NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

College of Computing and Data Science

Software tools for CA6000 – Local LLM and Jupyter AI

Continuing with our discussion of low code programming using AI coding assistant, we will install a large language model (LLM) and integrate it with the Jupyter AI in our Jupyter Lab on our laptop, such that we can have our own local AI coding assistant running on our laptop without the need to connect to the LLM hosted on the cloud (i.e. internet)

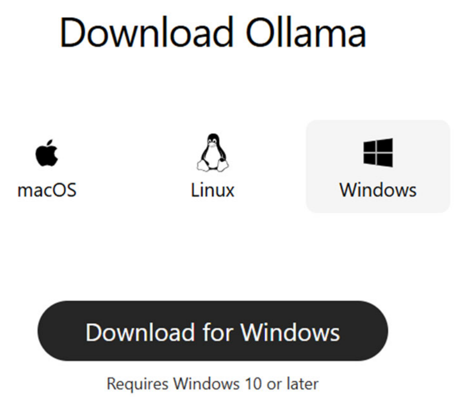
We will be using the following packages in the installation.

- Ollama + selected LLM
- Jupyter Lab install in a separate environment
- Jupyter AI

A. Installing Ollama (Procedure and screenshots are as of Nov 2025)

Ollama is an open-source tool that can allow us to run LLMs locally on our computer.

- Go to the Ollama download website (https://ollama.com/download) and choose the appropriate version for your computer to download its installer program.

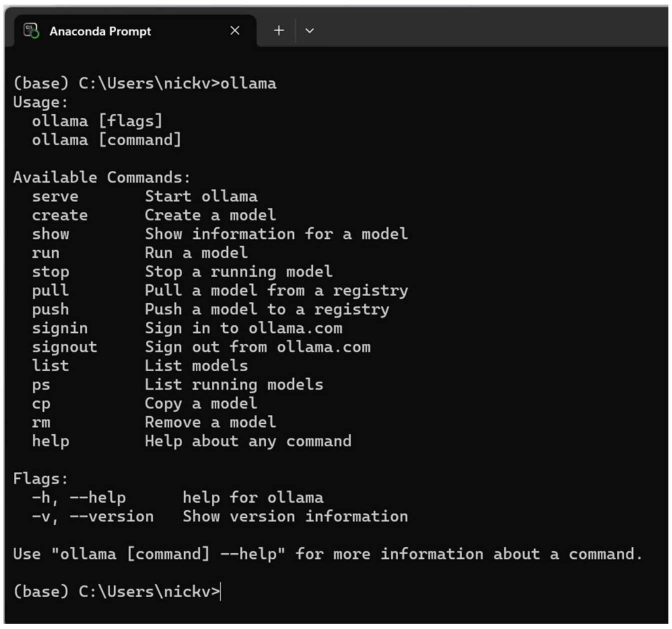


- Run the installer program - you will see a message similar to the following

```
Extracting files...
C:\Users\nickv\AppData\Local\Programs\Ollama\lib\ollama\cuda_v12\cublasLt64_12.dll
```

- Following the on-screen instruction to complete the installation. (A small Ollama icon may appear at the corner of your taskbar that will launch a GUI based interface if you click on it. But we will not be using this interface)

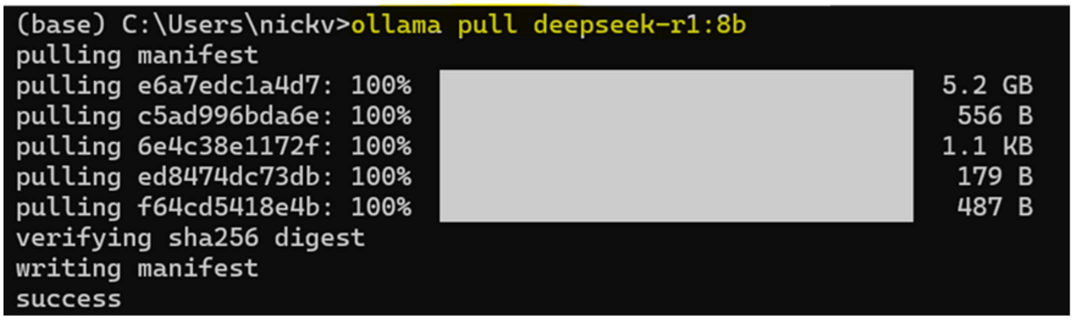
- After the installation, the Ollama server should execute automatically - you can check that it is running by opening a terminal (such as the anaconda prompt) and type the command "ollama"



```
(base) C:\Users\nickv>ollama
Usage:
  ollama [flags]
  ollama [command]

Available Commands:
  serve       Start ollama
  create      Create a model
  show        Show information for a model
  run         Run a model
  stop        Stop a running model
  pull        Pull a model from a registry
  push        Push a model to a registry
  signin      Sign in to ollama.com
  signout     Sign out from ollama.com
  list        List models
  ps          List running models
  cp          Copy a model
  rm          Remove a model
  help        Help about any command

Flags:
  -h, --help      help for ollama
  -v, --version   Show version information

Use "ollama [command] --help" for more information about a command.

(base) C:\Users\nickv>
```

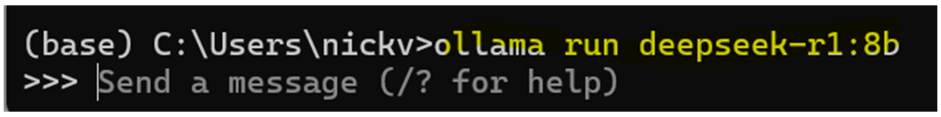
- You can use the "ollama list" command to view all the downloaded models. It should be empty at this stage as this is the first time you use Ollama.

- We will next install an LLM model to run with the Ollama. The available LLM models are listed at https://ollama.com/library. To download a specific model, use the "ollama pull" command. E.g. ollama pull deepseek-r1:8b



```
(base) C:\Users\nickv>ollama pull deepseek-r1:8b
pulling manifest
pulling e6a7edc1a4d7: 100%                          5.2 GB
pulling c5ad996bda6e: 100%                           556 B
pulling 6e4c38e1172f: 100%                          1.1 KB
pulling ed8474dc73db: 100%                           179 B
pulling f64cd5418e4b: 100%                           487 B
verifying sha256 digest
writing manifest
success
```

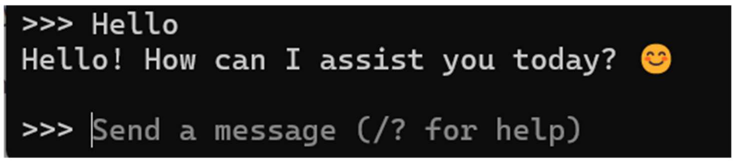
- Once the model is downloaded, we can run the model by using the "ollama run" command. E.g. ollama run deepseek-r1:8b



```
(base) C:\Users\nickv>ollama run deepseek-r1:8b
>>> Send a message (/? for help)
```

Note: You will use this command to launch the Ollama server with the selected model in future run of your system.

- You can test run the installation by typing a message, which you should see a response such as the following . E.g., type "Hello"

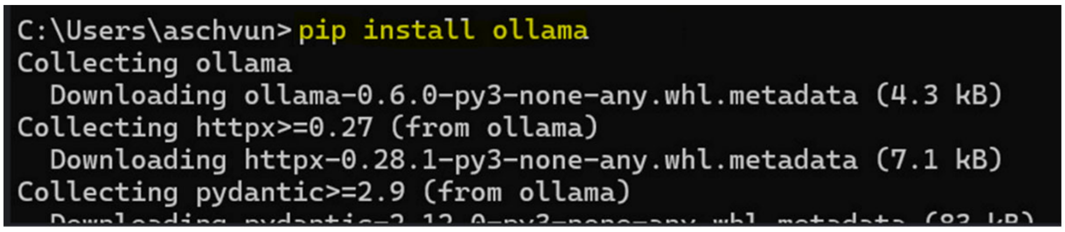


```
>>> Hello
Hello! How can I assist you today? 😊

>>> Send a message (/? for help)
```

B. Optional: Installing Python Library

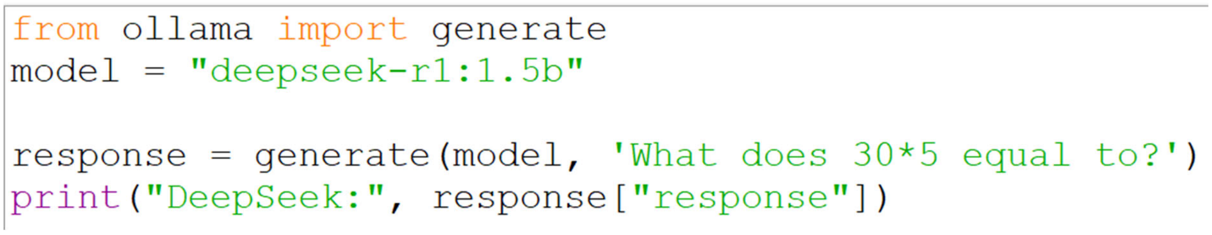
We can also interact with the Ollama LLM using Python code. To do so, we will install the Ollama's Python library (which is called 'ollama' - a bit confusing, I know)

- Open a cmd console, or an Anaconda Prompt console and type the command 'pip install ollama'

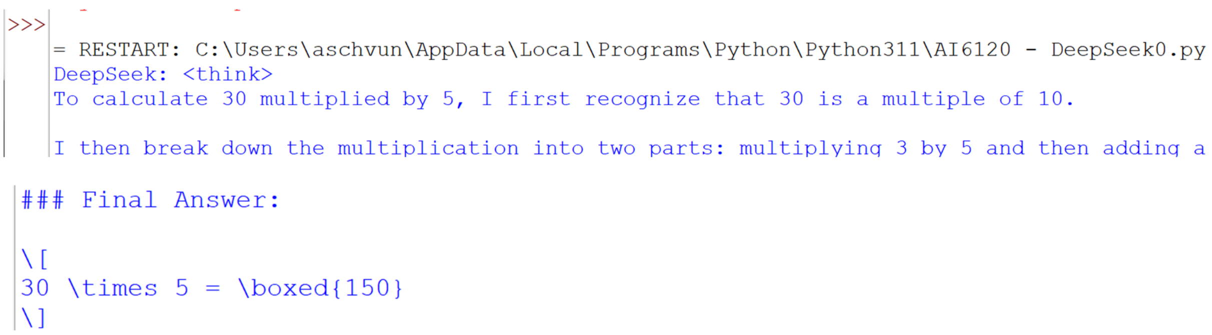


```
C:\Users\aschvun>pip install ollama
Collecting ollama
  Downloading ollama-0.6.0-py3-none-any.whl.metadata (4.3 kB)
Collecting httpx>=0.27 (from ollama)
  Downloading httpx-0.28.1-py3-none-any.whl.metadata (7.1 kB)
Collecting pydantic>=2.9 (from ollama)
```

The Ollama Python library provides functions such as **`generate()`, `chat()`** that you can use in a Python program to 'chat' with the LLM.



```python
from ollama import generate
model = "deepseek-r1:1.5b"

response = generate(model, 'What does 30*5 equal to?')
print("DeepSeek:", response["response"])
```



```
>>>
= RESTART: C:\Users\aschvun\AppData\Local\Programs\Python\Python311\AI6120 - DeepSeek0.py
DeepSeek: <think>
To calculate 30 multiplied by 5, I first recognize that 30 is a multiple of 10.

I then break down the multiplication into two parts: multiplying 3 by 5 and then adding a

### Final Answer:

\[
30 \times 5 = \boxed{150}
\]
```

C. Installing Jupyter Lab in a new Environment

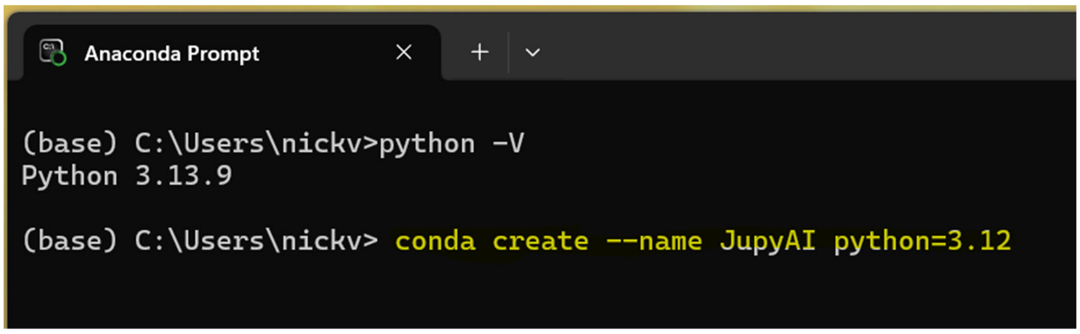
Jupyter AI is an extension package that enable the integration of generative AI capabilities directly in the Jupyter Lab (and Jupyter Notebook) platform. Once it is installed, we can use it to interact with a LLM model that is running remotely in the cloud through internet, or in our case, running locally on our computer through the Ollama.

Due to rapid changes in software versions that may not always be 100% backward compatible, it is common to have situation that a package can only work with certain version of another package (known as dependency). As such, we need to be aware of such dependency constraint when installing packages. For example, based on the latest (Nov 2025) information, Jupyter AI will only work with Python version 3.12.

Python provides the concept of 'Environment' where Python code can be run in a self-contained space, with its own interpreter, libraries and installed packages. So we can install multiple versions of the same software on the same computer, but with each running in its own environment. As such, it is recommended that we create a new Environment when we install the Jupyter AI with the Jupyter Lab as described below.

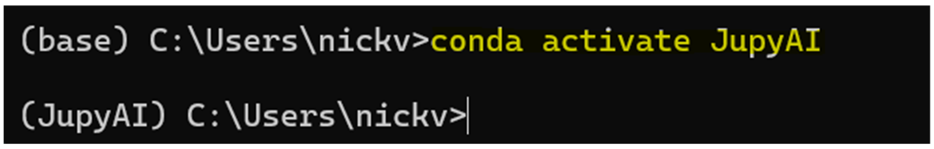
To begin with, we will create an Environment 'JupyAI' to install the Jupyter AI and Jupyter Lab (even though you had already installed the Jupyter Lab earlier. But that version will be running in a different environment – by default, the 'base') .

- Launch the Anaconda prompt to open a terminal (i.e. console). Check the version of Python that is already installed in our computer (and notice that it is in the environment "base"). We then create a new environment "JupyAI" and install it with Python 3.12.



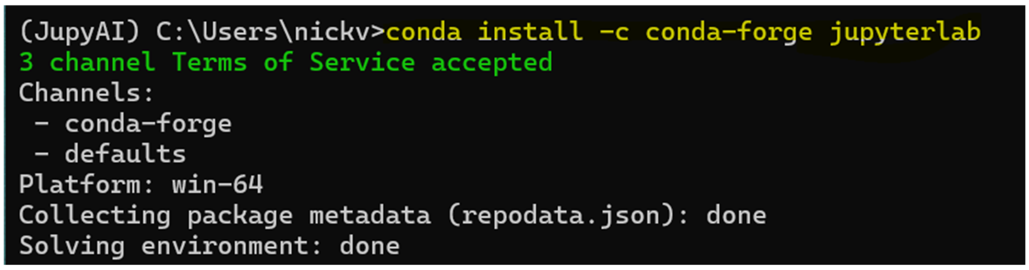
```
(base) C:\Users\nickv>python -V
Python 3.13.9

(base) C:\Users\nickv> conda create --name JupyAI python=3.12
```

- Follow the installation instruction, and once it is done, we can change to the new environment by activating it. Notice that we are now in the 'JupyAI' environment.



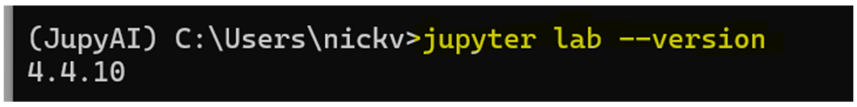
```
(base) C:\Users\nickv>conda activate JupyAI

(JupyAI) C:\Users\nickv>
```

- We will next install the Jupyter lab in this JupyAI environment, using the conda install instruction and specifying the conda-forge channel.



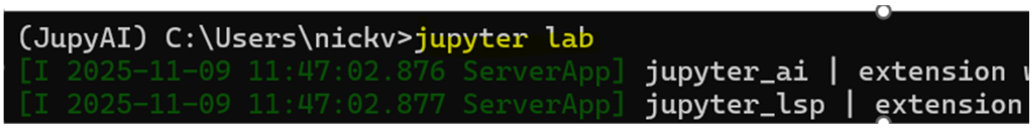
```
(JupyAI) C:\Users\nickv>conda install -c conda-forge jupyterlab
3 channel Terms of Service accepted
Channels:
 - conda-forge
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done
```

- Following the instructions and check the version that is installed



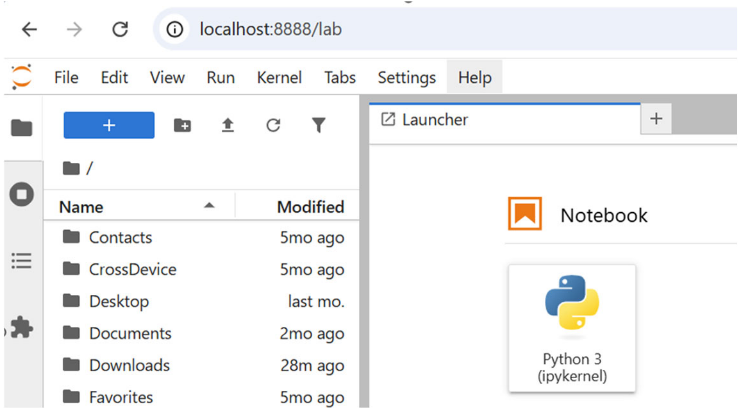
```
(JupyAI) C:\Users\nickv>jupyter lab --version
4.4.10
```

- Start the Jupyter Server program in this JupyAI environment



```
(JupyAI) C:\Users\nickv>jupyter lab
[I 2025-11-09 11:47:02.876 ServerApp] jupyter_ai | extension w
[I 2025-11-09 11:47:02.877 ServerApp] jupyter_lsp | extension
```

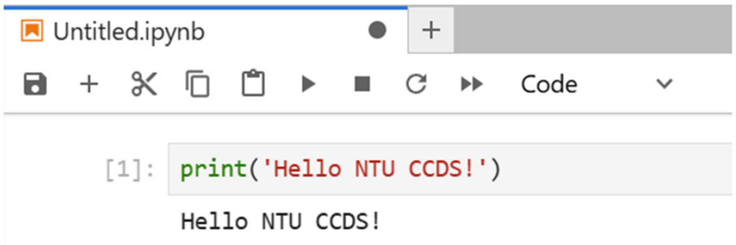
- You should see that your default browser is launched with a webpage with url at localhost:8888



localhost:8888/lab

Name	Modified
Contacts	5mo ago
CrossDevice	5mo ago
Desktop	last mo.
Documents	2mo ago
Downloads	28m ago
Favorites	5mo ago

- Select Python 3 Notebook and ken in a code to check that it is working as expected



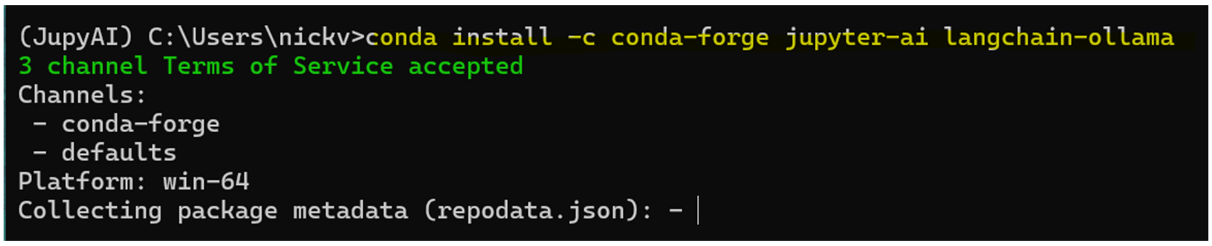
```
[1]: print('Hello NTU CCDS!')

Hello NTU CCDS!
```

- Terminate the Jupyter Lab server and close the browser (to get ready for installing and integration of Jupyter AI)

D. Installing Jupyter AI (in the new Environment)

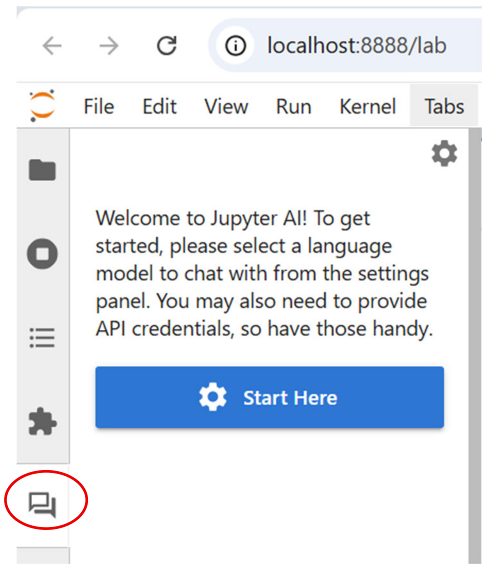
Here are the steps to install and integrate the Jupyter AI into the Jupyter Lab.

- Use conda install instruction to install the Jupyter AI with LangChain for Ollama



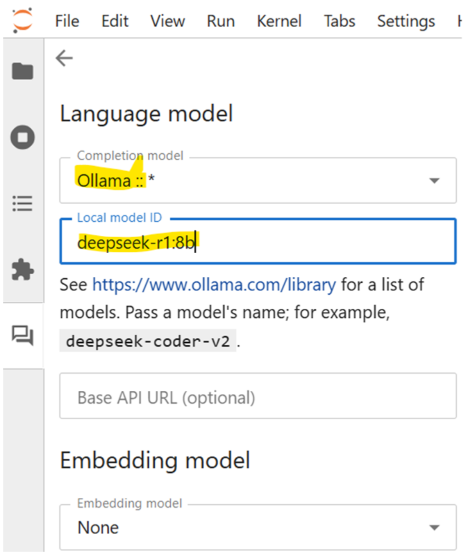
```
(JupyAI) C:\Users\nickv>conda install -c conda-forge jupyter-ai langchain-ollama
3 channel Terms of Service accepted
Channels:
 - conda-forge
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): - |
```

- Launch the Jupyter Lab and you should see the Jupyter AI Tab become available in the Tabs panel. Click the Start Here button and configure it with the elected LLM. In this case, Ollama and the earlier installed LLM deepseek-r1:8b.



localhost:8888/lab

Welcome to Jupyter AI! To get started, please select a language model to chat with from the settings panel. You may also need to provide API credentials, so have those handy.



Language model

Completion model: Ollama :: *

Local model ID: deepseek-r1:8b

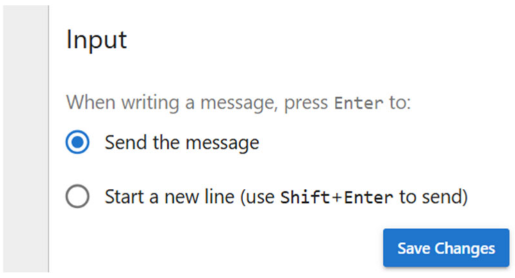
See https://www.ollama.com/library for a list of models. Pass a model's name; for example, `deepseek-coder-v2`.

Base API URL (optional)

Embedding model

Embedding model: None

- Click Save changes button.



Input

When writing a message, press `Enter` to:

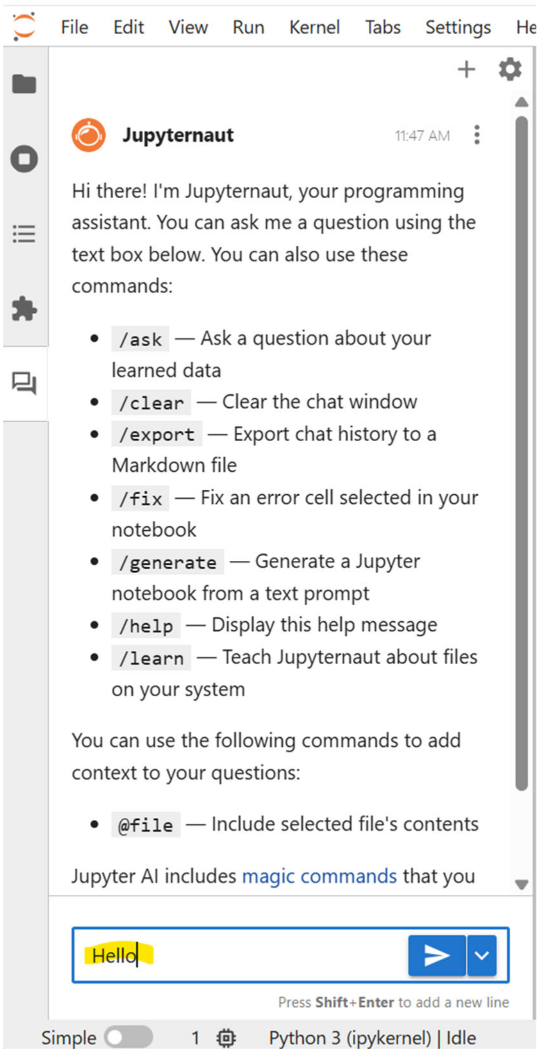
[✓ Send the message]

Start a new line (use `Shift+Enter` to send)

- Terminate the Jupyter Lab (server and browser).

Start a new session of Jupyter Lab server

- Now the Jupyter AI will display its interface (Jupyternaut) that allows you to key in messages (E.g. 'Hello').



Jupyternaut 11:47 AM

Hi there! I'm Jupyternaut, your programming assistant. You can ask me a question using the text box below. You can also use these commands:

- `/ask` — Ask a question about your learned data
- `/clear` — Clear the chat window
- `/export` — Export chat history to a Markdown file
- `/fix` — Fix an error cell selected in your notebook
- `/generate` — Generate a Jupyter notebook from a text prompt
- `/help` — Display this help message
- `/learn` — Teach Jupyternaut about files on your system

You can use the following commands to add context to your questions:

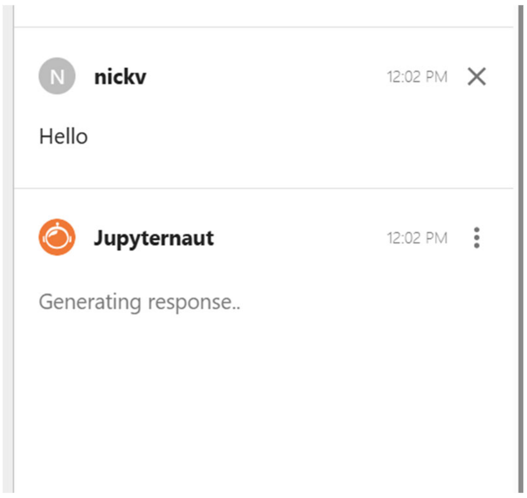
- `@file` — Include selected file's contents

Jupyter AI includes magic commands that you

Hello

Press **Shift+Enter** to add a new line

Simple 1 Python 3 (ipykernel) | Idle

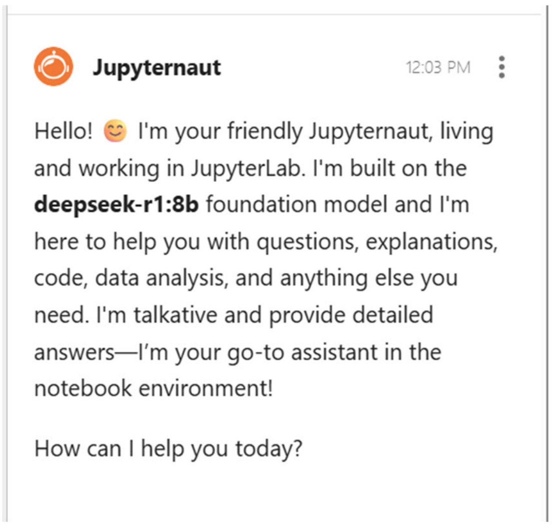


nickv 12:02 PM

Hello

Jupyternaut 12:02 PM

Generating response..

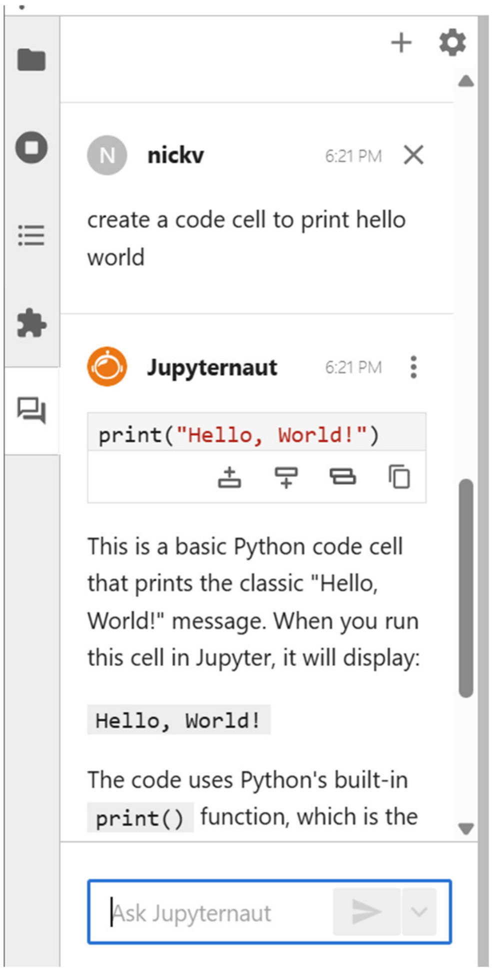


Jupyternaut 12:03 PM

Hello! 😊 I'm your friendly Jupyternaut, living and working in JupyterLab. I'm built on the **deepseek-r1:8b** foundation model and I'm here to help you with questions, explanations, code, data analysis, and anything else you need. I'm talkative and provide detailed answers—I'm your go-to assistant in the notebook environment!

How can I help you today?

- Another example – generating a python code: "create a code cell to print hello world"



nickv 6:21 PM

create a code cell to print hello world

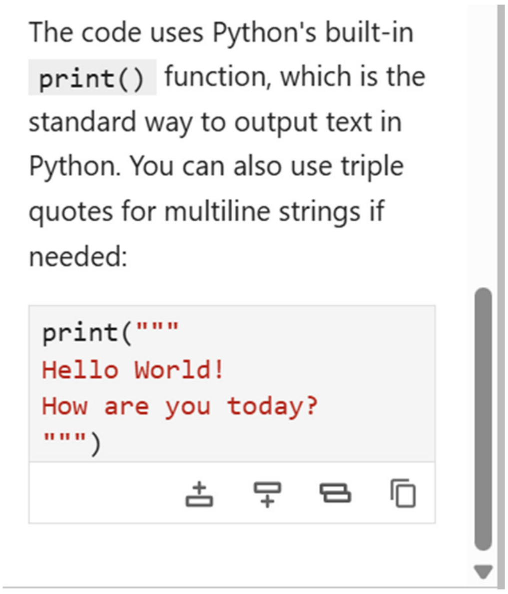
Jupyternaut 6:21 PM

```
print("Hello, World!")
```

This is a basic Python code cell that prints the classic "Hello, World!" message. When you run this cell in Jupyter, it will display:

```
Hello, World!
```

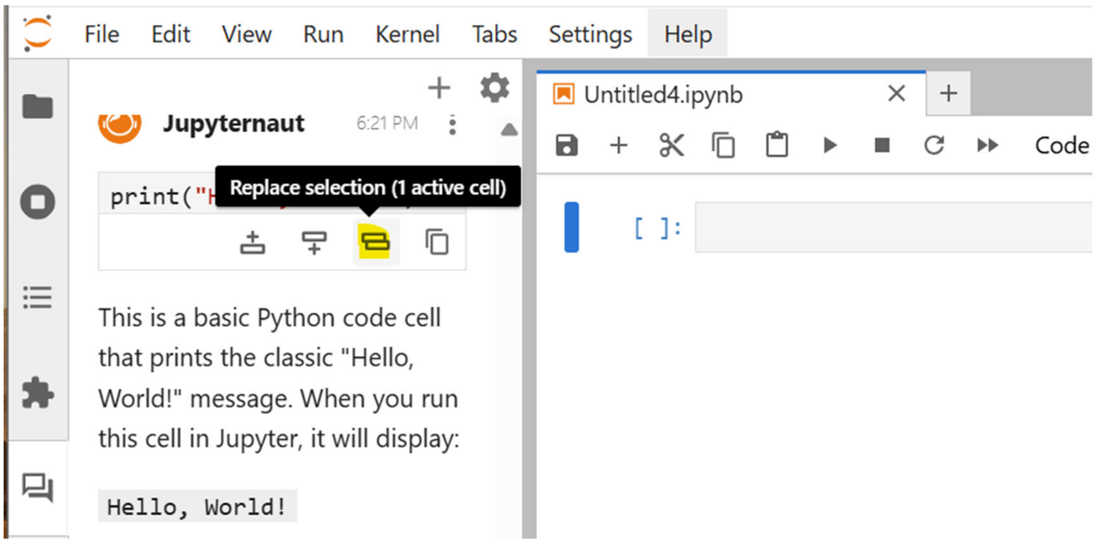
The code uses Python's built-in `print()` function, which is the



The code uses Python's built-in `print()` function, which is the standard way to output text in Python. You can also use triple quotes for multiline strings if needed:

```
print("""
Hello World!
How are you today?
""")
```

- The generated code can be directly copied to the active cell by clicking on the button "Replace selection"

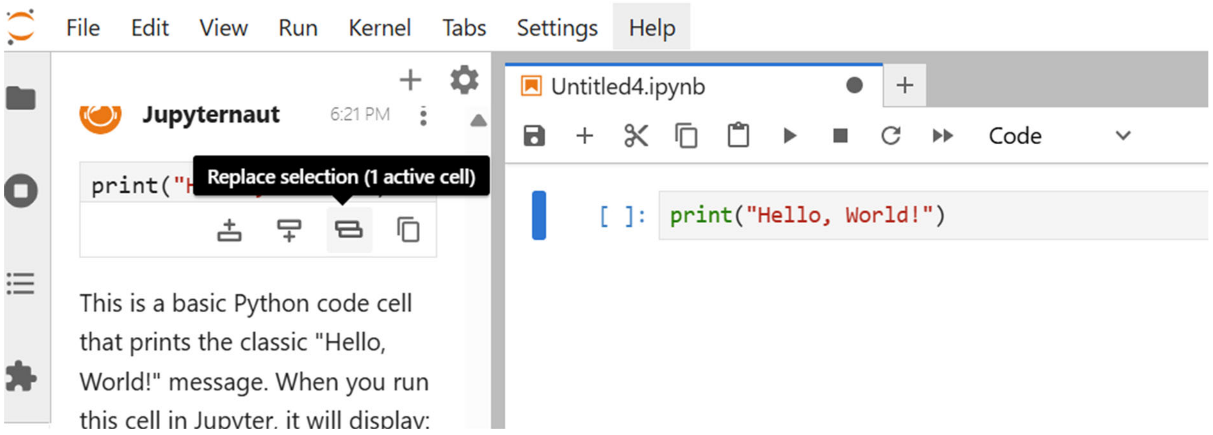


```
print("H
```

This is a basic Python code cell that prints the classic "Hello, World!" message. When you run this cell in Jupyter, it will display:

```
Hello, World!
```

-

- 

```
print("H
```

This is a basic Python code cell that prints the classic "Hello, World!" message. When you run this cell in Jupyter, it will display:

```
print("Hello, World!")
```