PYTHON ASSIGNMENT 1

1. I noticed that some numpy operations take an argument called shape, such as np.zeros, whereas some others take an argument called size, such as np.random.randint. To me, those arguments have the same function and the fact that they have different names is a bit confusing. Actually, size seems a bit off since it really specifies the .shape of the output.

Is there a reason for having different names, do they convey a different meaning even though they both end up being equal to the .shape of the output?
Because you are working with a numpy array, which was seen as a C array, size refers to how big your array will be. Moreover, if you can pass np.zeros(10) or np.zeros((10)). While the difference is subtle, size passed this way will create you a 1D array. You can give size=(n1, n2, ..., nn) which will create an nD array.

However, because python users want multi-dimensional arrays, array.reshape allows you to get from 1D to an nD array. So, when you call shape, you get the N dimension shape of the array, so you can see exactly how your array looks like.

In essence, size is equal to the product of the elements of shape.

EDIT: The difference in name can be attributed to 2 parts: firstly, you can initialise your array with a size. However, you do not know the shape of it. So size is only for total number of elements. Secondly, how numpy was developed, different people worked on different parts of the code, giving different names to roughly the same element, depending on their personal vision for the code.

2. If the dimensions of two arrays are dissimilar, element-to-element operations are not possible. However, operations on arrays of non-similar shapes is still possible in NumPy, because of the broadcasting capability. The smaller array is broadcast to the size of the larger array so that they have compatible shapes.

Broadcasting is possible if the following rules are satisfied −

Array with smaller ndim than the other is prepended with '1' in its shape.

Size in each dimension of the output shape is maximum of the input sizes in that dimension.

An input can be used in calculation, if its size in a particular dimension matches the output size or its value is exactly 1.

If an input has a dimension size of 1, the first data entry in that dimension is used for all calculations along that dimension.

A set of arrays is said to be broadcastable if the above rules produce a valid result and one of the following is true −

Arrays have exactly the same shape.

Arrays have the same number of dimensions and the length of each dimension is either a common length or 1.

Array having too few dimensions can have its shape prepended with a dimension of length 1, so that the above stated property is true.

The following program shows an example of broadcasting.

Example 2
Live Demo

```
import numpy as np
a = np.array([[0.0,0.0,0.0],[10.0,10.0,10.0],[20.0,20.0,20.0],[30.0,30.0,30.0]])
b = np.array([1.0,2.0,3.0])

print 'First array:'
print a
print '\n'

print 'Second array:'
print b
print '\n'

print 'First Array + Second Array'
print a + b
```

3. As such, the programming language has numerous applications and has been widely adopted by all sorts of communities, from data science to business. These communities value Python for its precise and efficient syntax, relatively flat learning curve, and good integration with other languages (e.g. C/C++).

The language's popularity has resulted in a plethora of Python packages being produced for data visualization, machine learning, natural language processing, complex data analysis, and more.

Here is our list of the most popular Python libraries.

4. NumPy introduces a simple file format for ndarray objects. This . npy file stores data, shape, dtype and other information required to reconstruct the ndarray in a disk file such that the array is correctly retrieved even if the file is on another machine with different architecture

5.

empty() function
The empty() function is used to create a new array of given shape and type, without initializing entries.

Syntax:

```
numpy.empty(shape, dtype=float, order='C')
```

NumPy array: empty() function
Version: 1.15.0

Parameter:

Name	Description	Required / Optional
shape	Shape of the empty array, e.g., (2, 3) or 2.	Required
dtype	Desired output data-type for the array, e.g, numpy.int8. Default is numpy.float64.	optional
order	Whether to store multi-dimensional data in row-major (C-style) or column-major (Fortran-style) order in memory.	optional

Return value:

[ndarray] Array of uninitialized (arbitrary) data of the given shape, dtype, and order. Object arrays will be initialized to None.

Example: numpy.empty() function