Lesson 13 Python Numpy Basic Operation

1. Brief Introduction

NumPy (Numerical Python) is an extension library of Python, which supports a large number of dimensional array and matrix operations, and also provides a large number of mathematical function libraries for array operations.

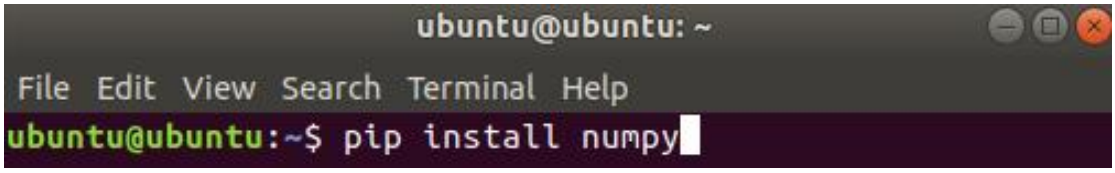
1) In some ways, Numpy arrays are greatly similar to built-in lists in Python. It is an essential tool for Python data science.

2) A numpy one-dimensional array can be considered as a vector, and a two-dimensional array is a data frame (precisely corresponding to the matrix in R).

3) The biggest difference between numpy arrays and lists in python is that the type of the data contained in the former should be the same, but there is no restriction on the latter.

2. Numpy Importing

Numpy is a third-party library that needs to be installed first. Open the command line terminal and enter the command to install "pip install numpy".



Then we need to import it in python with "import numpy". We can use "import numpy as np" to take "np" as an alias.

3. Create Array

1) np.array(): Create directly

```
# one-dimensional array
x1 = np.array([1,2,3])
x1.shape
# (3,) represents a one-dimensional array of length 3

# two-dimensional array
x2 = np.array([[1,3,5],[11,12,13]])
x2.shape
# (2,3) represent the 2-d matrix of two rows and three columns
```

3.1 Take Fixed Value

1) np.zeros(n, dtype=int): Create a one-dimensional array of integers of length n and all zeros

2) np.ones((a,b), dtype=float): Create a two-dimensional floating-point array with "**a**" row and "**b**" column, and all 1.

3) np.full((a,b),x, dtype=int/float): Create a two-dimensional array with "**a**" row and "**b**" column, and all **x**.

```
np.full((3,3),1.11)
# array([[1.11, 1.11, 1.11],
#        [1.11, 1.11, 1.11],
#        [1.11, 1.11, 1.11]])
```

3.2 Regular Distribution

1) np.arange(): arithmetic progression

```
In [144]: np.arange(10)
Out[144]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [145]: np.arange(1,10,2)
Out[145]: array([1, 3, 5, 7, 9])
```

2) np.random.randint(): Random integer

```
In [148]: np.random.randint(0,100,10)
Out[148]: array([85, 48, 49, 69, 41, 35, 64, 95, 69, 94])

In [149]: np.random.randint(0,10,(3,3))
Out[149]:
array([[0, 1, 2],
       [4, 2, 0],
       [3, 2, 0]])
```

3) np.random.random(10) : A one-dimensional array with a length of 10 and distributed uniformly from 0 to 1.

4) np.random.normal(0, 1, c(2,3)): A two-dimensional array normally distributed with two rows and three columns, a mean of 0 and a standard deviation of 1.

4. Take Subset

4.1 Index Starts from 0

1) Similar to the way list takes subset in Python, the index of numpy array also starts from 0.

2) The index of negative can be seen as the n^{th} last value.

4.2 Take Subset of Single Value

View with the element subscript.

```
In [151]: x1 = np.arange(10)

In [152]: x1
Out[152]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [153]: x1[9]
Out[153]: 9
```

4.3 Slice Takes Subset

1) For the one-dimensional array x1 of the length of 10, x1[0:5] represents the subset from index0 to index4, and the step size is 1 by default.

2) When a of [a:b:c] is 0, a can be omitted. When both a and b are omitted, it represents the complete set.

3) When the step size is negative, it represents take subset in reversed order.

```
In [162]: x1[1:7]
Out[162]: array([1, 2, 3, 4, 5, 6])

In [163]: x1[:5]
Out[163]: array([0, 1, 2, 3, 4])

In [164]: x1[2:8:2]
Out[164]: array([2, 4, 6])

In [165]: x1[9:5:-1]
Out[165]: array([9, 8, 7, 6])
```

4.4 Multiple Index of the List Take Subset

Take the subset in sequence according to the index.

```
In [38]: x1
Out[38]: array([2, 3, 6, 7, 0, 8, 2, 7, 6, 8])

In [39]: x1[[0,2,6]]
Out[39]: array([2, 6, 2])

In [40]: x2
Out[40]:
array([[11, 11,  2,  4],
       [ 0,  9,  9,  7],
       [ 2,  5,  0, 10]])

In [41]: x2[:,[1,2]]
Out[41]:
array([[11,  2],
       [ 9,  9],
       [ 5,  0]])
```

4.4 Logical Boolean Takes Subset

```
In [35]: x1
Out[35]: array([2, 3, 6, 7, 0, 8, 2, 7, 6, 8])

In [36]: x1[(x1 < 4) | (x1 > 7)]
Out[36]: array([2, 3, 0, 8, 2, 8])

In [37]: x1[~((x1 >= 4) & (x1 <= 7))]
Out[37]: array([2, 3, 0, 8, 2, 8])
```

4.5 Create Unrelated Subset Replicas

copy() can create unrelated subset replicas

```
In [185]: x2_sub = x2[:2, :3].copy()

In [186]: x2_sub[0,0] = 10000

In [187]: x2_sub
Out[187]:
array([[10000,  1000,     3],
       [   18,    14,     9]])

In [188]: x2
Out[188]:
array([[   18,  1000,     3,    17],
       [   18,    14,     9,     1],
       [    4,    10,    11,     8]])
```

5.Array Dimension Transforming

1) Transpose matrix with T()：Transform row and column

```
In [200]: x2
Out[200]:
array([[   18,  1000,     3,    17],
       [   18,    14,     9,     1],
       [    4,    10,    11,     8]])

In [201]: x2.T
Out[201]:
array([[   18,    18,     4],
       [1000,    14,    10],
       [    3,     9,    11],
       [   17,     1,     8]])
```

2) Transform dimension with reshaple(): transform the dimension according to your desire.

```
In [204]: x2
Out[204]:
array([[  18, 1000,    3,   17],
       [  18,   14,    9,    1],
       [   4,   10,   11,    8]])

In [205]: x2.reshape((2,6))
Out[205]:
array([[  18, 1000,    3,   17,   18,   14],
       [   9,    1,    4,   10,   11,    8]])

In [206]: x2.reshape((1,12))
Out[206]:
array([[  18, 1000,    3,   17,   18,   14,    9,    1,    4,   10,   11,
          8]])
```

6. Arrays Merging and Splitting

6.1 Array Merging

1) concatenate() : Connect two one-dimensional array or merge two two-dimensional arrays according to row and column.

```
In [207]: x1_1 = np.random.randint(10, size = 3)

In [208]: x1_2 = np.random.randint(20, size = 5)

In [209]: np.concatenate([x1_1,x1_2])
Out[209]: array([ 2,  3,  0,  0,  0,  6, 19, 14])

In [210]: np.concatenate([x1_1,x1_2,x1_2])
Out[210]: array([ 2,  3,  0,  0,  0,  6, 19, 14,  0,  0,  6, 19, 14])
```

2) vstack(): Merge two arrays vertically, as long as these two arrays have the same number of columns.

```
In [222]: np.vstack([x2_1, x2_2])
Out[222]:
array([[ 8,  0,  8],
       [ 5,  9,  0],
       [60, 61, 83],
       [33, 32, 70]])
```

3) hstack(): Merge two arrays horizontally, as long as these two arrays

have the same number of rows.

```
In [223]: np.hstack([x2_1, x2_2])
Out[223]:
array([[ 8,  0,  8, 60, 61, 83],
       [ 5,  9,  0, 33, 32, 70]])
```

6.2 Split Array

1) split(): Split one-dimensional array. Where to split will be designated by the index of the parameter. When the function returns multiple objects, we need to set the corresponding numbers of object names in front of the equal sign. When the function result returns multiple objects, set the corresponding number of object names before the equal sign

```
In [224]: x1 = np.random.randint(10, size = 10)

In [225]: x1_1, x1_2, x1_3 = np.split(x1, [2,5,])

In [226]: x1_1
Out[226]: array([5, 7])

In [227]: x1_2
Out[227]: array([8, 8, 9])

In [228]: x1_3
Out[228]: array([2, 8, 6, 6, 9])
```

2) vsplit(): Split a two-dimensional array horizontally. And hsplit() splits a two-dimensional array vertically.

```
In [229]: x2 = np.random.randint(12, size = (3,4))

In [230]: x2_up, x2_down = np.vsplit(x2,[1])

In [231]: x2_up
Out[231]: array([[1, 6, 8, 8]])

In [232]: x2_down
Out[232]:
array([[ 3,  2,  3, 10],
       [ 6,  3,  6,  5]])

In [233]: x2_left, x2_right = np.hsplit(x2,[1])

In [234]: x2_left
Out[234]:
array([[1],
       [3],
       [6]])

In [235]: x2_right
Out[235]:
array([[ 6,  8,  8],
       [ 2,  3, 10],
       [ 3,  6,  5]])
```