numpy.fromfile

`numpy.` **`fromfile`** (*file*, *dtype=float*, *count=-1*, *sep=''*)

Construct an array from data in a text or binary file.

A highly efficient way of reading binary data with a known data-type, as well as parsing simply formatted text files. Data written using the *tofile* method can be read using this function.

Parameters:

file *: file or str*

Open file object or filename.

dtype *: data-type*

Data type of the returned array. For binary files, it is used to determine the size and byte-order of the items in the file.

count *: int*

Number of items to read. `-1` means all items (i.e., the complete file).

sep *: str*

Separator between items if file is a text file. Empty (“”) separator means the file should be treated as binary. Spaces (” ”) in the separator match zero or more whitespace characters. A separator consisting only of spaces must match at least one whitespace.

See also:

`load` (numpy.load.html#numpy.load), `save` (numpy.save.html#numpy.save), `ndarray.tofile` (numpy.ndarray.tofile.html#numpy.ndarray.tofile)

`loadtxt` **(numpy.loadtxt.html#numpy.loadtxt)** More flexible way of loading data from a text file.

Previous topic

numpy.frombuffer (numpy.frombuffer.html)

Next topic

numpy.fromfunction (numpy.fromfunction.html)

Notes

Do not rely on the combination of *tofile* and `fromfile` for data storage, as the binary files generated are are not platform independent. In particular, no byte-order or data-type information is saved. Data can be stored in the platform independent `.npy` format using `save` (numpy.save.html#numpy.save) and `load` (numpy.load.html#numpy.load) instead.

Examples

Construct an ndarray:

```
>>> dt = np.dtype([('time', [('min', int), ('sec', int)]),
...                ('temp', float)])
>>> x = np.zeros((1,), dtype=dt)
>>> x['time']['min'] = 10; x['temp'] = 98.25
>>> x
array([((10, 0), 98.25)],
      dtype=[('time', [('min', '<i4'), ('sec', '<i4')]), ('temp', '<f8')])
```

Save the raw data to disk:

```
>>> import os
>>> fname = os.tmpnam()
>>> x.tofile(fname)
```

Read the raw data from disk:

```
>>> np.fromfile(fname, dtype=dt)
array([((10, 0), 98.25)],
      dtype=[('time', [('min', '<i4'), ('sec', '<i4')]), ('temp', '<f8')])
```

The recommended way to store and load data:

```
>>> np.save(fname, x)
>>> np.load(fname + '.npy')
array([((10, 0), 98.25)],
      dtype=[('time', [('min', '<i4'), ('sec', '<i4')]), ('temp', '<f8')])
```