From MATLAB to Python

A quick-start guide for Civil Engineering students (Python 3.12+)

Works in Jupyter Notebook or Google Colab

Last updated: 2026-02-22

Goal: help you transfer what you already know in MATLAB into beginner-friendly Python you can run in a notebook.

This guide avoids advanced syntax on purpose. Once you are comfortable, you can learn faster and more idiomatic styles.

Contents

1. Getting set up

Fastest start: use Google Colab in a browser. No installation. Your notebook runs on Google's hosted machines.

Local option: install Python 3.12+ and Jupyter. A common beginner setup is:
1) Install Python 3.12+ (from python.org or a distribution like Anaconda).
2) In a terminal: `pip install notebook`
3) Start Jupyter: `jupyter notebook`

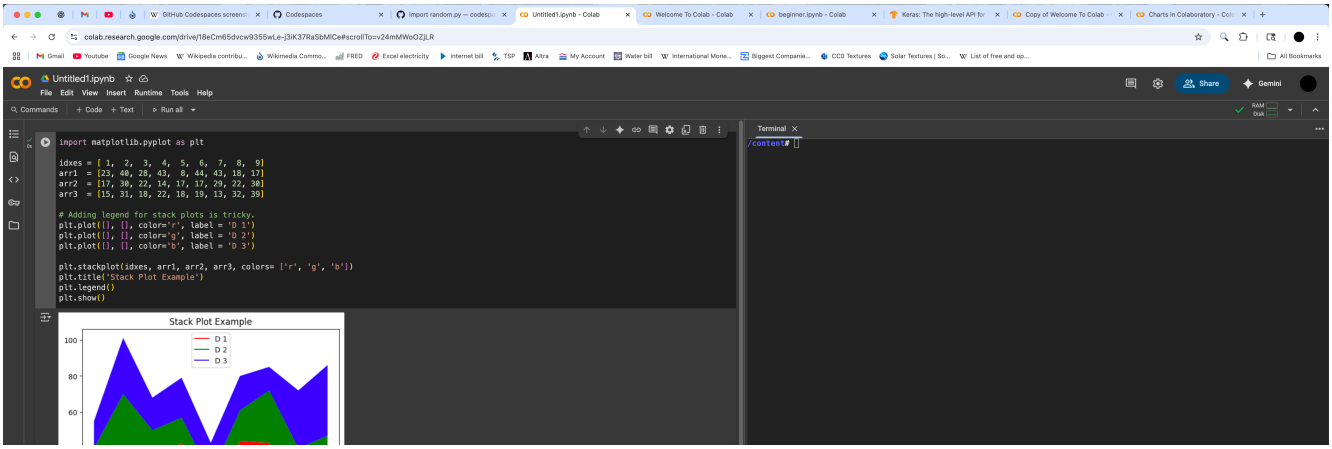

Figure 1. Google Colab notebook interface (example). Source: Wikimedia Commons, "Google Colab screenshot" (CC0 1.0) [1].

Notebook mindset

A notebook is interactive: you write a small piece, run it, inspect the result, then continue. This is ideal for engineering calculations and quick plots.

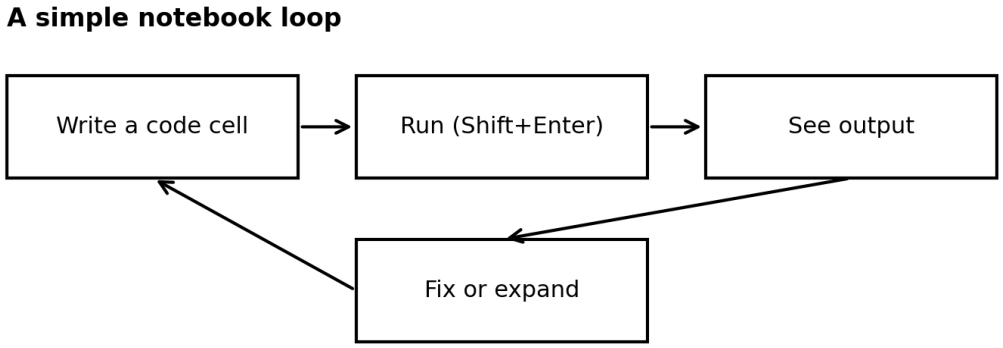

Figure 2. A simple notebook loop (generated for this guide).

2. Notebook basics

Whether you are in Jupyter or Colab, the core ideas are the same:

- **Cells:** code cells run Python; markdown cells hold text and equations.
- **Run:** Shift+Enter runs the current cell and moves to the next.
- **Kernel/runtime:** the Python process that remembers variables.
- **Restart:** clears memory and reruns from scratch (useful when things get messy).

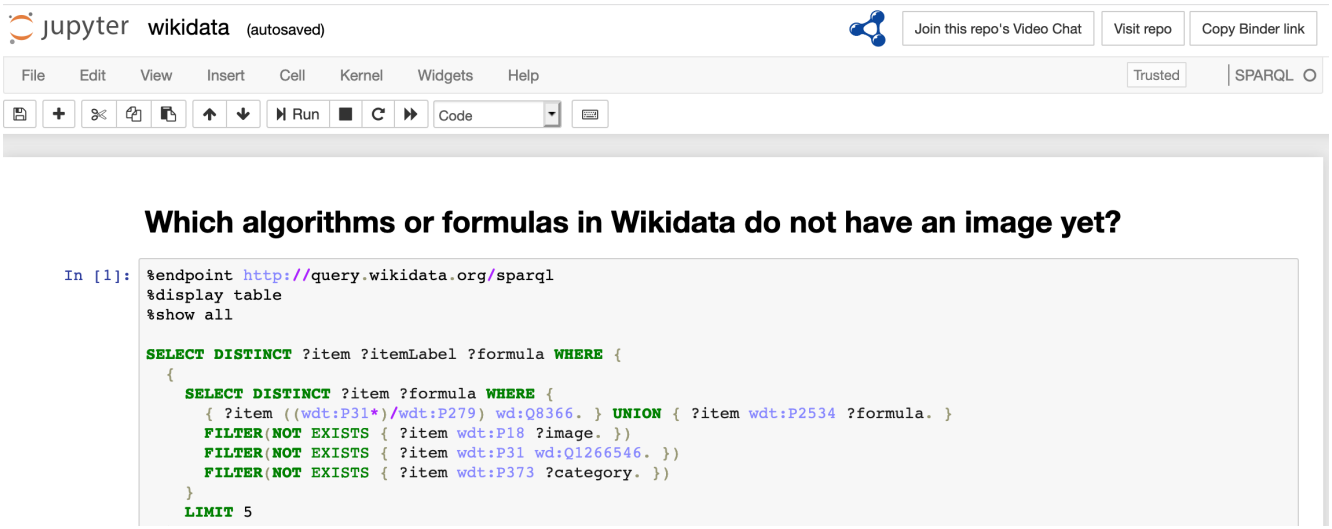

Figure 3. Jupyter Notebook interface (example). Source: Wikimedia Commons, "Screenshot of a Jupyter Notebook with SPARQL kernel..." (CC0 1.0) [2].

Tip: if a notebook stops behaving as expected, restart the runtime and run cells from top to bottom.

3. MATLAB vs Python: the mental model

You already know how to think like an engineer-programmer. The biggest shift is syntax and a few defaults (especially indexing).

Concept	MATLAB	Python (beginner)	Notes
Comments	% this is a comment	# this is a comment	Same idea
Assignment	x = 3	x = 3	No type declarations
Strings	'hi' or "hi"	"hi" or 'hi'	Both quote styles work
Indexing	a(1) is first	a[0] is first	Python is 0-based
Block end	end	indentation	Whitespace matters
Loop	for i = 1:n	for i in range(n):	range(n) gives 0..n-1
If	if ... elseif ... else	if ... elif ... else:	Use colon + indent
Logical AND	&&	and	Also: or, not
Elementwise mult.	.*	*	In NumPy, * is elementwise

Table 1. Quick translation table (MATLAB to beginner-friendly Python).

Indexing is the #1 gotcha. In MATLAB, the first element is 1. In Python, the first element is 0.

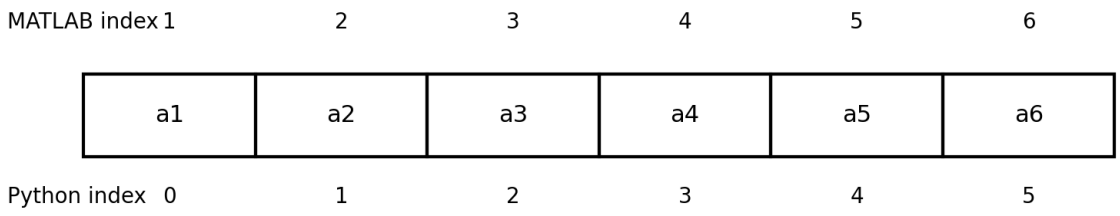

Figure 4. Indexing difference (generated for this guide).

4. Data types and built-in functions

Python values have types. You usually do not write them explicitly; Python figures them out.

```
x = 3 # int
y = 3.5 # float
ok = True # bool
name = "Bridge" # str (string)

print(type(x), type(y), type(ok), type(name))
```

Type	Example	When you use it	MATLAB analogy
int	3	counts, indices	integer
float	3.5	measurements, units	double
bool	True / False	conditions	logical
str	"abc"	labels, file paths	char / string
list	[1, 2, 3]	a changeable collection	cell array (often)
tuple	(1, 2)	fixed group of values	small fixed vector
dict	{'E':30e9}	key-value lookups	struct / containers.Map
None	None	means 'no value yet'	[] or empty

Table 2. Common Python data types (beginner view).

A few built-in functions you will use constantly:

- `print(x)` shows a value
- `len(a)` length of a list or string
- `sum(a)`, `min(a)`, `max(a)`
- `abs(x)`, `round(x)`
- `type(x)` tells you the type
- `range(n)` numbers 0..n-1 for loops

Reference: Python tutorial [3].

5. Logic, loops, and functions

Indentation replaces `end`. Every `if`, `for`, and `while` block ends when the indentation goes back.

```
grade = 82

if grade >= 90:
letter = "A"
elif grade >= 80:
letter = "B"
else:
letter = "C"

print(letter)
```

Loops look like this:

```
# for-loop
total = 0
for i in range(5): # 0,1,2,3,4
total = total + i
print(total)

# while-loop
n = 3
while n > 0:
print(n)
n = n - 1
```

Functions are like MATLAB function files, but you can define them inside a notebook cell.

```
def area_rectangle(b, h):
"""Return area of a rectangle."""
A = b * h
return A

print(area_rectangle(3, 4))
```

6. Arrays with NumPy (MATLAB-like matrices)

Most engineering work uses **NumPy** arrays. Think of them as MATLAB matrices with Python syntax.

```
import numpy as np

a = np.array([10, 20, 30]) # 1D array
b = np.array([[1, 2], [3, 4]]) # 2D array (matrix)

print(a[0]) # first element
print(b[1, 0]) # row 2, col 1 (0-based)
```

Common creations (similar to MATLAB):

```
x = np.arange(0, 5) # like 0:4
t = np.linspace(0, 10, 101) # like linspace(0,10,101)
Z = np.zeros((3, 4)) # 3x4 zeros
O = np.ones((2, 2)) # 2x2 ones
```

Task	MATLAB	NumPy (Python)
Zeros	zeros(m,n)	np.zeros((m, n))
Ones	ones(m,n)	np.ones((m, n))
Identity	eye(n)	np.eye(n)
Size	size(A)	A.shape
Transpose	A'	A.T
Matrix multiply	A*B	A @ B
Elementwise multiply	A.*B	A * B

Table 3. A few MATLAB array operations and the closest NumPy equivalents.

Reference: NumPy quickstart [4].

7. Plotting with Matplotlib

Matplotlib's `pyplot` is intentionally similar to MATLAB plotting. A typical pattern is:

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

x = np.linspace(0, 10, 100)
y = x**2

plt.plot(x, y)
plt.xlabel("x")
plt.ylabel("y")
plt.title("Simple plot")
plt.grid(True)
plt.show()
```

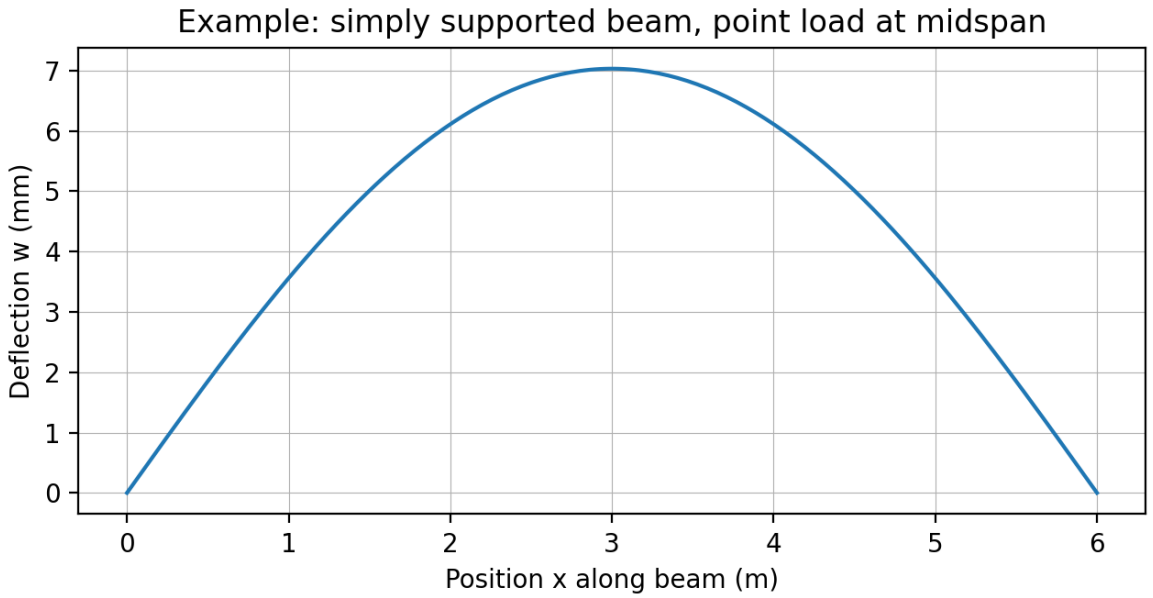

Figure 5. Example plot: beam deflection along span (generated for this guide).

Reference: Matplotlib pyplot tutorial [5].

8. Reading and writing simple data files

Start with CSV. In Colab, you can upload files from your computer or mount Google Drive. Locally, keep your CSV next to the notebook.

```
import csv

rows = []
with open("data.csv", "r", newline="") as f:
reader = csv.reader(f)
for row in reader:
rows.append(row)

print(rows[0]) # first row (often a header)
```

If you are ready for a friendlier table tool, **pandas** is common in engineering data work:

```
import pandas as pd

df = pd.read_csv("data.csv")
print(df.head())
```

Tip: keep your first goal simple: read data, plot it, and compute one summary value.

9. From scripts to classes (a simple Beam class)

A class bundles **data** (properties) and **methods** (functions) together. Start small.

- `__init__` runs when you create an object (like a constructor).
- `self` is the object itself, used to store and access properties.
- Methods look like functions, but they live inside the class.

```
class Beam:
def __init__(self, L, E, I):
self.L = L
self.E = E
self.I = I

def midspan_deflection_point_load(self, P):
# Simply supported beam, point load at midspan
w = P * (self.L**3) / (48 * self.E * self.I)
return w

b = Beam(L=6.0, E=30e9, I=8e-4)
print(b.midspan_deflection_point_load(P=20e3)) # meters
```

Reference: Python tutorial, Classes [6].

10. Practice checklist and next steps

Try these mini-tasks in a notebook. Each one should take 5-10 minutes.

- Create a NumPy array of 101 points from 0 to 10 and compute its mean.
- Write a loop that sums numbers 1 to 100 (then try `sum`).
- Write a function that converts kN to N and MPa to Pa.
- Make a plot of y = sin(x) over 0 to 2*pi.
- Create a small class (like `Beam`) with two properties and one method.

Notebook cheat sheet (works in Colab and Jupyter)

Action	Shortcut (typical)	Notes
Run cell and move on	Shift+Enter	Most common
Run cell and stay	Ctrl+Enter	Cmd+Enter on macOS
Command vs edit mode	Esc / Enter	Jupyter classic
Restart runtime/kernel	Menu: Runtime or Kernel	Clears variables
Find text	Ctrl+F	Search in the current page

Table 4. A few useful notebook actions and shortcuts.

When you want more depth, these official references are excellent starting points:

- Jupyter documentation for notebook basics [7].
- Python tutorial (start at the introduction and data structures) [3].
- NumPy beginner and quickstart guides [4].
- Matplotlib pyplot tutorial [5].
- Colab documentation and FAQ [8].

References

[1] Wikimedia Commons. "Google Colab screenshot.webp" (CC0 1.0). Uploaded by Wikideas1, 5 Aug 2025. Accessed Feb 2026. commons.wikimedia.org/wiki/File:Google_Colab_screenshot.webp

[2] Wikimedia Commons. "Screenshot of a Jupyter Notebook with SPARQL kernel after running a query to the Wikidata Query Service as of 21 September 2020.png" (CC0 1.0). Uploaded by Daniel Mietchen, 21 Sep 2020. Accessed Feb 2026. commons.wikimedia.org/wiki/File:Screenshot_of_a_Jupyter_Notebook_with_SPARQL_kernel_after_running_a_query_to_the_Wikidata_Query_Service_as_of_21_September_2020.png

[3] Python Software Foundation. "The Python Tutorial" (Python 3 documentation). docs.python.org/3/tutorial/index.html

[4] NumPy Developers. "NumPy quickstart". numpy.org/doc/stable/user/quickstart.html

[5] Matplotlib Developers. "Pyplot tutorial". matplotlib.org/stable/tutorials/pyplot.html

[6] Python Software Foundation. "Classes" (Python tutorial). docs.python.org/3/tutorial/classes.html

[7] Project Jupyter. "Project Jupyter Documentation". docs.jupyter.org/

[8] Google. "Colab" and "Colaboratory FAQ". developers.google.com/colab and research.google.com/colaboratory/faq.html