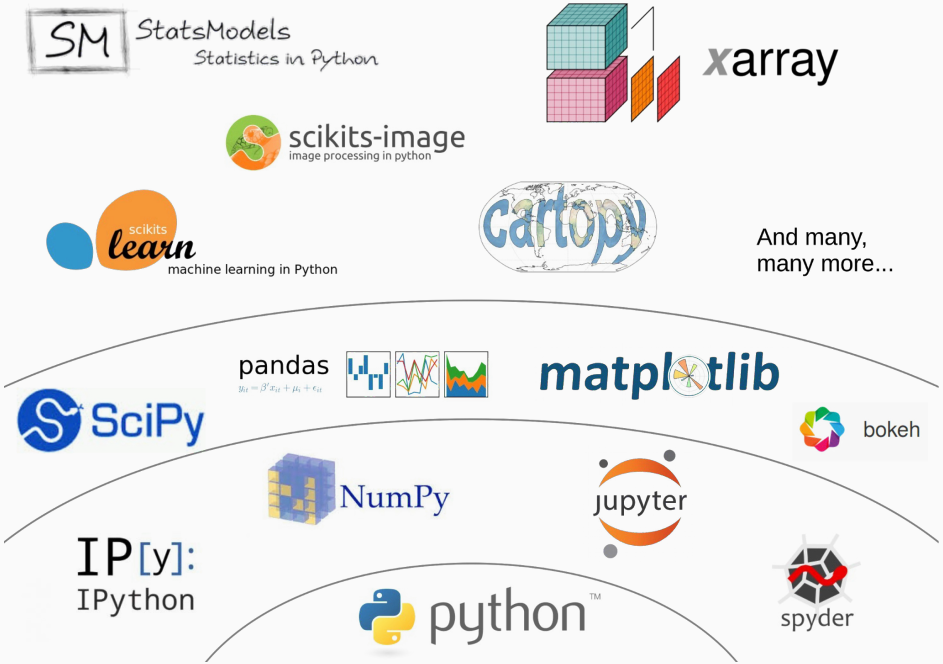
NumPy Multiplication

Nan

Why NumPy


SM
StatsModels
Statistics in Python
xarray
scikits-image
image processing in python
scikits
learn
machine learning in Python
cartopy
And many,
many more...
pandas
matplotlib
SciPy
bokeh
NumPy
jupyter
IP[y]:
IPython
python
spyder

95%

of the AI Engineers claim they know numpy well

Quiz Time!

1 * 2 = ?

Case 1 - Scalar

- a = 1
- b = 2

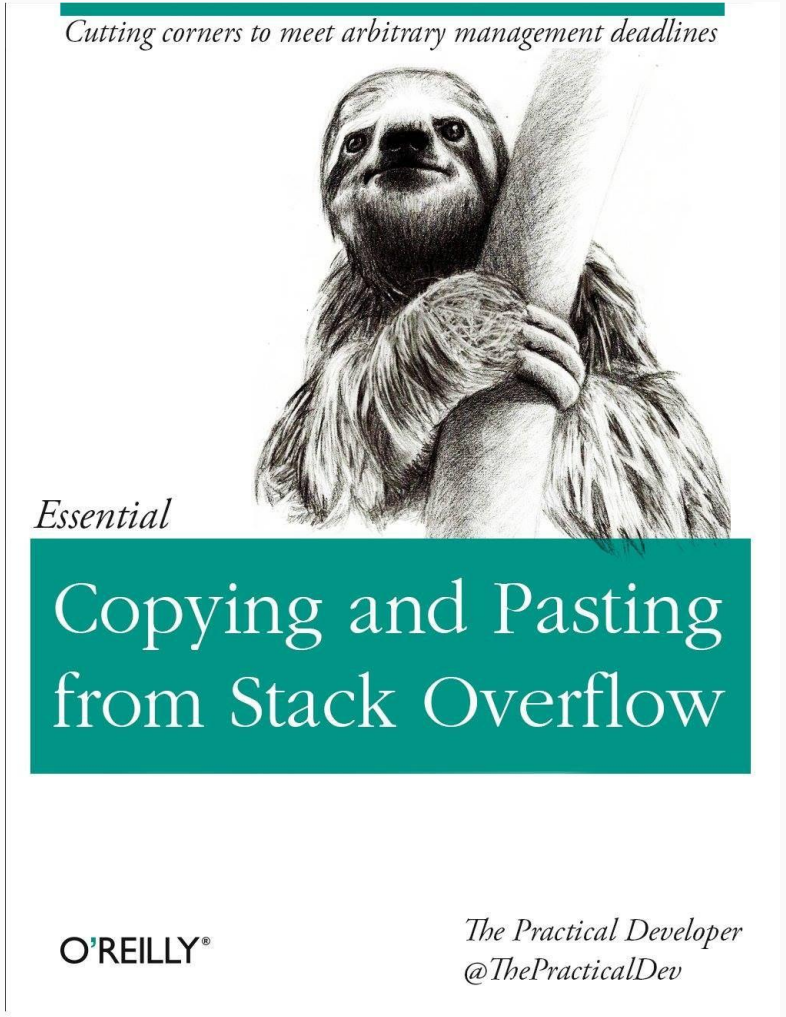
1. a * b = ?
2. np.multiply(a, b) = ?
3. np.dot(a,b) = ?
4. a @ b = ?
5. np.matmul(a,b)

Key point

matmul differs from dot in two important ways.

Multiplication by scalars is not allowed.

Stacks of matrices are broadcast together as if the matrices were elements.



[1,2] * 3 = ?

Case 2 - 1D & scalar

- a = [1, 2]
- b = 3

1. a * b = ?
2. np.multiply(a, b) = ?
3. np.dot(a,b) = ?
4. np.array(a) * b
5. a @ b = ?
6. np.matmul(a,b)
7. np.array(a).shape
8. np.array(a)[:,None].shape
9. np.matmul(np.array(a)[:,None], [b])

Getting interesting

[x, y] * [m, n]

Case 3 - 1D & 1D

- a = [1, 2]
- b = [2, 3]
- c = [3, 4, 5]

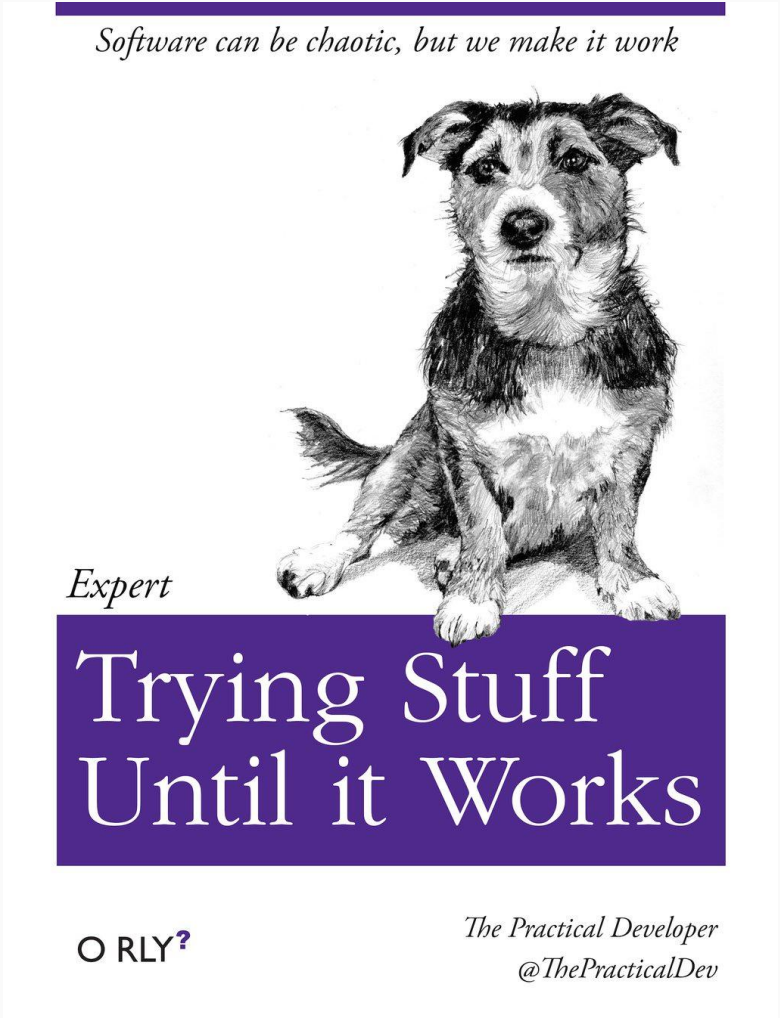
1. a * b = ?
2. a * c = ?
3. np.multiply(a, b) = ?
4. np.dot(a,b) = ?
5. np.array(a) * b
6. a @ b = ?
7. np.matmul(a,b)

Key points

Careful about broadcasting.

Though we don't care normally, np.array is different from np.asarray.

Good practice to convert numerical python list to numpy array first.



More fun to come

[[x, y], [p, q]] * [m, n]

Case 4 - 2D & 1D

- a = np.array([[1,2], [3,4]])
- b = np.array([5,6]

1. a * b = ?
2. np.multiply(a, b) = ?
3. np.dot(a,b) = ?
4. a.dot(b) = ?
5. b.dot(a) = ?
6. b @ a

Error is good

[[x, y], [p, q]] * [[m, n], [u, v]]

Case 5 - 2D & 2D

- a = np.array([[1,2], [3,4]])
- b = np.array([[5,6], [7,8]])

1. a * b = ?
2. np.multiply(a, b) = ?
3. np.dot(a,b) = ?
4. b @ a

Case 6 - 2D & 2D

- a = np.matrix([[1,2], [3,4]])
- b = np.matrix([[5,6], [7,8]])

[[x, y], [p, q], [u, v]] * [[m, n]]

Case 7 - 3D & 1D

- a = np.array([[[1,2,3], [2,3,4]], [[3,4,5], [4,5,6]]])
- b = np.array([1,2,3])

1. a * b = ?
2. a.dot(b)
3. a @ b

Case 8 - 1D & 3D

- c = np.array([1,2])

1. b * a = ?
2. c.dot(a)
3. c @ a

Getting real

[[x, y], [p, q], [u, v]] * [[m, n], [i, j]]

Case 9 - 3D & 2D or 2D & 3D

- a = np.arange(12).reshape((2, 2, 3))
- b = np.arange(6).reshape((3,2))

1. a * b = ?
2. a * b.T = ?
3. a.dot(b)
4. a @ b
5. b.T * a = ?
6. b.dot(a)
7. b @ a

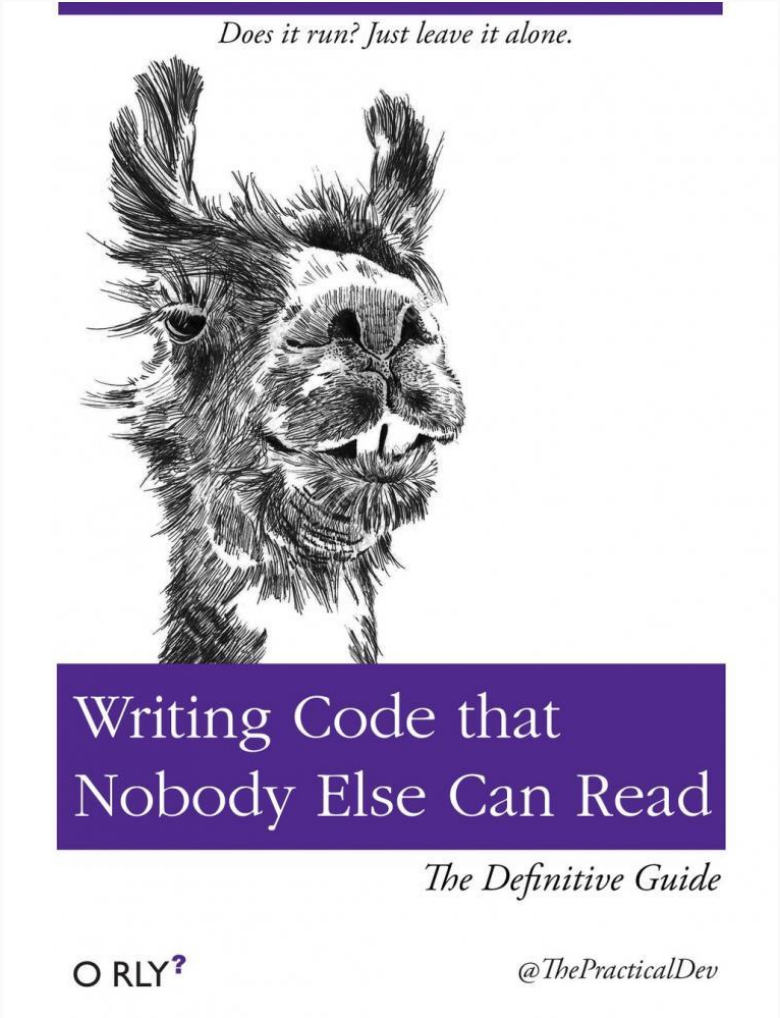
Key points

np.dot and np.matmul behavior differently when it comes to ND array

matmul -> Stacks of matrices are broadcast together as if the matrices were elements.

It only treats last 2 columns as matrix and do 2D matrix multiplication

dot -> it is a sum product over the last axis of a and the second-to-last of b



One last example

Case 10

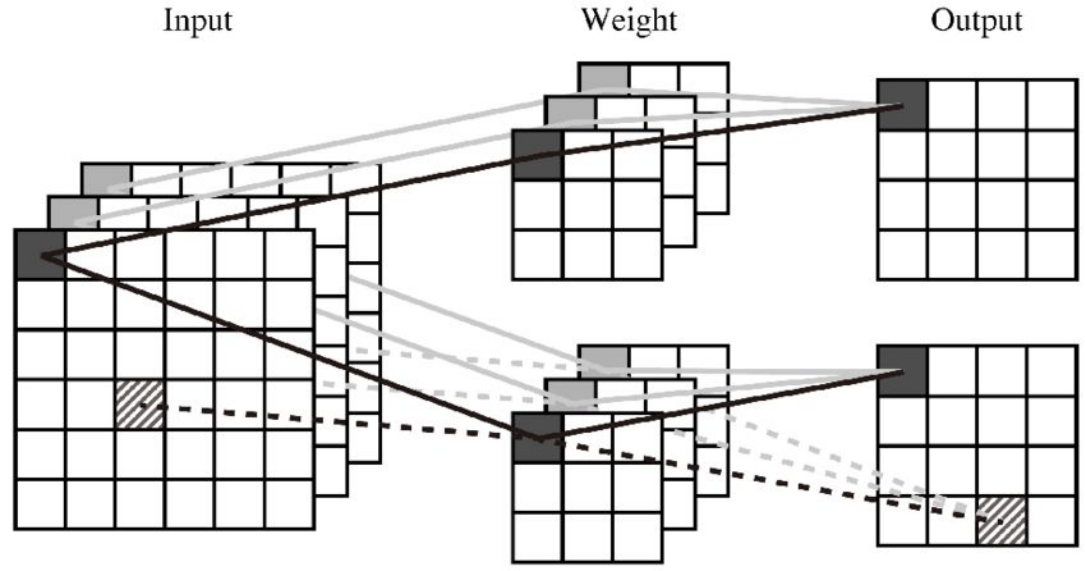
- a = np.random.rand(2,2,2)
- b = np.random.rand(2,2)

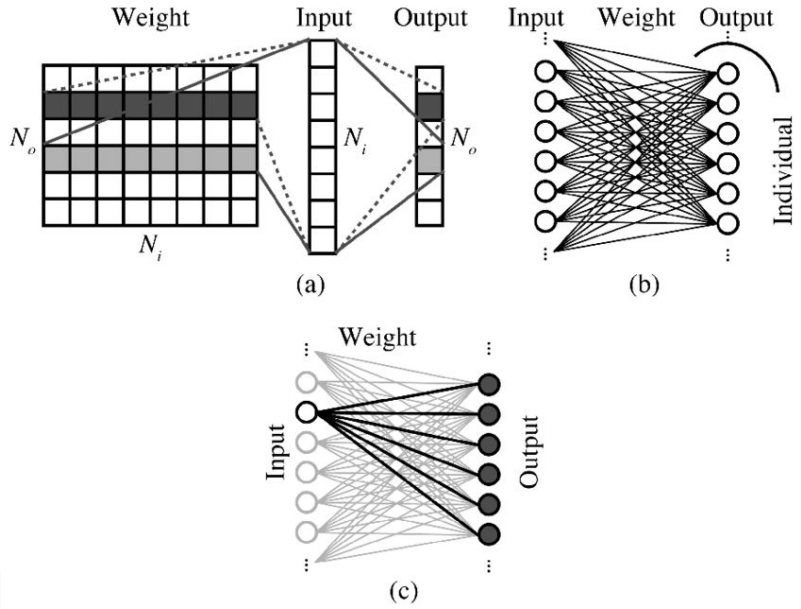
1. a @ b
2. a.dot(b)
3. b @ a
4. b.dot(a)

Both return results.

It might introduces errors that you have a hard time to debug.

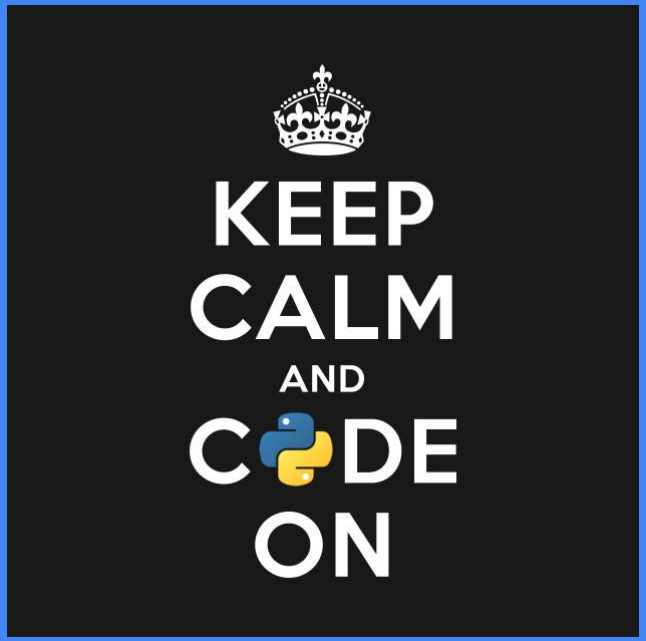
… yet another example





Takeaway

Always know what you are doing



“If you torture the data long enough, it will confess.”

- Ronald Coase

Thanks!

Made in rush with ❤️@ AI Lab

by Nan

<xiao.n@outlook.com>