1. What makes NumPy.shape() different from NumPy.size()?

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.
Shape relates to the size of the dimensions of an N-dimensional array.

Size regarding arrays, relates to the amount (or count) of elements that are contained in the array (or sometimes, at the top dimension of the array - when used as length).

For example, let a be a matrix

1 2 3 4
5 6 7 8
9 10 11 12
the shape of a is (3, 4), the size of a is 12 and the size of a[1] is 4.

2. In NumPy, describe the idea of broadcasting.

The term broadcasting refers to the ability of NumPy to treat arrays of different shapes during arithmetic operations. Arithmetic operations on arrays are usually done on corresponding elements. If two arrays are of exactly the same shape, then these operations are smoothly performed

Example :
Live Demo
```
import numpy as np

a = np.array([1,2,3,4])
b = np.array([10,20,30,40])
c = a * b
print c
```
Its output is as follows −

```
[10   40   90   160]
```
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.

3. What makes Python better than other libraries for numerical computation?

One of the key features of Python is its numerous libraries and packages. In this article, we will list down the popular packages and libraries in Python that are being widely used for numeric and scientific applications.

(The list is in no particular order).

1| SciPy (Scientific Numeric Library)
Officially released in 2000-01, SciPy is free and open source library used for scientific computing and technical computing. The library consists of modules for optimisation,

image processing, FFT, special functions and signal processing.

The SciPy package includes algorithms and functions which are the crux of Python scientific computing capabilities. The sub-package includes:

io: used for the standard input and output
lib: this function is used to wrap python external libraries
signal: used for processing signal tools
sparse: used for algorithms related to sparse matrix

spatial: widely used to determine paths in KD-trees, nearest neighbor and distance functions.
optimise: used to optimise algorithms which include linear programming.
linals: used for the regular linear algebra applications.
interpolate: used for the integration of tools
intergate: applied for integration of numerical tools
fftpack: this subpackage helps for the discretion Fourier to transform algorithms
cluster: the package consists of hierarchical clustering, vector quantisation, and K-means.
misc: used for the miscellaneous utility applications.
special: used to switch in special functions.
weave: a tool to convert C/C++ codes to python programming.
ndimage: used for wide range of functions in multi-dimensional image processing.
stats: used for better understanding and analysing of statistical functions.
constants: this algorithm includes physical specification and conversion components.

2| Pandas (Data Analytics Library)
Pandas is the most important data analysis library of Python. Being open source, it is used for analysing data with Python. It can take data formats of CSV or TSV files, or a SQL database and convert it into Python data frames with rows and columns which is similar to tables in statistical formats. The package makes comparisons with dictionaries with the aid of 'for' loops which are very easy to understand and operate.

Python 2.7 and above versions are required to install Pandas package. We need to import the Panda's library into the memory to work with it. The following codes can be run to implement different operations on pandas.

Import pandas as pd (importing pandas library to memory), it is highly suggested to import the library as pd because next time when we want to use the application we need not mention the package full name instead we can name as pd, this avoids confusion.
pd.read_filetype() (to open the desired file)
pd.DataFrame() (to convert a specified python object)
df.to_filetype (filename) (to save a data frame you are currently working with)
The advantage of using Pandas is that it can perform a bunch of functions on the tables we have created. The following are some functions that can be performed on selected data frames.

df.median()-to get the median of each column
df.mean()-to get the mean of all columns
df.max()-to get the highest value of a column
df.min()-to get the minimum value of a column
df.std()-to get the standard deviation of each column.
df.corr()-to specify the relationship between columns of a data frame.
df.count()-to get the number of non-null values in each column of the data frame.

3| IPython (Command Shell)

Developed by Fernando Perez in the year 2001, IPython is a command shell which is designed for interactive calculation in various programming languages. It offers self-examination, rich media, shell syntax, tab completion, and history.

IPython is a browser-based notebook interface which supports code, text, mathematical expressions, inline plots and various media for interactive data visualisation with the use of GUI (Graphic User Interface) toolkits for flexible and rectifiable interpreters to load into one's own projects.

IPython architecture contributes to parallel and distributed computing. It facilitates for the enhanced parallel applications of various styles of parallelism such as:

Customer user defined prototypes
Task Parallelism
Data Parallelism
Message cursory using M.P.I (Message Passing Interface)
Multiple programs, multiple data (MIMD) parallelism
A single program, multiple data (SPMD) parallelism

4| Numeric Python (Fundamental Numeric Package)
Better known as Numpy, numeric Python has developed a module for Python, mostly written in C. Numpy guarantees swift execution as it is accumulated with mathematical and numerical functions.

Robust Python with its dynamic data structures, efficient implementation of multi-dimensional arrays and matrices, Numpy assures accurate calculations with matrices and arrays.

We need to import Numpy into memory to perform numerical operations.

Import numpy as np (to import Numpy into memory)
A_values=[20,30,40,50] (defining a list)
A=np.array(A_values) (to convert list into one dimensional numpy array)
print(A) (to get one dimensional array displayed)
print(A*9/5 +32) (to turn values in the list into degrees fahrenheit)

5| Natural Language Toolkit (Library For Mathematical And Text Analysis)
Simply known as NLP, Natural Language Processing library is used to build applications and services that can understand and analyse human languages and data. One of the sub-libraries which are widely used in NLP is NLTK (Natural Language Toolkit). It has an active discussion forum through which they give hands-on guidance on programming basic topics such as computational linguistics, comprehensive API documentation, linguistics to engineers, students, industries and researchers. NLTK is an open source free community-driven project which is accessible for operating systems such as Windows, MAC OS X, and Linux. The implementations of NLP are:

Search engines (eg: Yahoo, Google, firefox etc) they use NLP to optimise the search results for users.
Social websites like Facebook, Twitter use NLP for the news feed. The NLP algorithms understand the interests of the users and show related posts.
Spam filters: unlike the traditional spam filters, the NLP has driven spam filters to understand what the mail is about and decides whether it is a spam or not.
NLP includes well known and advanced sub-libraries which are very effective in mathematical calculations.

NLTK, which handles text analysis and related problems. Having over 50 corpora and lexicons, 9 stemmers and handful of algorithms NLTK is very popular for education and research. The application involves a deep learning and analysing process which makes it one of the tough libraries in NLP
TextBlob, which is a simple library for text analysis
Stanford core NLP, a library that includes entity recognition, pattern understanding, parsing, tagging etc.
SpaCy, which presents the best algorithm for the purpose
Gensim, which is used for topic prototypes and document similarity analysis

4. How does NumPy deal with files?

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.

Numpy is an acronym for ‘Numerical Python’. It is a library in python for supporting n-dimensional arrays. But have you ever wondered about loading data into NumPy from text files. Don’t worry we will discuss the same in this article. To import Text files into Numpy Arrays, we have two functions in Numpy:

numpy.loadtxt( ) – Used to load text file data
numpy.genfromtxt( ) – Used to load data from a text file, with missing values handled as defined.
Note: numpy.loadtxt( ) is equivalent function to numpy.genfromtxt( ) when no data is missing.

Method 1 : numpy.loadtxt()

Syntax :

numpy.loadtxt(fname, dtype = float, comments=’#’, delimiter=None, converters=None, skiprows=0, usecols=None, unpack=False, ndmin=0, encoding=’bytes’, max_rows=None, *, like= None)

The default data type(dtype) parameter for numpy.loadtxt( ) is float.

Example 1: Importing Text file into Numpy arrays

The following ‘example1.txt’ text file is considered in this example.

Python3

```
import numpy as np


# Text file data converted to integer data type
File_data = np.loadtxt("example1.txt", dtype=int)
print(File_data)
```

Output :

```
[[ 1  2]
[ 3  4]
[ 5  6]
[ 7  8]
[ 9 10]]
```

Example 2: Importing text file into NumPy array by skipping first row

Python3

```
import numpy as np


# skipping first row
# converting file data to string
data = np.loadtxt("example2.txt", skiprows=1, dtype='str')
print(data)
```

Output :

```
[['2' 'Bunty']
['3' 'Tinku']
['4' 'Rina']]
```

Example 3: Importing only the first column(Names) of text file into numpy arrays

The indexing in NumPy arrays starts from 0. Hence, the Roll column in the text file is the 0th column, Names column is the 1st column and the Marks are the 2nd column in the text file ‘example3.txt’.

Python3
```
import numpy as np


# only column1 data is imported into numpy
# array from text file
data = np.loadtxt("example3.txt", usecols=1, skiprows=1, dtype='str')

for each in data:
    print(each)
```
Output :

Ankit
Bunty
Tinku
Rina
Rajesh
Method 2 : numpy.genfromtxt()

Syntax :

numpy.genfromtxt(fname, dtype=float, comments='#', delimiter=None, skip_header=0, skip_footer=0, converters=None, missing_values=None, filling_values=None, usecols=None, names=None,excludelist=None, deletechars=" !#$%&'()*+, -./:;<=>?@[\\]^{|}~", replace_space='_', autostrip=False, case_sensitive=True, defaultfmt='f%i', unpack=None, usemask=False, loose=True, invalid_raise=True, max_rows=None, encoding='bytes', *, like=None)

Except the fname(filename) in numpy.genfromtxt( ), all the other parameters are optional.

Example 1:

Python3
```
import numpy as np

Data = np.genfromtxt("example4.txt", dtype=str,
                     encoding=None, delimiter=",")
print(Data)
```
Output :

[['a' 'b' 'c' 'd']
['e' 'f' 'g' 'h']]
Example 2: Importing text file into numpy arrays by skipping last row

Python3
```
import numpy as np


# skipping last line in the file
Data = np.genfromtxt("example5.txt", dtype=str,
                     encoding=None, skip_footer=1)
print(Data)
```
Output :

[['This' 'is' 'GeeksForGeeks' 'Website']
['How' 'are' 'You' 'Geeks?']
['Geeks' 'for' 'Geeks' 'GFG']]

5. Mention the importance of NumPy.empty().

The numpy module of Python provides a function called numpy. empty(). This function is used to create an array without initializing the entries of given shape and type.

Just like numpy.zeros(), the numpy.empty() function doesn't set the array values to zero, and it is quite faster than the numpy.zeros(). This function requires the user to set all the values in the array manually and should be used with caution.

Syntax

numpy.empty(shape, dtype=float, order='C')
Parameters:
shape: int or tuple of ints

This parameter defines the shape of the empty array, such as (3, 2) or (3, 3).

dtype: data-type(optional)

This parameter defines the data type, which is desired for the output array.

order: {'C', 'F'}(optional)

This parameter defines the order in which the multi-dimensional array is going to be stored either in row-major or column-major. By default, the order parameter is set to 'C'.

Returns:
This function returns the array of uninitialized data that have the shape, dtype, and order defined in the function.

Example 1:

```
import numpy as np
x = np.empty([3, 2])
x
```

Output:

```
array([[7.56544226e-316, 2.07617768e-316],
       [2.02322570e-316, 1.93432036e-316],
       [1.93431918e-316, 1.93431799e-316]])
```

In the above code

We have imported numpy with alias name np.
We have declared the variable 'x' and assigned the returned value of the np.empty() function.
We have passed the shape in the function.
Lastly, we tried to print the value of 'x' and the difference between elements.
Example 2:

```
import numpy as np
x = np.empty([3, 3], dtype=float)
x
```

Output:

```
array([[ 2.94197848e+120, -2.70534020e+252, -4.25371363e+003],
       [ 1.44429964e-088,  3.12897830e-053,  1.11313317e+253],
       [-2.28920735e+294, -5.11507284e+039,  0.00000000e+000]])
```

Example 3:

```
import numpy as np
x = np.empty([3, 3], dtype=float, order='C')
x
```

Output:

```
array([[ 2.94197848e+120, -2.70534020e+252, -4.25371363e+003],
       [ 1.44429964e-088,  3.12897830e-053,  1.11313317e+253],
       [-2.28920735e+294, -5.11507284e+039,  0.00000000e+000]])
```

In the above code

We have imported numpy with alias name np.
We have declared the variable 'x' and assigned the returned value of the np.empty() function.
We have passed the shape, data-type, and order in the function.
Lastly, we tried to print the value of 'x' and the difference between elements.
In the output, it shows an array of uninitialized values of defined shape, data type, and order.

Example 4:

```
import numpy as np
x = np.empty([3, 3], dtype=float, order='F')
x
```

Output:

```
array([[ 2.94197848e+120,  1.44429964e-088, -2.28920735e+294],
       [-2.70534020e+252,  3.12897830e-053, -5.11507284e+039],
       [-4.25371363e+003,  1.11313317e+253,  0.00000000e+000]])
```