Google Developer Student Clubs

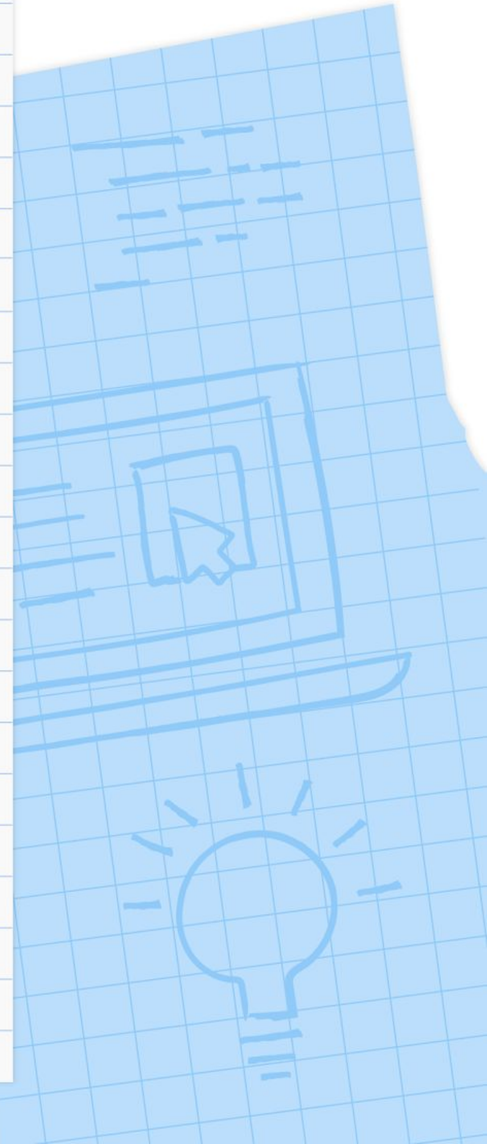
# Python NumPy

## Linear Algebra and Scientific Computing

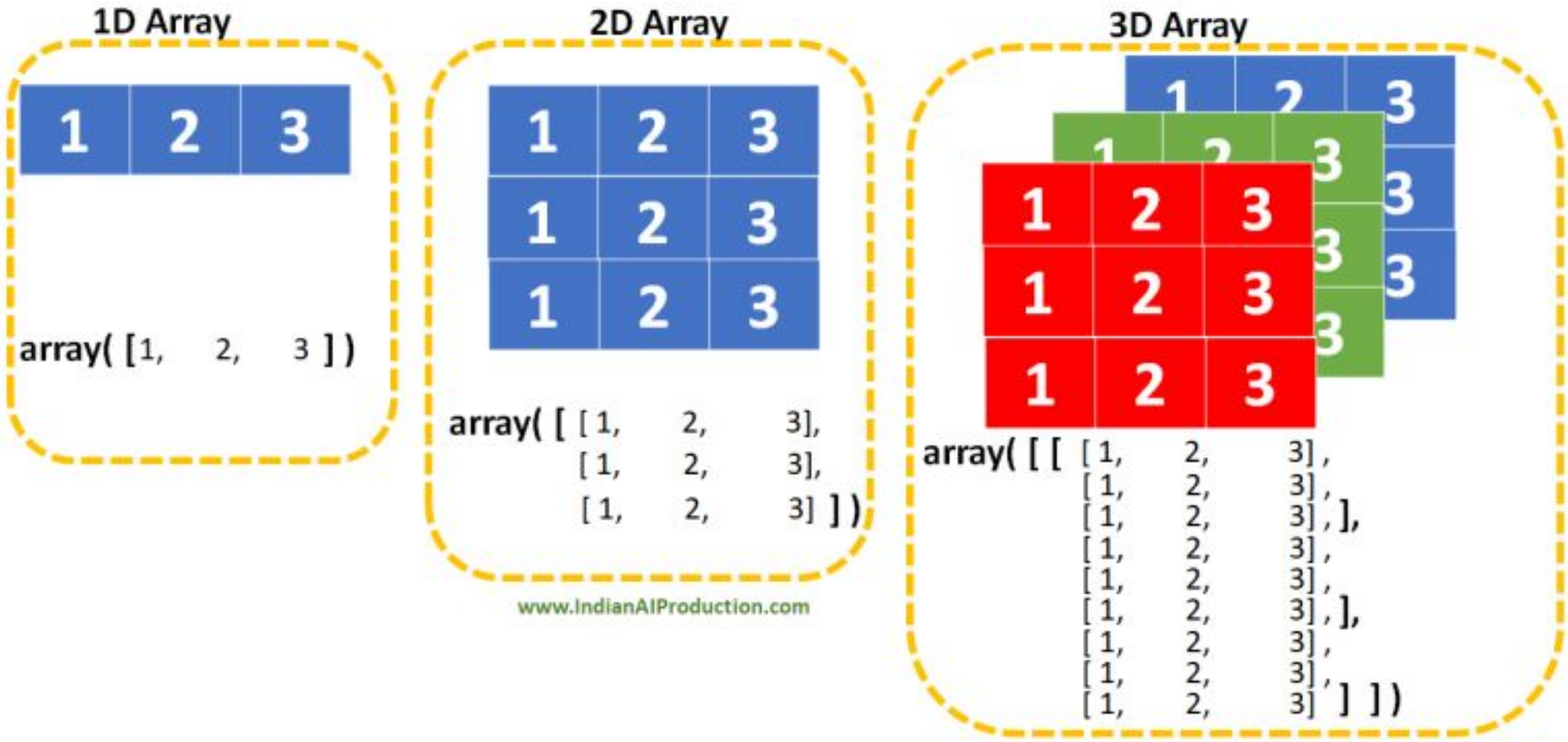
By: Mouafak Dakhlaoui

# What is NumPy?

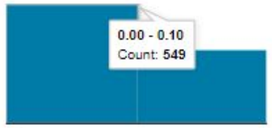
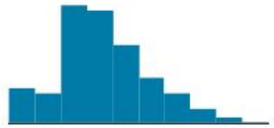
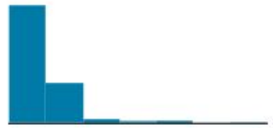
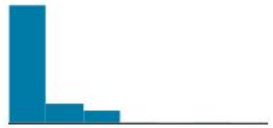
NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.



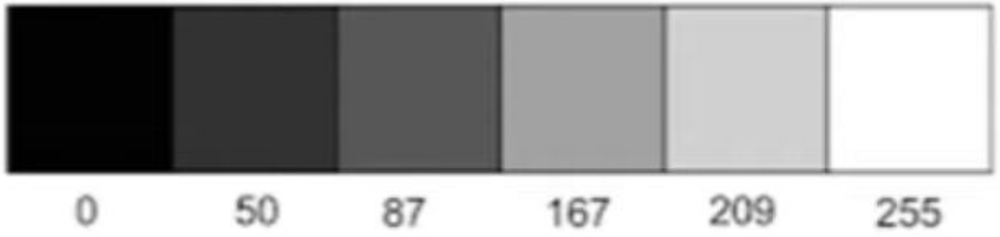
# NumPy - ndarray

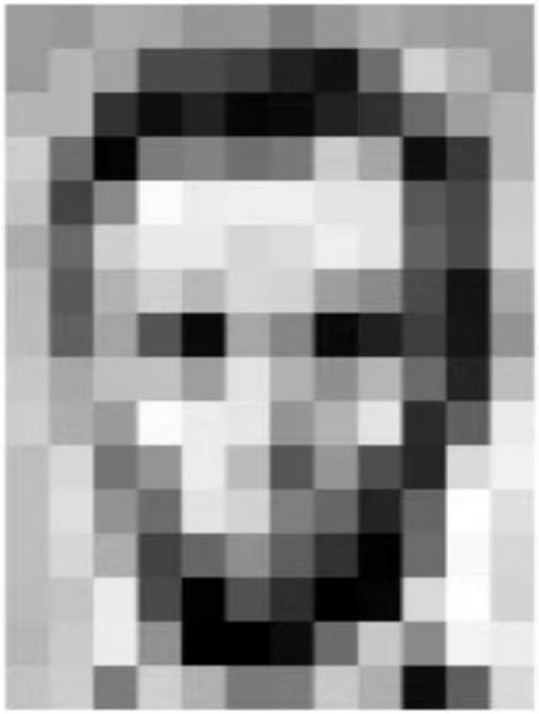


# NumPy - 2D Array (1)

| PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket |
|---|---|---|---|---|---|---|---|---|
|  1 891 |   0 1 |  1 3 | 891 unique values | male 65%<br>female 35% |  0.42 80 |  0 8 |  0 6 | 6 unique |
| 1 | 0 | 3 | Braund, Mr. Owen Harris | male | 22 | 1 | 0 | A/5 21171 |
| 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Thayer) | female | 38 | 1 | 0 | PC 17599 |
| 3 | 1 | 3 | Heikkinen, Miss. Laina | female | 26 | 0 | 0 | STON/O2. 3 |
| 4 | 1 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | female | 35 | 1 | 0 | 113803 |
| 5 | 0 | 3 | Allen, Mr. William Henry | male | 35 | 0 | 0 | 373450 |
| 6 | 0 | 3 | Moran, Mr. James | male | | 0 | 0 | 330877 |
| 7 | 0 | 1 | McCarthy, Mr. Timothy J | male | 54 | 0 | 0 | 17463 |
| 8 | 0 | 3 | Palsson, Master. Gosta Leonard | male | 2 | 3 | 1 | 349909 |
| 9 | 1 | 3 | Johnson, Mrs. Oscar | female | 27 | 0 | 2 | 347742 |

# NumPy - 2D Array (2)

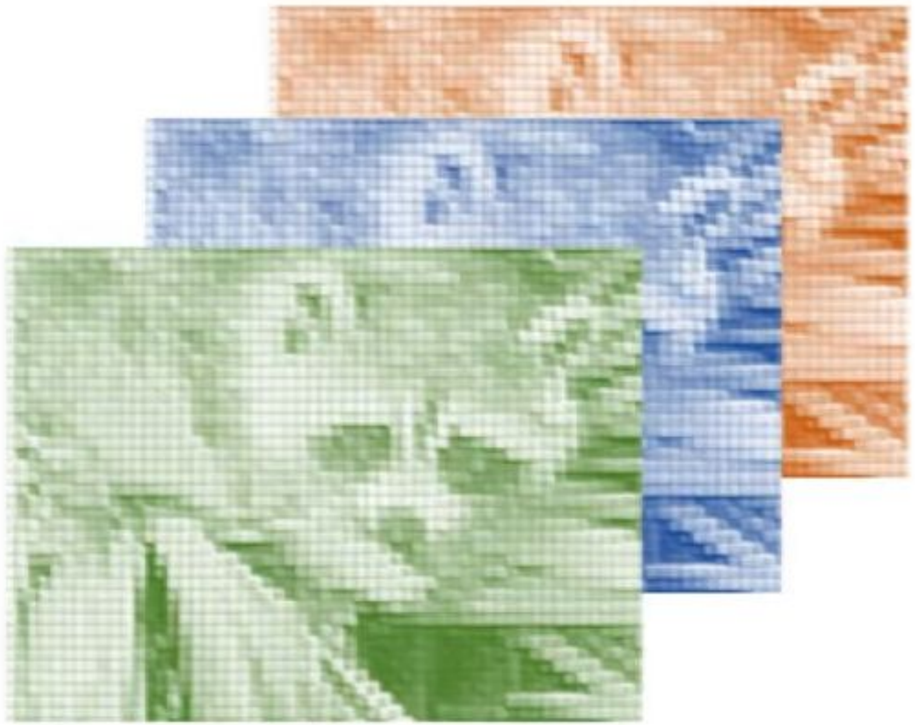


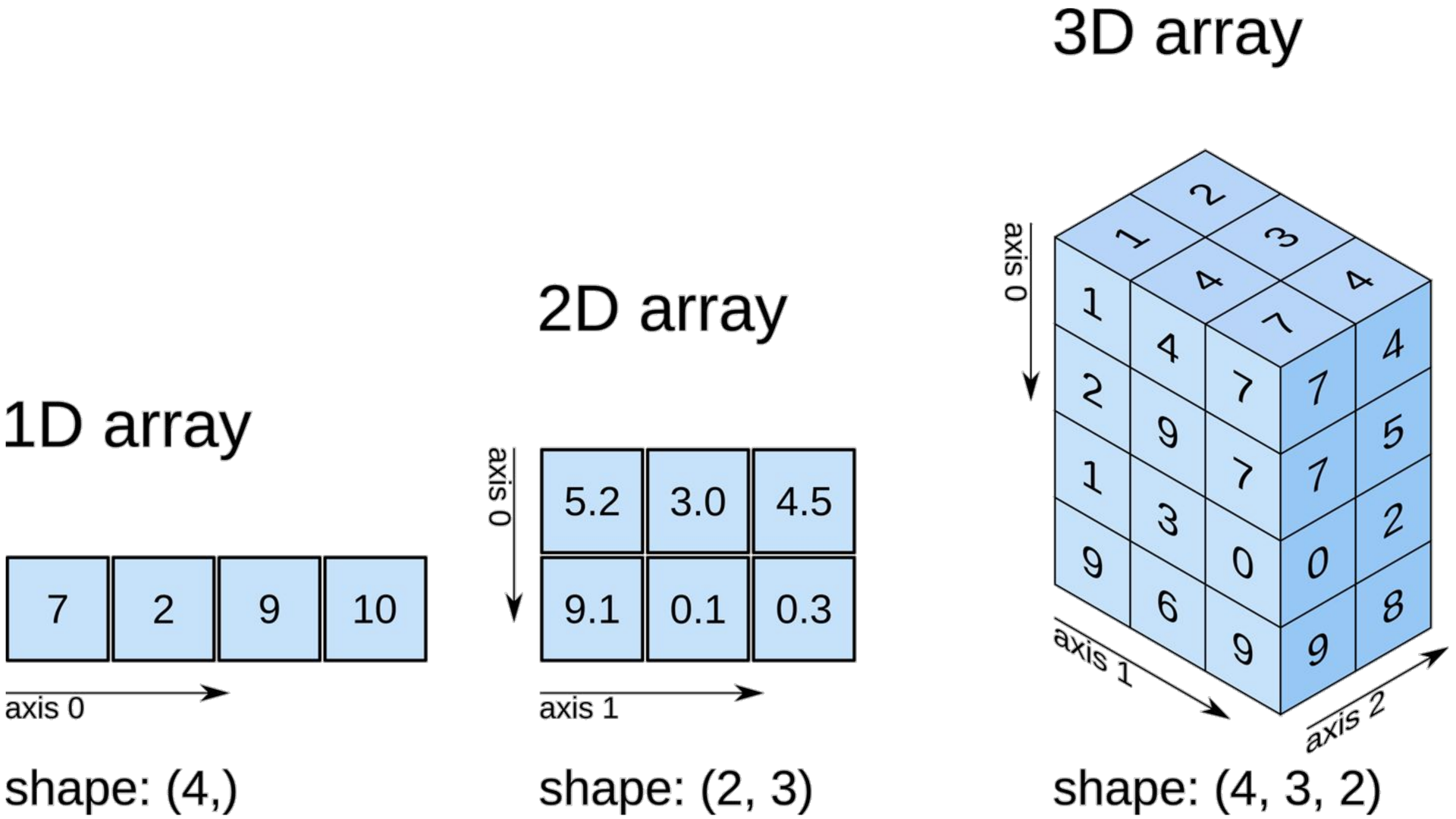
| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 157 | 153 | 174 | 168 | 150 | 152 | 129 | 151 | 172 | 161 | 155 | 156 |
| 155 | 182 | 163 | 74 | 75 | 62 | 33 | 17 | 110 | 210 | 180 | 154 |
| 180 | 180 | 50 | 14 | 34 | 6 | 10 | 33 | 48 | 106 | 159 | 181 |
| 206 | 109 | 5 | 124 | 131 | 111 | 120 | 204 | 166 | 15 | 56 | 180 |
| 194 | 68 | 137 | 251 | 237 | 239 | 239 | 228 | 227 | 87 | 71 | 201 |
| 172 | 105 | 207 | 233 | 233 | 214 | 220 | 239 | 228 | 98 | 74 | 206 |
| 188 | 88 | 179 | 209 | 185 | 215 | 211 | 158 | 139 | 75 | 20 | 169 |
| 189 | 97 | 165 | 84 | 10 | 168 | 134 | 11 | 31 | 62 | 22 | 148 |
| 199 | 168 | 191 | 193 | 158 | 227 | 178 | 143 | 182 | 106 | 36 | 190 |
| 205 | 174 | 155 | 252 | 236 | 231 | 149 | 178 | 228 | 43 | 95 | 234 |
| 190 | 216 | 116 | 149 | 236 | 187 | 86 | 150 | 79 | 38 | 218 | 241 |
| 190 | 224 | 147 | 108 | 227 | 210 | 127 | 102 | 36 | 101 | 255 | 224 |
| 190 | 214 | 173 | 66 | 103 | 143 | 96 | 50 | 2 | 109 | 249 | 215 |
| 187 | 196 | 235 | 75 | 1 | 81 | 47 | 0 | 6 | 217 | 255 | 211 |
| 183 | 202 | 237 | 145 | 0 | 0 | 12 | 108 | 200 | 138 | 243 | 236 |
| 195 | 206 | 123 | 207 | 177 | 121 | 123 | 200 | 175 | 13 | 96 | 218 |


| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 157 | 153 | 174 | 168 | 150 | 152 | 129 | 151 | 172 | 161 | 155 | 156 |
| 155 | 182 | 163 | 74 | 75 | 62 | 33 | 17 | 110 | 210 | 180 | 154 |
| 180 | 180 | 50 | 14 | 34 | 6 | 10 | 33 | 48 | 106 | 159 | 181 |
| 206 | 109 | 5 | 124 | 131 | 111 | 120 | 204 | 166 | 15 | 56 | 180 |
| 194 | 68 | 137 | 251 | 237 | 239 | 239 | 228 | 227 | 87 | 71 | 201 |
| 172 | 105 | 207 | 233 | 233 | 214 | 220 | 239 | 228 | 98 | 74 | 206 |
| 188 | 88 | 179 | 209 | 185 | 215 | 211 | 158 | 139 | 75 | 20 | 169 |
| 189 | 97 | 165 | 84 | 10 | 168 | 134 | 11 | 31 | 62 | 22 | 148 |
| 199 | 168 | 191 | 193 | 158 | 227 | 178 | 143 | 182 | 106 | 36 | 190 |
| 205 | 174 | 155 | 252 | 236 | 231 | 149 | 178 | 228 | 43 | 95 | 234 |
| 190 | 216 | 116 | 149 | 236 | 187 | 86 | 150 | 79 | 38 | 218 | 241 |
| 190 | 224 | 147 | 108 | 227 | 210 | 127 | 102 | 36 | 101 | 255 | 224 |
| 190 | 214 | 173 | 66 | 103 | 143 | 96 | 50 | 2 | 109 | 249 | 215 |
| 187 | 196 | 235 | 75 | 1 | 81 | 47 | 0 | 6 | 217 | 255 | 211 |
| 183 | 202 | 237 | 145 | 0 | 0 | 12 | 108 | 200 | 138 | 243 | 236 |
| 195 | 206 | 123 | 207 | 177 | 121 | 123 | 200 | 175 | 13 | 96 | 218 |

# NumPy - 3D Array



# NumPy - shape


1D array
7
2
9
10
axis 0
shape: (4,)
2D array
axis 0
5.2
3.0
4.5
9.1
0.1
0.3
axis 1
shape: (2, 3)
3D array
axis 0
axis 1
axis 2
shape: (4, 3, 2)

# numpy.array

## Create an array

```
1  import numpy as np
2
3  A = np.array([1, 2, 3, 4])
4
5  print(A)
6  print(A.shape)
```

```
[1 2 3 4]
(4,)
```

Google Developer Student Clubs

# NumPy - other ndarray Constructors

numpy.**full**(**shape**, **fill_value**)

*Return a new array of given shape, filled with fill_value.*

numpy.**zeros**(**shape**)

*Return a new array of given shape, filled with zeros.*

numpy.**ones**(**shape**)

*Return a new array of given shape, filled with ones.*

numpy.**eye**()

*Return a 2-D array with ones on the diagonal and zeros elsewhere.*

numpy.**random**.**randn**(**shape**)

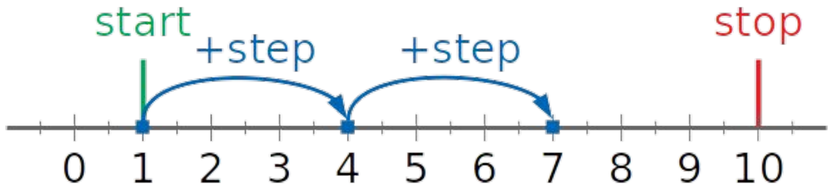
*Return a sample (or samples) from the "standard normal" distribution.*

# NumPy - 1D array Constructors

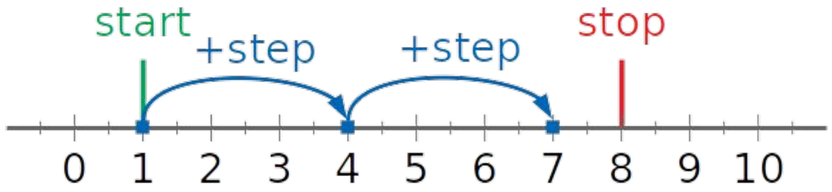
numpy.**linspace**(**start**, **stop**, **num**)
numpy.**arange**(**start**, **stop**, **step**)

Both **linspace** and **arange** return evenly spaced numbers over a specified interval.

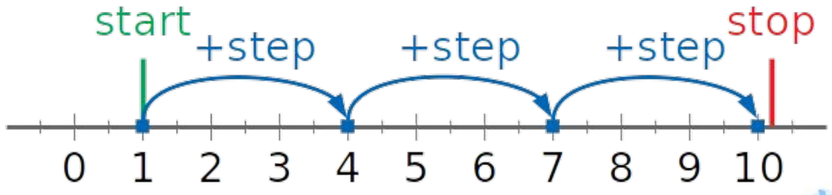
```
>>> np.arange(1, 10, 3)
array([1, 4, 7])
```




```
>>> np.arange(1, 8, 3)
array([1, 4, 7])
```




```
>>> np.arange(1, 10.1, 3)
array([1., 4., 7., 10.])
```



# NumPy - dtypes

## Array types and conversions between types

NumPy supports a much greater variety of numerical types than Python does. This section shows which are available, and how to modify an array's data-type.

The primitive types supported are tied closely to those in C:

| Numpy type | C type | Description |
|---|---|---|
| `numpy.bool_` | `bool` | Boolean (True or False) stored as a byte |
| `numpy.byte` | `signed char` | Platform-defined |
| `numpy.ubyte` | `unsigned char` | Platform-defined |
| `numpy.short` | `short` | Platform-defined |
| `numpy.ushort` | `unsigned short` | Platform-defined |
| `numpy.intc` | `int` | Platform-defined |
| `numpy.uintc` | `unsigned int` | Platform-defined |
| `numpy.int` | `long` | Platform-defined |

# NumPy - np.hstack, np.vstack

a

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |

c

| | |
|---|---|
| 1 | 2 |
| 3 | 4 |
| 5 | 6 |

np.hstack((a, c))

| | | | | | |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 1 | 2 |
| 5 | 6 | 7 | 8 | 3 | 4 |
| 9 | 10 | 11 | 12 | 5 | 6 |

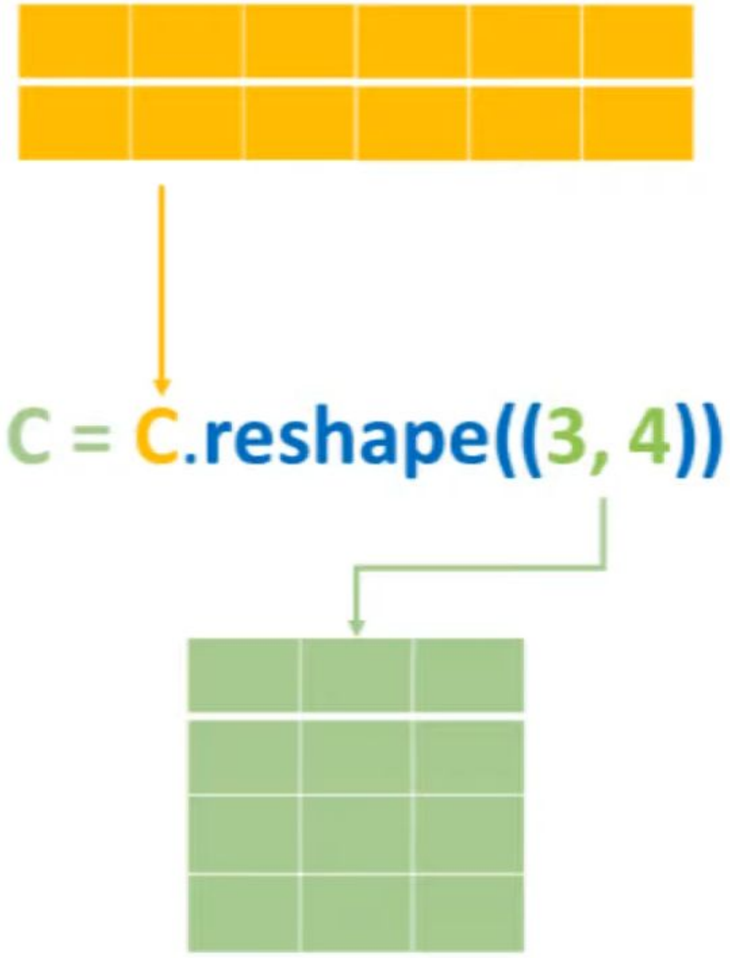
b

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |

np.vstack((a, b))

| | | | |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |

# NumPy - ndarray.reshape

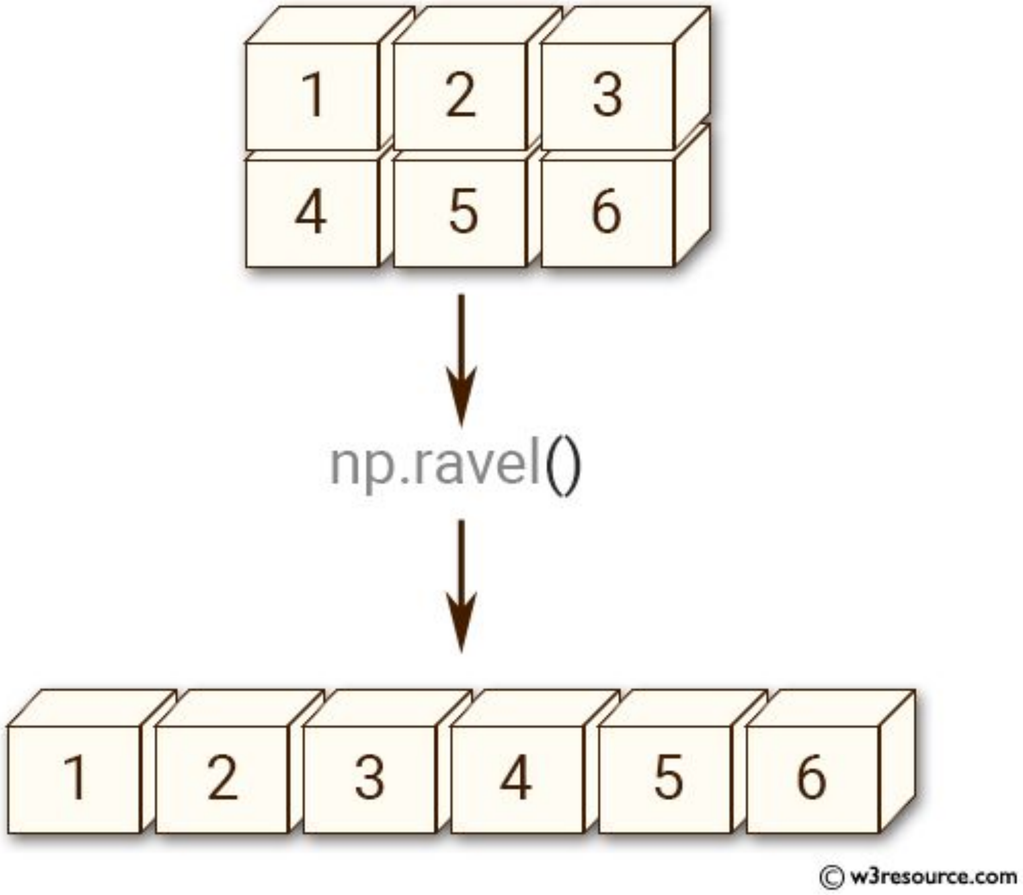



ndarray.**reshape**(**shape**)

*Returns an array containing the same data with a new shape.*

Google Developer Student Clubs

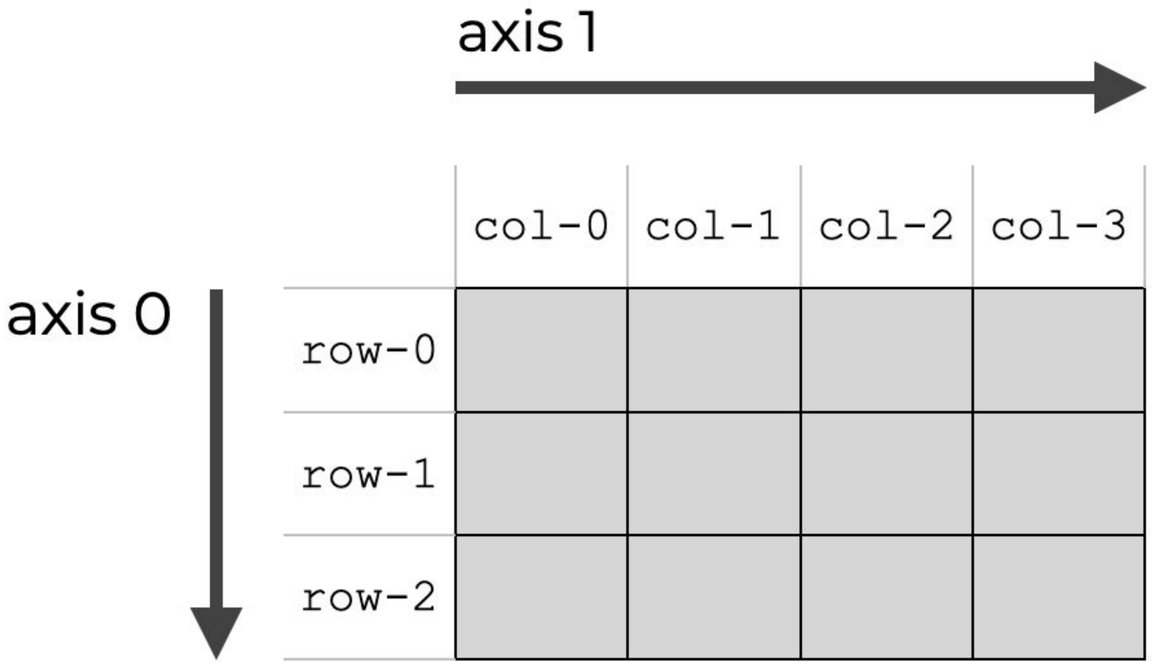
# NumPy - ndarray.ravel




ndarray.**ravel**()

*Returns a flattened array.*

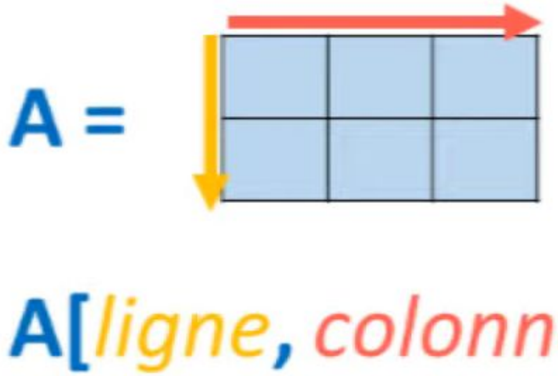
# NumPy - Axes



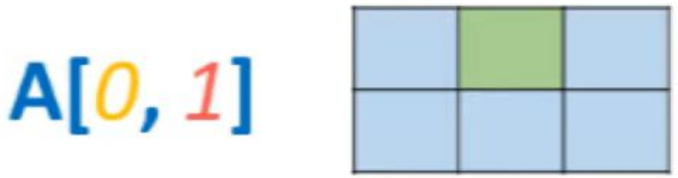

NumPy axes are **the directions along the rows and columns**. Just like coordinate systems, NumPy arrays also have axes. In a 2-dimensional NumPy array, the axes are the directions along the rows and columns.

Google Developer Student Clubs

# NumPy - Indexing

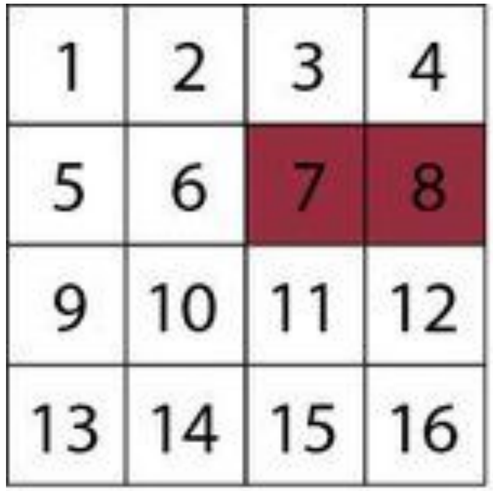



**Array indexing is the same as accessing an array element.** You can access an array element by referring to its index number. The indexes in NumPy arrays start with 0, meaning that the first element has index 0, and the second has index 1 etc.

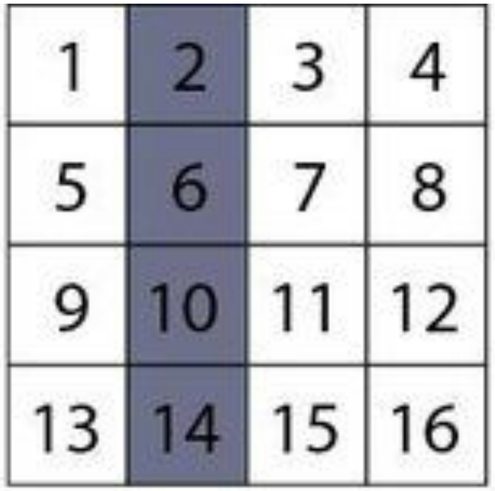

Google Developer Student Clubs

# NumPy - Slicing



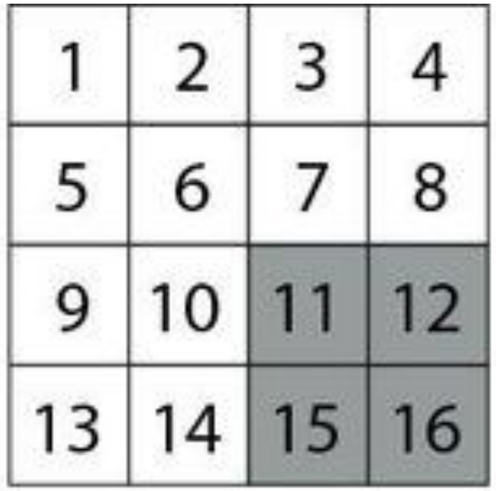

Row one, columns two to four

```
>>> arr[1, 2:4]
array([7, 8])
```



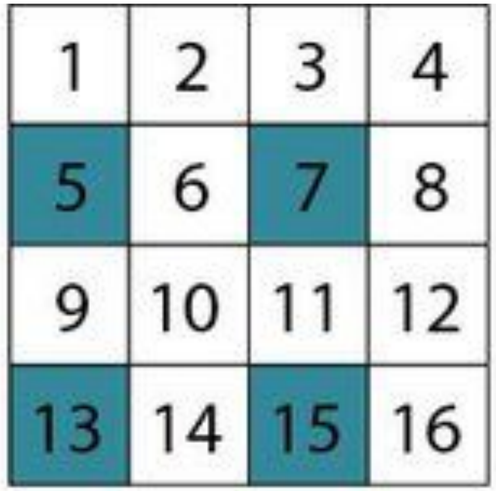

All rows in column one

```
>>> arr[:, 1]
array([2,  6, 10, 14])
```




All rows after row two,
all columns after column two

```
>>> arr[2:, 2:]
array([[11, 12],
       [15, 16]])
```




Every other row after row one,
every other column

```
>>> arr[1::2, ::2]
array([[5, 7],
       [13, 15]])
```

Google Developer Student Clubs

# NumPy - Boolean Indexing

**A =**

| | | |
|---|---|---|
| 5 | 4 | 5 |
| 4 | 4 | 4 |
| 5 | 4 | 5 |

**A < 5** →

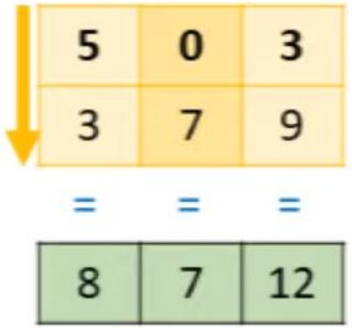
| | | |
|---|---|---|
| F | T | F |
| T | T | T |
| F | T | F |

**A[A < 5] = 10** →

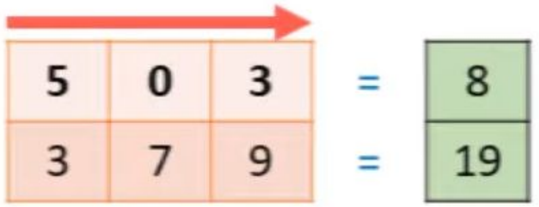
| | | |
|---|---|---|
| 5 | 10 | 5 |
| 10 | 10 | 10 |
| 5 | 10 | 5 |

You can index specific values from a NumPy array using another NumPy array of Boolean values on one axis to specify the indices you want to access.

# NumPy - ndarray.sum

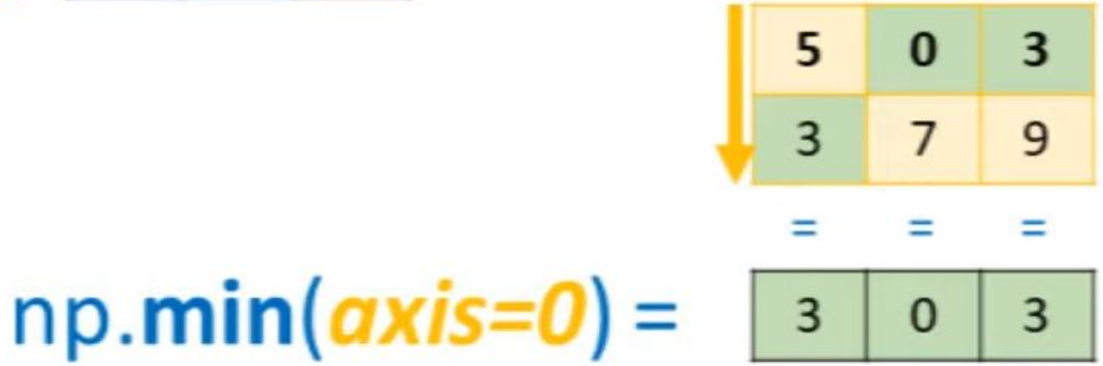



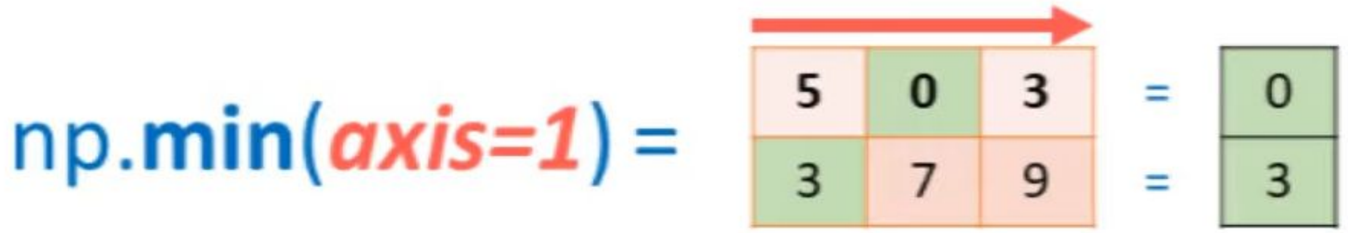

A.sum(*axis=0*) =




A.sum(*axis=1*) =

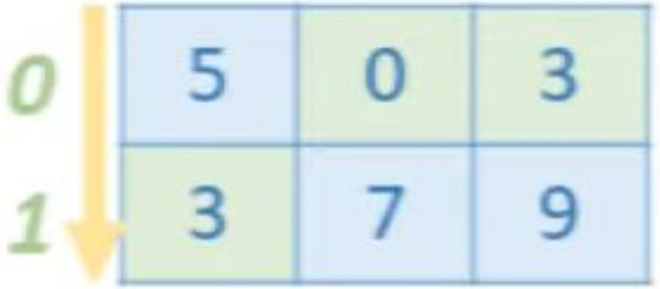
# NumPy - ndarray.min, ndarray.max

# NumPy - ndarray.argmin, ndarray.argmax


Return indices of the minimum values along the given **axis**.

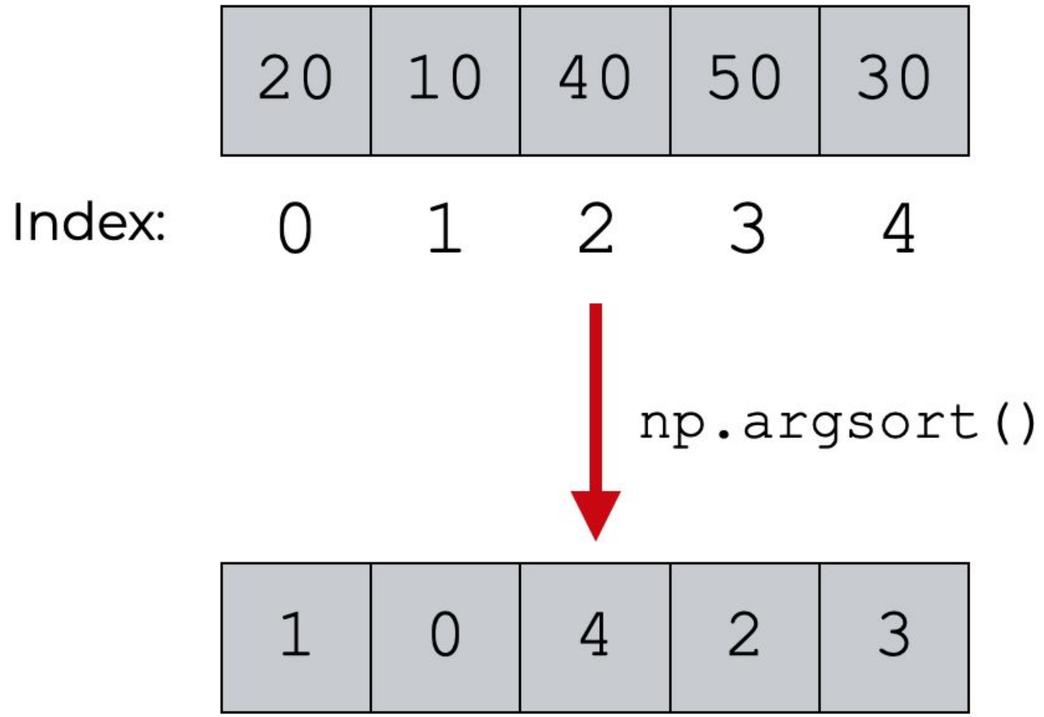



**argmin(*axis=0*) =**

| | | |
|---|---|---|
| 1 | 0 | 0 |

# NumPy - argsort

`np.argsort()` RETURNS THE INDEX VALUES
IN AN ORDER THAT WOULD SORT THE INPUT ARRAY



# NumPy - Mathematical Functions

## Trigonometric functions

| | |
|---|---|
| **sin**(x, /[, out, where, casting, order, ...]) | Trigonometric sine, element-wise. |
| **cos**(x, /[, out, where, casting, order, ...]) | Cosine element-wise. |
| **tan**(x, /[, out, where, casting, order, ...]) | Compute tangent element-wise. |
| **arcsin**(x, /[, out, where, casting, order, ...]) | Inverse sine, element-wise. |
| **arccos**(x, /[, out, where, casting, order, ...]) | Trigonometric inverse cosine, element-wise. |
| **arctan**(x, /[, out, where, casting, order, ...]) | Trigonometric inverse tangent, element-wise. |
| **hypot**(x1, x2, /[, out, where, casting, ...]) | Given the "legs" of a right triangle, return its hypotenuse. |

## Hyperbolic functions

| | |
|---|---|
| **sinh**(x, /[, out, where, casting, order, ...]) | Hyperbolic sine, element-wise. |
| **cosh**(x, /[, out, where, casting, order, ...]) | Hyperbolic cosine, element-wise. |
| **tanh**(x, /[, out, where, casting, order, ...]) | Compute hyperbolic tangent element-wise. |
| **arcsinh**(x, /[, out, where, casting, order, ...]) | Inverse hyperbolic sine element-wise. |
| **arccosh**(x, /[, out, where, casting, order, ...]) | Inverse hyperbolic cosine, element-wise. |
| **arctanh**(x, /[, out, where, casting, order, ...]) | Inverse hyperbolic tangent element-wise. |

## Exponents and logarithms #

| | |
|---|---|
| **exp**(x, /[, out, where, casting, order, ...]) | Calculate the exponential of all elements in the input array. |
| **expm1**(x, /[, out, where, casting, order, ...]) | Calculate `exp(x) - 1` for all elements in the array. |
| **exp2**(x, /[, out, where, casting, order, ...]) | Calculate *2**p* for all *p* in the input array. |
| **log**(x, /[, out, where, casting, order, ...]) | Natural logarithm, element-wise. |
| **log10**(x, /[, out, where, casting, order, ...]) | Return the base 10 logarithm of the input array, element-wise. |
| **log2**(x, /[, out, where, casting, order, ...]) | Base-2 logarithm of *x*. |
| **log1p**(x, /[, out, where, casting, order, ...]) | Return the natural logarithm of one plus the input array, element-wise. |

## Sums, products, differences

| | |
|---|---|
| **prod**(a[, axis, dtype, out, keepdims, ...]) | Return the product of array elements over a given axis. |
| **sum**(a[, axis, dtype, out, keepdims, ...]) | Sum of array elements over a given axis. |
| **nanprod**(a[, axis, dtype, out, keepdims, ...]) | Return the product of array elements over a given axis treating Not a Numbers (NaNs) as ones. |
| **nansum**(a[, axis, dtype, out, keepdims, ...]) | Return the sum of array elements over a given axis treating Not a Numbers (NaNs) as zero. |
| **cumprod**(a[, axis, dtype, out]) | Return the cumulative product of elements along a given axis. |
| **cumsum**(a[, axis, dtype, out]) | Return the cumulative sum of the elements along a given axis. |

Mathematical functions — NumPy v1.23 Manual

# NumPy - Statistics

## Order statistics

| | |
|---|---|
| **ptp**(a[, axis, out, keepdims]) | Range of values (maximum - minimum) along an axis. |
| **percentile**(a, q[, axis, out, ...]) | Compute the q-th percentile of the data along the specified axis. |
| **nanpercentile**(a, q[, axis, out, ...]) | Compute the qth percentile of the data along the specified axis, while ignoring nan values. |
| **quantile**(a, q[, axis, out, overwrite_input, ...]) | Compute the q-th quantile of the data along the specified axis. |
| **nanquantile**(a, q[, axis, out, ...]) | Compute the qth quantile of the data along the specified axis, while ignoring nan values. |

## Correlating

| | |
|---|---|
| **corrcoef**(x[, y, rowvar, bias, ddof, dtype]) | Return Pearson product-moment correlation coefficients. |
| **correlate**(a, v[, mode]) | Cross-correlation of two 1-dimensional sequences. |
| **cov**(m[, y, rowvar, bias, ddof, fweights, ...]) | Estimate a covariance matrix, given data and weights. |

## Averages and variances

| | |
|---|---|
| **median**(a[, axis, out, overwrite_input, keepdims]) | Compute the median along the specified axis. |
| **average**(a[, axis, weights, returned, keepdims]) | Compute the weighted average along the specified axis. |
| **mean**(a[, axis, dtype, out, keepdims, where]) | Compute the arithmetic mean along the specified axis. |
| **std**(a[, axis, dtype, out, ddof, keepdims, where]) | Compute the standard deviation along the specified axis. |
| **var**(a[, axis, dtype, out, ddof, keepdims, where]) | Compute the variance along the specified axis. |
| **nanmedian**(a[, axis, out, overwrite_input, ...]) | Compute the median along the specified axis, while ignoring NaNs. |
| **nanmean**(a[, axis, dtype, out, keepdims, where]) | Compute the arithmetic mean along the specified axis, ignoring NaNs. |

## Histograms #

| | |
|---|---|
| **histogram**(a[, bins, range, normed, weights, ...]) | Compute the histogram of a dataset. |
| **histogram2d**(x, y[, bins, range, normed, ...]) | Compute the bi-dimensional histogram of two data samples. |
| **histogramdd**(sample[, bins, range, normed, ...]) | Compute the multidimensional histogram of some data. |
| **bincount**(x, /[, weights, minlength]) | Count number of occurrences of each value in array of non-negative ints. |
| **histogram_bin_edges**(a[, bins, range, weights]) | Function to calculate only the edges of the bins used by the **histogram** function. |
| **digitize**(x, bins[, right]) | Return the indices of the bins to which each value in input array belongs. |

Statistics — NumPy v1.23 Manual

# NumPy - Linear Algebra

## Matrix and vector products

| | |
|---|---|
| **dot**(a, b[, out]) | Dot product of two arrays. |
| **linalg.multi_dot**(arrays, *[, out]) | Compute the dot product of two or more arrays in a single function call, while automatically selecting the fastest evaluation order. |
| **vdot**(a, b, /) | Return the dot product of two vectors. |
| **inner**(a, b, /) | Inner product of two arrays. |
| **outer**(a, b[, out]) | Compute the outer product of two vectors. |
| **matmul**(x1, x2, /[, out, casting, order, ...]) | Matrix product of two arrays. |
| **tensordot**(a, b[, axes]) | Compute tensor dot product along specified axes. |
| **einsum**(subscripts, *operands[, out, dtype, ...]) | Evaluates the Einstein summation convention on the operands. |

## Decompositions

| | |
|---|---|
| **linalg.cholesky**(a) | Cholesky decomposition. |
| **linalg.qr**(a[, mode]) | Compute the qr factorization of a matrix. |
| **linalg.svd**(a[, full_matrices, compute_uv, ...]) | Singular Value Decomposition. |

## Matrix eigenvalues

| | |
|---|---|
| **linalg.eig**(a) | Compute the eigenvalues and right eigenvectors of a square array. |
| **linalg.eigh**(a[, UPLO]) | Return the eigenvalues and eigenvectors of a complex Hermitian (conjugate symmetric) or a real symmetric matrix. |
| **linalg.eigvals**(a) | Compute the eigenvalues of a general matrix. |
| **linalg.eigvalsh**(a[, UPLO]) | Compute the eigenvalues of a complex Hermitian or real symmetric matrix. |

## Solving equations and inverting matrices

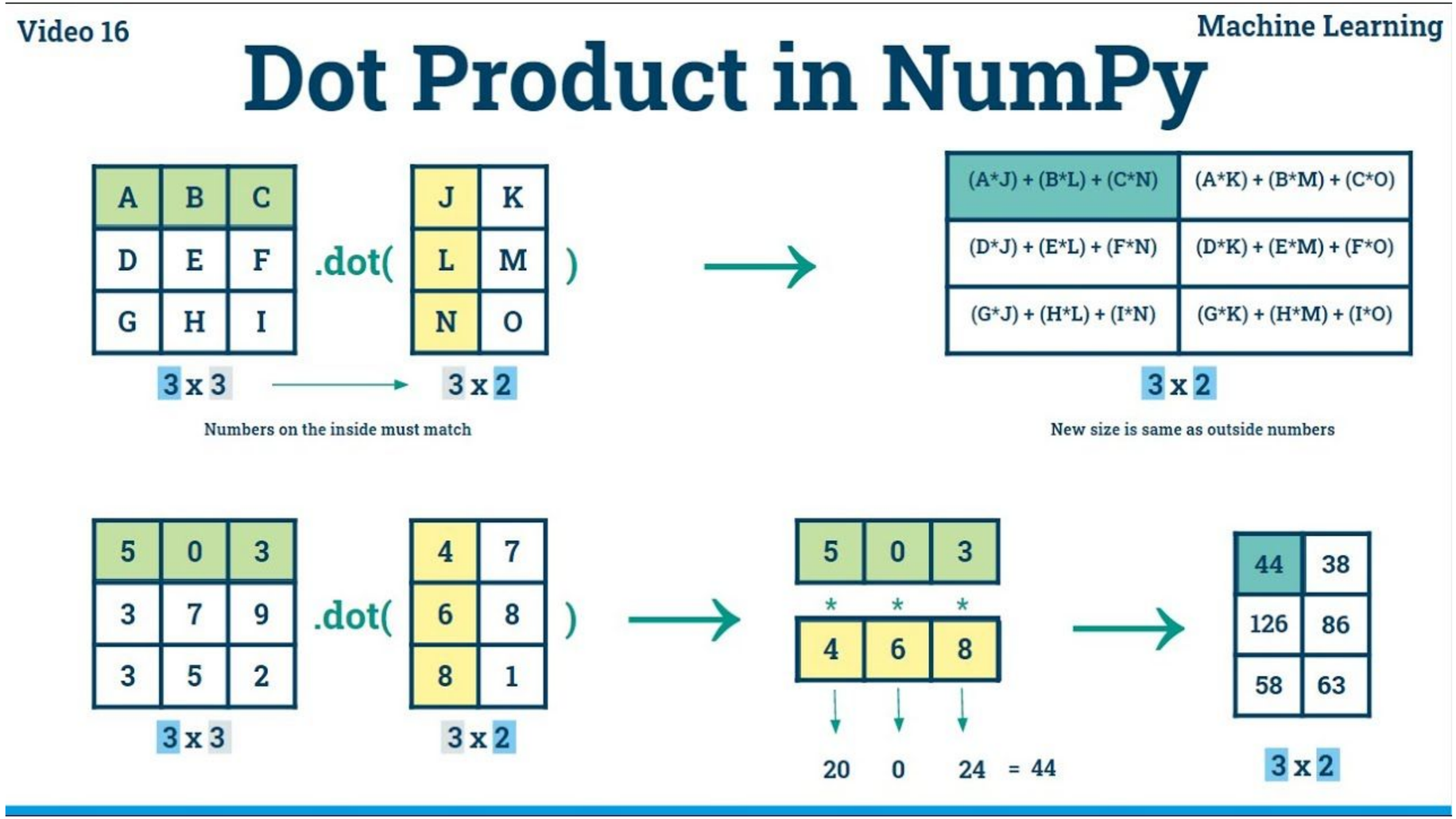
| | |
|---|---|
| **linalg.solve**(a, b) | Solve a linear matrix equation, or system of linear scalar equations. |
| **linalg.tensorsolve**(a, b[, axes]) | Solve the tensor equation `a x = b` for x. |
| **linalg.lstsq**(a, b[, rcond]) | Return the least-squares solution to a linear matrix equation. |
| **linalg.inv**(a) | Compute the (multiplicative) inverse of a matrix. |
| **linalg.pinv**(a[, rcond, hermitian]) | Compute the (Moore-Penrose) pseudo-inverse of a matrix. |
| **linalg.tensorinv**(a[, ind]) | Compute the 'inverse' of an N-dimensional array. |

## Norms and other numbers

| | |
|---|---|
| **linalg.norm**(x[, ord, axis, keepdims]) | Matrix or vector norm. |
| **linalg.cond**(x[, p]) | Compute the condition number of a matrix. |
| **linalg.det**(a) | Compute the determinant of an array. |
| **linalg.matrix_rank**(A[, tol, hermitian]) | Return matrix rank of array using SVD method |
| **linalg.slogdet**(a) | Compute the sign and (natural) logarithm of the determinant of an array. |
| **trace**(a[, offset, axis1, axis2, dtype, out]) | Return the sum along diagonals of the array. |

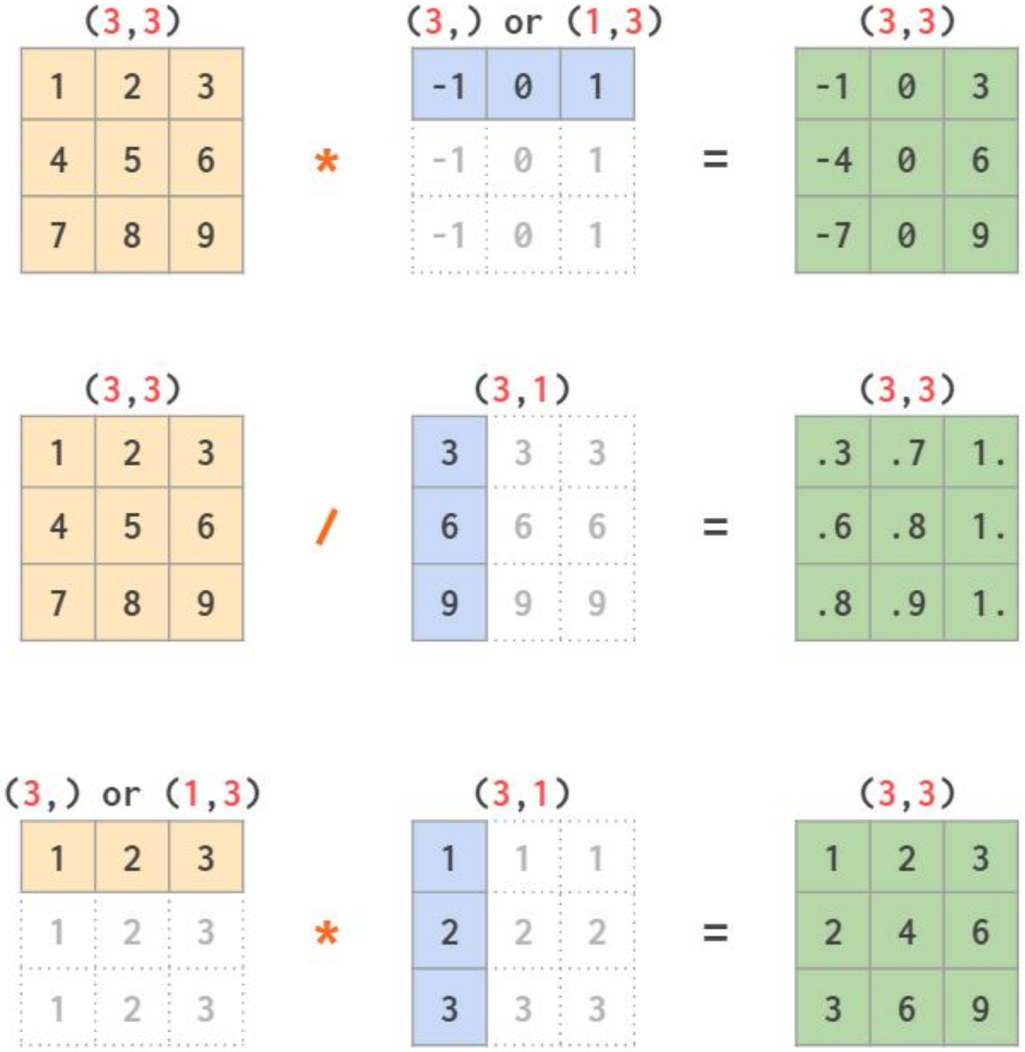
[Linear algebra (numpy.linalg) — NumPy v1.23 Manual]

# NumPy - Dot Product




Google Developer Student Clubs

# NumPy - Broadcasting




Google Developer Student Clubs