Numpy Where

Numpy where() function returns elements, either from x or y array_like objects, depending on condition.

The syntax of where() function is:

```
numpy.where(conditio
n[, x, y])
```

```
numpy.where(condition[, x, y])
```

If the condition is true x is chosen. If the condition is false y is chosen. If only condition is given, return condition.nonzero().

Parameters	**condition : array_like, bool**	When True, yield x, otherwise yield y.
	x, y : array_like, optional	Values from which to choose. x, y and condition need to be broadcastable to some shape.
Returns	**out : ndarray or tuple of ndarrays**	If both x and y are specified, the output array contains elements of x where condition is True, and elements from yelsewhere. If only condition is given, return the tuple condition.nonzero(), the indices where condition is True.

Numpy Where – Examples

Numpy Where with a condition and two array_like variables

In the example, we provide demonstrate the two cases: when condition is true and when the condition is false.

```
>>> a =
np.random.randint(1,10
```

```
>>> a = np.random.randint(1,10,8).reshape(2,4)
>>> b = np.random.randint(1,10,8).reshape(2,4)
>>> a
array([[6, 8, 8, 8],
       [1, 3, 9, 2]])
>>> b
array([[9, 7, 6, 8],
       [7, 8, 2, 7]])
>>> np.where(4<5, a+2, b+2)
array([[ 8, 10, 10, 10],
       [ 3, 5, 11, 4]])
>>> np.where(4>5, a+2, b+2)
array([[11, 9, 8, 10],
       [ 9, 10, 4, 9]])
>>>
```

In the first case, `np.where(4<5, a+2, b+2)`, the condition is true, hence `a+2` is yielded as output.

In the first case, `np.where(4>5, a+2, b+2)`, the condition is false, hence `b+2` is yielded as output.

Now we will look into some examples where only the condition is provided. These scenarios can be useful when we would like to find out the indices or number of places in an array where the condition is true.

Numpy Where with Two-Dimensional Array

Now let us see what numpy.where() function returns when we apply the condition on a two dimensional array.

In this example, we will create a random integer array with 8 elements and reshape it to of shape (2,4) to get a two-dimensional array. Then we shall call the where() function with the condition a%2==0, in other words where the number is even.

```
>>> a =
np.random.randint(1,10
```

```
>>> a = np.random.randint(1,10,8).reshape(2,4)
>>> a
array([[9, 8, 1, 4],
       [9, 1, 5, 4]])
>>> w = np.where(a%2==0)
>>> w
(array([0, 0, 1], dtype=int32), array([1, 3, 3], dtype=int32))
>>>
```

The result is also a two dimensional array. The first array represents the indices in first dimension and the second array represents the indices in the second dimension.

For example, a%2==0 for 8, 4, 4 and their indices are (0,1), (0,3), (1,3). Now if we separate these indices based on dimension, we get [0, 0, 1], [1, 3, 3], which is ofcourse our returned value from numpy.where().

Numpy Where with multiple conditions passed

Now let us see what numpy.where() function returns when we provide multiple conditions array as argument.

In this example, we will create two random integer arrays `a` and `b` with 8 elements each and reshape them to of shape (2,4) to get a two-dimensional array. Then we shall call the where() function with the condition `a>10` and `b<5`.

```
>>> a =
np.random.randint(1,10
```

```
>>> a = np.random.randint(1,10,8).reshape(2,4)
>>> b = np.random.randint(1,10,8).reshape(2,4)
>>> a
array([[6, 8, 8, 8],
       [1, 3, 9, 2]])
>>> b
array([[9, 7, 6, 8],
       [7, 8, 2, 7]])
>>> np.where([a>10,b<8])
(array([1, 1, 1, 1, 1], dtype=int32), array([0, 0, 1, 1, 1], dtype=int32), array([1, 2, 0, 2, 3], dtype=int32))
```

Let us analyse the output. As we have provided two conditions, and there is no result for the first condition, the returned list of arrays represent the result for second array.

`(array([1, 1, 1, 1, 1], dtype=int32)` represents that all the results are for the second condition. index `1` mean second.

`array([0, 0, 1, 1, 1], dtype=int32)` represents the first dimensional indices.

`array([1, 2, 0, 2, 3], dtype=int32)` represents the second dimensional indices.

Numpy Where function with One-dimensional Array

```
>>> import numpy as
np
```

```
>>> import numpy as np
>>> a = np.random.randint(1,10,8)
>>> a
array([6, 2, 9, 1, 8, 4, 6, 4])
>>> w = np.where(a>5)
>>> w
(array([0, 2, 4, 6], dtype=int32),)
>>>
```

The `numpy.where()` function returns an array with indices where the specified condition is true. The given condition is `a>5`. So, the result of `numpy.where()` function contains indices where this condition is satisfied. Since, `a = [6, 2, 9, 1, 8, 4, 6, 4]`, the indices where `a>5` is `0,2,4,6`.

numpy.where() kind of oriented for two dimensional arrays. So, the returned value has a non-empty array followed by nothing (after comma): `(array([0, 2, 4, 6], dtype=int32),)`. You will get more clarity on this when we go through where function for two dimensional arrays.

You can store this result in a variable and access the elements using index.

```
>>> w[0][3]
6
```

```
>>> w[0][3]
6
>>> w[0][1]
2
```

Python Library Tutorials

- Python Numpy Tutorial
- Python SciPy Tutorial
- Python Matplotlib Tutorial
- Python 3D Tutorial
- Python GUI - tkinter Tutorial
- NLTK Tutorial