

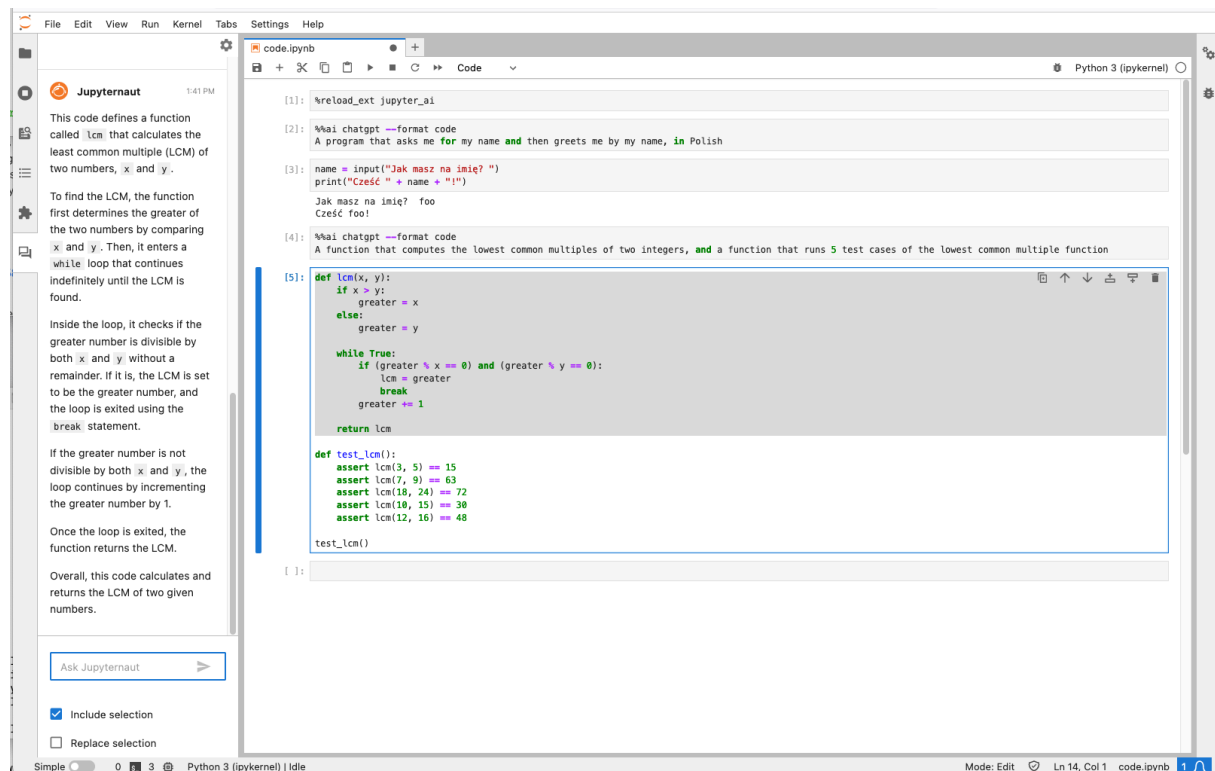
Jupyter AI

Jupyter AI is under incubation as part of the JupyterLab organization.

Jupyter AI connects generative AI with Jupyter notebooks. Jupyter AI provides a user-friendly and powerful way to explore generative AI models in notebooks and improve your productivity in JupyterLab and the Jupyter Notebook. More specifically, Jupyter AI offers:

- An `%%ai` magic that turns the Jupyter notebook into a reproducible generative AI playground. This works anywhere the IPython kernel runs (JupyterLab, Jupyter Notebook, Google Colab, Kaggle, VSCode, etc.).
- A native chat UI in JupyterLab that enables you to work with generative AI as a conversational assistant.
- Support for a wide range of generative model providers, including AI21, Anthropic, AWS, Cohere, Gemini, Hugging Face, NVIDIA, and OpenAI.
- Local model support through GPT4All, enabling use of generative AI models on consumer grade machines with ease and privacy.

Documentation is available on [ReadTheDocs](#).



Requirements

You will need to have installed the following software to use Jupyter AI:

- Python 3.8 - 3.11
- JupyterLab 4

In addition, you will need access to at least one model provider.

[!IMPORTANT] JupyterLab 3 will reach its end of maintenance date on May 15, 2024, anywhere on Earth. As a result, we will not backport new features to the v1 branch supporting JupyterLab 3 after this date. Fixes for critical issues will still be backported until December 31, 2024. If you are still using JupyterLab 3, we strongly encourage you to **upgrade to JupyterLab 4 as soon as possible**. For more information, see JupyterLab 3 end of maintenance on the Jupyter Blog.

Setting Up Model Providers in a Notebook

To use any AI model provider within this notebook, you'll need the appropriate credentials, such as API keys.

Obtain the necessary credentials, such as API keys, from your model provider's platform.

You can set your keys using environment variables or in a code cell in your notebook. In a code cell, you can use the `%env` magic command to set the credentials as follows:

```
1 # NOTE: Replace 'PROVIDER_API_KEY' with the credential key's name,  
2 # and replace 'YOUR_API_KEY_HERE' with the key.  
3 %env PROVIDER_API_KEY=YOUR_API_KEY_HERE
```

For more specific instructions for each model provider, refer to the model providers documentation.

Installation

Below is a simplified overview of the installation and usage process. See our official documentation for details on installing and using Jupyter AI.

With pip

If you want to install both the `%%ai` magic and the JupyterLab extension, you can run:

```
1 $ pip install jupyter-ai
```

If you are not using JupyterLab and you only want to install the Jupyter AI `%%ai` magic, you can run:

```
1 $ pip install jupyter-ai-magics
```

With conda

As an alternative to using `pip`, you can install `jupyter-ai` using Conda from the `conda-forge` channel, using one of the following two commands:

```
1 $ conda install -c conda-forge jupyter-ai # or,  
2 $ conda install conda-forge::jupyter-ai
```

The `%%ai` magic command

The `%%ai` magic works anywhere the IPython kernel runs, including JupyterLab, Jupyter Notebook, Google Colab, and Visual Studio Code.

Once you have installed the `%%ai` magic, you can enable it in any notebook or the IPython shell by running:

```
1 %load_ext jupyter_ai_magics
```

or:

```
1 %load_ext jupyter_ai
```

The screenshots below are from notebooks in the `examples/` directory of this package.

Then, you can use the `%%ai` magic command to specify a model and natural language prompt:

```
[15]: %%ai chatgpt
Please generate the Python code to solve the 2D Laplace equation in cartesian coordinates.
Solve the equation on the square domain x=(0,1) and y=(0,1) with vanishing boundary conditions.
Plot the solution using Matplotlib.
Please also provide an explanation.
```

Here's the Python code to solve the 2D Laplace equation in Cartesian coordinates:

```
import numpy as np
import matplotlib.pyplot as plt

# Set up grid
nx = 101
ny = 101
nt = 100

dx = 1. / (nx - 1)
dy = 1. / (ny - 1)

x = np.linspace(0, 1, nx)
y = np.linspace(0, 1, ny)

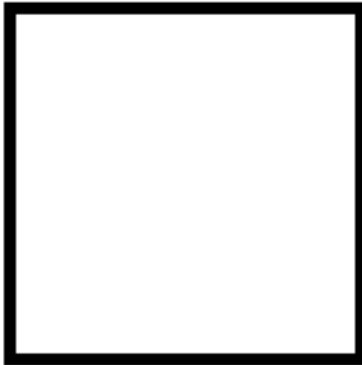
# Initialize solution
u = np.zeros((nx, ny))

# Set boundary conditions
u[0,:] = 0
u[-1,:] = 0
u[:,0] = 0
u[:,nt-1] = 0
```

Jupyter AI can also generate HTML and math to be rendered as cell output.

```
[3]: %%ai anthropic:claude-v1.2 -f html
Create a square using SVG with a black border and white fill.
```

```
[3]:
```



```
[4]: %%ai chatgpt -f math
Generate the 2D heat equation.
```

```
[4]:
```

$$\frac{\partial u}{\partial t} = \alpha \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)$$

Jupyter AI can interpolate IPython expressions, allowing you to run prompts that include variable values.

```
[9]: poet = "Walt Whitman"
```

```
[10]: %%ai chatgpt  
Write a poem in the style of {poet}
```

O Captain, my Captain, you have led us through the fray
Through the storm and the turmoil, through the darkest of days
With your unwavering courage, your steadfast heart ablaze
You have lifted us up where the eagles soar and play

Your voice is as thunder, your soul a raging fire
You guide us through the tempest, you lift us ever higher
With you at our helm, we shall never falter or tire
For you are the Captain, and we are your loyal choir

Together we shall march, to the beat of our own drum
We shall conquer every fear, we shall overcome

JupyterLab extension

The Jupyter AI extension for JupyterLab offers a native UI that enables multiple users to chat with the Jupyter AI conversational assistant. If you have JupyterLab installed, this should be installed and activated when you install the `jupyter_ai` package.

Using

For help with installing and using Jupyter AI, please see our user documentation on ReadTheDocs.

Contributing

If you would like to contribute to Jupyter AI, see our contributor documentation on ReadTheDocs.