

Installing Libraries, NumPy, and Matplotlib

Python's standard library covers a lot of ground, but for specialized tasks — like fast numeric computation or making plots — you'll reach for external packages instead. In this activity you'll install two of the most widely used scientific Python libraries, use NumPy to work with arrays of numbers, and use Matplotlib to turn those numbers into a picture.

Content Learning Objectives

After completing this activity, students should be able to:

- Install an external Python package with pip and confirm that it is available.
- Create NumPy arrays and perform element-wise arithmetic on them.
- Generate a Matplotlib line plot with labeled axes.
- Combine NumPy and Matplotlib to visualize a mathematical function.

Process Skill Goals

During the activity, students should make progress toward:

- Reading and interpreting an error message when a package or import is missing. (Information Processing)
- Following a multi-step technical procedure as a team, and confirming each step before moving on. (Teamwork)

Model 1 Installing Libraries

Some tasks call for code too specialized to ship with Python itself. NumPy (fast numeric arrays) and Matplotlib (plotting) are two of the most popular such packages, but neither one is built in the way `turtle` or `random` are — you have to install them yourself, once per environment.

Command (typed in a terminal, or the Thonny Shell)	What Happens
<code>pip list</code>	Prints every package currently installed – probably a short list, without <code>numpy</code> or <code>matplotlib</code> yet
<code>import numpy</code> (before installing)	<code>ModuleNotFoundError: No module named 'numpy'</code>
<code>pip install numpy</code>	Downloads and installs <code>numpy</code> ; ends with a line like <code>Successfully installed numpy-2.1.0</code>
<code>import numpy</code> (after installing)	(no output at all – a successful import is silent)
<code>pip list</code> (again)	<code>numpy</code> now appears somewhere in the list

Do not install anything yet! Answer the first few questions before you touch the keyboard.

Questions (15 min)

Start time: _____

1. Based on the table above, what type of error do you get when you try to `import` a library that hasn't been installed yet?

2. What is the command to install the `numpy` library from a terminal?

3. In Thonny, what menu option lets you install a package without typing a `pip` command at all?

Now actually install `numpy` and `matplotlib` (using either method above), and confirm both show up in `pip list`. Ask for help if you get stuck!

4. After installing, how can you confirm a library is installed without writing a single line of Python code?

5. Type `import numpy` in the Shell. What output do you see?

6. Consider the command `pip install numpy` and the statement `import numpy`.

a) What is the name of the library as given to `pip` (the *package* name)?

b) What is the name used in the `import` statement (the *import* name)?

c) In this case, are the two names the same?

7. Not every package's `pip` name matches its `import` name — for example, you run `pip install pillow` but then write `import PIL`. Give one reason a library's install name and `import` name might end up different.

8. A classmate says "I don't need to install numpy — I already have Python installed." Explain what's wrong with this reasoning.

Model 2 NumPy Basics

NumPy's core object is the **array** — a grid of numbers you can do math on all at once, instead of one number at a time in a loop. The conventional way to import it gives it a short alias:

```
1 import numpy as np
```

Python code	Shell output
values = [10, 20, 30]	
values * 2	
arr = np.array([10, 20, 30])	
arr * 2	
arr + 1	
arr.sum()	
arr.mean()	
arr.max()	
np.arange(0, 10, 2)	
np.linspace(0, 1, 5)	

Do not type anything yet! Read the questions first.

Questions (20 min)

Start time: _____

9. What is the conventional alias used when importing numpy (as in `import numpy as ____`)?

Now open the Shell, type each line of Model 2 one at a time, and record the output above (in the table). Place an asterisk next to anything you didn't expect.

10. Compare rows 1–2 with rows 3–4 above (the plain Python list versus the NumPy array). What is the difference between what `* 2` does to a list and what it does to an array?

11. Based on the table, list three NumPy array methods (called like `arr.method()`) that summarize the values in an array.

12. `np.arange(0, 10, 2)` takes three arguments. What do the first, second, and third arguments represent, in order?

13. How is `np.linspace` different from `np.arange`?

14. Predict the output of `np.arange(1, 5)` (only two arguments given).

Test your prediction in the Shell — were you right?

15. Predict the output of `arr / 0`, where `arr = np.array([1, 2, 3])`. Then test it in the Shell. What's different from what plain Python does with `1 / 0`?

16. Why might it be useful that array operations don't require you to write your own loop?

Model 3 Matplotlib Basics

Matplotlib turns arrays of numbers into pictures. Its most commonly used piece is `pyplot`, almost always imported as `plt`.

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 x = np.linspace(0, 10, 100)
5 y = np.sin(x)
6
7 plt.plot(x, y)
8 plt.xlabel("x")
9 plt.ylabel("sin(x)")
10 plt.title("My First Plot")
11 plt.show()
```

Create a new file `plot1.py`, enter the code above, and run it.

Questions (10 min)

Start time: _____

17. What appeared when you ran the program?

18. Which single line of code is responsible for actually opening the plot window?

19. Which two lines create the arrays of numbers that get plotted?

20. What are the two arguments to `plt.plot()` in this example, and what does each one represent?

Now delete (or comment out) the three lines starting with `plt.xlabel`, `plt.ylabel`, and `plt.title`, and run the program again.

21. What does the plot look like now — what's missing compared to before?

22. Why is it good practice to always label your axes?

Now change the line `y = np.sin(x)` to each of the following in turn, re-running each time, and describe the resulting shape.

Code for y	Shape of the plot
------------	-------------------

<code>y = x</code>	
<code>y = x ** 2</code>	
<code>y = np.exp(-x)</code>	

23. Based on the last row of the table, describe in your own words what `np.exp(-x)` does to the values as `x` increases.

24. If you wanted two different curves — say `np.sin(x)` and `np.cos(x)` — on the *same* plot, how many times would you need to call `plt.plot()` before calling `plt.show()`?