KOÇ ÜNİVERSİTESİ

14 – NumPy

COMP125 Programming with Python

Recording Disclaimer

The synchronous sessions are recorded (audiovisual recordings). The students are not required to keep their cameras on during class.

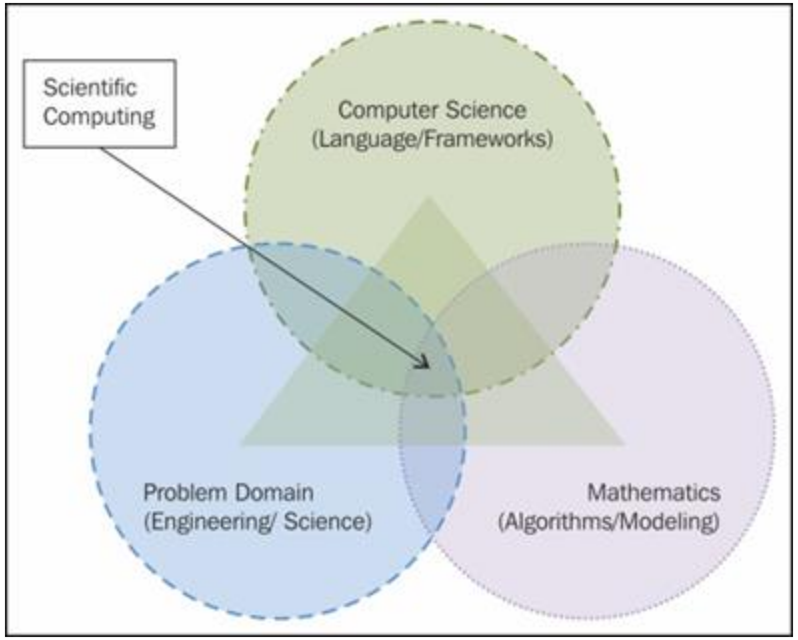
The audiovisual recordings, presentations, readings and any other works offered as the course materials aim to support remote and online learning. They are only for the personal use of the students. Further use of course materials other than the personal and educational purposes as defined in this disclaimer, such as making copies, reproductions, replications, submission and sharing on different platforms including the digital ones or commercial usages are strictly prohibited and illegal.

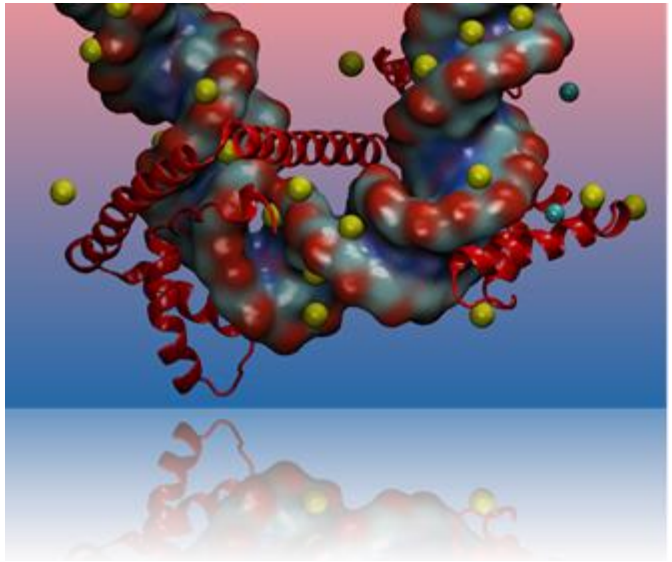
The persons violating the above-mentioned prohibitions can be subject to the administrative, civil, and criminal sanctions under the Law on Higher Education Nr. 2547, the By-Law on Disciplinary Matters of Higher Education Students, the Law on Intellectual Property Nr. 5846, the Criminal Law Nr. 5237, the Law on Obligations Nr. 6098, and any other relevant legislation.

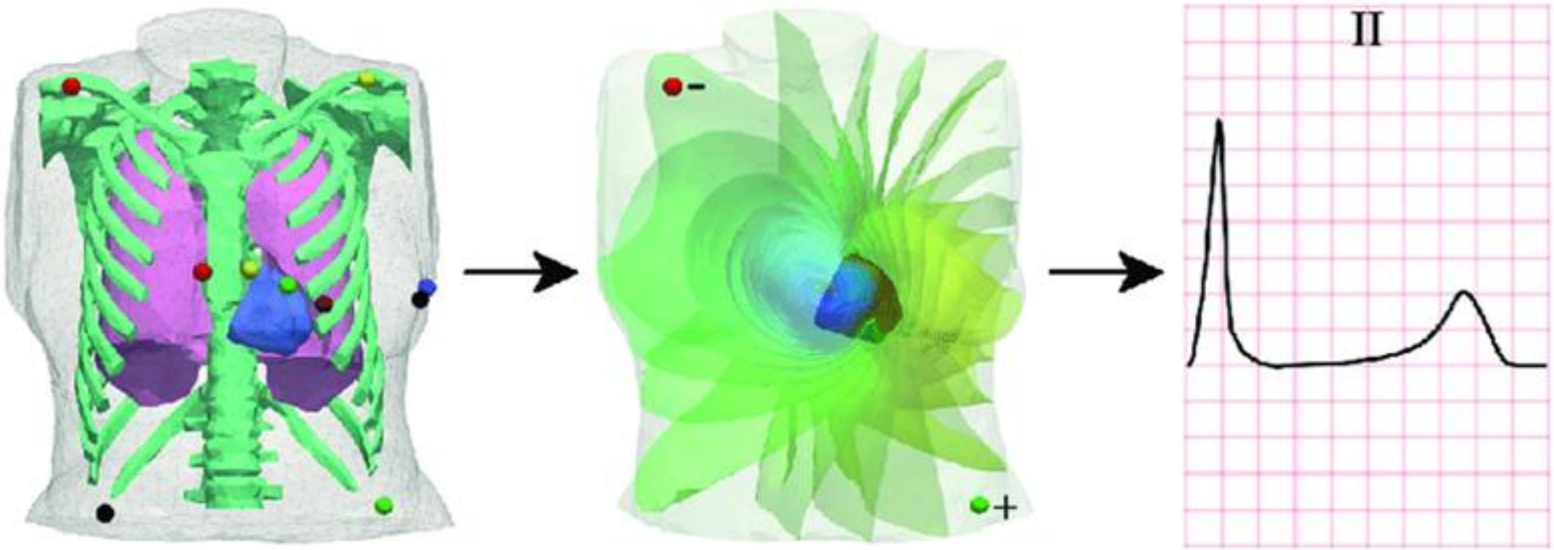
The academic expressions, views, and discussions in the course materials including the audio-visual recordings fall within the scope of the freedom of science and art.

Scientific Computing

- Scientific Computing is the collection of tools, techniques, and theories required to solve mathematical models of problems in Science and Engineering on a computer.

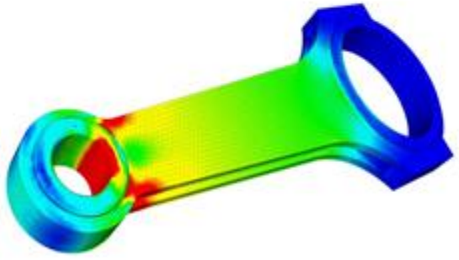




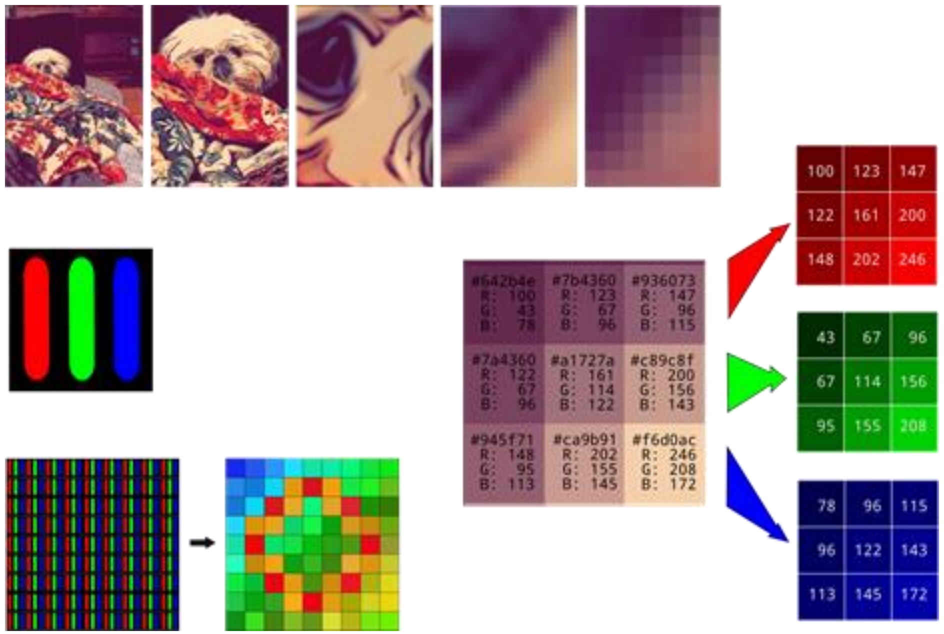


Matrix Operations

- Linear Algebra, Probability and Statistics, and Calculus show ubiquitously in scientific and engineering problems
- Require computers for solving big problems
- Matrix operations are an important subset



Finite Element/Volume Methods



Representing and working on Images

System of linear equations

$$
\begin{array}{ll}
\text{Joint } 2: & \begin{cases} f_2 = f_6 \\ f_3 = 10 \end{cases} \\
\text{Joint } 3: & \begin{cases} \alpha f_1 = f_4 + \alpha f_5 \\ \alpha f_1 + f_3 + \alpha f_5 = 0 \end{cases} \\
\text{Joint } 4: & \begin{cases} f_4 = f_8 \\ f_7 = 0 \end{cases} \\
\text{Joint } 5: & \begin{cases} \alpha f_5 + f_6 = \alpha f_9 + f_{10} \\ \alpha f_5 + f_7 + \alpha f_9 = 15 \end{cases} \\
\text{Joint } 6: & \begin{cases} f_{10} = f_{13} \\ f_{11} = 20 \end{cases} \\
\text{Joint } 7: & \begin{cases} f_8 + \alpha f_9 = \alpha f_{12} \\ \alpha f_9 + f_{11} + \alpha f_{12} = 0 \end{cases} \\
\text{Joint } 8: & \left\{ f_{13} + \alpha f_{12} = 0 \right.
\end{array}
$$

Matrix Operations

- So far, we have seen a few simple examples (Remember Lab09) with
  - list of lists (dense matrices)
  - dictionaries (sparse matrices)
- However, these are inefficient – why?
  - Too general, causing overheads
  - All the axes are not equal (remember getting a column from a list of lists?)
  - Gets worse when generalizing to Tensors
  - Looping over items, difficult to parallelize
- Need a specialized implementation - NumPy

NumPy

The fundamental package for scientific computing with Python

GET STARTED

NumPy v1.19.0 First Python 3 only release - Cython interface to numpy.random complete

POWERFUL N-DIMENSIONAL ARRAYS

Fast and versatile, the NumPy vectorization, indexing, and broadcasting concepts are the de-facto standards of array computing today.

NUMERICAL COMPUTING TOOLS

NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.

INTEROPERABLE

NumPy supports a wide range of hardware and computing platforms, and plays well with distributed, GPU, and sparse array libraries.

PERFORMANT

The core of NumPy is well-optimized C code. Enjoy the flexibility of Python with the speed of compiled code.

EASY TO USE

NumPy's high level syntax makes it accessible and productive for programmers from any background or experience level.

OPEN SOURCE

Distributed under a liberal BSD license, NumPy is developed and maintained publicly on GitHub by a vibrant, responsive, and diverse community.

https://numpy.org/devdocs/user/absolute_beginners.html

NumPy (Numerical Python)

- Implemented in C (means fast)
- NumPy `ndarrays`: Efficient storage of a **single type of data**
- These arrays can be of any dimension (e.g., vector 1, matrix 2, tensor n)
- Efficient mathematical and logical operations on arrays
- Efficient linear algebra operations on arrays
- Random number (array) generation
- Foundation of the Python scientific computation stack

NumPy Module

- Typically imported as

  `import numpy as np`

- Its main data structure, ndarray, is created as:

  `x = np.array(array_like)`

```python
import numpy as np

# 1D ndarray
x1D = np.array((3, 4))

# 2D ndarray
x2D = np.array([[3.3, 4.9], [-1, 3.2], [7.1, -5.7]])

print('Contents of 1D ndarray: ')
print(x1D)

print('Contents of 2D ndarray: ')
print(x2D)
```

```
In [1]: runfile('/Users/c
Contents of 1D ndarray:
[3 4]
Contents of 2D ndarray:
[[ 3.3  4.9]
 [-1.   3.2]
 [ 7.1 -5.7]]
```

```python
import numpy as np
L1 = [[1, 2, 3], [4, 5]]
A1 = np.array(L1)
```

```
In [1]: runfile('/Users/cigdem/Desktop/comp125/py Files/untitled3.py', wdir='/
Users/cigdem/Desktop/comp125/py Files')
/Users/cigdem/Desktop/comp125/py Files/untitled3.py:3: VisibleDeprecationWarning:
Creating an ndarray from ragged nested sequences (which is a list-or-tuple of
lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If
you meant to do this, you must specify 'dtype=object' when creating the ndarray
  A1 = np.array(L1)
```

NumPy Arrays – Basic properties

- **ndim** : Number of dimensions
- **shape** : Size of each dimension
- **size** : Total number of elements
- **dtype** : Type of stored data

```
import numpy as np

x = np.random.randint(10, size = (3, 4, 5))
print("ndim:", x.ndim)
print("shape:", x.shape)
print("size:", x.size)
print("dtype:", x.dtype)

y = np.array([3.4, 5])
print("ndim:", y.ndim)
print("shape:", y.shape)
print("size:", y.size)
print("dtype:", y.dtype)
```

```
ndim: 3
shape: (3, 4, 5)
size: 60
dtype: int64


ndim: 1
shape: (2,)
size: 2
dtype: float64
```

NumPy Arrays – Indexing (1D arrays)

```python
import numpy as np

L = [10, 20, 30, 40]
for i in range(len(L)):
    print(L[i])

A = np.array(L)
for i in range(len(A)):
    print(A[i])

B = np.array(L)
for i in range(B.shape[0]):
    print(B[i])
```

Nam	Type	Size	Value
A	Array of int64	(4,)	[10 20 30 40]
B	Array of int64	(4,)	[10 20 30 40]
i	int	1	3
L	list	4	[10, 20, 30, 40]

NumPy Arrays – Indexing (2D arrays)

```python
import numpy as np

L = [[11, 22, 33], [44, 55, 66]]
for i in range(len(L)):
    for j in range(len(L[i])):
        print(L[i][j])

A = np.array(L)
for i in range(len(A)):
    for j in range(len(A[i])):
        print(A[i][j])

B = np.array(L)
for i in range(B.shape[0]):
    for j in range(B.shape[1]):
        print(B[i, j])
```

Nan	Type	Size	Value
A	Array of int64	(2, 3)	[[11 22 33] [44 55 66]]
B	Array of int64	(2, 3)	[[11 22 33] [44 55 66]]
i	int	1	1
j	int	1	2
L	list	2	[[11, 22, 33], [44, 55, 66]]

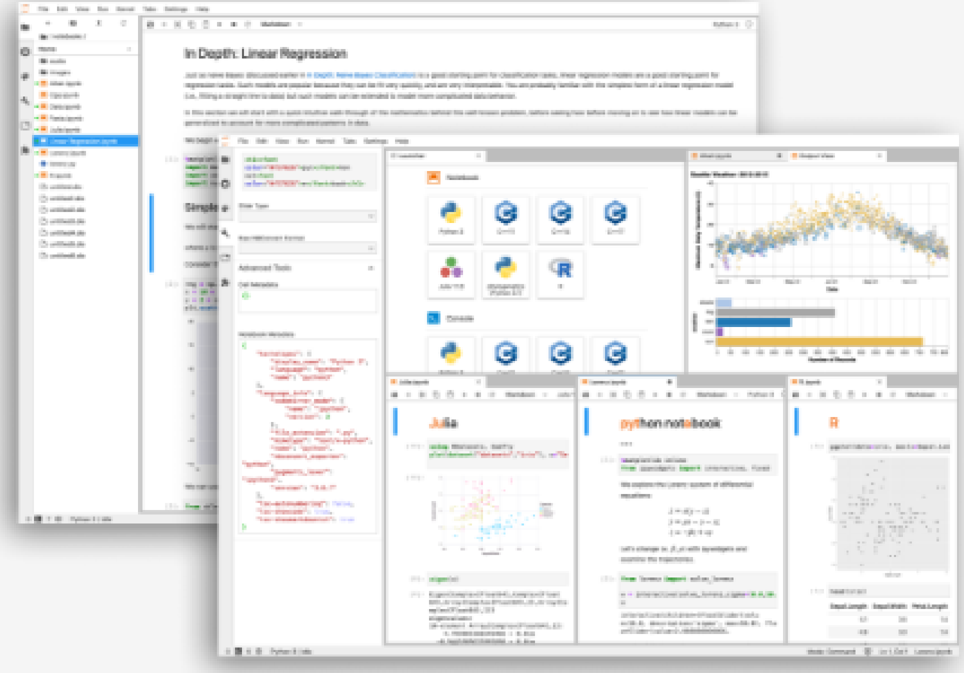
We may also use another way of indexing

JupyterLab

https://jupyter.org/

jupyter

Install About Us Get Involved Events Documentation Widgets News Security



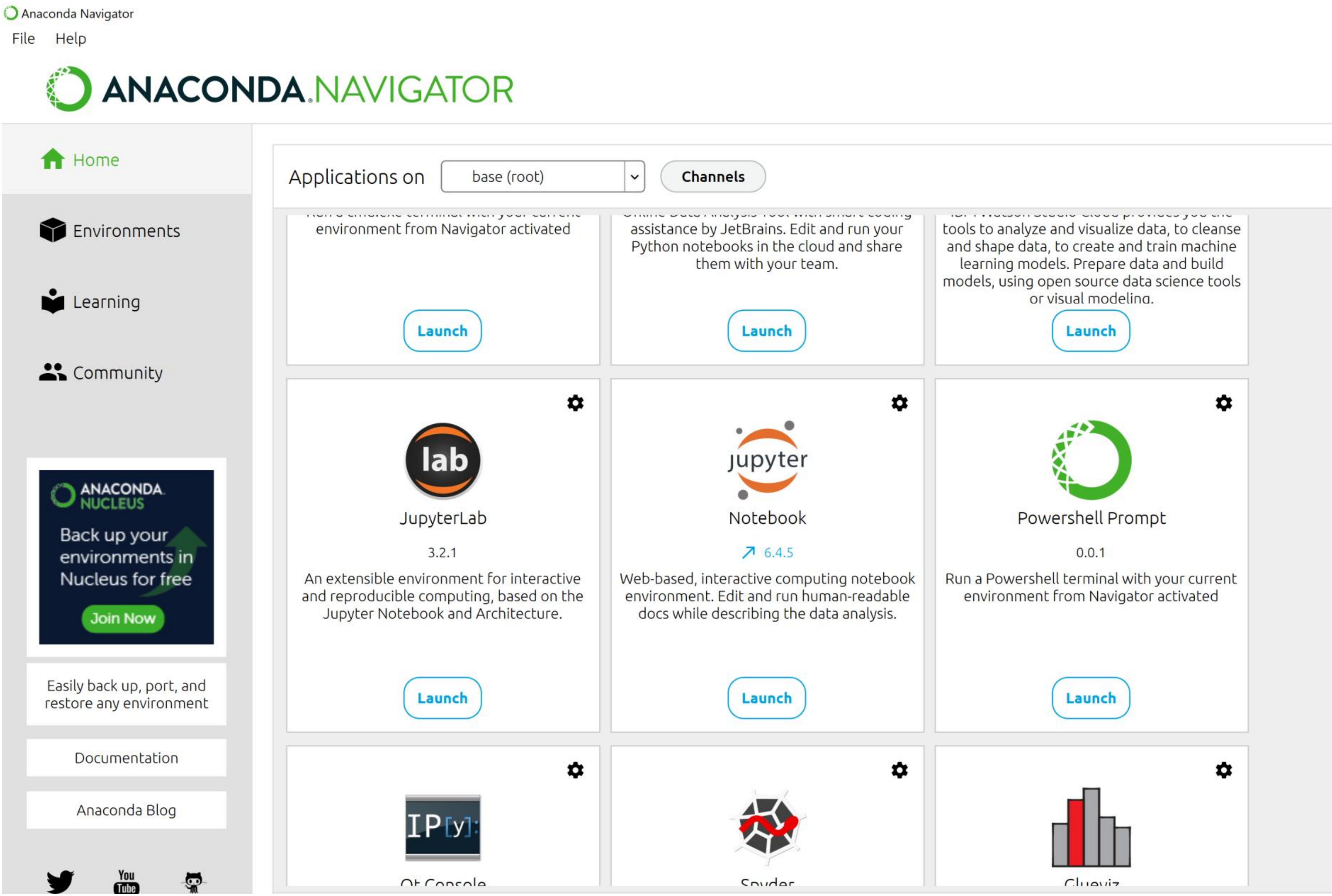
JupyterLab: A Next-Generation Notebook Interface

JupyterLab is a web-based interactive development environment for notebooks, code, and data. Its flexible interface allows users to configure and arrange workflows in data science, scientific computing, computational journalism, and machine learning. A modular design allows for extensions that expand and enrich functionality.

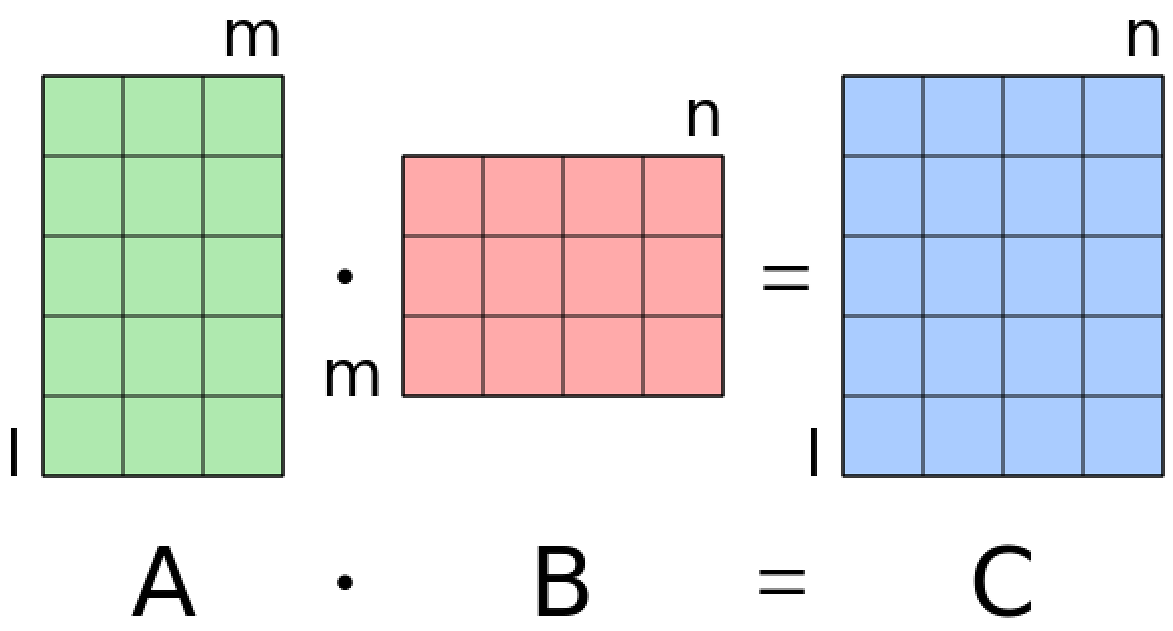
Try it in your browser

Install JupyterLab

Starting JupyterLab from Anaconda



Matrix Multiplication



NumPy:
- `np.dot(A,B)`
- `A.dot(B)`
- `A@B`

$$\begin{pmatrix}0&1\\2&3\end{pmatrix}\begin{pmatrix}4&5\\6&7\end{pmatrix}=\begin{pmatrix}\boxed{6}&7\\26&31\end{pmatrix}$$

$$\begin{pmatrix}0&1\\2&3\end{pmatrix}\begin{pmatrix}4&5\\6&7\end{pmatrix}=\begin{pmatrix}6&\boxed{7}\\26&31\end{pmatrix}$$

$$\begin{pmatrix}0&1\\2&3\end{pmatrix}\begin{pmatrix}4&5\\6&7\end{pmatrix}=\begin{pmatrix}6&7\\\boxed{26}&31\end{pmatrix}$$

$$\begin{pmatrix}0&1\\2&3\end{pmatrix}\begin{pmatrix}4&5\\6&7\end{pmatrix}=\begin{pmatrix}6&7\\26&\boxed{31}\end{pmatrix}$$