

Langage Python

ISUP, Sorbonne Université

Etienne Guével

Ingénieur de Recherche - SCAI

etienne.guevel@sorbonne-universite.fr

Septembre-Novembre 2025

Objectifs du cours

Thématiques abordées

- Fondamentaux du langage et outils associés
- Programmation orientée objet
- Pratique des bibliothèques scientifiques

Chaque semaine se déroulera en 2 parties :

- CM
- TP sur machine

Évaluation finale sur projet

Contenu du cours

Partie 1 – Fondamentaux

- Revue des outils, installation et prise en main
- Syntaxe, structures de contrôle
- Types, structures de données

Partie 2 – Programmation orientée objet

- Classe, objets
- Attributs, méthodes
- Héritage

Partie 3 – Bibliothèques scientifiques : NumPy, Pandas, Matplotlib

Partie 4 – Packaging, tests unitaires et bonnes pratiques

Quelques références

- The Python Language Reference
docs.python.org/3/reference/index.html
- Luciano Ramalho (2019) Fluent Python: Clear, concise and effective programming, O'Reilly.
- www.w3schools.com/python
- realpython.com

Introduction

Historique

Le langage Python a été créé par Guido van Rossum en 1989 et rendu public en 1991. Le nom fait référence aux *Monty Python*.

G. van Rossum a été jusqu'à 2018 "Benevolent Dictator for Life".

www.python.org

Un langage facile à interpréter

- Simplificité d'écriture et flexibilité

```
duck.py
part_1 > example_code > duck.py > ...
1  class Duck:
2      def __init__(self):
3          self.name = 'duck'
4
5      def quack(self):
6          print('Duck says quack!')
7
8      def fly(self):
9          print('Duck flies away!')
10
11 duck = Duck()
12 duck.quack()
13 duck.fly()
14
```

Un langage facile à interpréter

- Simplificité d'écriture et flexibilité
- Exécution par un interpréteur ligne par ligne

```
duck.py
part_1 > example_code > duck.py > ...
1  class Duck:
2      def __init__(self):
3          self.name = 'duck'
4
5      def quack(self):
6          print('Duck says quack!')
7
8      def fly(self):
9          print('Duck flies away!')
10
11  duck = Duck()
12  duck.quack()
13  duck.fly()
14
```


Très adaptable

- Exécution interactive ou par script

```
etiennequevel ~/cours_2025/part_1/example_code v3.13.3 12:35 > python3 duck.py
Duck says quack!
Duck flies away!
```

```
duck.ipynb x
part_1 > example_code > duck.ipynb > duck = Duck()
Generate + Code + Markdown | Run All Restart Clear All Outputs

class Duck:
    def __init__(self):
        self.name = 'duck'

    def quack(self):
        print('Duck says quack!')

    def fly(self):
        print('Duck flies away!')

[3] ✓ 0.0s

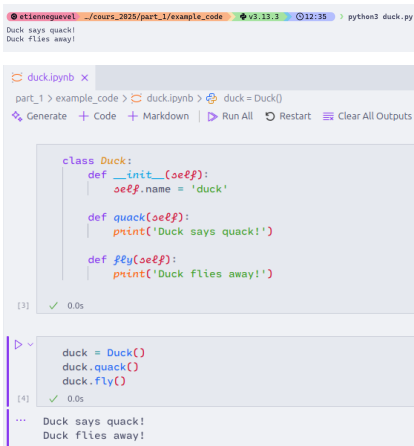
duck = Duck()
duck.quack()
duck.fly()

[4] ✓ 0.0s

...
Duck says quack!
Duck flies away!
```

Très adaptable

- Exécution interactive ou par script
- Multiplateforme et open source



The image shows two screenshots. The top screenshot is a terminal window with the command `python3 duck.py` and its output: `Duck says quack!` and `Duck flies away!`. The bottom screenshot is a Jupyter Notebook interface. It shows a code cell with the following Python code:

```
class Duck:
    def __init__(self):
        self.name = 'duck'

    def quack(self):
        print('Duck says quack!')

    def fly(self):
        print('Duck flies away!')
```

Below the code cell, the execution output is shown:

```
[3] ✓ 0.0s
```

The next cell shows the execution of the code:

```
duck = Duck()
duck.quack()
duck.fly()
```

Below this cell, the execution output is shown:

```
[4] ✓ 0.0s
```

Finally, the output of the code execution is displayed:

```
... Duck says quack!
Duck flies away!
```

Très adaptable

- Exécution interactive ou par script
- Multiplateforme et open source
- Correction de bugs **relativement simple**

```
etiennequevel ~/cours_2025/part_1/example_code v3.13.3 12:35 > python3 duck.py
Duck says quack!
Duck flies away!
```

```
duck.ipynb x
part_1 > example_code > duck.ipynb > duck = Duck()
Generate + Code + Markdown | Run All Restart Clear All Outputs

class Duck:
    def __init__(self):
        self.name = 'duck'

    def quack(self):
        print('Duck says quack!')

    def fly(self):
        print('Duck flies away!')

[3] ✓ 0.0s

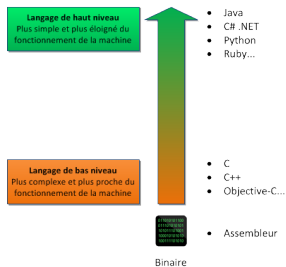
duck = Duck()
duck.quack()
duck.fly()

[4] ✓ 0.0s

... Duck says quack!
Duck flies away!
```

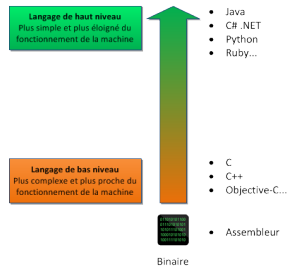
Mais relativement peu performant

- Interprétation vs compilation



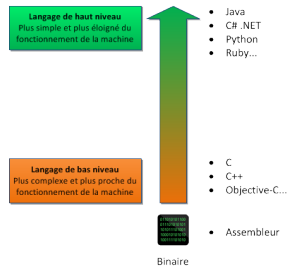
Mais relativement peu performant

- Interprétation vs compilation
- Gestion de données et bibliothèques de haut niveau



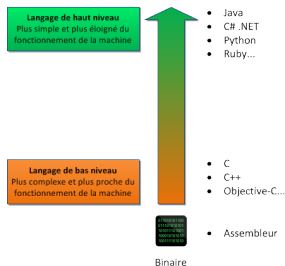
Mais relativement peu performant

- Interprétation vs compilation
- Gestion de données et bibliothèques de haut niveau
- Typage dynamique

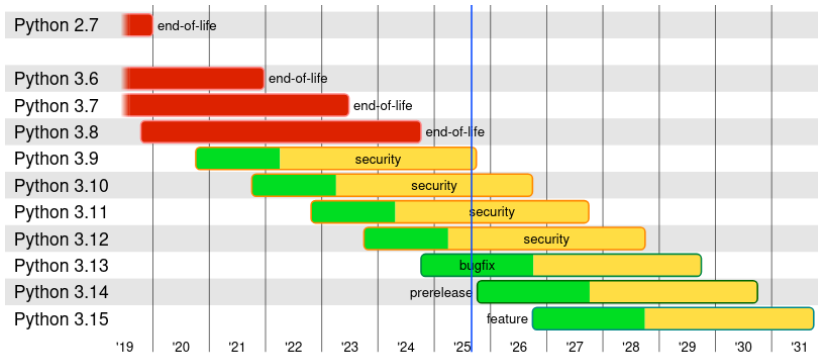


Mais relativement peu performant

- Interprétation vs compilation
- Gestion de données et bibliothèques de haut niveau
- Typage dynamique
- Gestion automatique de mémoire



Une histoire de versions



Installation et revue des outils

Pré-requis pour le cours

- Un terminal

Pré-requis pour le cours

- Un terminal
 - Linux / MacOS : le terminal par défaut

Pré-requis pour le cours

- Un terminal
 - Linux / MacOS : le terminal par défaut
 - Windows : Powershell, intégré à Anaconda

Pré-requis pour le cours

- Un terminal
 - Linux / MacOS : le terminal par défaut
 - Windows : Powershell, intégré à Anaconda
 - Windows 10 ou 11 (conseil) : Windows Subsystem for Linux (wsl)

Pré-requis pour le cours

- Un terminal
 - Linux / MacOS : le terminal par défaut
 - Windows : Powershell, intégré à Anaconda
 - Windows 10 ou 11 (conseil) : Windows Subsystem for Linux (wsl)
- Python, Anaconda, Miniconda version ≥ 3.10

Pré-requis pour le cours

- Un terminal
 - Linux / MacOS : le terminal par défaut
 - Windows : Powershell, intégré à Anaconda
 - Windows 10 ou 11 (conseil) : Windows Subsystem for Linux (wsl)
- Python, Anaconda, Miniconda version ≥ 3.10
- jupyter-lab ou jupyter-notebook

Pré-requis pour le cours

- Un terminal
 - Linux / MacOS : le terminal par défaut
 - Windows : Powershell, intégré à Anaconda
 - Windows 10 ou 11 (conseil) : Windows Subsystem for Linux (wsl)
- Python, Anaconda, Miniconda version ≥ 3.10
- jupyter-lab ou jupyter-notebook
- Un IDE (ex. VSCode, PyCharm)

Commandes Linux à connaître

Linux command	Description	Linux command example
cd	Change directory with a specified path	<code>cd /path/directory1</code>
clear	Clear the screen	<code>clear</code>
cp	Copy file(s)	<code>cp /path1/file1 /path2/file1</code>
diff	Compare the contents of files	<code>diff file1 file2</code>
exit	Log out of Linux	<code>exit</code>
grep	Find a string of text in a file	<code>grep "word or phrase" file1</code>
head	Display beginning of a file	<code>head file1</code>
less	View a file	<code>less file1</code>
ls	List contents of a directory	<code>ls /path/directory1</code>
mv	Move file(s) or rename file(s)	<code>mv /path1/file1 /path2/file2</code>
mkdir	Create a directory	<code>mkdir directory</code>
rm	Delete file(s)	<code>rm file1</code>
rmdir	Remove a directory	<code>rmdir directory</code>
tail	Display end of a file	<code>tail file1</code>
tar	Store, list or extract files in an archive	<code>tar file1</code>
vi	Edit file(s) with simple text editor	<code>vi file1</code>

Installation de Python

Python3 est en général déjà installé, sous Linux **Python2 est systématiquement installé.**

Il existe plusieurs façons d'installer python (et ses outils) :

1/ Paquets Python

Pour Linux (ou wsl) :

```
$ sudo apt-get install python3 python3-pip
```

Pour MacOS :

```
$ brew install python
```

Installation de Python

Python3 est en général déjà installé, sous Linux **Python2 est systématiquement installé.**

Il existe plusieurs façons d'installer python (et ses outils) :

2/ Anaconda

- Installation complète de l'environnement
- Un très grand nombre de librairies pré-installées
- anaconda navigator installé
- Multiplateforme : Windows, Linux, MacOS

Voir : <https://docs.anaconda.com/anaconda/install>

Installation de Python

Python3 est en général déjà installé, sous Linux **Python2 est systématiquement installé.**

Il existe plusieurs façons d'installer python (et ses outils) :

3/ Miniconda

- Installation minimale de l'environnement (480 MB contre 4.4GB)
- Aucune librairie installée
- anaconda navigator non installé
- Multiplateforme : Windows, Linux, MacOS

Voir : <https://docs.anaconda.com/miniconda/miniconda-install/>

Que Choisir ?

- Python officiel : pour avoir un environnement avec plus de contrôle sur les outils installés
- Anaconda : pour avoir un environnement clé en main
- Miniconda : pour avoir un environnement minimal avec la possibilité de créer des environnements virtuels avec conda

Installation de Python

Une fois Python installé, ouvrir un terminal et taper

```
$ python --version  
$ python3 --version
```

Installation de Python via Anaconda

Home

Anaconda Individual Edition

Installation

Installing on Windows

Installing on macOS

Installing on Linux

Installing on Linux-aarch64 (arm64)

Installing on AWS Graviton2 (arm64)

Installing on Linux-s390x (IBM Z)

Installing on Linux POWER

Installing in silent mode

Installing for multiple users

Verifying your installation

Anaconda installer file hashes

Updating from older versions

Uninstalling Anaconda

User guide

Reference

End User License Agreement - Anaconda Individual Edition



Installation

Review the system requirements listed below before installing Anaconda Individual Edition. If you don't want the hundreds of packages included with Anaconda, you can [install Miniconda](#), a mini version of Anaconda that includes just conda, its dependencies, and Python.



Looking for Python 3.5 or 3.6? See our [FAQ](#).

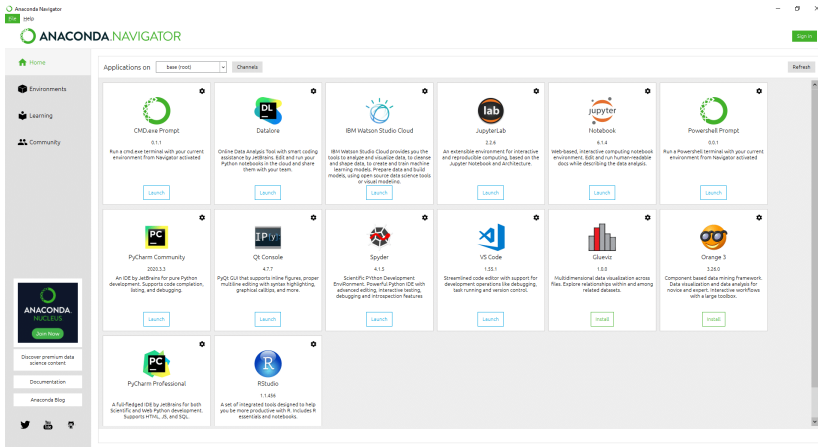
System requirements

- License: Free use and redistribution under the terms of the [EULA for Anaconda Individual Edition](#).
- Operating system: Windows 8 or newer, 64-bit macOS 10.13+, or Linux, including Ubuntu, RedHat, CentOS 7+, and others.
- If your operating system is older than what is currently supported, you can find older versions of the Anaconda installers in our [archive](#) that might work for you. See [Using Anaconda on older operating systems](#) for version recommendations.
- System architecture: Windows- 64-bit x86, 32-bit x86; MacOS- 64-bit x86; Linux- 64-bit x86, 64-bit aarch64 (AWS Graviton2 / arm64), 64-bit Power8/Power9, s390x (Linux on IBM Z & LinuxONE).
- Minimum 5 GB disk space to download and install.

On Windows, macOS, and Linux, it is best to install Anaconda for the local user, which does not require administrator permissions and is the most robust type of installation. However, if you need to, you can install Anaconda system wide, which does require administrator permissions.

- Installing on Windows
- Installing on macOS
- Installing on Linux
- Installing on Linux-aarch64 (arm64)
- Installing on AWS Graviton2 (arm64)
- Installing on Linux-s390x (IBM Z)
- Installing on Linux POWER

Installation de Python via Anaconda



Environnements virtuels

Pour gérer des environnements virtuels, avec une installation spécifique

```
$ python -m venv myenv  
$ source myenv/bin/activate  
$ pip install numpy  
$ deactivate
```

Voir ce [lien](#) pour les venvs.

Certaines installations de python viennent sans venv (plutôt pour linux), pour l'installer :

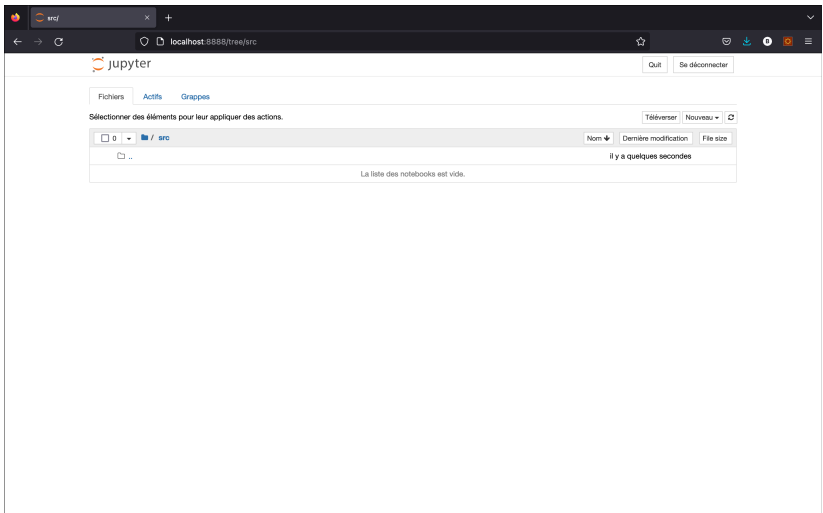
```
$ sudo apt install python3-venv # pour linux
```

Environnements conda

Pour gérer des environnements virtuels, avec une installation spécifique

```
$ conda create --name myenv python=3.12
$ conda activate myenv
$ conda install pip
$ pip install numpy
$ conda deactivate
```

Voir ce [lien](#) pour les envs conda.



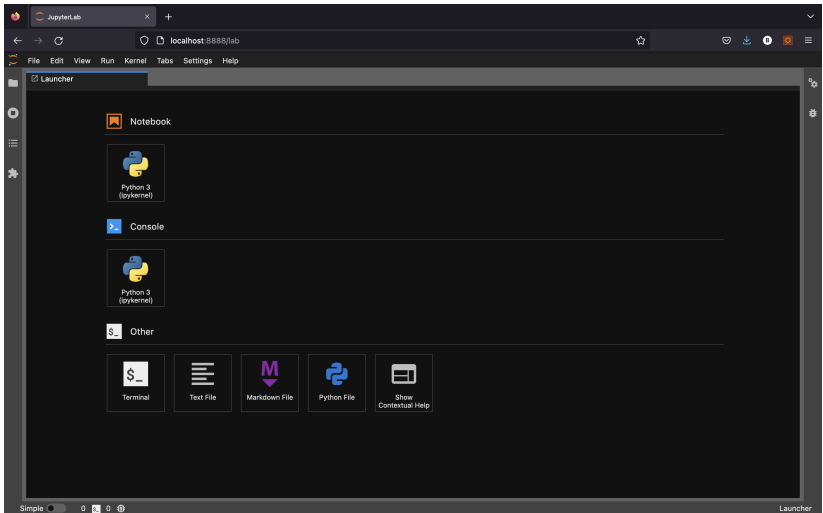
Pour installer jupyter dans un environnement python via pip :

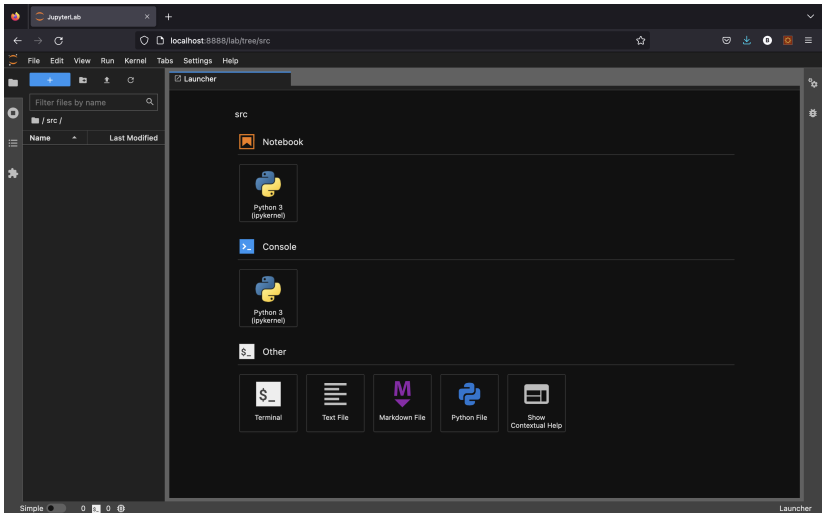
```
$ source my_env/bin/activate  
$ pip install jupyter # ou notebook ou jupyterlab  
$ pip install ipython # package minimal pour vscode notebook
```

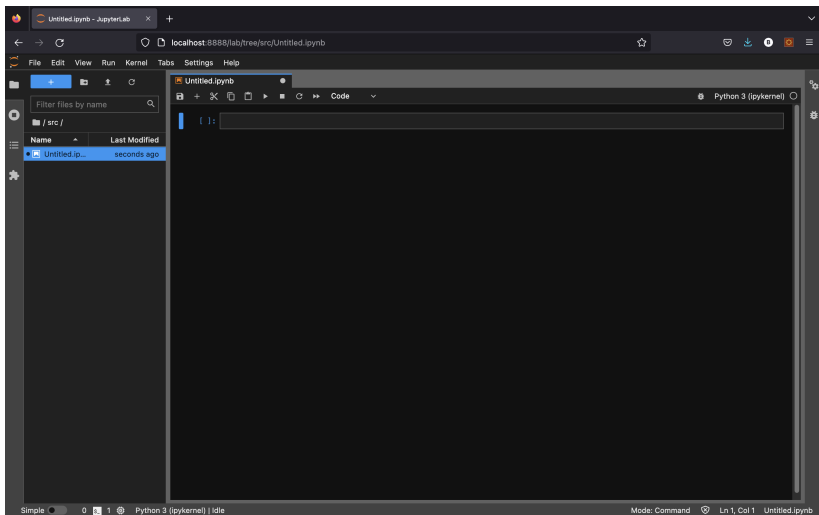
```
$ jupyter lab # ou jupyter notebook
```

Cela va ensuite ouvrir jupyter dans un navigateur. Si ce n'est pas le cas, un lien apparaîtra dans le terminal : CTRL + click dessus.

Jupyter







Exécution d'un programme python

Cas 1 : via un shell python ou ipython

Exécution à la volée d'instructions python. Ouvrir un shell

```
$ python
```

Ecrire des instructions

```
>>> text = "my name is Etienne Guevel"  
>>> print(text)
```

```
❶ etienneguevel .../cours_2025/part_1/example_code v3.13.3(#env) 12:54 » python3  
Python 3.13.3 (main, Aug 14 2025, 11:53:40) [GCC 14.2.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> text = "my name is Etienne Guevel"  
>>> print(text)  
my name is Etienne Guevel  
>>> █
```


Exécution d'un programme python

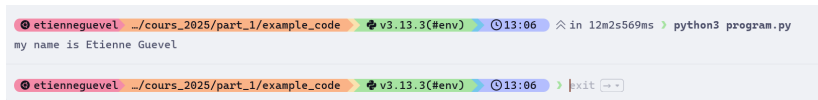
Cas 2 : via un terminal

Dans un éditeur de texte, créer un fichier `program.py` et écrire

```
text = "my name is Etienne Guevel"  
print(text)
```

Exécuter la ligne de commande

```
$ python program.py
```



The screenshot shows a terminal window with a light blue background. The prompt is `etienneguevel` and the current directory is `~/cours_2025/part_1/example_code`. The terminal shows the command `python3 program.py` being executed, which outputs `my name is Etienne Guevel`. The terminal also shows the command `exit` being entered, which results in a `↩` prompt.

```
etienneguevel ~/cours_2025/part_1/example_code v3.13.3(#env) 13:06 in 12m2s569ms > python3 program.py  
my name is Etienne Guevel  
  
etienneguevel ~/cours_2025/part_1/example_code v3.13.3(#env) 13:06 > |exit ↩
```

Exécution d'un programme python

Cas 3 : via Jupyter

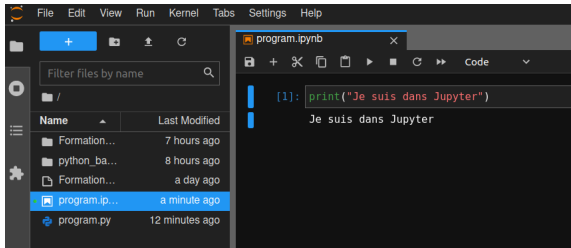
Lancer Jupyter depuis un terminal

```
$ jupyter notebook // ou jupyter lab
```

Dans un Jupyter notebook, écrire

```
print("Je suis dans Jupyter")
```

et exécuter la cellule correspondante



Outils de base :

- python : exécuter du code à la volée
- pip : installer des packages
- pytest : tests unitaires
- pylint : vérification de la qualité du code source

Outils de base :

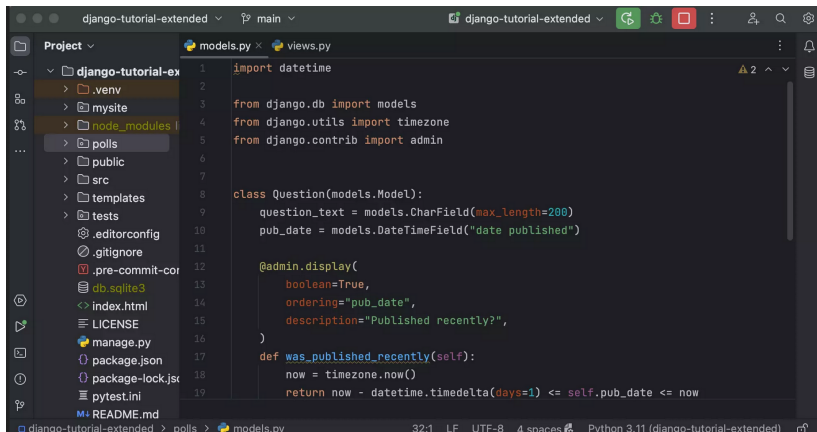
- python : exécuter du code à la volée
- pip : installer des packages
- pytest : tests unitaires
- pylint : vérification de la qualité du code source

Intégré au projet Anaconda :

- ipython : version interactive du shell python
- conda : installer des packages, gérer des environnements d'exécutions (voir ici), etc.
- Jupyter : environnement de développement interactif et flexible

Installer des IDEs

Les IDEs sont des environnements de développement intégrés (Integrated Development Environment) qui permettent de faciliter la création de programmes.



The screenshot shows a code editor interface with a dark theme. On the left, a sidebar displays the project structure for 'django-tutorial-extended'. The main editor area shows the content of 'models.py', which defines a 'Question' model with a 'question_text' field and a 'pub_date' field, along with a 'was_published_recently' method. The status bar at the bottom indicates the file is at line 32, column 1, using UTF-8 encoding with 4 spaces, and is running Python 3.11.

```
1 import datetime
2
3 from django.db import models
4 from django.utils import timezone
5 from django.contrib import admin
6
7
8 class Question(models.Model):
9     question_text = models.CharField(max_length=200)
10    pub_date = models.DateTimeField("date published")
11
12    @admin.display(
13        boolean=True,
14        ordering="pub_date",
15        description="Published recently?",
16    )
17    def was_published_recently(self):
18        now = timezone.now()
19        return now - datetime.timedelta(days=1) <= self.pub_date <= now
```

Installer des IDEs

Les IDEs sont des environnements de développement intégrés (Integrated Development Environment) qui permettent de faciliter la création de programmes.

