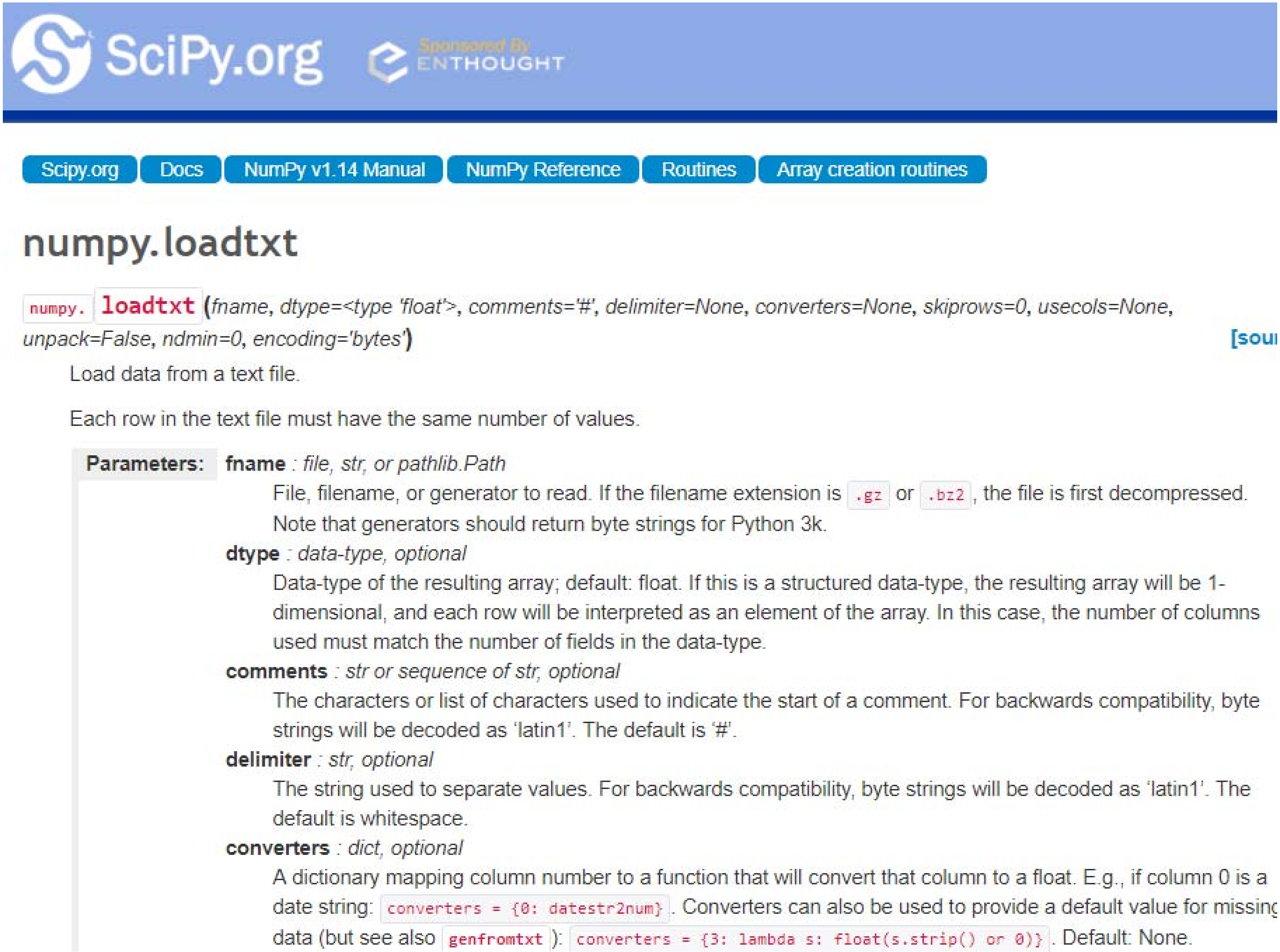


numpy.loadtxt

`numpy.loadtxt`(*fname*, *dtype=<type 'float'>*, *comments='#'*, *delimiter=None*, *converters=None*, *skiprows=0*, *usecols=None*, *unpack=False*, *ndmin=0*, *encoding='bytes'*) **[sou**

Load data from a text file.

Each row in the text file must have the same number of values.

Parameters: **fname** : *file, str, or pathlib.Path*

File, filename, or generator to read. If the filename extension is `.gz` or `.bz2`, the file is first decompressed. Note that generators should return byte strings for Python 3k.

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, optional*

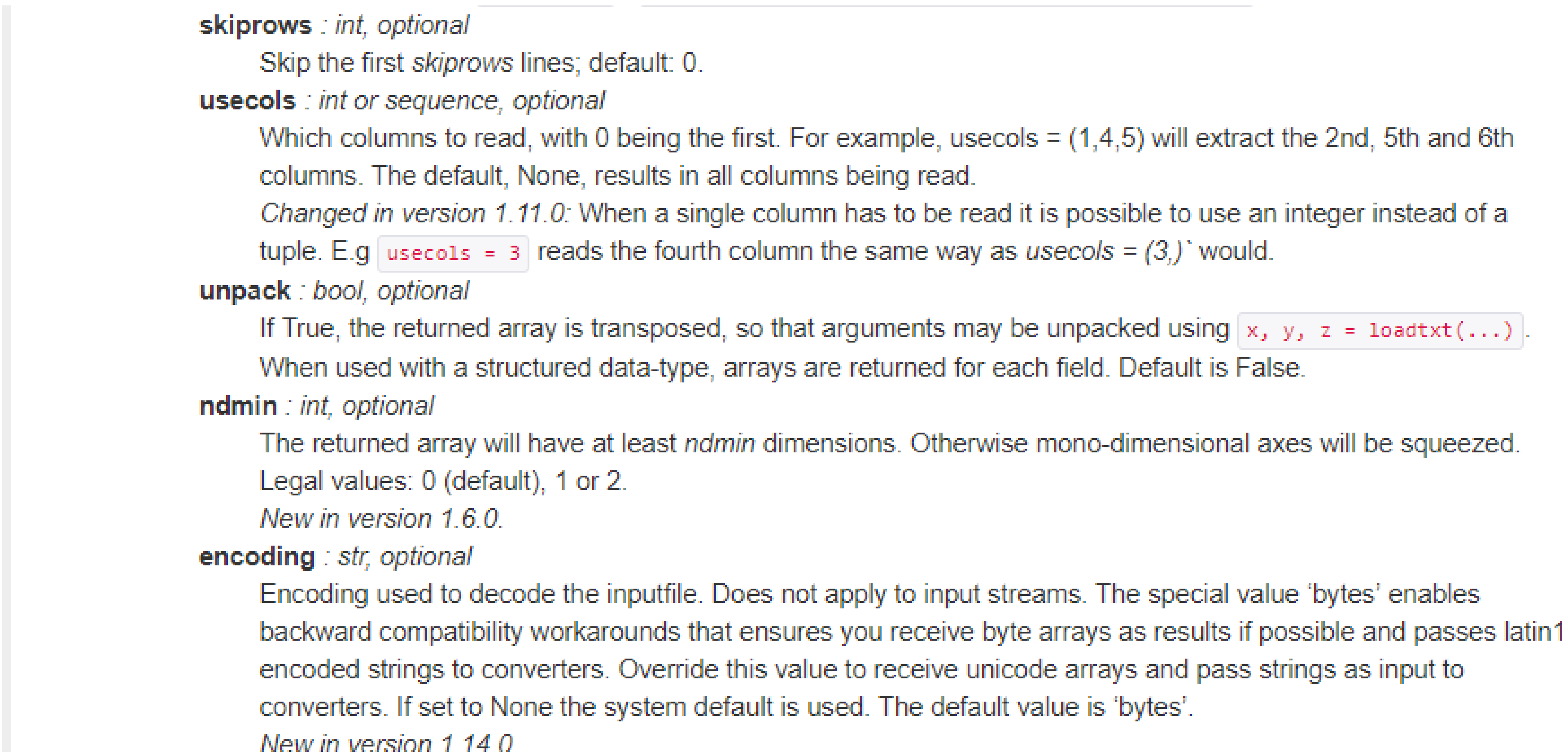
The characters or list of characters used to indicate the start of a comment. For backwards compatibility, byte strings will be decoded as 'latin1'. The default is '#'.

delimiter : *str, optional*

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

converters : *dict, optional*

A dictionary mapping column number to a function that will convert that column to a float. E.g., if column 0 is a date string: `converters = {0: datestr2num}`. Converters can also be used to provide a default value for missing data (but see also `genfromtxt`): `converters = {3: lambda s: float(s.strip() or 0)}`. Default: None.



skiprows : *int, optional*

Skip the first *skiprows* lines; default: 0.

usecols : *int or sequence, optional*

Which columns to read, with 0 being the first. For example, usecols = (1,4,5) will extract the 2nd, 5th and 6th columns. The default, None, results in all columns being read.

Changed in version 1.11.0: When a single column has to be read it is possible to use an integer instead of a tuple. E.g `usecols = 3` reads the fourth column the same way as *usecols = (3,)*` would.

unpack : *bool, optional*

If True, the returned array is transposed, so that arguments may be unpacked using `x, y, z = loadtxt(...)`. When used with a structured data-type, arrays are returned for each field. Default is False.

ndmin : *int, optional*

The returned array will have at least *ndmin* dimensions. Otherwise mono-dimensional axes will be squeezed. Legal values: 0 (default), 1 or 2.

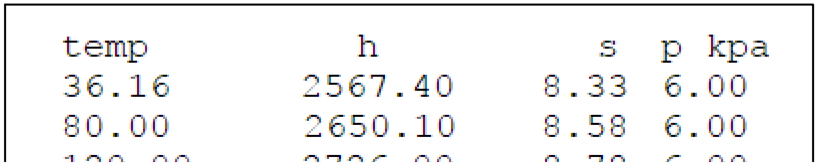
New in version 1.6.0.

encoding : *str, optional*

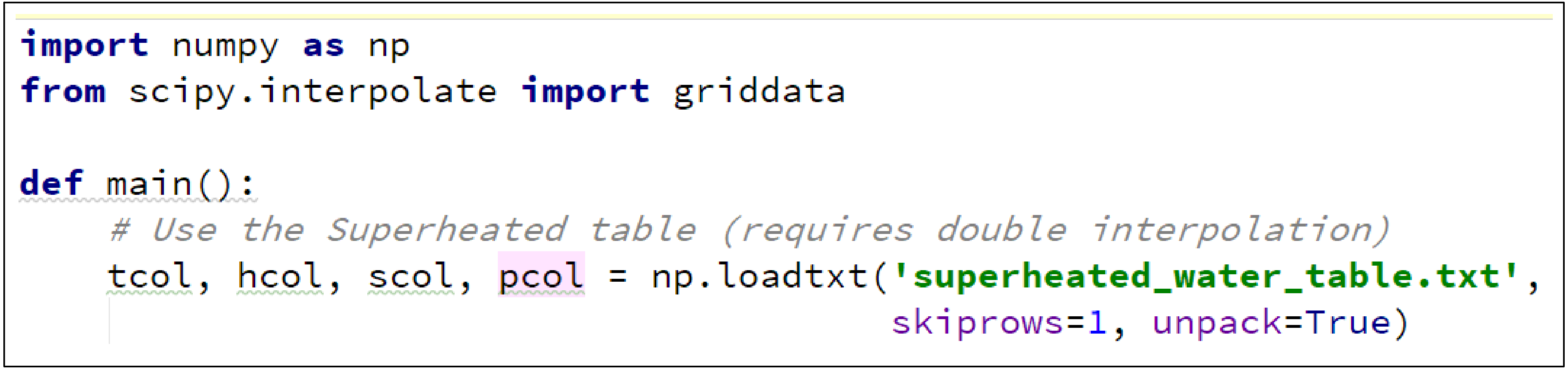
Encoding used to decode the inputfile. Does not apply to input streams. The special value 'bytes' enables backward compatibility workarounds that ensures you receive byte arrays as results if possible and passes latin1 encoded strings to converters. Override this value to receive unicode arrays and pass strings as input to converters. If set to None the system default is used. The default value is 'bytes'.

New in version 1.14.0

Superheated_water_table.txt

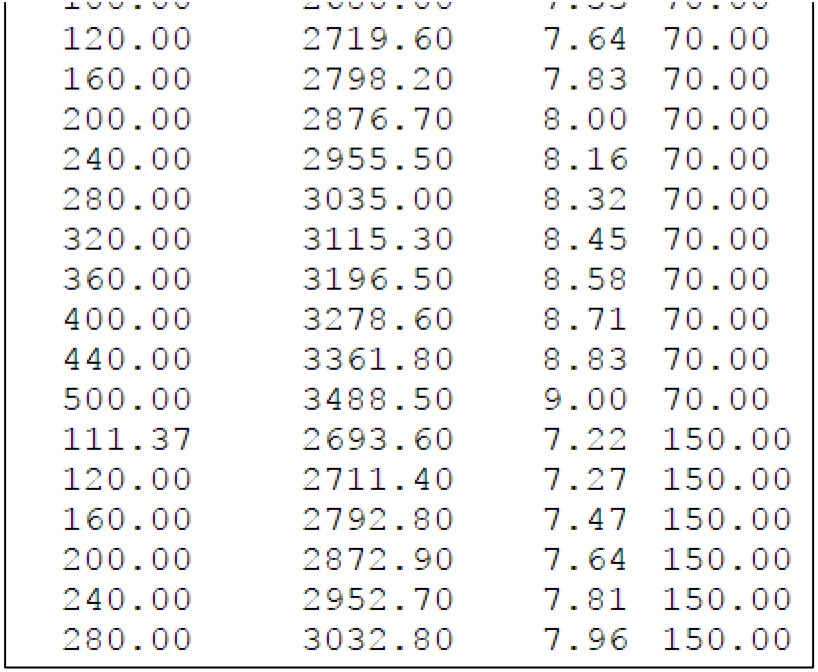


```
temp            h            s  p kpa
36.16        2567.40      8.33  6.00
80.00        2650.10      8.58  6.00
```



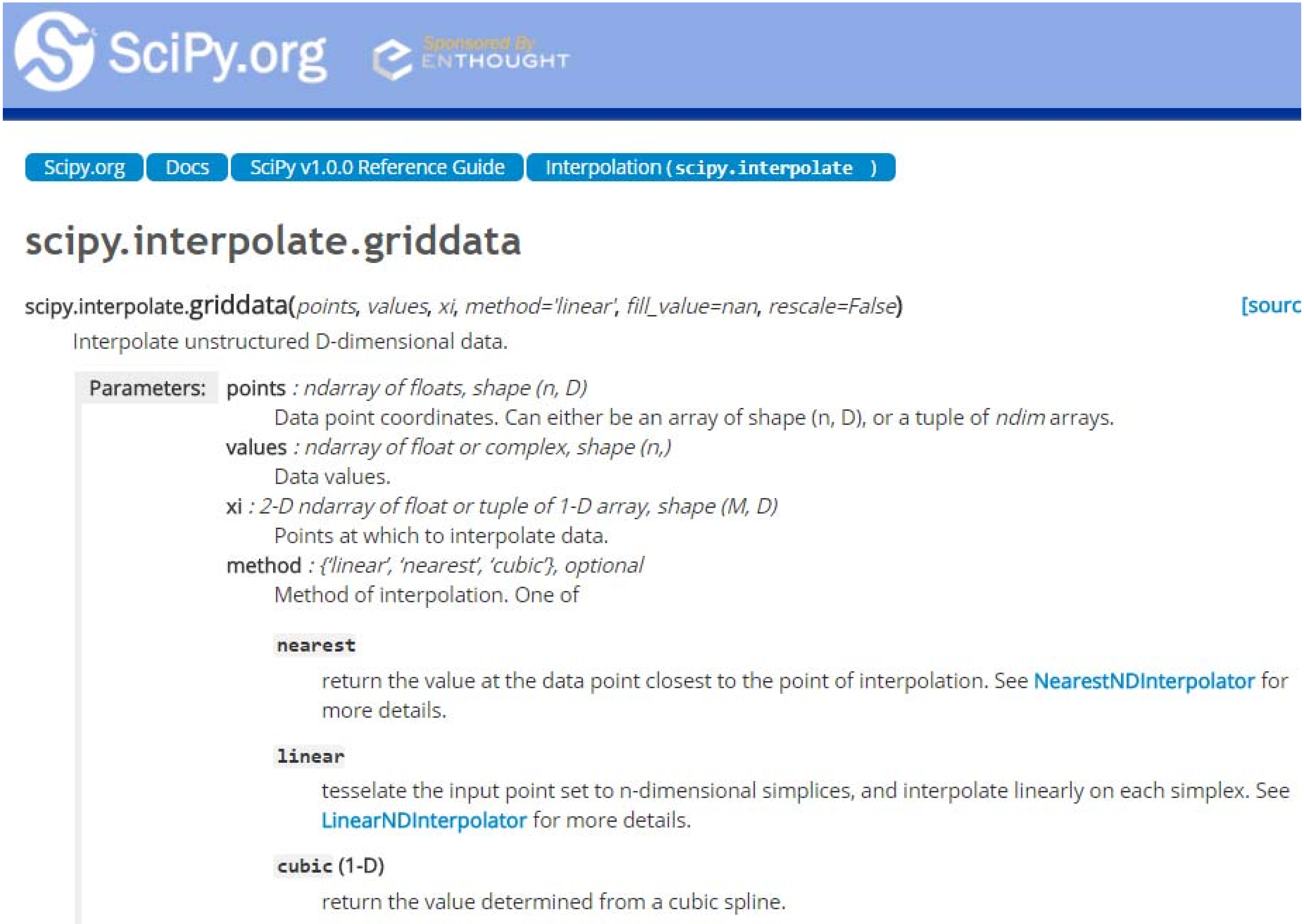
```python
import numpy as np
from scipy.interpolate import griddata

def main():
    # Use the Superheated table (requires double interpolation)
    tcol, hcol, scol, pcol = np.loadtxt('superheated_water_table.txt',
                                         skiprows=1, unpack=True)
```



```
120.00       2719.60      7.64  70.00
160.00       2798.20      7.83  70.00
200.00       2876.70      8.00  70.00
240.00       2955.50      8.16  70.00
280.00       3035.00      8.32  70.00
320.00       3115.30      8.45  70.00
360.00       3196.50      8.58  70.00
400.00       3278.60      8.71  70.00
440.00       3361.80      8.83  70.00
500.00       3488.50      9.00  70.00
111.37       2693.60      7.22  150.00
120.00       2711.40      7.27  150.00
160.00       2792.80      7.47  150.00
200.00       2872.90      7.64  150.00
240.00       2952.70      7.81  150.00
280.00       3032.80      7.96  150.00
```

Note - unpack = True


SciPy.org
Sponsored By
ENTHOUGHT
Scipy.org
Docs
SciPy v1.0.0 Reference Guide
Interpolation (`scipy.interpolate`)

scipy.interpolate.griddata

scipy.interpolate.**griddata**(*points*, *values*, *xi*, *method='linear'*, *fill_value=nan*, *rescale=False*) [sourc

Interpolate unstructured D-dimensional data.

Parameters: **points** : *ndarray of floats, shape (n, D)*

Data point coordinates. Can either be an array of shape (n, D), or a tuple of *ndim* arrays.

values : *ndarray of float or complex, shape (n,)*

Data values.

xi : *2-D ndarray of float or tuple of 1-D array, shape (M, D)*

Points at which to interpolate data.

method : *{'linear', 'nearest', 'cubic'}, optional*

Method of interpolation. One of

`nearest`

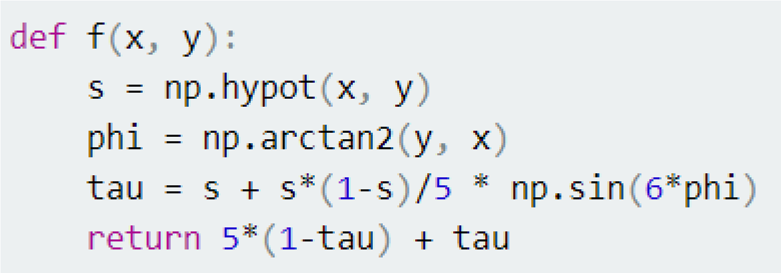
return the value at the data point closest to the point of interpolation. See NearestNDInterpolator for more details.

`linear`

tesselate the input point set to n-dimensional simplices, and interpolate linearly on each simplex. See LinearNDInterpolator for more details.

`cubic` (1-D)

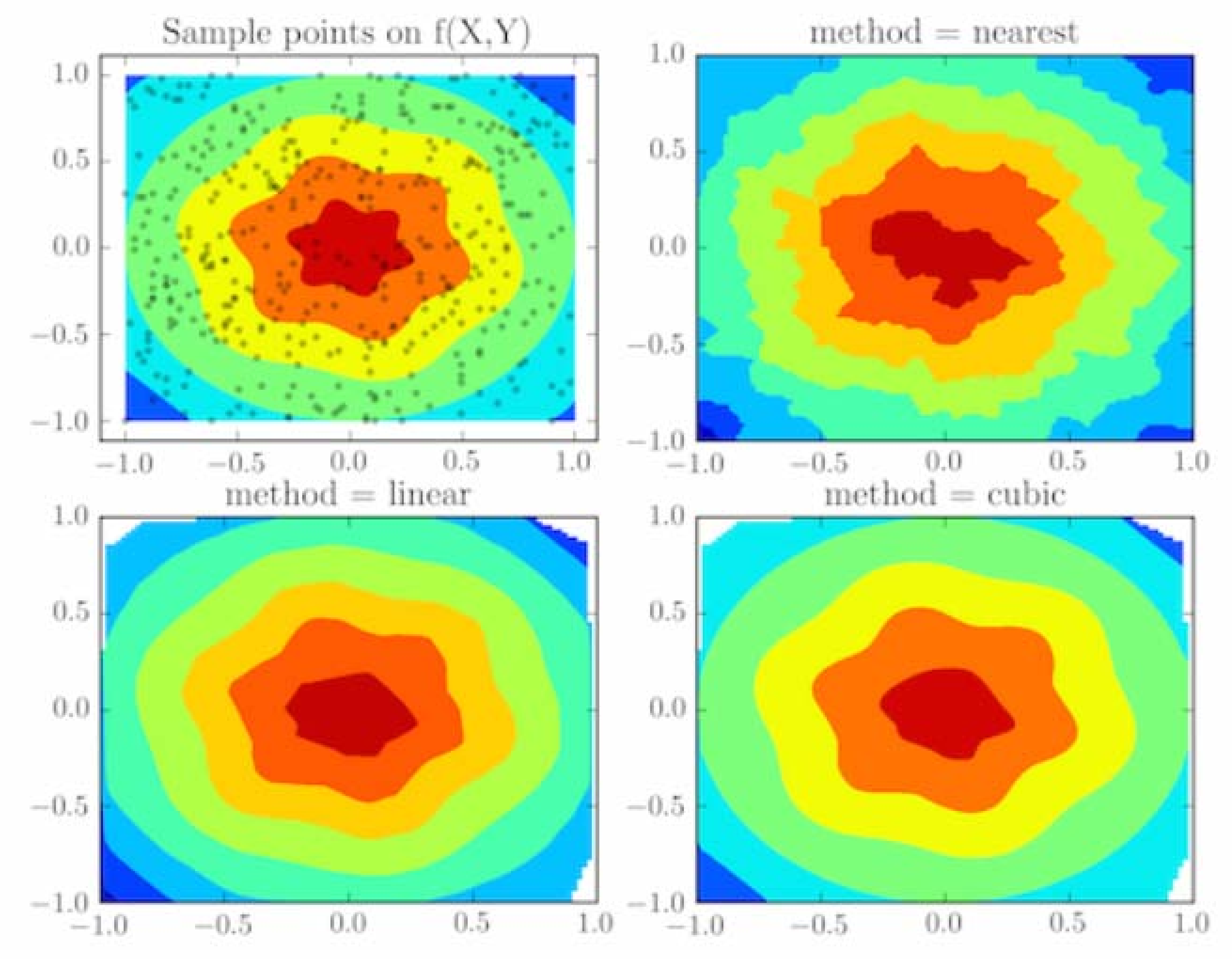
return the value determined from a cubic spline.



```python
def f(x, y):
    s = np.hypot(x, y)
    phi = np.arctan2(y, x)
    tau = s + s*(1-s)/5 * np.sin(6*phi)
    return 5*(1-tau) + tau
```

Griddata Interpolating from unorganized points.

but it works for organized data as well!



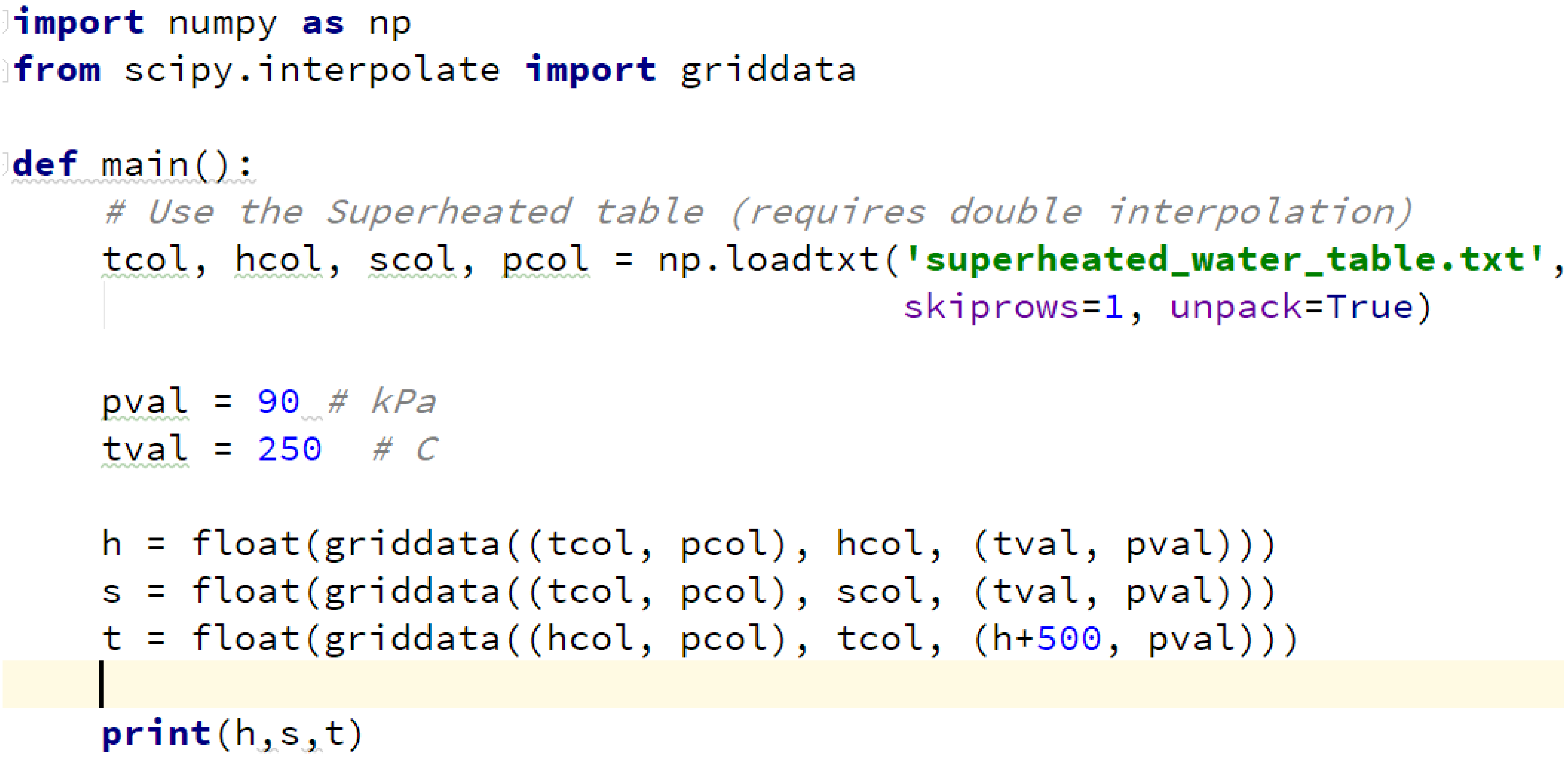
Sample points on f(X,Y)

method = nearest

method = linear

method = cubic

Using loadtxt and griddata to interpolate saturated steam properties



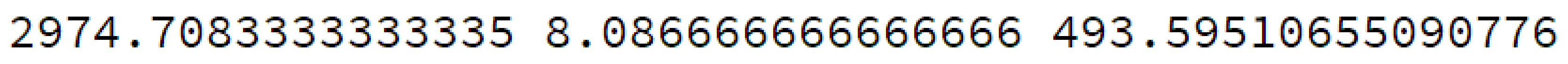
```python
import numpy as np
from scipy.interpolate import griddata

def main():
    # Use the Superheated table (requires double interpolation)
    tcol, hcol, scol, pcol = np.loadtxt('superheated_water_table.txt',
                                        skiprows=1, unpack=True)

    pval = 90 # kPa
    tval = 250  # C

    h = float(griddata((tcol, pcol), hcol, (tval, pval)))
    s = float(griddata((tcol, pcol), scol, (tval, pval)))
    t = float(griddata((hcol, pcol), tcol, (h+500, pval)))

    print(h,s,t)
```



```
2974.7083333333335 8.086666666666666 493.59510655090776
```