Create Your Own ufunc

‹ Previous Next ›

How To Create Your Own ufunc

To create you own ufunc, you have to define a function, like you do with normal functions in Python, then you add it to your NumPy ufunc library with the `frompyfunc()` method.

The `frompyfunc()` method takes the following arguments:

1. *`function`* - the name of the function.
2. *`inputs`* - the number of input arguments (arrays).
3. *`outputs`* - the number of output arrays.

Example

Create your own ufunc for addition:

```
import numpy as np

def myadd(x, y):
  return x+y

myadd = np.frompyfunc(myadd, 2, 1)

print(myadd([1, 2, 3, 4], [5, 6, 7, 8]))
```

Try it Yourself »

Check if a Function is a ufunc

Check the *type* of a function to check if it is a ufunc or not.

A ufunc should return `<class 'numpy.ufunc'>`.

Example

Check if a function is a ufunc:

```
import numpy as np

print(type(np.add))
```

Try it Yourself »

If it is not a ufunc, it will return another type, like this built-in NumPy function for joining two or more arrays:

Example

Check the type of another function: concatenate():

```
import numpy as np

print(type(np.concatenate))
```

Try it Yourself »

If the function is not recognized at all, it will return an error:

Example

Check the type of something that does not exist. This will produce an error:

```
import numpy as np

print(type(np.blahblah))
```

Try it Yourself »

To test if the function is a ufunc in an if statement, use the `numpy.ufunc` value (or `np.ufunc` if you use np as an alias for numpy):

Example

Use an if statement to check if the function is a ufunc or not:

```
import numpy as np

if type(np.add) == np.ufunc:
  print('add is ufunc')
else:
  print('add is not ufunc')
```

Try it Yourself »

‹ Previous Next ›

XML Certificate

Get Certified »