algoritma
data science education center

Efficient Information Extraction: Q&A and Summarization over PDF Documents using LLM

Pre-Workshop Guide

May 2024

Contents

Introduction

This guide is a resource for students at Algoritma to use in setting up their laptop or environment prior to the scheduled workshops. In this guide, students can find a list of prerequisites that will be consistently used throughout the entire course. These prerequisites are required to be **completed before** the start of the workshop.

For new students, we will run through the installation process to ensure that the necessary programming languages and tools - such as Python - are installed. The next section will then talk about methods on how to verify whether the installs were completed successfully.

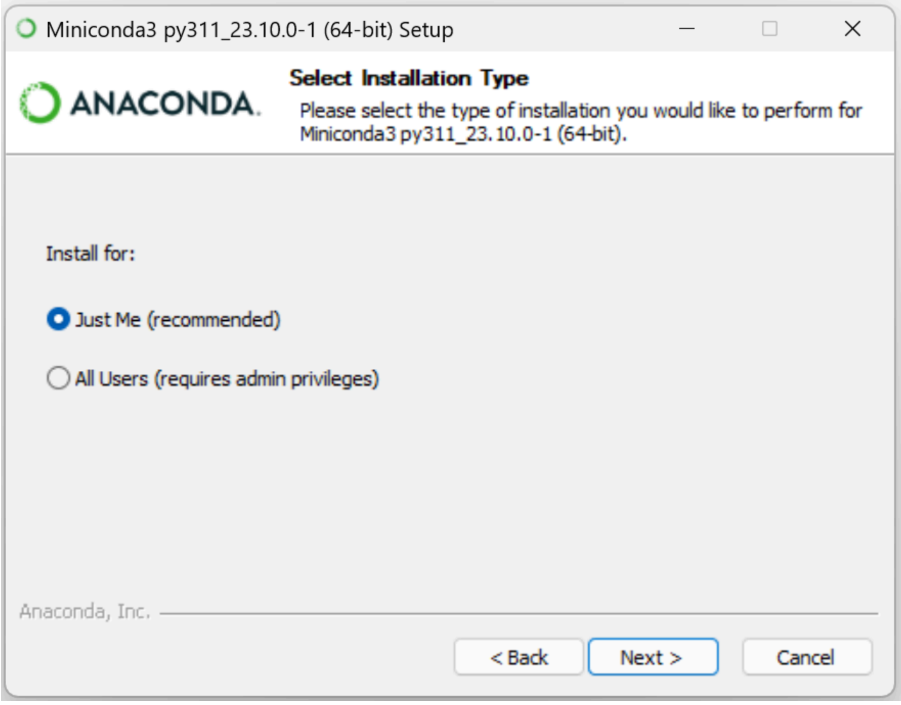
For recurring students, we recommend repeating the System Verification section once more to confirm past completed installations.

Installation

Installing Python using *Miniconda*

To install Python, we will use **Miniconda,** a minimal version of Anaconda that includes only *conda*, Python, and the essential packages they rely on. Miniconda provides not only Python but also the required packages (such as numpy and pandas) used in our workshops. Please follow the steps below to install Miniconda:

1. Open your web browser and **go to the official Miniconda download page** using this link*:* https://docs.conda.io/projects/miniconda/en/latest/.
2. On the download page, you'll find options for Windows, macOS, and Linux. **Select the appropriate download link that matches your operating system.**
3. Once the download is complete, launch the Miniconda installer file.
4. Follow the installation instructions.
5. When you're on the Select Installation Type menu, select **Just Me (recommended).**



Miniconda3 py311_23.10.0-1 (64-bit) Setup

Select Installation Type

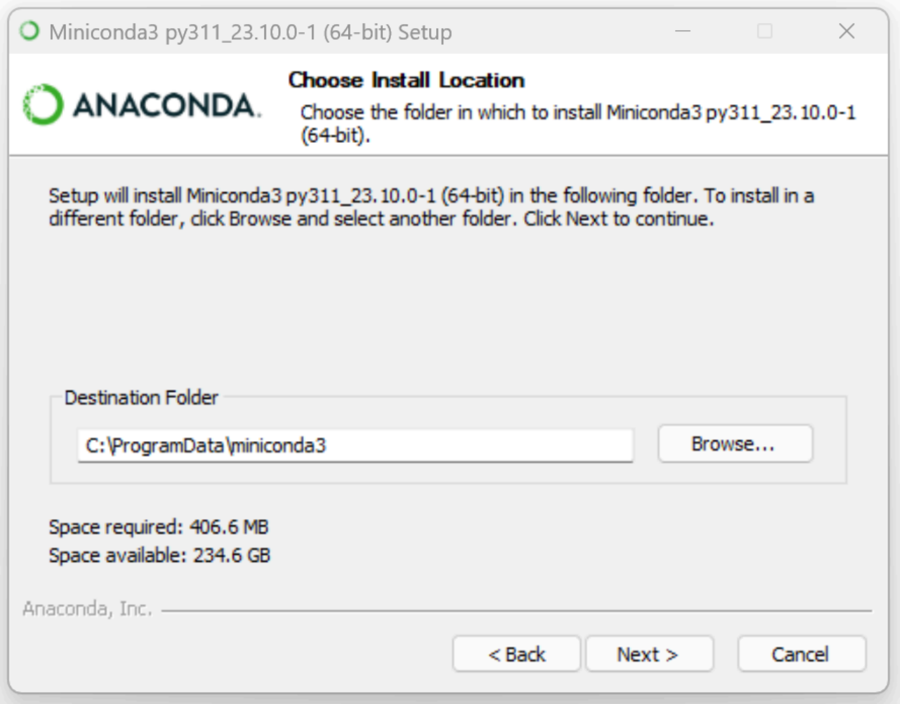
Please select the type of installation you would like to perform for Miniconda3 py311_23.10.0-1 (64-bit).

Install for:

● Just Me (recommended)

○ All Users (requires admin privileges)

6. Choose the installation location. It's **recommended to accept the default location.**



Miniconda3 py311_23.10.0-1 (64-bit) Setup

Choose Install Location

Choose the folder in which to install Miniconda3 py311_23.10.0-1 (64-bit).

Setup will install Miniconda3 py311_23.10.0-1 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder

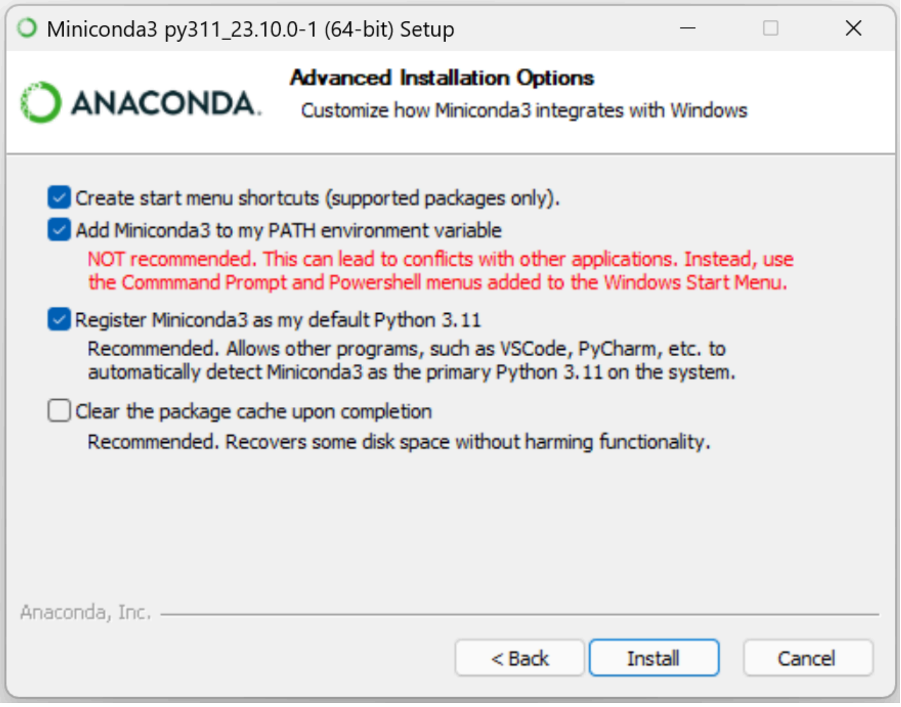
C:\ProgramData\miniconda3

Space required: 406.6 MB

Space available: 234.6 GB

Anaconda, Inc.

7. ⚠️**IMPORTANT** : **Select the checkbox to add Miniconda to my PATH environment variable**. This makes it easier to use Miniconda in other applications.



Miniconda3 py311_23.10.0-1 (64-bit) Setup

Advanced Installation Options

Customize how Miniconda3 integrates with Windows

☑ Create start menu shortcuts (supported packages only).

☑ Add Miniconda3 to my PATH environment variable

NOT recommended. This can lead to conflicts with other applications. Instead, use the Commmand Prompt and Powershell menus added to the Windows Start Menu.

☑ Register Miniconda3 as my default Python 3.11

Recommended. Allows other programs, such as VSCode, PyCharm, etc. to automatically detect Miniconda3 as the primary Python 3.11 on the system.

☐ Clear the package cache upon completion

Recommended. Recovers some disk space without harming functionality.

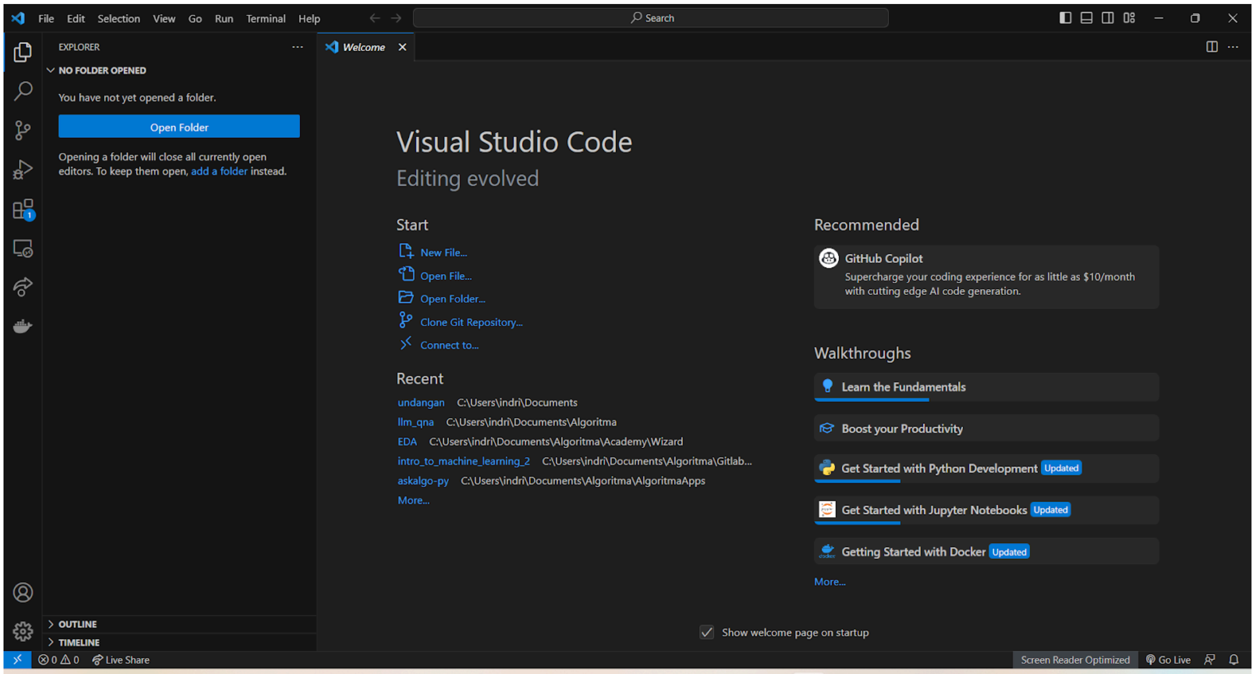
Anaconda, Inc.

8. Press the **Install** button to start the installation.

Installing Visual Studio Code

To set up a code editor, we’ll be using Microsoft’s Visual Studio Code, often referred to as **VS Code.** This robust and versatile code editor supports multiple programming languages and offers a wide range of extensions to enhance its functionality. Follow the steps below to install VS Code:

1. Open your web browser and **go to the official Visual Studio Code download page** using this link: https://code.visualstudio.com/download
2. On the download page, you'll find options for Windows, macOS, and Linux. **Select the appropriate download link that matches your operating system.**
3. **Once the installation is complete, launch VS Code.** You'll be greeted with a welcome screen like the picture below.



You have not yet opened a folder.

Opening a folder will close all currently open editors. To keep them open, add a folder instead.

Visual Studio Code

Editing evolved

undangan C:\Users\indri\Documents
llm_qna C:\Users\indri\Documents\Algoritma
EDA C:\Users\indri\Documents\Algoritma\Academy\Wizard
intro_to_machine_learning_2 C:\Users\indri\Documents\Algoritma\Gitlab...
askalgo-py C:\Users\indri\Documents\Algoritma\AlgoritmaApps

Supercharge your coding experience for as little as $10/month with cutting edge AI code generation.

☑ Show welcome page on startup

Figure 1: Visual Studio Code’s Welcome Page

Installing Python & Jupyter Extension in VS Code

To set up VS Code with Miniconda, we'll require the ‘Python’ and ‘Jupyter’ extensions in VS Code. The 'Python' extension is necessary for managing Python environments and providing code assistance. Moreover, the 'Jupyter' extension is essential for editing, running, and interacting with Jupyter notebooks within VS Code. Follow the steps below to install VS Code:

1. Open VSCode and **navigate to the ‘Extension’ menu on the sidebar** (*highlighted in green boxes in the image below*)**.** Alternatively, you can also open this menu using the shortcut **Ctrl+Shift+X** on Windows or **Cmd+Option+X** on macOS.

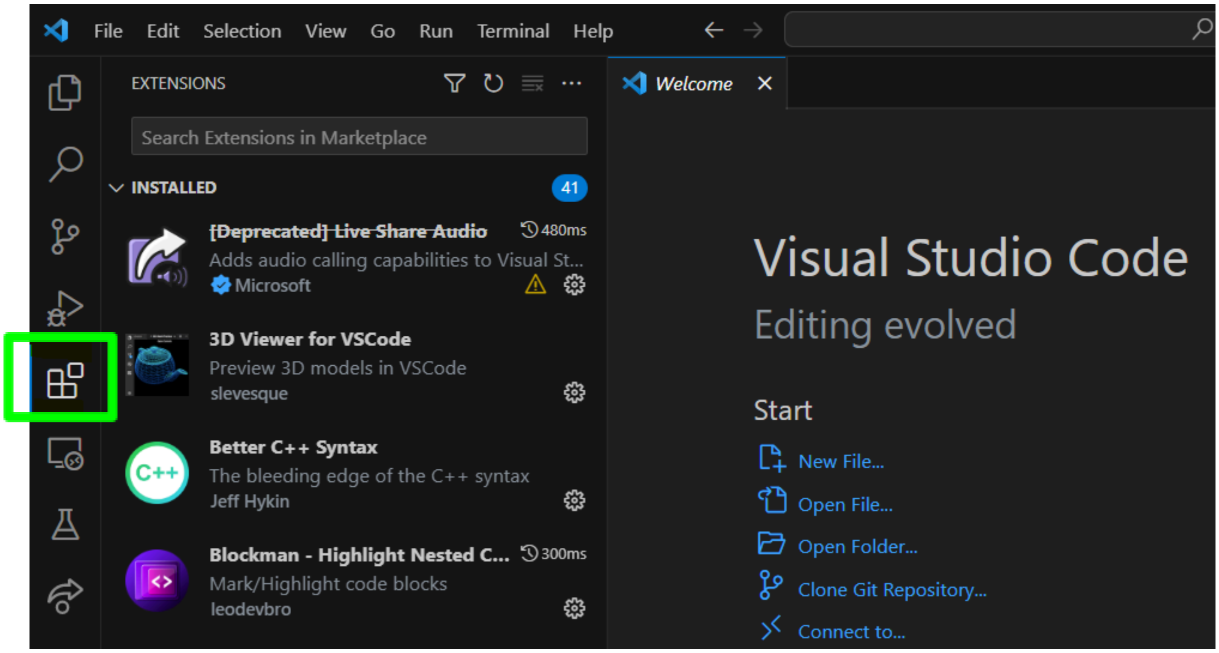


Figure 2: VS Code Extensions on the Activity Bar

Python Extension Installation

2. **Search ‘Python’ on Extension** search bar. Then, press **Install.**

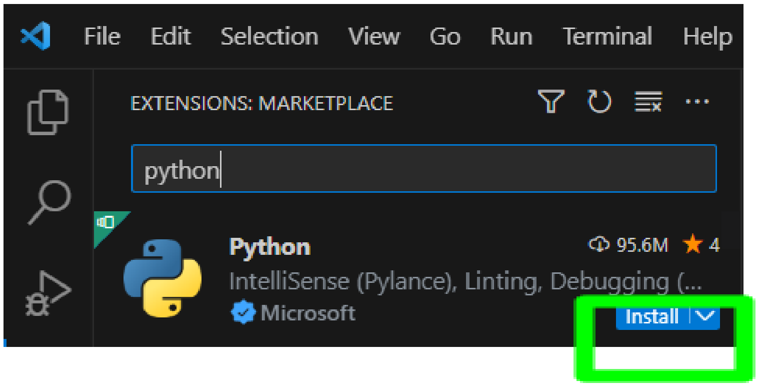
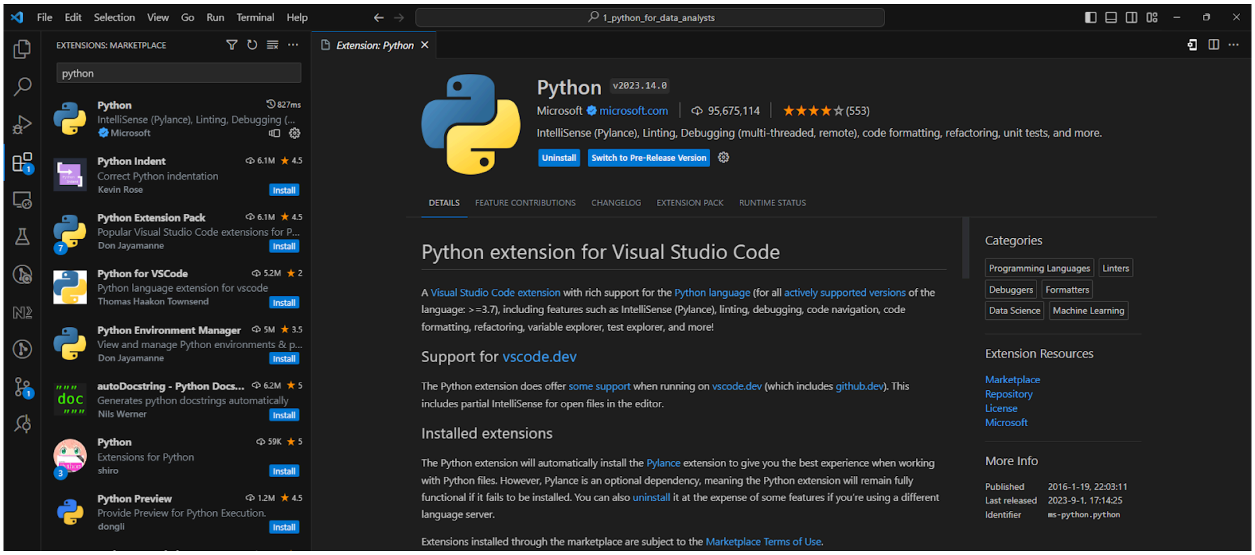


Figure 3: Searching for `Python` Extension in VS Code Extensions

3. **Once the 'Python' extension installation is complete,** the page view is like picture below.



IntelliSense (Pylance), Linting, Debugging (multi-threaded, remote), code formatting, refactoring, unit tests, and more.

Python extension for Visual Studio Code

A Visual Studio Code extension with rich support for the Python language (for all actively supported versions of the language: >=3.7), including features such as IntelliSense (Pylance), linting, debugging, code navigation, code formatting, refactoring, variable explorer, test explorer, and more!

Support for vscode.dev

The Python extension does offer some support when running on vscode.dev (which includes github.dev). This includes partial IntelliSense for open files in the editor.

Installed extensions

The Python extension will automatically install the Pylance extension to give you the best experience when working with Python files. However, Pylance is an optional dependency, meaning the Python extension will remain fully functional if it fails to be installed. You can also uninstall it at the expense of some features if you're using a different language server.

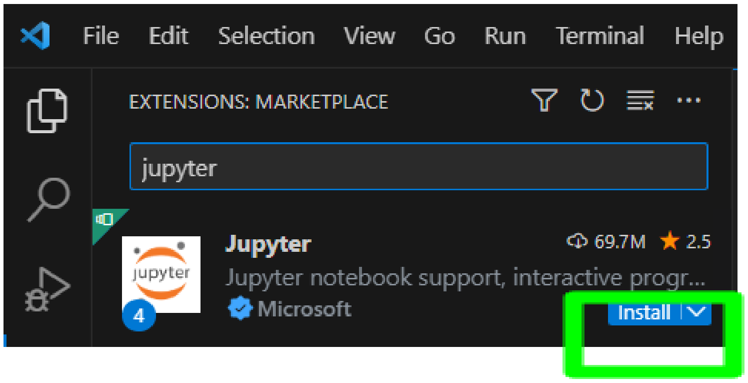
Extensions installed through the marketplace are subject to the Marketplace Terms of Use.

Published	2016-1-19, 22:03:11
Last released	2023-9-1, 17:14:25
Identifier	ms-python.python

Figure 4: `Python` Extension Installed in VS Code Extensions

Jupyter Extension Installation

4. **Search 'Jupyter' on Extension** search bar. Then, click **Install.**

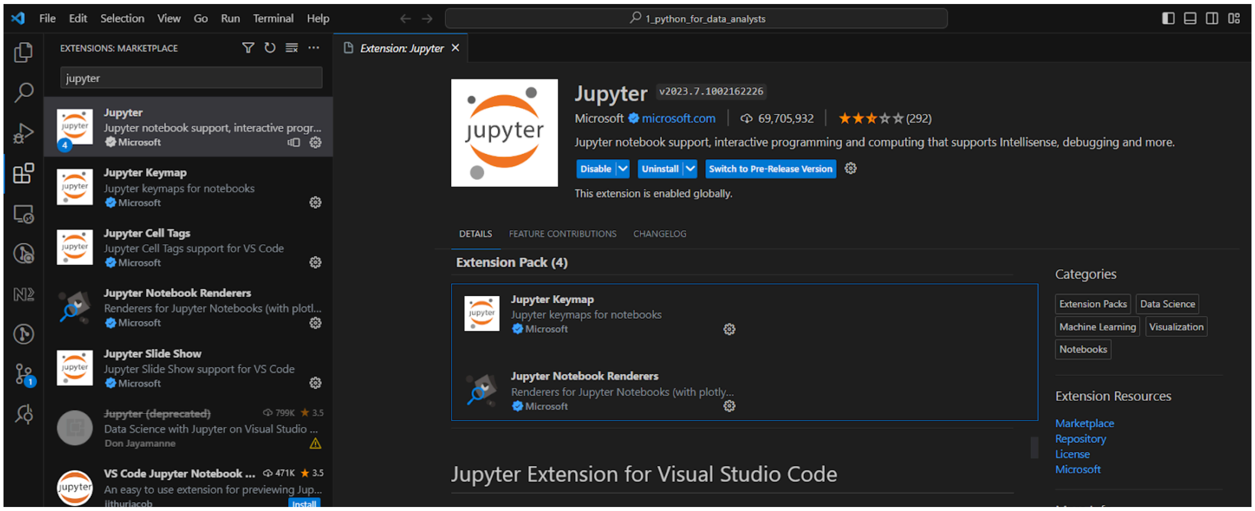


jupyter

Jupyter notebook support, interactive progr...

Figure 5: Searching for `Jupyter` Extension in VS Code Extensions

5. **Once the ‘Jupyter’ extension installation is complete,** the page view is like picture below.



jupyter

Jupyter notebook support, interactive progr...

Jupyter keymaps for notebooks

Jupyter Cell Tags support for VS Code

Renderers for Jupyter Notebooks (with plotl...

Jupyter Slide Show support for VS Code

Data Science with Jupyter on Visual Studio ...

An easy to use extension for previewing Jup...

Jupyter notebook support, interactive programming and computing that supports Intellisense, debugging and more.

This extension is enabled globally.

Jupyter keymaps for notebooks

Renderers for Jupyter Notebooks (with plotly...

Jupyter Extension for Visual Studio Code

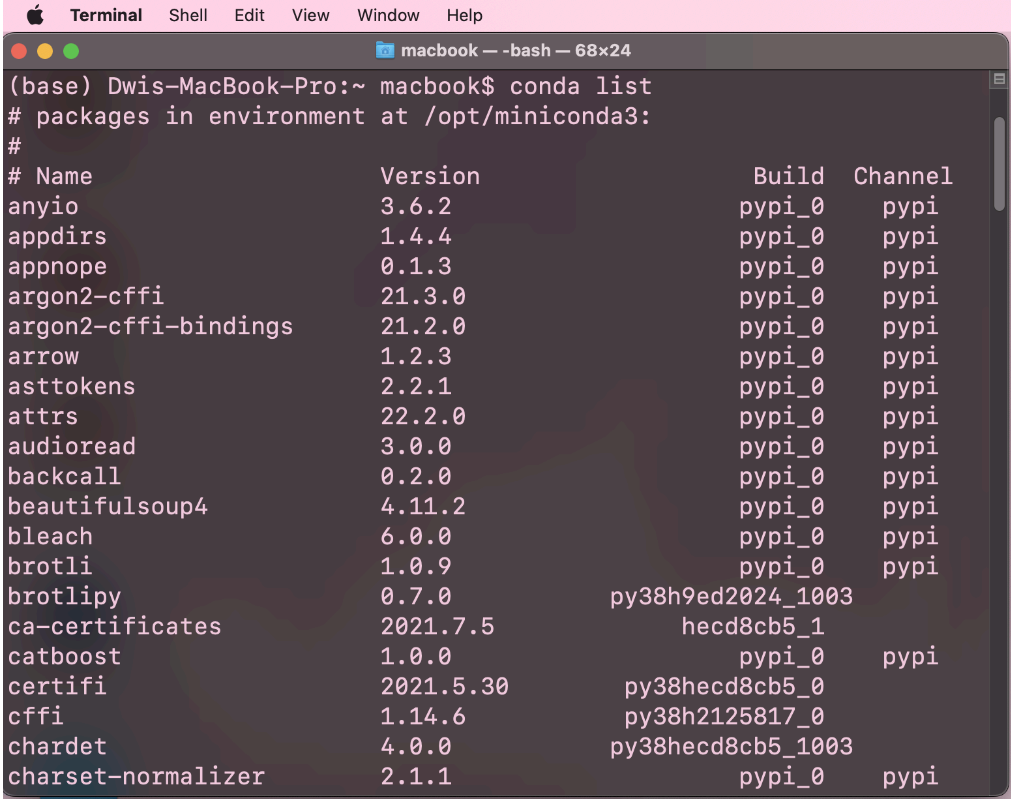
Figure 6: `Jupyter` Extension Installed in VS Code Extensions

System Verification

- **For Mac OS X and Linux-based OS:** Open "*Terminal*"
- **For Windows:** Open "*Anaconda Prompt (miniconda3)*"

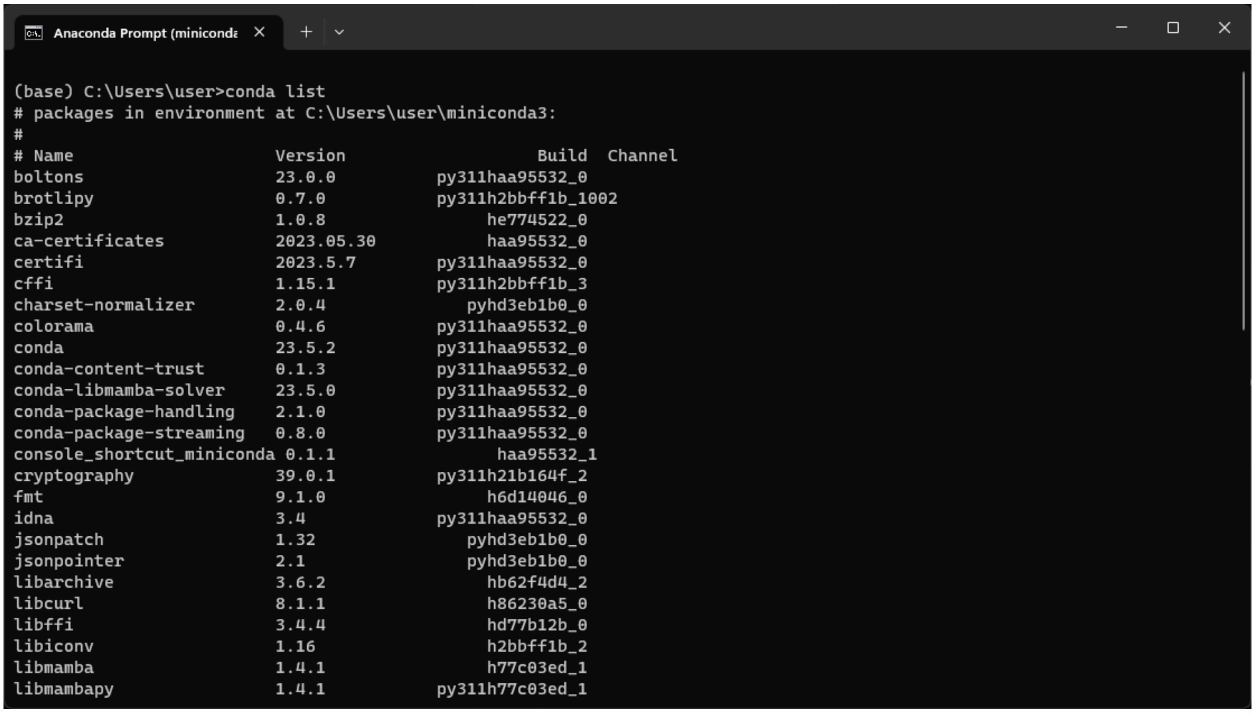
Verify Anaconda Installation

1. Type the command `conda list` in your *"Terminal"* or *"Anaconda Prompt (miniconda3)"*.
2. If the installation was completed successfully, your terminal will give a response of list of packages like the example below.
3. If your terminal does not give any response, please check the installation section's Warning, if the problem still persists, kindly reach out for further assistance via email at mentor@algorit.ma.



```
(base) Dwis-MacBook-Pro:~ macbook$ conda list
# packages in environment at /opt/miniconda3:
#
# Name                    Version                   Build  Channel
anyio                     3.6.2                    pypi_0    pypi
appdirs                   1.4.4                    pypi_0    pypi
appnope                   0.1.3                    pypi_0    pypi
argon2-cffi               21.3.0                   pypi_0    pypi
argon2-cffi-bindings      21.2.0                   pypi_0    pypi
arrow                     1.2.3                    pypi_0    pypi
asttokens                 2.2.1                    pypi_0    pypi
attrs                     22.2.0                   pypi_0    pypi
audioread                 3.0.0                    pypi_0    pypi
backcall                  0.2.0                    pypi_0    pypi
beautifulsoup4            4.11.2                   pypi_0    pypi
bleach                    6.0.0                    pypi_0    pypi
brotli                    1.0.9                    pypi_0    pypi
brotlipy                  0.7.0           py38h9ed2024_1003
ca-certificates           2021.7.5             hecd8cb5_1
catboost                  1.0.0                    pypi_0    pypi
certifi                   2021.5.30        py38hecd8cb5_0
cffi                      1.14.6           py38h2125817_0
chardet                   4.0.0           py38hecd8cb5_1003
charset-normalizer        2.1.1                    pypi_0    pypi
```

Figure 7: `conda list` Response on Mac OS X Terminal

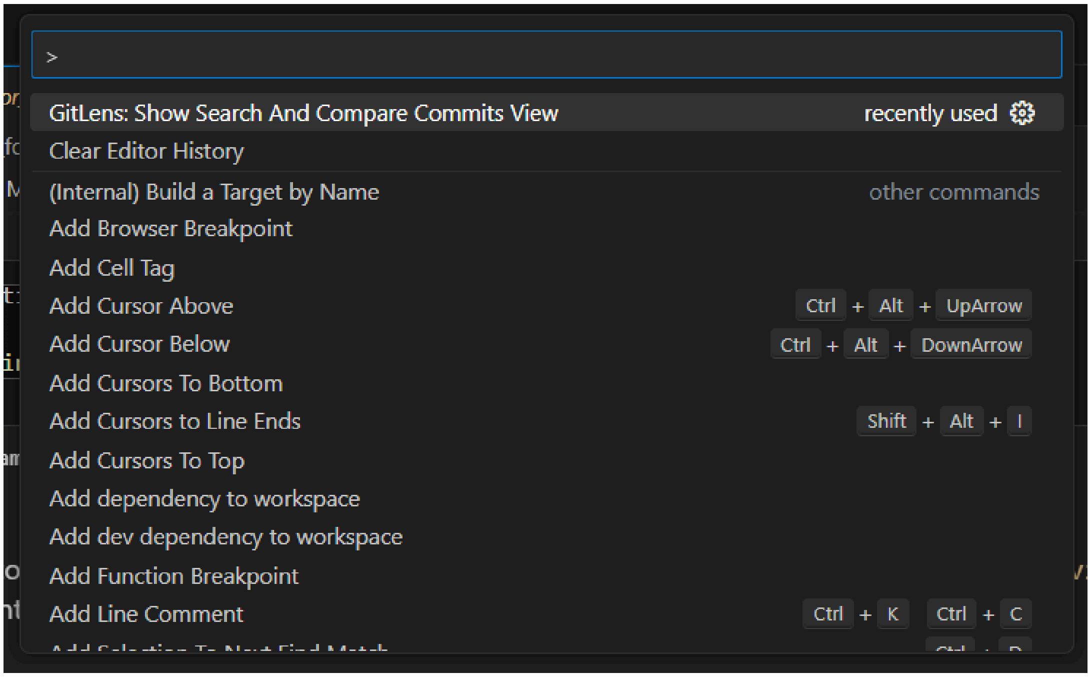


```
(base) C:\Users\user>conda list
# packages in environment at C:\Users\user\miniconda3:
#
# Name                    Version                   Build  Channel
boltons                   23.0.0          py311haa95532_0
brotlipy                  0.7.0           py311h2bbff1b_1002
bzip2                     1.0.8                he774522_0
ca-certificates           2023.05.30           haa95532_0
certifi                   2023.5.7        py311haa95532_0
cffi                      1.15.1          py311h2bbff1b_3
charset-normalizer        2.0.4              pyhd3eb1b0_0
colorama                  0.4.6           py311haa95532_0
conda                     23.5.2          py311haa95532_0
conda-content-trust       0.1.3           py311haa95532_0
conda-libmamba-solver     23.5.0          py311haa95532_0
conda-package-handling    2.1.0           py311haa95532_0
conda-package-streaming   0.8.0           py311haa95532_0
console_shortcut_miniconda 0.1.1                haa95532_1
cryptography              39.0.1          py311h21b164f_2
fmt                       9.1.0                h6d14046_0
idna                      3.4             py311haa95532_0
jsonpatch                 1.32               pyhd3eb1b0_0
jsonpointer               2.1                pyhd3eb1b0_0
libarchive                3.6.2                hb62f4d4_2
libcurl                   8.1.1                h86230a5_0
libffi                    3.4.4                hd77b12b_0
libiconv                  1.16                 h2bbff1b_2
libmamba                  1.4.1                h77c03ed_1
libmambapy                1.4.1           py311h77c03ed_1
```

Figure 8: `conda list` Response on Anconda Prompt (miniconda3)

Verify 'Python' and 'Jupyter' Extension on VS Code Installation

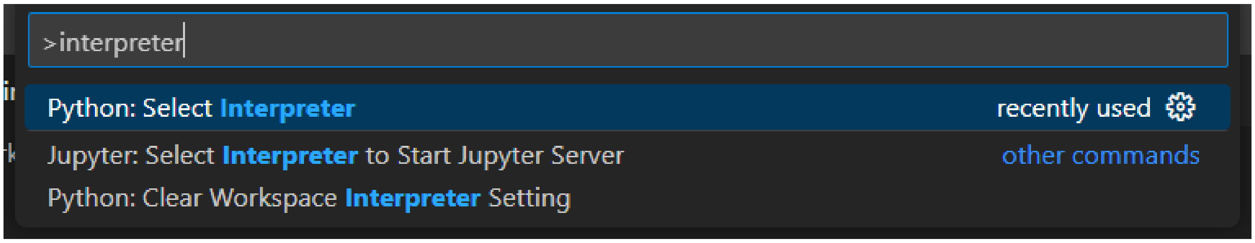
1. Open VS Code and press **Ctrl+Shift+P** on Windows or **Cmd+Option+P** on MacOS until a modal, like the one pictured below, appears. This modal is called the Command Palette, and from here, we can access all the functionality of VS Code.



```
>
GitLens: Show Search And Compare Commits View        recently used
Clear Editor History
(Internal) Build a Target by Name                     other commands
Add Browser Breakpoint
Add Cell Tag
Add Cursor Above                                      Ctrl + Alt + UpArrow
Add Cursor Below                                      Ctrl + Alt + DownArrow
Add Cursors To Bottom
Add Cursors to Line Ends                              Shift + Alt + I
Add Cursors To Top
Add dependency to workspace
Add dev dependency to workspace
Add Function Breakpoint
Add Line Comment                                      Ctrl + K  Ctrl + C
```

Figure 9: Opening the Command Palette in VS Code

2. Type 'interpreter' to the search bar.



```
>interpreter
Python: Select Interpreter                                 recently used
Jupyter: Select Interpreter to Start Jupyter Server        other commands
Python: Clear Workspace Interpreter Setting
```

Figure 10: Typing 'Interpreter' to Find the 'Python: Select Interpreter' Menu

3. Click on "Python: Select Interpreter" then you can select the **base** environment (~\miniconda3\python.exe).

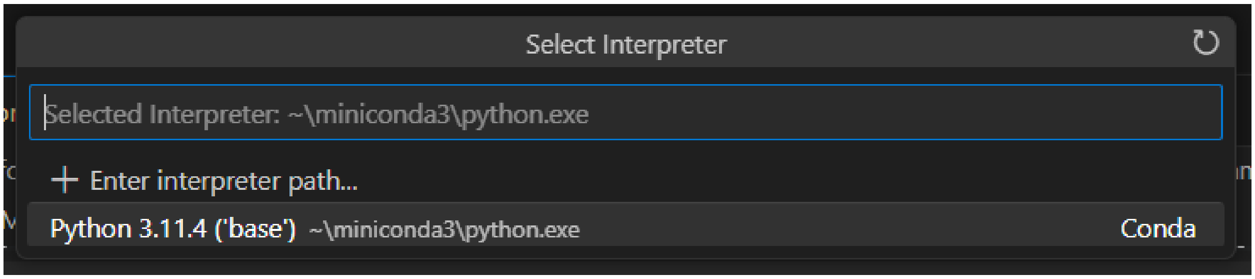
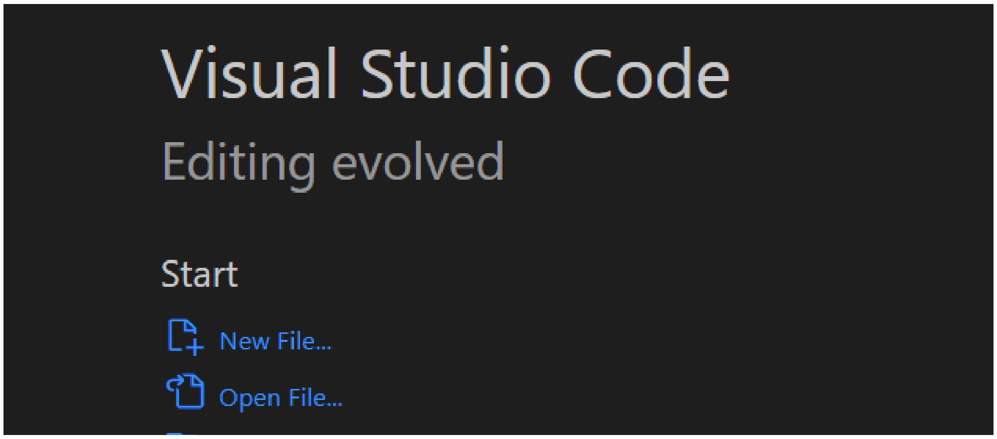

Select Interpreter
Selected Interpreter: ~\miniconda3\python.exe
+ Enter interpreter path...
Python 3.11.4 ('base') ~\miniconda3\python.exe
Conda

Figure 11: Selecting 'Python 3 (`base`) ~\miniconda\python.exe'

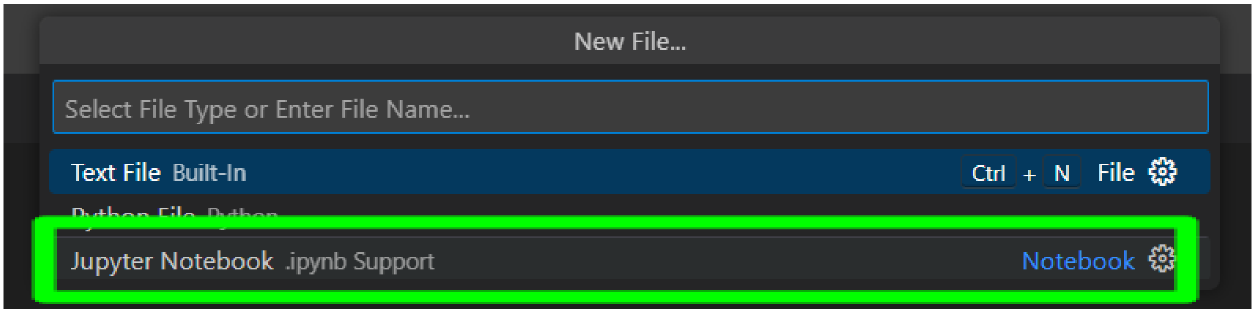
Alternative Kernel Selection

Instead of manually using Ctrl+Shift+P or Cmd+Shift+P in VSCode, you can select the kernel by clicking on "Select Kernel" in the upper right-hand corner of your notebook.

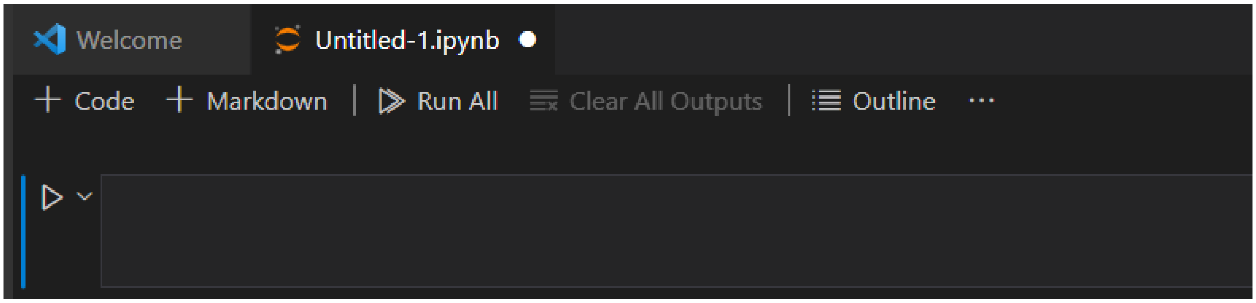
1. Create **a new notebook** by clicking the **'New File'** button.



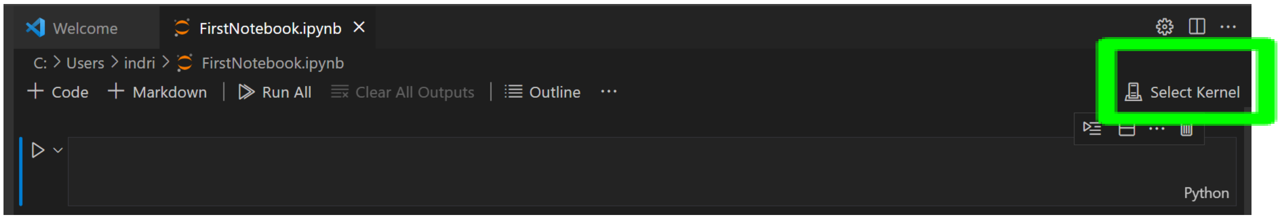
2. A modal will appear, prompting you to choose a file type. **Select the 'Jupyter Notebook .ipynb Support' file.**


New File...
Select File Type or Enter File Name...
Text File Built-In
Ctrl + N
File
Python File Python
Jupyter Notebook .ipynb Support
Notebook

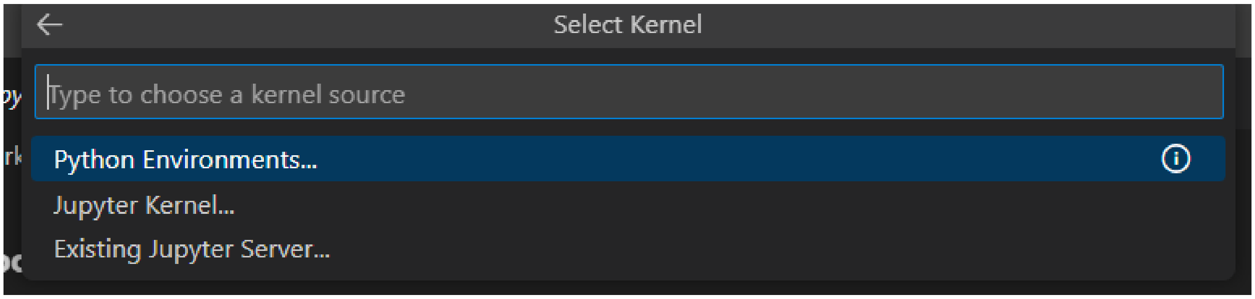
3. Afterward, an 'Untitled.ipynb' file will appear as shown below. Save it as **'FirstNotebook.ipynb.'"**



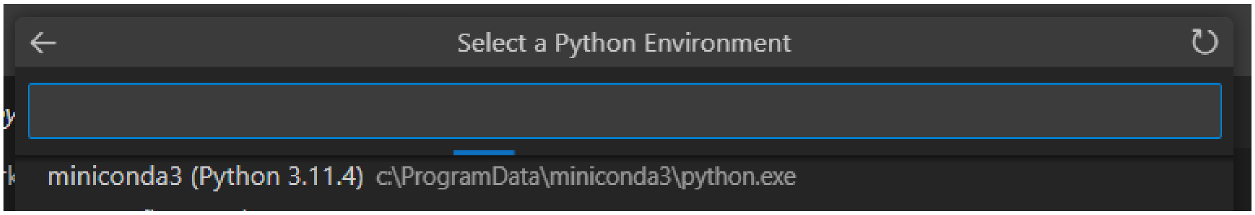
4. Look for the Select Kernel option on the upper right-hand corner of your notebook interface. It's typically a dropdown menu or a button with a kernel icon.



5. Click on it to see a list of available kernels. Kernels are associated with specific programming languages or environments, so choose the one that matches your requirements.
   a. Select "**Python Environments**"



b. After installing Miniconda for the first time, you can select the **'miniconda3'** kernel, which is the default Python 3.X.X kernel provided by Miniconda.



Select a Python Environment

miniconda3 (Python 3.11.4) c:\ProgramData\miniconda3\python.exe

6. Once you've selected the desired kernel, your notebook will start using that kernel for code execution.

This makes it easy to switch between different programming languages or environments within a Jupyter Notebook without having to remember keyboard shortcuts.

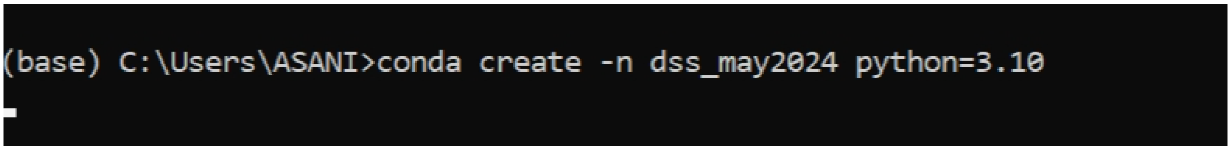
Virtual Environment and Package Preparation *

This step is required to prepare our environment throughout the course and prevent any conflicting dependencies. Make sure you have downloaded **requirements.txt** file attached alongside this handbook in the Google Classroom post.

1. Search for **Anaconda Prompt (miniconda3)** application for Windows or **Terminal** for Mac.
2. Create a new virtual environment named **dss_may2024** with Python version 3.10 using the following syntax. Press **Enter.**

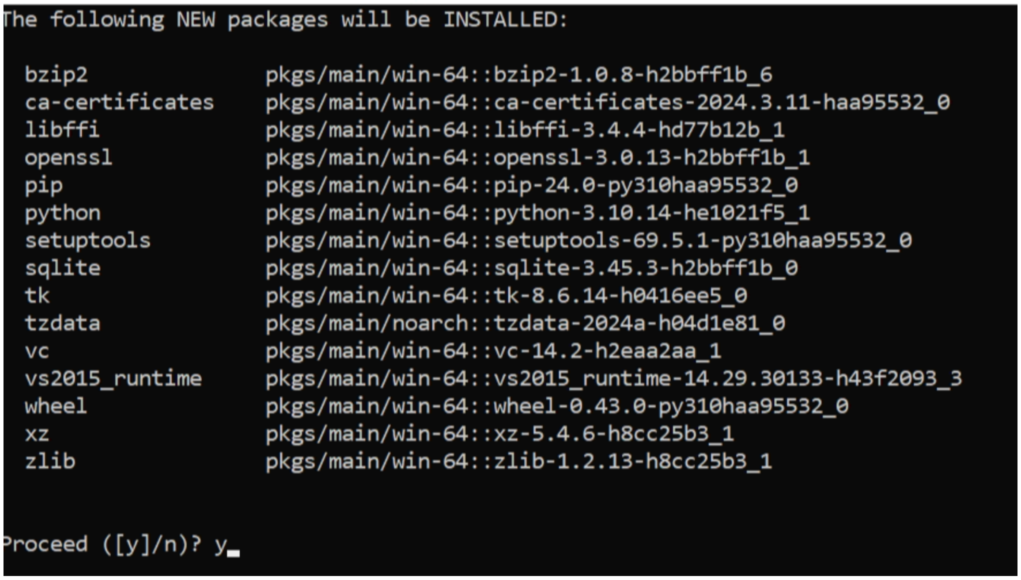
Unset

```
conda create -n dss_may2024 python=3.10
```



```
(base) C:\Users\ASANI>conda create -n dss_may2024 python=3.10
```

3. During the process, you will be asked to proceed by typing either **y** or **n**. Type **y** and press **Enter**.

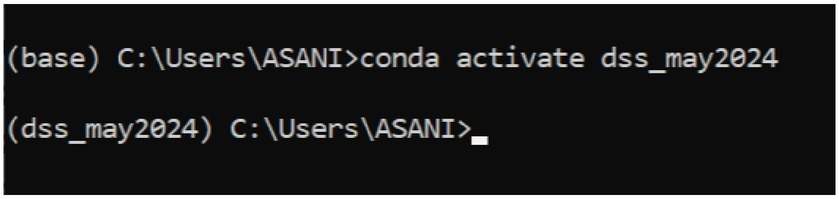


```
The following NEW packages will be INSTALLED:

  bzip2              pkgs/main/win-64::bzip2-1.0.8-h2bbff1b_6
  ca-certificates    pkgs/main/win-64::ca-certificates-2024.3.11-haa95532_0
  libffi             pkgs/main/win-64::libffi-3.4.4-hd77b12b_1
  openssl            pkgs/main/win-64::openssl-3.0.13-h2bbff1b_1
  pip                pkgs/main/win-64::pip-24.0-py310haa95532_0
  python             pkgs/main/win-64::python-3.10.14-he1021f5_1
  setuptools         pkgs/main/win-64::setuptools-69.5.1-py310haa95532_0
  sqlite             pkgs/main/win-64::sqlite-3.45.3-h2bbff1b_0
  tk                 pkgs/main/win-64::tk-8.6.14-h0416ee5_0
  tzdata             pkgs/main/noarch::tzdata-2024a-h04d1e81_0
  vc                 pkgs/main/win-64::vc-14.2-h2eaa2aa_1
  vs2015_runtime     pkgs/main/win-64::vs2015_runtime-14.29.30133-h43f2093_3
  wheel              pkgs/main/win-64::wheel-0.43.0-py310haa95532_0
  xz                 pkgs/main/win-64::xz-5.4.6-h8cc25b3_1
  zlib               pkgs/main/win-64::zlib-1.2.13-h8cc25b3_1


Proceed ([y]/n)? y
```

4. Wait until the environment creation is complete.
5. Activate the newly created environment, namely **dss_may2024.** Press **Enter.**

Unset

```
conda activate dss_may2024
```

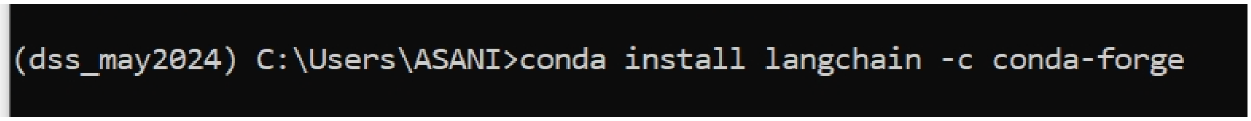


```
(base) C:\Users\ASANI>conda activate dss_may2024

(dss_may2024) C:\Users\ASANI>_
```

6. After activating your environment, install LangChain library by using the following command. Press **Enter.** If you are asked to proceed by typing either **y** or **n,** type **y** and press **Enter.** Wait until the installation is complete.

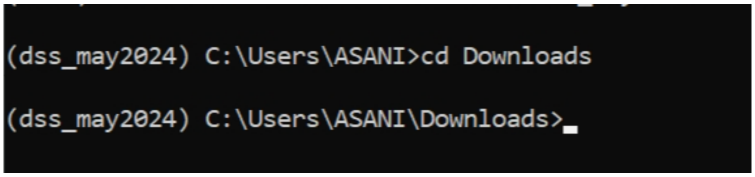
Unset

```
conda install langchain -c conda-forge
```



```
(dss_may2024) C:\Users\ASANI>conda install langchain -c conda-forge
```

7. Navigate to the directory where you saved **requirements.txt** file. We will use this file to install the remaining packages. For example, if you saved it in your **Downloads** directory, navigate to that directory using the **cd** command and press **Enter**.

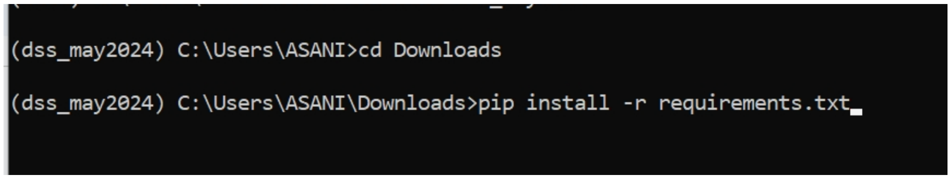


```
(dss_may2024) C:\Users\ASANI>cd Downloads

(dss_may2024) C:\Users\ASANI\Downloads>_
```

8. Install all of the remaining packages in **requirements.txt** using the following command. Wait until the packages installation process is complete and your environment is ready!

Unset

```
pip install -r requirements.txt
```



```
(dss_may2024) C:\Users\ASANI>cd Downloads

(dss_may2024) C:\Users\ASANI\Downloads>pip install -r requirements.txt
```

Here is the list of package in **requirements.txt:**

Unset

```
    python-dotenv
    huggingface-hub==0.22.2
    langchain-community
    langchain-chroma==0.1.0
    langchain-google-genai==1.0.3
    langchain-openai==0.1.6
    pypdf==4.2.0
    PyPDF2==3.0.1
    sentence-transformers==2.7.0
    rapidocr-onnxruntime
```

Alternatively to the package installation using **requirements.txt,** you can also install the package above one by one using the following command. For example, we want to langchain-chroma:

Unset

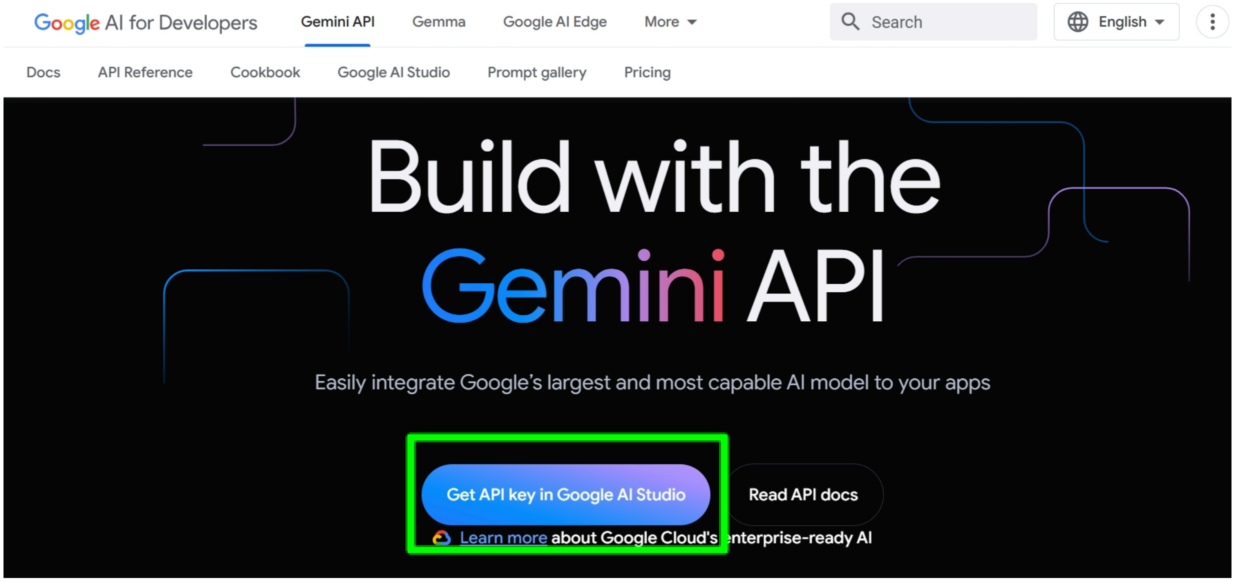
```
pip install langchain-chroma==0.10
```

* : will be discussed later in day 1 class

Gemini API Key Guideline

During our workshop, we will primarily utilize the LLM developed by Google, Gemini API. The usage of this Gemini API requires us to have an API key. To obtain your Gemini API key, follow these steps:

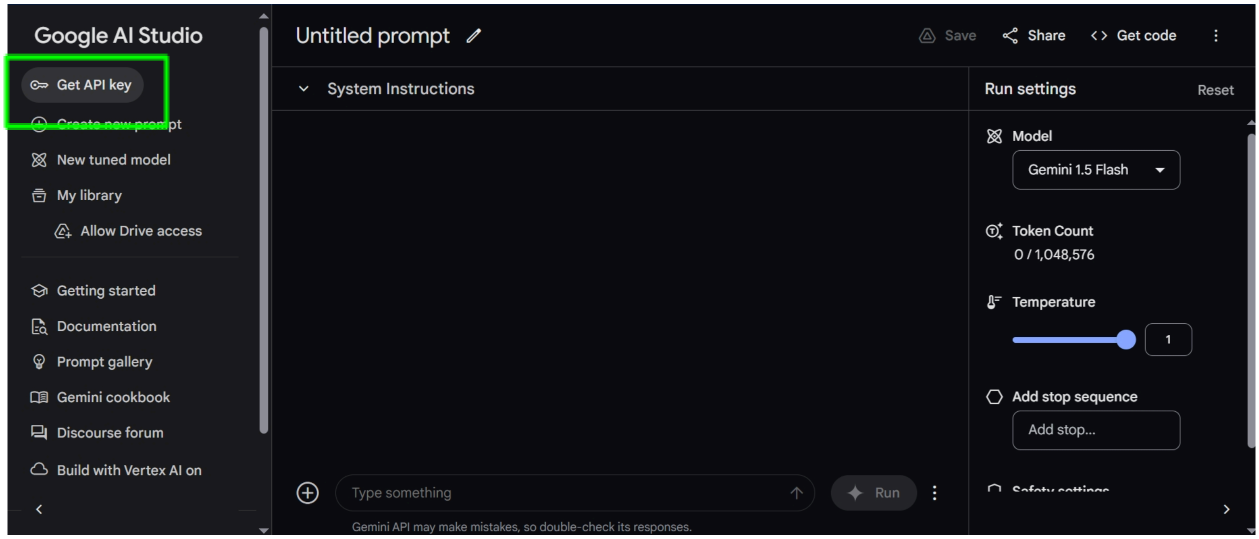
1. Open this link from your browser: https://ai.google.dev/gemini-api (make sure you have signed in to your Gmail account) .



Easily integrate Google's largest and most capable AI model to your apps

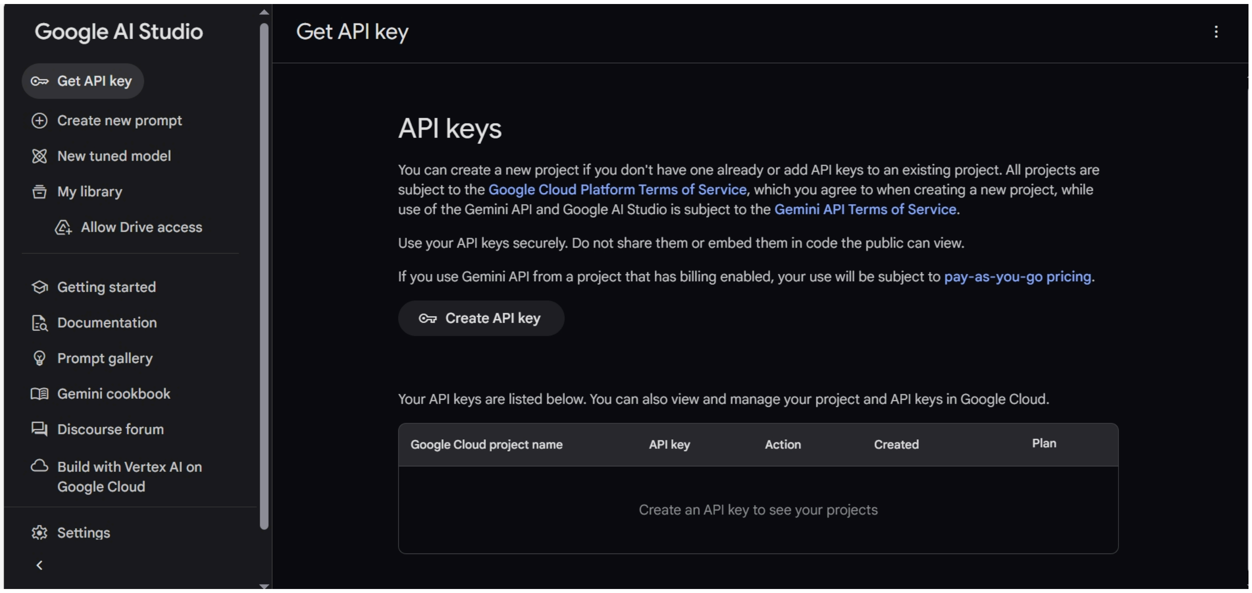
Learn more about Google Cloud's enterprise-ready AI

2. Choose **GetAPI key in Google AI Studio** menu and you will be directed to this page. Choose **Get API key** menu in the top left corner.



Gemini API may make mistakes, so double-check its responses.

3. Choose the menu **Create API Key** > **Create API Key in New Project** and your API key will be generated. Copy your API key and save it somewhere safe.



API keys

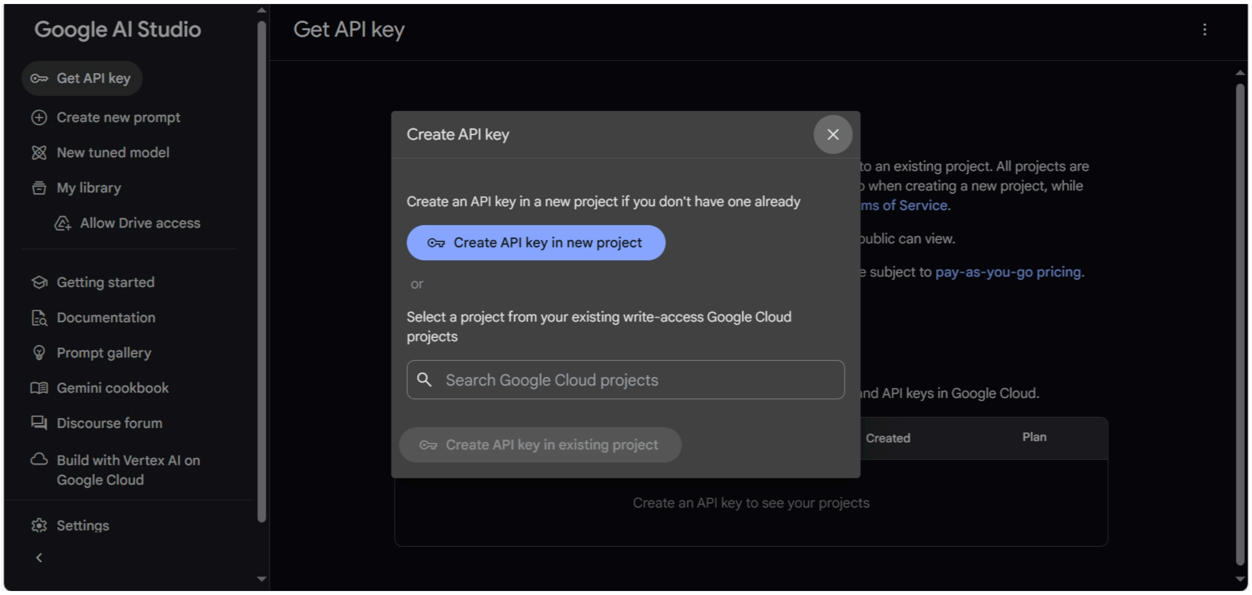
You can create a new project if you don't have one already or add API keys to an existing project. All projects are subject to the Google Cloud Platform Terms of Service, which you agree to when creating a new project, while use of the Gemini API and Google AI Studio is subject to the Gemini API Terms of Service.

Use your API keys securely. Do not share them or embed them in code the public can view.

If you use Gemini API from a project that has billing enabled, your use will be subject to pay-as-you-go pricing.

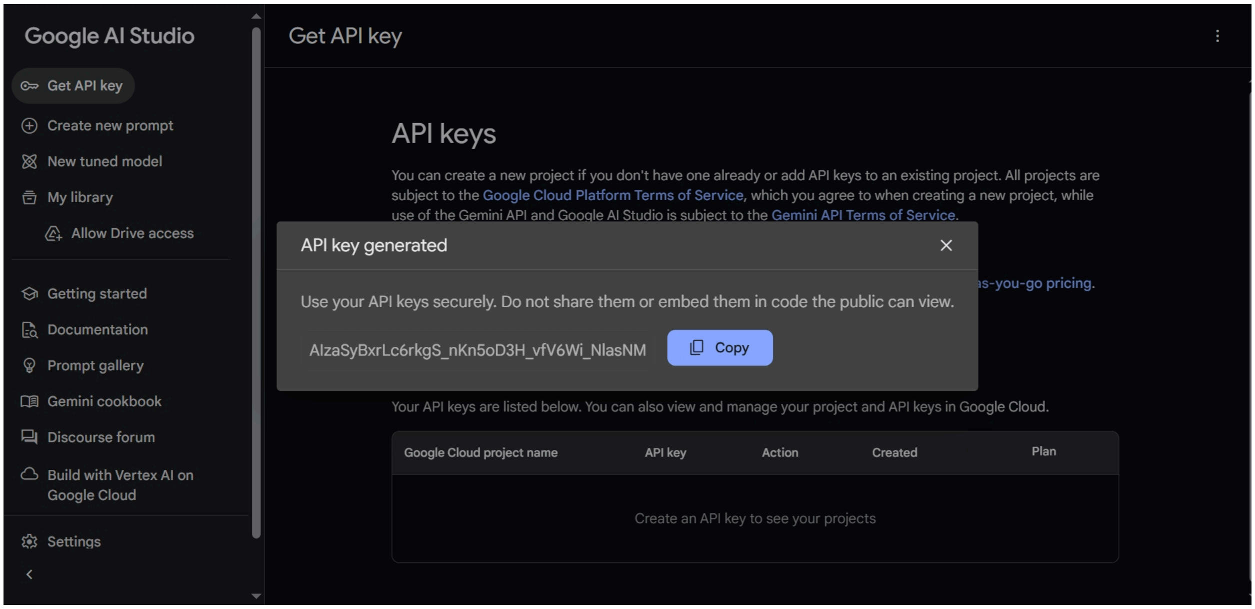
Your API keys are listed below. You can also view and manage your project and API keys in Google Cloud.

Google Cloud project name	API key	Action	Created	Plan
Create an API key to see your projects				



Create an API key in a new project if you don't have one already

Select a project from your existing write-access Google Cloud projects



API keys

You can create a new project if you don't have one already or add API keys to an existing project. All projects are subject to the Google Cloud Platform Terms of Service, which you agree to when creating a new project, while use of the Gemini API and Google AI Studio is subject to the Gemini API Terms of Service.

as-you-go pricing.

API key generated

Use your API keys securely. Do not share them or embed them in code the public can view.

AIzaSyBxrLc6rkgS_nKn5oD3H_vfV6Wi_NlasNM

Your API keys are listed below. You can also view and manage your project and API keys in Google Cloud.

Google Cloud project name	API key	Action	Created	Plan

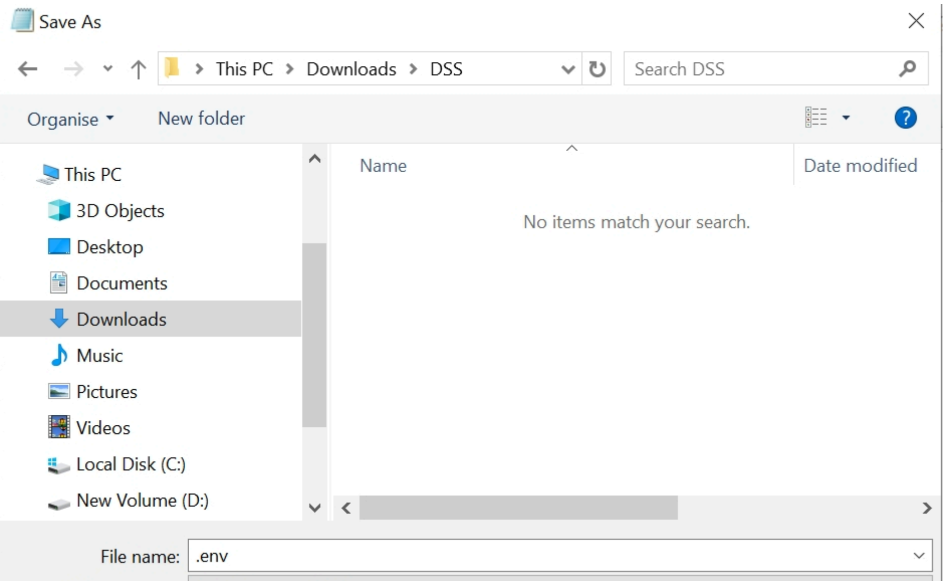
Create an API key to see your projects

4. For Windows users, you can paste the API key in Notepad with the following format and save your file with the name **.env**

   For Mac users, you can follow the same steps using TextEdit.

Unset

```
GOOGLE_API_KEY = "paste_API_key_here"
```



No items match your search.

File name: .env