NumPy

numpy.ravel: Flatten a NumPy Array

If you want to get a 1-D array of a multi-dimensional array, try `numpy.ravel(arr)`. You can either read the elements in the same row first or read the elements in the same column first.

```
>>> import numpy as np

>>> arr = np.array([[1, 2], [3, 41]])
>>> arr
```

```
array([[ 1,  2],
       [ 3, 41]])
```

```
>>> np.ravel(arr)
```

```
array([ 1,  2,  3, 41])
```

```
>>> np.ravel(arr, order="F")
```

```
array([ 1,  3,  2, 41])
```

Use List to Change the Positions of Rows or Columns in a NumPy Array

If you want to change the positions of rows or columns in a NumPy array, simply use a list to specify the new positions as shown below.

```
>>> arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
>>> arr
```

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

```
>>> new_row_position = [1, 2, 0]
>>> new_arr = arr[new_row_position, :]
>>> new_arr
```

```
array([[4, 5, 6],
       [7, 8, 9],
       [1, 2, 3]])
```

Difference Between NumPy's All and Any Methods

If you want to get the row whose ALL values satisfy a certain condition, use NumPy's `all` method.

```
>>> a = np.array([[1, 2, 1], [2, 2, 5]])

# get the rows whose all values are fewer than 3
>>> mask_all = (a < 3).all(axis=1)
>>> a[mask_all]
```

```
array([[1, 2, 1]])
```

To get the row whose AT LEAST one value satisfies a certain condition, use NumPy's `any` method.

```
>>> mask_any = (a < 3).any(axis=1)
>>> a[mask_any]
```

```
array([[1, 2, 1],
       [2, 2, 5]])
```

Double numpy.argsort: Get Rank of Values in an Array

If you want to get the index of the sorted list for the original list, apply `numpy.argsort()` twice.

```
>>> a = np.array([2, 1, 4, 7, 3])

# Get rank of values in an array
>>> a.argsort().argsort()
```

```
array([1, 0, 3, 4, 2])
```

In the example above, 1 is the smallest value so it is indexed 0. 2 is the second-largest value to it is indexed 1.

Get the Index of the Max Value in a NumPy Array

To get the index of the max value in a NumPy array, use `np.argmax`. This can be helpful to get the highest probability in an array of probabilities.

```
>>> a = np.array([0.2, 0.4, 0.7, 0.3])
>>> np.argmax(a)
```

```
2
```

np.where: Replace Elements of a NumPy Array Based on a Condition

If you want to replace elements of a NumPy array based on a condition, use `numpy.where`.

```
>>> arr = np.array([[1, 4, 10, 15], [2, 3, 8, 9]])

# Multiply values that are less than 5 by 2
>>> np.where(arr < 5, arr * 2, arr)
```

```
array([[ 2,  8, 10, 15],
       [ 4,  6,  8,  9]])
```

array-to-latex: Turn a NumPy Array into Latex

```
!pip install array-to-latex
```

Sometimes you might want to use latex to write math. You can turn a NumPy array into latex using array-to-latex.

```
>>> import array_to_latex as a2l

>>> a = np.array([[1, 2, 3], [4, 5, 6]])
>>> latex = a2l.to_ltx(a)
>>> latex
```

```
\begin{bmatrix}
  1.00 &  2.00 &  3.00\\
  4.00 &  5.00 &  6.00
\end{bmatrix}
```

I copied and pasted the output of array-to-latex to the Markdown cell of Jupyter Notebook, and below is the output.

$$\begin{bmatrix} 1.00 & 2.00 & 3.00 \\ 4.00 & 5.00 & 6.00 \end{bmatrix}$$

Link to array-to-latex.

NumPy Comparison Operators

If you want to get elements of a NumPy array that are greater, smaller, or equal to a value or an array, simply use comparison operators such as <, <=, >, >=, ==.

```
>>> a = np.array([1, 2, 3])
>>> b = np.array([4, 1, 2])

>>> a < 2
```

```
array([ True, False, False])
```

```
>>> a < b
```

```
array([ True, False, False])
```

```
>>> a[a < b]
```

```
array([1])
```

NumPy.linspace: Get Evenly Spaced Numbers Over a Specific Interval

If you want to get evenly spaced numbers over a specific interval, use `numpy.linspace(start, stop, num)`. The code below shows a use case of the `numpy.linspace` method.

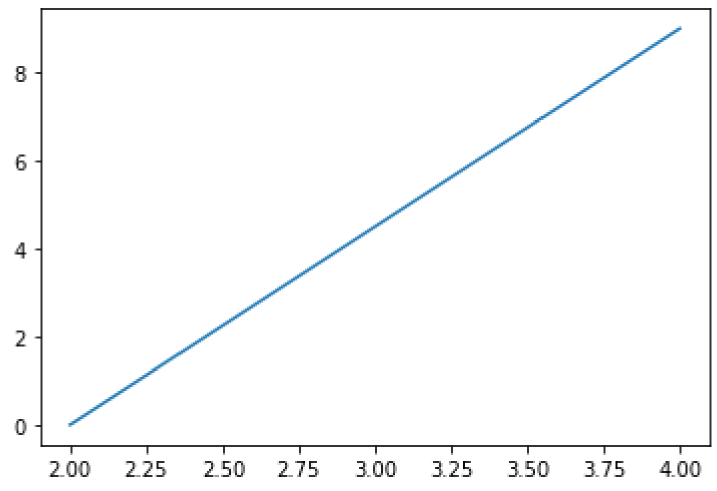
```
>>> import matplotlib.pyplot as plt

>>> x = np.linspace(2, 4, num=10)
>>> x
```

```
array([2.        , 2.22222222, 2.44444444, 2.66666667,
2.88888889,
       3.11111111, 3.33333333, 3.55555556, 3.77777778, 4.
   ])
```

```
>>> y = np.arange(10)

>>> plt.plot(x, y)
>>> plt.show()
```



NumPy.testing.assert_almost_equal: Check if Two Arrays Are Equal up to a Certain Precision

Sometimes, you might only want to check if two arrays are equal up to a certain precision. If so, use `numpy.testing.assert_almost_equal`.

```
>>> from numpy.testing import assert_almost_equal,
assert_array_equal

>>> a = np.array([[1.222, 2.222], [3.222, 4.222]])
>>> test = np.array([[1.221, 2.221], [3.221, 4.221]])
>>> assert_almost_equal(a, test, decimal=2)

>>> assert_array_equal(a, test)
```

```
AssertionError:
Arrays are not equal

Mismatched elements: 4 / 4 (100%)
Max absolute difference: 0.001
Max relative difference: 0.000819
 x: array([[1.222, 2.222],
       [3.222, 4.222]])
 y: array([[1.221, 2.221],
       [3.221, 4.221]])
```