## NumPy

NumPy is a Python package used for numerical computation. NumPy is one of the foundational packages for scientific computing with Python. NumPy's core data type is the array and NumPy functions operate on arrays.

## Installing NumPy

Before NumPy's functions and methods can be used, NumPy must be installed. Depending on which distribution of Python you use, the installation method is slightly different.

In [1]:

pip install numpy

Requirement already satisfied: numpy in c:\userdata\anaconda\lib\site-packages (1.19.5) Note: you may need to restart the kernel to use updated packages.

## Verify NumPy installation

In [2]:

*# Load library* **import** numpy **as** np np.version

Out[2]: <module 'numpy.version' from 'C:\\Userdata\\Anaconda\\lib\\site-packages\\numpy\\version.py'>

In [3]:

*# Create a vector as a row*

vector\_row **=** np.array([1, 2, 3])

*# Create a vector as a column*

vector\_column **=** np.array([[1],

[2],

[3]])

*# Create matrix*

matrix **=** np.array([[1, 2, 3],

[4, 5, 6],

[7, 8, 9]])

In [4]:

*# Show this vector*

print(vector\_row)

[1 2 3]

In [5]:

*# Show this column*

print(vector\_column)

[[1]

[2]

[3]]

In [6]:

*# Show this Matrix*

print(matrix)

[[1 2 3]

[4 5 6]

[7 8 9]]

# Describing a Matrix

In [7]:

*# Create matrix*

matrix **=** np.array([[1, 2, 3, 4],

[5, 6, 7, 8],

[9, 10, 11, 12]])

*# View number of rows and columns*

matrix.shape

Out[7]: (3, 4)

In [8]:

*# View number of elements (rows \* columns)*

matrix.size

Out[8]: 12

In [9]:

*# View number of dimensions*

matrix.ndim

Out[9]: 2

# Vector Addition & Scalar Multiplication

In [24]:

*# Vector Addition*

A**=**np.array([[3,2],

[7,5]])

B**=**np.array([[1,3],

[2,2]])

C**=**np.add(A,B)

In [26]:

print(C)

[[4 5]

[9 7]]

In [27]:

*# Scalar Multiplication*

A**=** np.array([[3,2],

[7,5]])

scalar **=** 3

new\_Matrix **=** A **\*** scalar print(new\_Matrix)

[[ 9 6]

[21 15]]

# Multiplication of two Matrices

In [34]:

*# input two matrices*

mat1 **=** ([1, 6, 5],[3 ,4, 8],[2, 12, 3])

mat2 **=** ([3, 4, 6],[5, 6, 7],[6,56, 7])

*## This will return dot product* res **=** np.dot(mat1,mat2) print(res)

[[ 63 320 83]

[ 77 484 102]

[ 84 248 117]]

In [37]:

*# same result will be obtained when we use @ operator # as shown below(only in python &gt;3.5)*

**import** numpy **as** np

*# input two matrices*

mat1 **=** np.array([[1, 6, 5],[3 ,4, 8],[2, 12, 3]])

mat2 **=** np.array([[3, 4, 6],[5, 6, 7],[6,56, 7]])

*# This will return matrix product of two array*

res **=** mat1 **@** mat2

*# print resulted matrix*

print(res)

[[ 63 320 83]

[ 77 484 102]

[ 84 248 117]]

In [38]:

*## Cross Product*

x **=** np.array([[1,2,3], [4,5,6]])

y **=** np.array([[4,5,6], [1,2,3]])

np.cross(x, y)

Out[38]: array([[-3, 6, -3],

[ 3, -6, 3]])

# Inverses and Determinants

In [31]:

A **=** np.array([[1,1],

[2,3]])

*# Calculating the inverse of the matrix*

B**=**np.linalg.inv(A) print(B)

[[ 3. -1.]

[-2. 1.]]

In [33]:

*# Check if B is a right inverse of A*

C**=**np.dot(A,B) print(C)

[[1. 0.]

[0. 1.]]

In [39]:

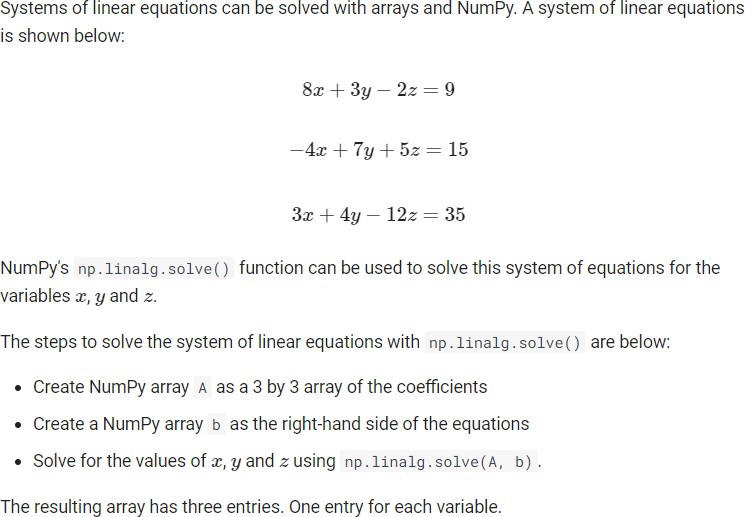
*# Determinants*

a **=** np.array([ [[1, 2], [3, 4]], [[1, 2], [2, 1]], [[1, 3], [3, 1]] ])

a.shape np.linalg.det(a)

Out[39]: array([-2., -3., -8.])

## Systems of linear equations



In [10]:

**import** numpy **as** np

*## creat matrix*

A **=** np.array([[8, 3, **-**2], [**-**4, 7, 5], [3, 4, **-**12]])

b **=** np.array([9, 15, 35])

*## solve this linear equation*

x **=** np.linalg.solve(A, b) x

Out[10]: array([-0.58226371, 3.22870478, -1.98599767])

# 4x + 3y + 2z = 25

**-2x + 2y + 3z = -10 3x -5y + 2z = -4**

Bulid your own code to solve this linear equation

Answer : [ 5. 3. -2.]

# Vector space, subspace, span, column space, row space, null space, left-null space, rank, basis

Imagine we have a matrix A, in python, A.T means transpose of A, @ means matrix multiplication.

In [ ]:

a **=** np.random.randn(2,3) a

*# array([[ 0.39, 0.54, -0.71],*

*# [-1.84, -0.6 , 0.53]])*

a.T

*# array([[ 0.39, -1.84],*

*# [ 0.54, -0.6 ],*

*# [-0.71, 0.53]])*

b **=** np.random.randn(3,2) b

*# array([[ 1., -0.],*

*# [ 1., -0.],*

*# [-3., 1.]])*

a **@** b

*# array([[-1.09762048, 0.96911979],*

What is a span? The span of a set of vectors is all linear combinations of these vectors. Think about vector (0,1) and (1,0), a span of these two vectors would be the whole x-y plane.

Then thinking about a x-y-z 3D space (Vector space R3), it is composed of basis — (1,0,0), (0,1,0), and (0,0,1), since every element in Vector space can be written as a linear combination of the elements in the basis.

Another concept is subspace, Span of Vector (1,0,0) and (0,1,0) constitutes a subspace of x-y-z 3D vector space R3. (A subset of a larger vector space)

Column space of A is the span of all column vectors of A, row space of A is the span of all row vectors of A. If A @ x=0, the span of all solutions x constitutes the null space of A. If A.T @ x= 0, span of all solutions x constitutes the left-null space of A. These four spaces will keep occuring in lots of linear algebra tutorials.

How many linearly independent column vectors in matrix A? This is rank of A. To better understand linearly independent, for instance (1,2,3) and (10,20,30) are linear dependent, because (10,20,30) is a multiple of (1,2,3)! Then how to compute the rank of a matrix?

In [ ]:

x **=** np.random.rand(2,3) np.linalg.matrix\_rank(x) *# 2*

*# So, x has two linearly independent column vectors*

In [ ]:

In [ ]: