutée, *Python* n'a pas connaissance de son contenu.

 Tant qu'une cellule n'a pas été exécutée, *Python* n'a pas connaissance de son contenu.

Conséquences :

 Tant qu'une cellule n'a pas été exécutée, *Python* n'a pas connaissance de son contenu.

Conséquences :

- quand on ouvre un Notebook précédemment fermé, il faut exécuter l'ensemble des cellules pour reprendre où vous en étiez.

 Tant qu'une cellule n'a pas été exécutée, *Python* n'a pas connaissance de son contenu.

Conséquences :

- quand on ouvre un Notebook précédemment fermé, il faut exécuter l'ensemble des cellules pour reprendre où vous en étiez.
- quand une cellule est en cours d'exécution, une petite *étoile* figure à la droite d'*Entrée*, entre les crochets. Tant que cette étoile est présente, *Python* est occupé !

 Tant qu'une cellule n'a pas été exécutée, *Python* n'a pas connaissance de son contenu.

Conséquences :

- quand on ouvre un Notebook précédemment fermé, il faut exécuter l'ensemble des cellules pour reprendre où vous en étiez.
- quand une cellule est en cours d'exécution, une petite *étoile* figure à la droite d'*Entrée*, entre les crochets. Tant que cette étoile est présente, *Python* est occupé !
- il est possible de fusionner des cellules, ou de couper une cellule en deux (voir le menu "Édition")

Python est en train de travailler...

Entrée [*]:

```
# génère un million d'entier aléatoiremnt entre 0 et 2000
import random
l, som = [], 0
for i in range(1,1000000):
    a = random.randint(1,2000)
    l.append(a)
    som = sum(l)  # somme de la liste à chaque itération
```

Python est en train de travailler...

Entrée [*]:

```
# génère un million d'entier aléatoiremnt entre 0 et 2000
import random
l, som = [], 0
for i in range(1,1000000):
    a = random.randint(1,2000)
    l.append(a)
    som = sum(l)  # somme de la liste à chaque itération
```

Le nombre qui figure après le mot "Entrée" indique le nombre d'exécution de la cellule (ici 6).

Entrée [6]:

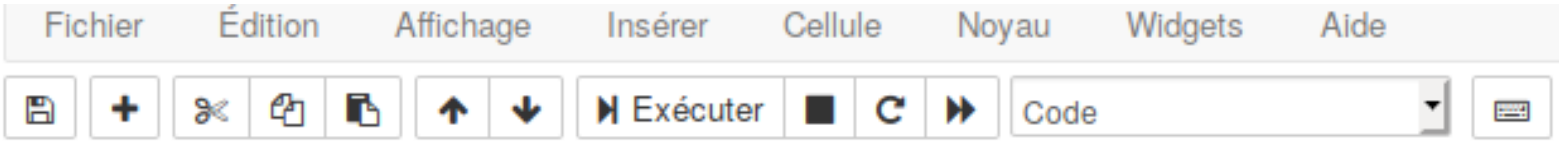
```
# génère un million d'entier aléatoiremnt entre 0 et 2000
import random
l, som = [], 0
for i in range(1,1000000):
    a = random.randint(1,2000)
    l.append(a)
    som = sum(l)  # somme de la liste à chaque itération
```

- Les possibilités du Notebook sont bien plus étendues.

- Les possibilités du Notebook sont bien plus étendues. Vous les découvrirez par la suite.

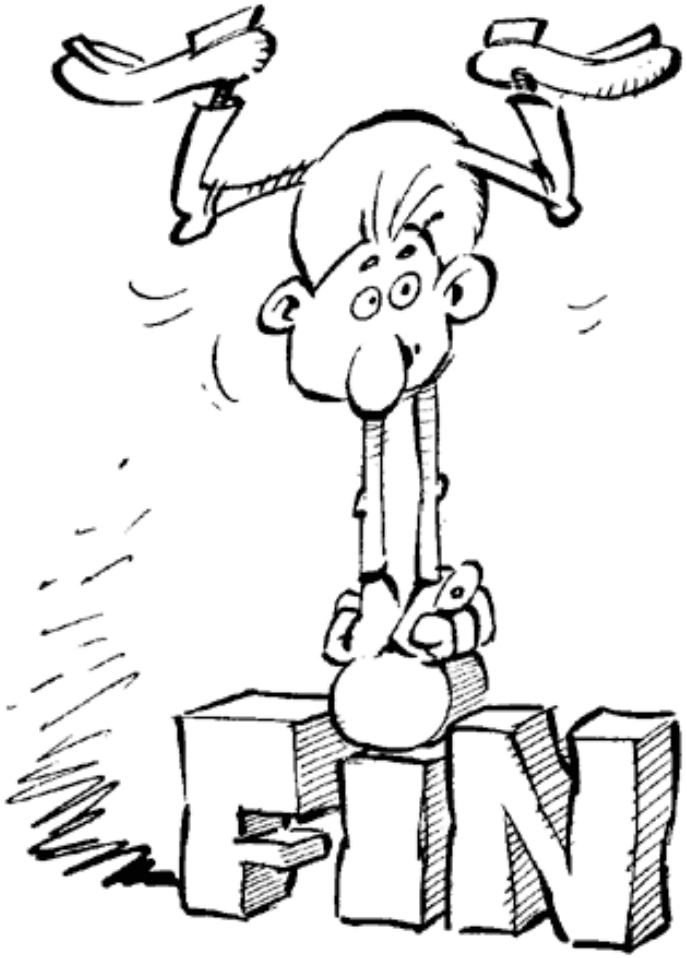
- Les possibilités du Notebook sont bien plus étendues. Vous les découvrirez par la suite.
- Il est conseillé de bien maîtriser les "raccourcis clavier".

- Les possibilités du Notebook sont bien plus étendues. Vous les découvrirez par la suite.
- Il est conseillé de bien maîtriser les "raccourcis clavier".
- En cas de plantage, ou de boucle infinie (nous verrons plus tard...), ou de routine trop longue, il faut interrompre et redémarrer le noyau (menu Noyau).



- Les possibilités du Notebook sont bien plus étendues. Vous les découvrirez par la suite.
- Il est conseillé de bien maîtriser les "raccourcis clavier".
- En cas de plantage, ou de boucle infinie (nous verrons plus tard...), ou de routine trop longue, il faut interrompre et redémarrer le noyau (menu Noyau).

- L'enregistrement de votre travail est automatique, mais peut être provoqué en cliquant sur l'icône le plus à gauche (disquette).


FIN