

第三章 利用Python读写数据

Markdown by 陈威霖

数据读写是气象工作的基本功之一，Python在进行多种数据格式读写前，需要安装不同的第三方库。

```
conda install -c conda-forge numpy pandas xlrd openpyxl xarray netcdf4
hdf5 pygrib cfgrib h5py
```

3.1 多维数据结构

Python若仅使用基础对象，不方便进行数组计算。并且多数数据的读写基于多维数组，下面介绍三种多维数组结构，包括Nddarray、DataFrame和DataArray。

a. Numpy的Nddarray

- Numpy最基本的数据结构是多维数组（N-dimensional arrays, Nddarray）
- 可利用Numpy 的“array()”函数建立。
- 使用Numpy中的函数处理后的返回数组也为Nddarray结构
- Numpy的主页：<https://numpy.org/>
- 一般缩写为np

用np.array()函数建立Nddarray结构

```
In [1]: import numpy as np
a = np.array([1,2,3,4,5])
print(type(a),a,sep='\n')
```

```
<class 'numpy.ndarray'>
[1 2 3 4 5]
```

建立一个3×2 的Nddarray

```
In [4]: a = [[1,2],[3,4],[5,6]]
b = np.array(a)
print(type(a),type(b),sep='\n')
```

```
<class 'list'>
<class 'numpy.ndarray'>
```

用shape查看其大小

- 使用Nddarray的附带函数
- 使用Numpy的shape函数

```
In [5]: print(b.shape,np.shape(b))
```

```
(3, 2) (3, 2)
```

索引方式

- Nddarray的索引方式与其它编程语言的数组索引方式相似

- 但是，每个维度的索引需遵循切片法则

例如：索引第2行第1列的数值

```
In [6]: print(b[1,0])
```

3

分段索引和使用负号索引

例如，引用前两行和后两列的元素

```
In [7]: b[:2,-2:]
```

```
Out[7]: array([[1, 2],
               [3, 4]])
```

数组间可以进行各类运算

```
In [8]: print(b*2+b/4+b%2)
```

```
[[ 3.25  4.5 ]
 [ 7.75  9.  ]
 [12.25 13.5 ]]
```

Ndarray可采用布尔值进行索引

- 在Python气象应用中被广泛使用
- 要求索引的布尔值序列要与数组的或维度的大小相符

例如：输出大于3的值

```
In [9]: msk = b>3
print(msk)
b[msk]
```

```
[[False False]
 [False  True]
 [ True  True]]
```

```
Out[9]: array([4, 5, 6])
```

索引数组中第一列大于3的数值

```
In [10]: msk = b[:,0]>3
print(msk)
print(b[msk,:])
```

```
[False False  True]
[[5 6]]
```

选取某一范围内的数据(合成分析)，或在绘图时将不显著的数值设成缺测

例如：利用布尔值索引将小于等于3 的值变成“np.nan”（numpy中的缺测值）

```
In [11]: b = b*1.
msk = b<=3
b[msk] = np.nan
print(b)
```

```
[[nan nan]
 [nan  4.]
 [ 5.  6.]]
```

对“msk”取反（使用“~”）

例如：利用上面定义的msk将大于3的值设为缺省

```
In [12]: b = np.array([[1,2],[3,4],[5,6]])*1.
b[~msk] = np.nan
print(msk,b,sep='\n')
```

```
[[ True  True]
 [ True False]
 [False False]]
[[ 1.  2.]
 [ 3. nan]
 [nan nan]]
```

布尔值间可以进行集合运算

例如：需要 $2 < b < 4$

```
In [13]: msk = (b>2) & (b<4)
print(b[msk])
```

```
[3.]
```

相当于

```
In [14]: msk0 = b>2
msk1 = b<4
msk = msk0 & msk1
b = b[msk]
print(b)
```

```
[3.]
```

b. Pandas的DataFrame

- Pandas特有的数据结构为DataFrame<https://pandas.pydata.org>
- 它附带读写二维表格的函数，包括Excel 表格和csv 数据。
- DataFrame 的数据结构与Excel 表格中看到的数据结构极为相似（二维）
- 由行和列组成，每行可以包括行索引，每列可以含有表头

	表头 1	表头 2	
索引 1				行
索引 2				
.....				
				列

图 3.1 DataFrame 的数据结构

- DataFrame 的数据结构可以利用字典建立，字典的key 值就是DataFrame 的表头，默认的索引是从0 开始的整数 - 若想使用其他索引，可以为Pandas 的“DataFrame()”函数设置“index”选项 - pandas一般缩写为pd

建立DataFrame 数据结构

例如：使用“pd.DataFrame()”函数建立DataFrame数据结构。

```
In [15]: import pandas as pd
score = {'A 同学': [65, 70, 99], 'B 同学': [50, 100, 100], 'C 同学': [75, 80, 59]}
print(score)
df = pd.DataFrame(score, index=['语文', '数学', '英语'])
print(type(df), df, sep='\n')
```

```
{'A 同学': [65, 70, 99], 'B 同学': [50, 100, 100], 'C 同学': [75, 80, 59]}
<class 'pandas.core.frame.DataFrame'>
   A 同学  B 同学  C 同学
语文    65    50    75
数学    70   100    80
英语    99   100    59
```

利用“iloc”函数，可对DataFrame 结构的二维数组进行索引

与二维的Ndarray 相似，第1个索引为行，第2个索引为列，索引的方式遵循切片式法则

```
In [16]: print(df.iloc[0,1])
```

50

或直接索引某一行

```
In [17]: print(df['B 同学'])
```

```
语文    50
数学   100
英语   100
Name: B 同学, dtype: int64
```

转换成Numpy的Ndarray

在实际计算中，不是每个数组都需要表头和索引，纯粹的数字更方便计算。

```
In [18]: print(df.values)
```

```
[[ 65  50  75]
 [ 70 100  80]
 [ 99 100  59]]
```

保存与读取

可以保存成excel (.xls或.xlsx) 或csv型数据

```
In [19]: print(df)
df.to_excel('成绩.xlsx')
df.to_csv('成绩.csv')
a = pd.read_excel('成绩.xlsx')
b = pd.read_csv('成绩.csv')
print(a,b,sep='\n')
```

	A 同学	B 同学	C 同学
语文	65	50	75
数学	70	100	80
英语	99	100	59


```

Unnamed: 0  A 同学  B 同学  C 同学
0          语文    65    50    75
1          数学    70   100    80
2          英语    99   100    59

```



```

Unnamed: 0  A 同学  B 同学  C 同学
0          语文    65    50    75
1          数学    70   100    80
2          英语    99   100    59

```

c. Xarray的DataArray

- Xarray是带标签 (label) 的多维数组<https://docs.xarray.dev/>
- 与Pandas的DataFrame有相似之处，但不再局限于二维
- xarray模块一般缩写为xr

下面建立一个三维的，包括时间、纬度、经度三个维度

```
In [23]: import xarray as xr
import numpy as np
```

利用列表循环和np.arange函数生成时间、经度和纬度，其中np.arange函数的用法与range函数相同，但是支持浮点数。

时间利用np.array函数，将字符串转换成Numpy的时间（将在第五章详细介绍）

```
In [30]: time = ['-'.join(['2100', '%02i' % (m), '01'])
               for m in range(1,13)]
time = np.array(time, dtype='datetime64[ns]') ## len(time):12
lon = np.arange(360)
lat = np.arange(-90,91)
print(time[[0,-1]],lon[[0,-1]],lat[[0,-1]],sep='\n')

['2100-01-01T00:00:00.000000000' '2100-12-01T00:00:00.000000000']
[ 0 359]
[-90 90]
```

利用np.meshgrid将一维的经纬度变成二维的（见第五章数组变形）

```
In [25]: x, y = np.meshgrid(lon,lat)
print(x.shape,y.shape)
```

(181, 360) (181, 360)

生成一个纬向和经向均是1波的数据

```
In [26]: n, m = (1,1)
A = np.cos(n*y/180*np.pi)**2
dat = [] ## 先定义一个空列表
for i in range(12):
    tmp = A*np.sin(i*2*np.pi/12-m*x/180*np.pi)
    dat.append(tmp) ## “append()”函数在每次循环时对列表元素进行增加。
dat = np.array(dat) ## np.shape(dat):(12, 181, 360)
da = xr.DataArray(dat,name='wave',dims=('time','lat','lon'),
                  coords={'time':time,'lat':lat,'lon':lon})
```

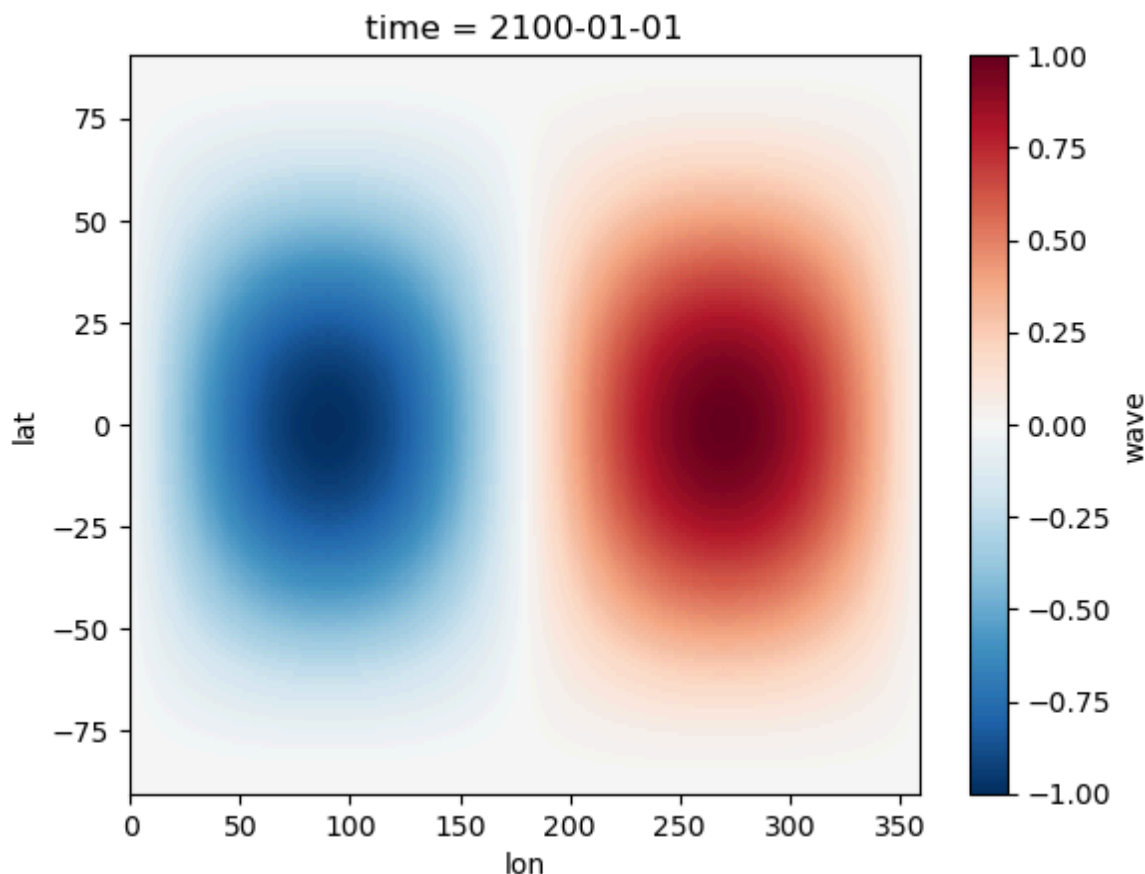
xr.DataArray函数的重要参数

- 第一个位置参数为数据
- name数据名称
- dims维度名称
- coords维度数据

Xarray还附带绘图功能（不进行介绍）

```
In [28]: da[0].plot()
```

```
Out[28]: <matplotlib.collections.QuadMesh at 0x2183e01e9c0>
```



索引方式

- 直接索引，与Nddarray一至
- 使用loc，使用维度的数值进行索引（需注意维度内数值的变化，例如由正到负，还是由负到正）
- 使用sel函数，与loc相似但需提供维度名称
- 使用维度名进行索引，与Pandas利用表头相似

```
In [29]: a = da.loc['2100-06-01',0,180]
b = da.loc['2100-06-01':'2100-12-1',0,100:102]
c = da.sel(time='2100-06-01',lon=np.arange(100,102),lat=0)
d = da['lon']
for x in [a,b,c,d]:
    print('分割线'+ '-'*20)
    print(x)
```

分割线-----

```
<xarray.DataArray 'wave' ()> Size: 8B
```

```
array(-0.5)
```

Coordinates:

```
time      datetime64[ns] 8B 2100-06-01
lat       int32 4B 0
lon       int32 4B 180
```

分割线-----

```
<xarray.DataArray 'wave' (time: 7, lon: 3)> Size: 168B
```

```
array([[ 0.76604444,  0.75470958,  0.74314483],
       [ 0.98480775,  0.98162718,  0.9781476 ],
       [ 0.93969262,  0.94551858,  0.95105652],
       [ 0.64278761,  0.65605903,  0.66913061],
       [ 0.17364818,  0.190809  ,  0.20791169],
       [-0.34202014, -0.32556815, -0.30901699],
       [-0.76604444, -0.75470958, -0.74314483]])
```

Coordinates:

```
* time      (time) datetime64[ns] 56B 2100-06-01 2100-07-01 ... 2100-12-01
lat       int32 4B 0
* lon      (lon) int32 12B 100 101 102
```

分割线-----

```
<xarray.DataArray 'wave' (lon: 2)> Size: 16B
```

```
array([0.76604444, 0.75470958])
```

Coordinates:

```
time      datetime64[ns] 8B 2100-06-01
lat       int32 4B 0
* lon      (lon) int32 8B 100 101
```

分割线-----

```
<xarray.DataArray 'lon' (lon: 360)> Size: 1kB
```

```
array([ 0,  1,  2, ..., 357, 358, 359])
```

Coordinates:

```
* lon      (lon) int32 1kB 0 1 2 3 4 5 6 7 ... 352 353 354 355 356 357 358 359
```

利用attrs函数增加属性说明

- 为使数据的说明更详细，可以自定义属性说明
- 所有的Xarray的结构均有attrs函数
- 有attrs函数就可以添加或修改说明

```
In [31]: da.attrs['units'] = 'unitless'
da.attrs['creator'] = 'Zhou Y.'
da.attrs['description'] = 'A sin wave'
da['lon'].attrs['units'] = 'degrees_east'
da['lat'].attrs['units'] = 'degrees_north'
print(da[0,0,0],da['lon'][0],da['lat'][0],sep='\n')
```

```

<xarray.DataArray 'wave' ()> Size: 8B
array(0.)
Coordinates:
  time      datetime64[ns] 8B 2100-01-01
  lat       int32 4B -90
  lon       int32 4B 0
Attributes:
  units:          unitless
  creator:        Zhou Y.
  destription:    A sin wave
<xarray.DataArray 'lon' ()> Size: 4B
array(0)
Coordinates:
  lon      int32 4B 0
Attributes:
  units:    degrees_east
<xarray.DataArray 'lat' ()> Size: 4B
array(-90)
Coordinates:
  lat      int32 4B -90
Attributes:
  units:    degrees_north

```

转换成Nddarray

并不是所有的数据都需要描述，纯粹的数字更方便计算

```

In [32]: print(type(da.to_numpy()))
         print(type(da.values))

```

```

<class 'numpy.ndarray'>
<class 'numpy.ndarray'>

```

建立Dataset

多个DataArray在一起，可以共享维度信息

```

In [33]: ds = xr.Dataset(data_vars={'wave':da})
         print(ds)

```

```

<xarray.Dataset> Size: 6MB
Dimensions:  (time: 12, lat: 181, lon: 360)
Coordinates:
  * time      (time) datetime64[ns] 96B 2100-01-01 2100-02-01 ... 2100-12-01
  * lat       (lat) int32 724B -90 -89 -88 -87 -86 -85 -84 ... 85 86 87 88 89 90
  * lon       (lon) int32 1kB 0 1 2 3 4 5 6 7 ... 352 353 354 355 356 357 358 359
Data variables:
  wave       (time, lat, lon) float64 6MB 0.0 -6.544e-35 ... -1.818e-33

```

存储数据成netCDF格式

DataArray和Dataset均可以存成netCDF格式，注意个别Xarray中文不友好。

```

In [34]: da.to_netcdf('wave.nc')
         ds.to_netcdf('wave.nc')

```