Importing Text Files with Numpy

Xavier Morera

Helping developers create amazing applications

@xmorera / www.xaviermorera.com / www.bigdatainc.org

NumPy

Numerical Python

Numpy

$$\begin{bmatrix} 4 & 7 \\ 1 & 5 \end{bmatrix}$$

Library for scientific computing

- Released in 2005, as open-source
- Built in C and Python

Wide range of mathematical functions

- Linear algebra, Fourier transforms...

Used as a base for other libraries

- Pandas, scikit-learn, Tensorflow...

Adopted by many industries

- Finance, healthcare, engineering...

Tasks Especially Suited for Using Numpy

Arrays

Matrices

Numerical computations

NumPy

The fundamental package for scientific computing with Python

Meet the new NumPy docs team leads

POWERFUL N-DIMENSIONAL ARRAYS

Fast and versatile, the NumPy vectorization, indexing, and broadcasting concepts are the de-facto standards of array computing today.

NUMERICAL COMPUTING TOOLS

NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.

OPEN SOURCE

Distributed under a liberal BSD license, NumPy is developed and maintained publicly on GitHub by a vibrant, responsive, and diverse community.

INTEROPERABLE

NumPy supports a wide range of hardware and computing platforms, and plays well with distributed, GPU, and sparse array libraries.

PERFORMANT

The core of NumPy is well-optimized C code. Enjoy the flexibility of Python with the speed of compiled code.

EASY TO USE

NumPy's high level syntax makes it accessible and productive for programmers from any background or experience level.

The Numpy ndarray

ndarray

n-dimensional array

Numpy ndarray

$$\begin{bmatrix} 4 & 7 \\ 1 & 5 \end{bmatrix}$$

Data structure in Numpy

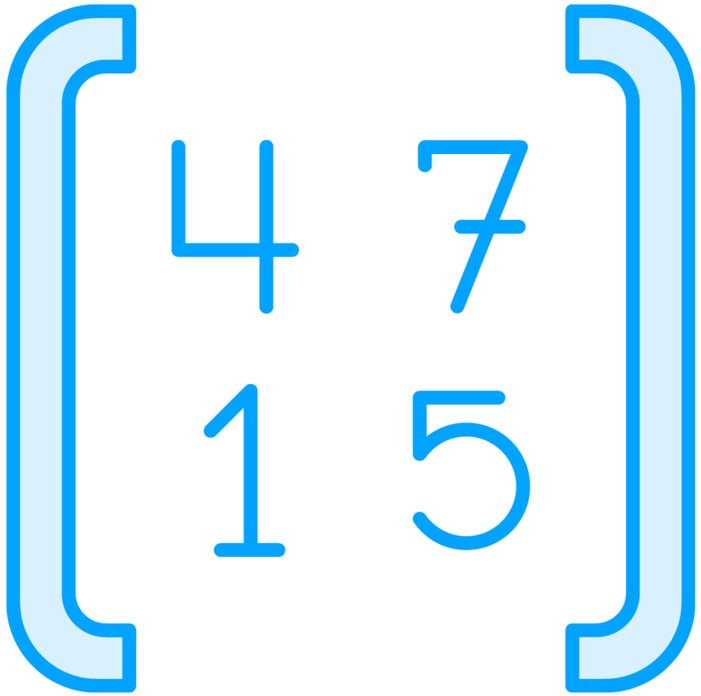
Table of elements

- Usually numbers, of the same type
- Number of axes, called dimensions

Homogeneous multidimensional array

- Allows for efficient storage and manipulation of data in multiple dimensions

Initializing a ndarray



Can be initialized in several ways

List or tuple

Nested lists

Loaded from files

- Possible to save to a file

Working with a ndarray

$$\begin{bmatrix} 4 & 7 \\ 1 & 5 \end{bmatrix}$$

Accessed using square brackets

- Indexing and slicing

Indexed by a tuple of positive integers

Number of dimensions of the array is the rank

A tuple of integers that give the size of the array along each dimension is called the shape

Working with a ndarray

$$\begin{bmatrix} 4 & 7 \\ 1 & 5 \end{bmatrix}$$

Provides multiple built-in operations

- Arithmetic, matrix, aggregation, broadcasting, reshaping, concatenation, splitting, transposing, and masking

Allows performing multiple mathematical and statistical tasks

Creating a ndarray

array()

zeros()

ones()

empty()

arange()

```
import numpy as np

sample_array = np.array([0,0,7])
```

Creating a ndarray using array()

Start by importing NumPy
Create a ndarray using array() from a list, tuple, or nested list
Type of the elements is known upfront
    Minimizes the need of growing arrays, which is an expensive operation

```
ones = np.ones((3, 2)
zeroes = np.zeros((2, 3)
empty_array = np.empty((2, 2)
a_range = np.arange(1, 11)
flat_array = np.flatten()
```

Other Ways of Creating a ndarray

Values may not be known upfront

Or a specific scenario is needed, i.e. 1s or 0s

A range may be needed, intervals, or a flattened array

Maybe uninitialized values are ok

How does a Numpy generated file look like?

A NumPy Created File

We may run into some files that look like this

```
1.0000000000000000e+00 1.0000000000000000e+00 2.0000000000000000e+00
2.0000000000000000e+00 3.0000000000000000e+00 4.0000000000000000e+00
4.0000000000000000e+00 5.0000000000000000e+00 5.0000000000000000e+00
8.0000000000000000e+00
```

A NumPy Created File

Or that looks like this

```
[[ 0., 0., 7.],
 [ 3., 1., 3.],
 [ 3., 0., 5.]]
```

How can I import and load data using Numpy?

Importing a ndarray

loadtxt()

genfromtxt()

recfromcsv()

recfromtxt()

numpy.loadtxt

numpy.loadtxt(*fname, dtype=<class 'float'>, comments='#', delimiter=None, converters=None, skiprows=0, usecols=None, unpack=False, ndmin=0, encoding='bytes', max_rows=None, *, quotechar=None, like=None*) [source]

Load data from a text file.

Parameters: **fname** : *file, str, pathlib.Path, list of str, generator*

File, filename, list, or generator to read. If the filename extension is `.gz` or `.bz2`, the file is first decompressed. Note that generators must return bytes or strings. The strings in a list or produced by a generator are treated as lines.

dtype : *data-type, optional*

Data-type of the resulting array; default: float. If this is a structured data-type, the resulting array will be 1-dimensional, and each row will be interpreted as an element of the array. In this case, the number of columns used must match the number of fields in the data-type.

comments : *str or sequence of str or None, optional*

The characters or list of characters used to indicate the start of a comment. None implies no comments. For backwards compatibility, byte strings will be decoded as 'latin1'. The default is '#'.

delimiter : *str, optional*

The character used to separate the values. For backwards compatibility, byte strings will be decoded as 'latin1'. The default is whitespace.

> *Changed in version 1.23.0:* Only single character delimiters are supported. Newline characters cannot be used as the delimiter.

converters : *dict or callable, optional*

Converter functions to customize value parsing. If *converters* is callable, the function is applied to all columns, else it must be a dict that maps column number to a parser function. See examples for further details. Default: None.

On this page

loadtxt

```
np.loadtxt('badges-five-numpy.txt')
```

loadtxt()

Load a file that was created with NumPy using loadtxt
Into a ndarray
Items of same type and same number of values

```
np.loadtxt('badges-five.txt', delimiter=‘,’, usecols=0)
```

loadtxt()

Can load files that were not generated with NumPy

- With delimiter, useful for files created by other means
- Options like usecols, skiprows, dtype, converter, and comments

Certain limitations, i.e. missing values

numpy.genfromtxt

```
numpy.genfromtxt(fname, dtype=<class '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', *, ndmin=0, like=None)
```
[source]

Load data from a text file, with missing values handled as specified.

Each line past the first *skip_header* lines is split at the *delimiter* character, and characters following the *comments* character are discarded.

Parameters:

fname : *file, str, pathlib.Path, list of str, generator*

File, filename, list, or generator to read. If the filename extension is `.gz` or `.bz2`, the file is first decompressed. Note that generators must return bytes or strings. The strings in a list or produced by a generator are treated as lines.

dtype : *dtype, optional*

Data type of the resulting array. If None, the dtypes will be determined by the contents of each column, individually.

comments : *str, optional*

The character used to indicate the start of a comment. All the characters occurring on a line after a comment are discarded.

delimiter : *str, int, or sequence, optional*

The string used to separate values. By default, any consecutive whitespaces act as delimiter. An integer or sequence of integers can also be provided as width(s) of each field.

skiprows : *int, optional*

skiprows was removed in numpy 1.10. Please use *skip_header* instead.

skip_header : *int, optional*

The number of lines to skip at the beginning of the file.

On this page

genfromtxt

```
np.genfromtxt('badges-five-missing-value.txt', delimiter=',',
skip_header=1)
```

genfromtext()

Load data from a file

Can handle missing values

Use missing_values to specify what is considered a missing value

And filling_values to specify how to handle

```
...                 filling_values={0:0, 'b':0, 2:-999})
>>> np.genfromtxt(StringIO(data), **kwargs)
array([(0, 2, 3), (4, 0, -999)],
      dtype=[('a', '<i8'), ('b', '<i8'), ('c', '<i8')])
```

usemask

We may also want to keep track of the occurrence of missing data by constructing a boolean mask, with `True` entries where data was missing and `False` otherwise. To do that, we just have to set the optional argument `usemask` to `True` (the default is `False`). The output array will then be a `MaskedArray`.

Shortcut functions

In addition to `genfromtxt`, the `numpy.lib.npyio` module provides several convenience functions derived from `genfromtxt`. These functions work the same way as the original, but they have different default values.

`numpy.lib.npyio.recfromtxt`

Returns a standard `numpy.recarray` (if `usemask=False`) or a `numpy.ma.mrecords.MaskedRecords` array (if `usemaske=True`). The default dtype is `dtype=None`, meaning that the types of each column will be automatically determined.

`numpy.lib.npyio.recfromcsv`

Like `numpy.lib.npyio.recfromtxt`, but with a default `delimiter=","`.

Previous
I/O with NumPy

Next
Data types

```
np.recfromtxt('files/people.txt')
np.recfromcsv('files/people.csv', delimiter=',')
```

recfromtxt and recfromcsv

Shortcut functions from numpy.lib.npyio, derived from genfromtxt, but with different defaults

recfromtxt used to read data of different types from a text file

- Creates a recarray, which allows field access using attributes

recfromcsv reads data from a CSV file

Importing Options with Numpy

Importing options with Numpy

Takeaway

Numpy is a widely used library for scientific computing

- Used as a base for other libraries

ndarray is Numpy's base object

- Homogeneous multidimensional array

Contains multiple built-in functionalities

- Arithmetic, matrix, aggregation, broadcasting, reshaping, concatenation, splitting, transposing, and masking

Takeaway

Can be initialized in several ways

In memory

- List, tuple, nested list

From a file

- loadtxt and genfromtxt
- recfromcsv and recfromtxt

Up Next:

Importing Text Files Using Pandas