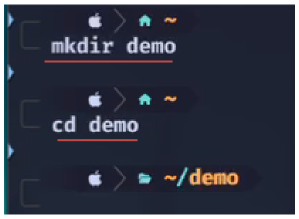
INSTALLING OPENCV AND RUNNING A SIMPLE PROJECT IN PYTHON

Step 1: Create a Project Folder

First, open your terminal and create a new project folder. Navigate to that folder using the following commands:

```
mkdir demo
cd demo
```



```
~
mkdir demo
~
cd demo
~/demo
```

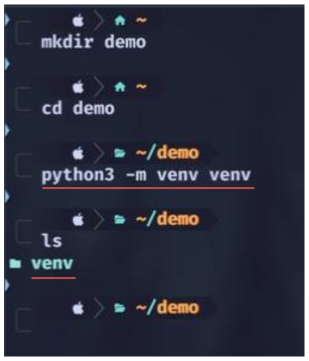
Step 2: Set Up a Virtual Environment

Before installing any Python packages, it is recommended to set up a virtual environment. This helps keep project dependencies isolated and avoids conflicts with system-wide packages.

To create a virtual environment, run:

```
python3 -m venv venv
```

This will create a new folder named `venv` inside your project directory.

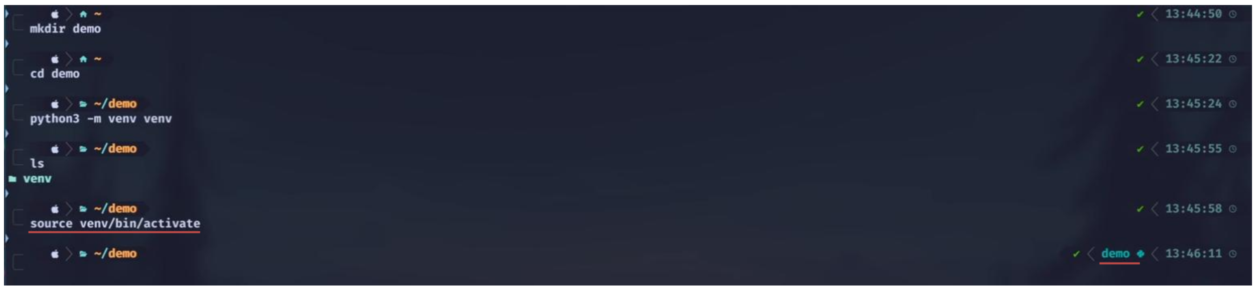


```
~
mkdir demo
~
cd demo
~/demo
python3 -m venv venv
~/demo
ls
venv
~/demo
```

Step 3: Activate the Virtual Environment

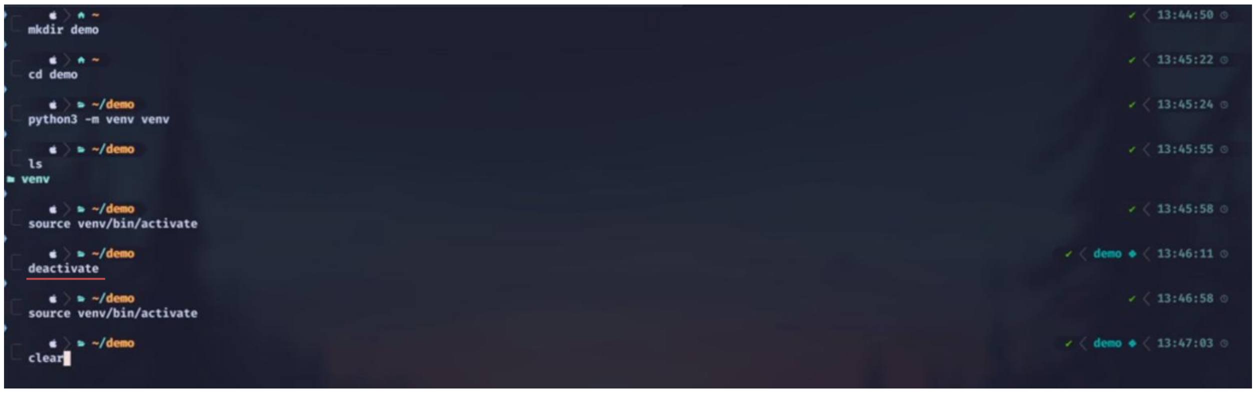
To activate the virtual environment, use the appropriate command for your operating system:

- **Mac/Linux**: `source venv/bin/activate`
- **Windows**: `venv\Scripts\activate`



```
~                                                              ✓ 13:44:50
mkdir demo
~                                                              ✓ 13:45:22
cd demo
~/demo                                                         ✓ 13:45:24
python3 -m venv venv
~/demo                                                         ✓ 13:45:55
ls
venv
~/demo                                                         ✓ 13:45:58
source venv/bin/activate
~/demo                                                  ✓ demo 13:46:11
```

Once activated, any package installed will be contained within this virtual environment. You can see on the right-hand side, inside the virtual environment, the prompt changes. On the machine, it may not show this, but you can verify you are inside the virtual environment by running: `deactivate`



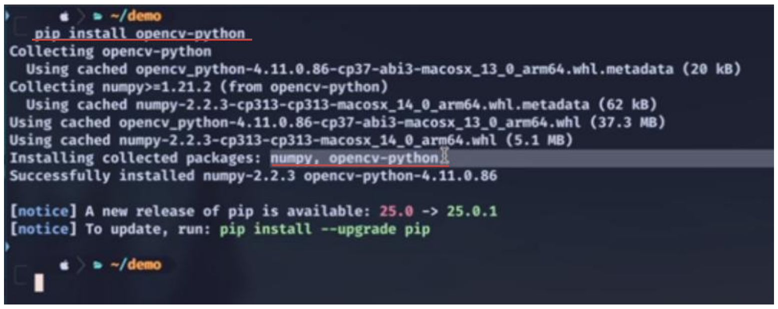
```
~
mkdir demo                                              13:44:50
~
cd demo                                                 13:45:22
~/demo
python3 -m venv venv                                    13:45:24
~/demo
ls                                                      13:45:55
venv
~/demo
source venv/bin/activate                                13:45:58
~/demo
deactivate                                   demo       13:46:11
~/demo
source venv/bin/activate                                13:46:58
~/demo
clear                                        demo       13:47:03
```

If the command auto-completes with Tab, then you are inside the virtual environment.

Step 4: Install OpenCV

Next, install OpenCV using pip:

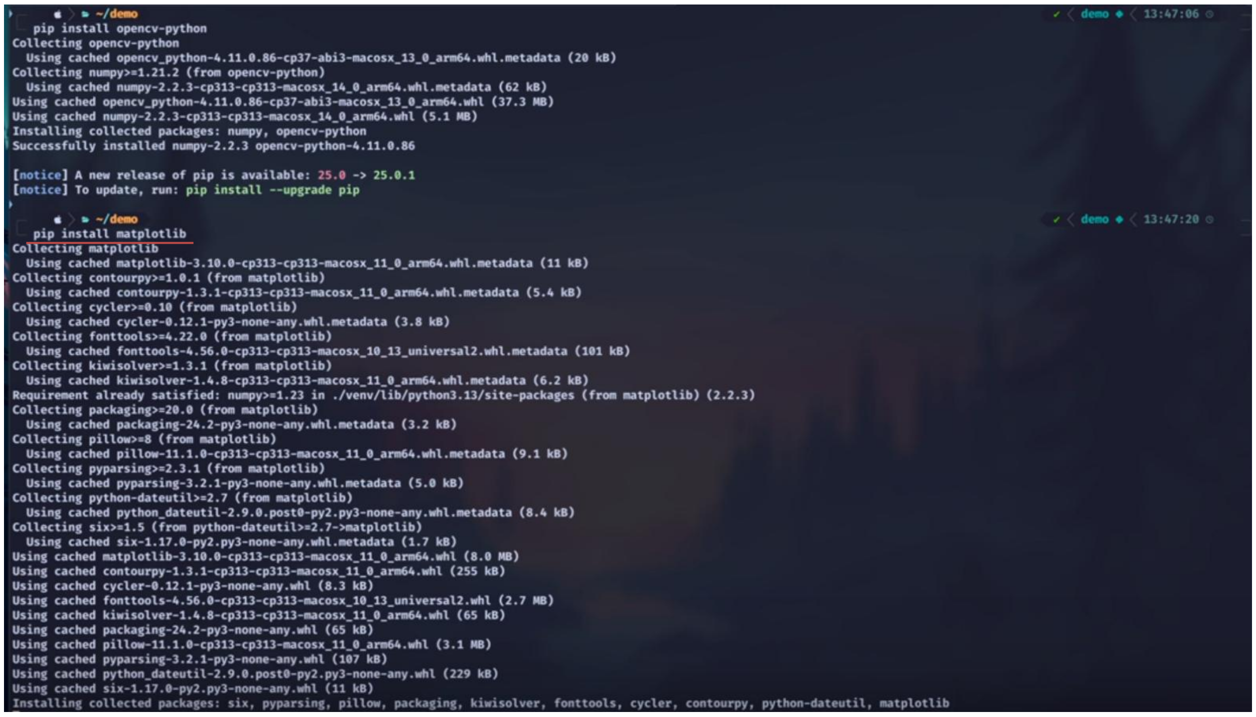
```
pip install opencv-python
```



```
~/demo
pip install opencv-python
Collecting opencv-python
  Using cached opencv_python-4.11.0.86-cp37-abi3-macosx_13_0_arm64.whl.metadata (20 kB)
Collecting numpy>=1.21.2 (from opencv-python)
  Using cached numpy-2.2.3-cp313-cp313-macosx_14_0_arm64.whl.metadata (62 kB)
Using cached opencv_python-4.11.0.86-cp37-abi3-macosx_13_0_arm64.whl (37.3 MB)
Using cached numpy-2.2.3-cp313-cp313-macosx_14_0_arm64.whl (5.1 MB)
Installing collected packages: numpy, opencv-python
Successfully installed numpy-2.2.3 opencv-python-4.11.0.86

[notice] A new release of pip is available: 25.0 -> 25.0.1
[notice] To update, run: pip install --upgrade pip

~/demo
```

OpenCV will automatically install NumPy as a dependency, but you may need additional libraries such as `matplotlib`:

```
pip install matplotlib
```



```
~/demo                                       demo       13:47:06
pip install opencv-python
Collecting opencv-python
  Using cached opencv_python-4.11.0.86-cp37-abi3-macosx_13_0_arm64.whl.metadata (20 kB)
Collecting numpy>=1.21.2 (from opencv-python)
  Using cached numpy-2.2.3-cp313-cp313-macosx_14_0_arm64.whl.metadata (62 kB)
Using cached opencv_python-4.11.0.86-cp37-abi3-macosx_13_0_arm64.whl (37.3 MB)
Using cached numpy-2.2.3-cp313-cp313-macosx_14_0_arm64.whl (5.1 MB)
Installing collected packages: numpy, opencv-python
Successfully installed numpy-2.2.3 opencv-python-4.11.0.86

[notice] A new release of pip is available: 25.0 -> 25.0.1
[notice] To update, run: pip install --upgrade pip

~/demo                                       demo       13:47:20
pip install matplotlib
Collecting matplotlib
  Using cached matplotlib-3.10.0-cp313-cp313-macosx_11_0_arm64.whl.metadata (11 kB)
Collecting contourpy>=1.0.1 (from matplotlib)
  Using cached contourpy-1.3.1-cp313-cp313-macosx_11_0_arm64.whl.metadata (5.4 kB)
Collecting cycler>=0.10 (from matplotlib)
  Using cached cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools>=4.22.0 (from matplotlib)
  Using cached fonttools-4.56.0-cp313-cp313-macosx_10_13_universal2.whl.metadata (101 kB)
Collecting kiwisolver>=1.3.1 (from matplotlib)
  Using cached kiwisolver-1.4.8-cp313-cp313-macosx_11_0_arm64.whl.metadata (6.2 kB)
Requirement already satisfied: numpy>=1.23 in ./venv/lib/python3.13/site-packages (from matplotlib) (2.2.3)
Collecting packaging>=20.0 (from matplotlib)
  Using cached packaging-24.2-py3-none-any.whl.metadata (3.2 kB)
Collecting pillow>=8 (from matplotlib)
  Using cached pillow-11.1.0-cp313-cp313-macosx_11_0_arm64.whl.metadata (9.1 kB)
Collecting pyparsing>=2.3.1 (from matplotlib)
  Using cached pyparsing-3.2.1-py3-none-any.whl.metadata (5.0 kB)
Collecting python-dateutil>=2.7 (from matplotlib)
  Using cached python_dateutil-2.9.0.post0-py2.py3-none-any.whl.metadata (8.4 kB)
Collecting six>=1.5 (from python-dateutil>=2.7->matplotlib)
  Using cached six-1.17.0-py2.py3-none-any.whl.metadata (1.7 kB)
Using cached matplotlib-3.10.0-cp313-cp313-macosx_11_0_arm64.whl (8.0 MB)
Using cached contourpy-1.3.1-cp313-cp313-macosx_11_0_arm64.whl (255 kB)
Using cached cycler-0.12.1-py3-none-any.whl (8.3 kB)
Using cached fonttools-4.56.0-cp313-cp313-macosx_10_13_universal2.whl (2.7 MB)
Using cached kiwisolver-1.4.8-cp313-cp313-macosx_11_0_arm64.whl (65 kB)
Using cached packaging-24.2-py3-none-any.whl (65 kB)
Using cached pillow-11.1.0-cp313-cp313-macosx_11_0_arm64.whl (3.1 MB)
Using cached pyparsing-3.2.1-py3-none-any.whl (107 kB)
Using cached python_dateutil-2.9.0.post0-py2.py3-none-any.whl (229 kB)
Using cached six-1.17.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: six, pyparsing, pillow, packaging, kiwisolver, fonttools, cycler, contourpy, python-dateutil, matplotlib
```

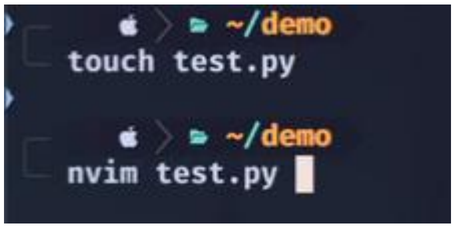
Step 5: Start Writing Python Code

After installing all the necessary packages, begin writing some Python code.

A new file will be created to include an example from the OpenCV website.

This example will open a window displaying an image. If the 's' key is pressed, the file will be saved with a different extension. For instance, if the original file is a JPEG, pressing 's' will save it as a PNG.

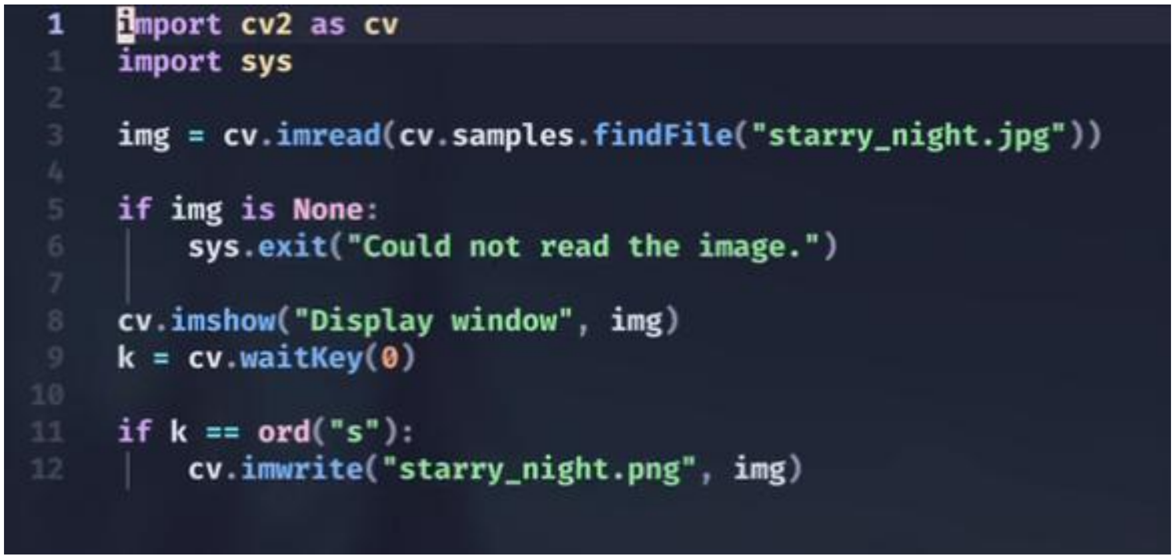
To begin, create a new Python file named `test.py`



```
~/demo
touch test.py

~/demo
nvim test.py
```

Adding the following code:

```
import cv2 as cv

import sys



img = cv.imread(cv.samples.findFile("starry_night.jpg"))



if img is None:

    sys.exit("Could not read the image.")



cv.imshow("Display window", img)

k = cv.waitKey(0)



if k == ord("s"):

    cv.imwrite("starry_night.png", img)
```

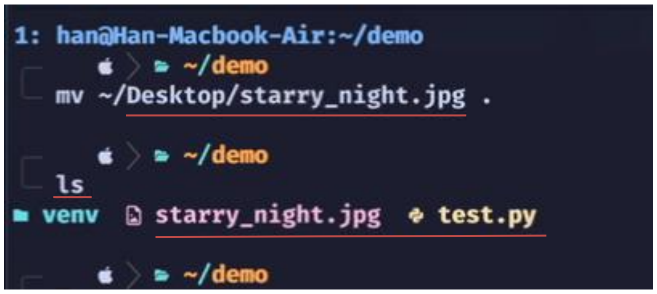


```python
import cv2 as cv
import sys

img = cv.imread(cv.samples.findFile("starry_night.jpg"))

if img is None:
    sys.exit("Could not read the image.")

cv.imshow("Display window", img)
k = cv.waitKey(0)

if k == ord("s"):
    cv.imwrite("starry_night.png", img)
```

Currently, `starry_night.jpg` is not in the project directory. This file must be downloaded and moved into the folder before executing the script.

Step 6: Get the Sample Image

Download `starry_night.jpg` from the [OpenCV GitHub]() repository and move it into your project folder.

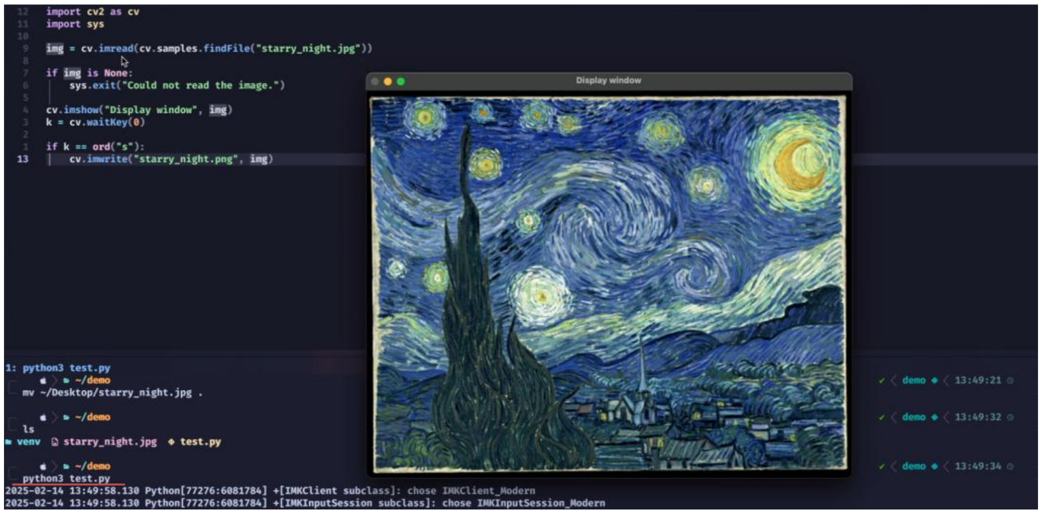


```
1: han@Han-Macbook-Air:~/demo
     ~/demo
 mv ~/Desktop/starry_night.jpg .

     ~/demo
 ls
 venv   starry_night.jpg   test.py

     ~/demo
```

Step 7: Run the Script

Execute the Python script with: `python3 test.py`

After a little bit, the image will show up in a new window.



```python
import cv2 as cv
import sys

img = cv.imread(cv.samples.findFile("starry_night.jpg"))

if img is None:
    sys.exit("Could not read the image.")

cv.imshow("Display window", img)
k = cv.waitKey(0)

if k == ord("s"):
    cv.imwrite("starry_night.png", img)
```

```
1: python3 test.py
     ~/demo                                                     ✓ demo ◆ 13:49:21
 mv ~/Desktop/starry_night.jpg .

     ~/demo                                                     ✓ demo ◆ 13:49:32
 ls
 venv   starry_night.jpg   test.py

     ~/demo                                                     ✓ demo ◆ 13:49:34
 python3 test.py
2025-02-14 13:49:58.130 Python[77276:6081784] +[IMKClient subclass]: chose IMKClient_Modern
2025-02-14 13:49:58.130 Python[77276:6081784] +[IMKInputSession subclass]: chose IMKInputSession_Modern
```

If the file does not exist, the program will throw an error and exit. Since we have the file, it will run successfully.

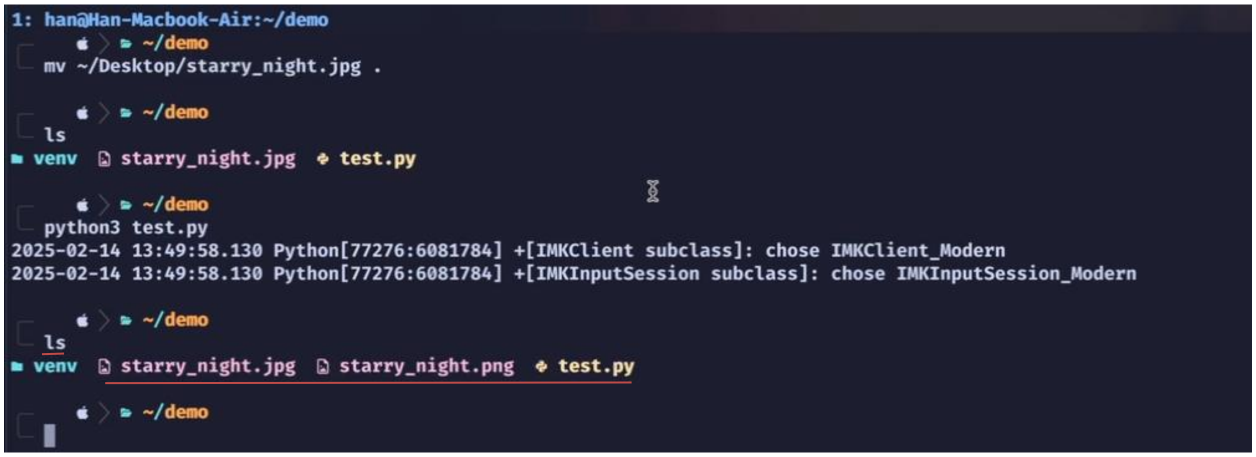
Using `cv2.imread()`, we read the file and store it in a variable.

Using `cv2.imshow()`, we display that image.

The `cv2.waitKey()` function:

- If set to `0`, it will never close the window until you press a key.
- If set to `10`, it will wait for 10 milliseconds and, if no key is pressed, move to the next step, which is saving the file.

Now, press the 's' key to save the file. If you hit 's', you will see a new file saved as `starry_night_copy.png`.



```
1: han@Han-Macbook-Air:~/demo
      ~/demo
 mv ~/Desktop/starry_night.jpg .

      ~/demo
 ls
 venv   starry_night.jpg   test.py

      ~/demo
 python3 test.py
2025-02-14 13:49:58.130 Python[77276:6081784] +[IMKClient subclass]: chose IMKClient_Modern
2025-02-14 13:49:58.130 Python[77276:6081784] +[IMKInputSession subclass]: chose IMKInputSession_Modern

      ~/demo
 ls
 venv   starry_night.jpg   starry_night.png   test.py

      ~/demo
```

This is just the basic setup. You can explore OpenCV's documentation or work on more advanced projects.

Step 8: Exit the Virtual Environment

Once you're done, deactivate the virtual environment by running: `deactivate`

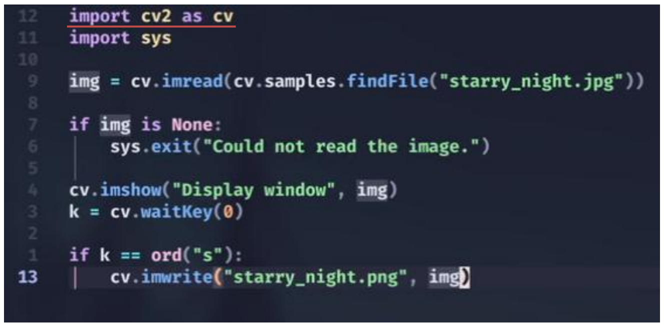
Summary

- Create a new virtual environment (`python3 -m venv venv`).
- Activate the virtual environment (`source venv/bin/activate` or `venv\Scripts\activate`).
- Install OpenCV (`pip install opencv-python`).
- Install additional packages if needed (`pip install matplotlib`).
- Create and run a Python script to display and save an image.
- Deactivate the virtual environment when finished (`deactivate`).

Additional Notes

Even though the package name is `opencv-python`, you import it as: `import cv2`

If you prefer, you can rename the import for shorter calls: `import cv2 as cv`



```
import cv2 as cv
import sys

img = cv.imread(cv.samples.findFile("starry_night.jpg"))

if img is None:
    sys.exit("Could not read the image.")

cv.imshow("Display window", img)
k = cv.waitKey(0)

if k == ord("s"):
    cv.imwrite("starry_night.png", img)
```

This helps simplify function calls within your script.

If you remain inside the virtual environment after finishing, simply run `deactivate` to exit.

That's it for this tutorial.