(http://www.pieriandata.com)

NumPy Indexing and Selection

In this lecture we will discuss how to select elements or groups of elements from an array.

In [1]:
```
import numpy as np
```

In [2]:
```
#Creating sample array
arr = np.arange(0,11)
```

In [3]:
```
#Show
arr
```

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

Bracket Indexing and Selection

The simplest way to pick one or some elements of an array looks very similar to python lists:

In [4]:
```
#Get a value at an index
arr[8]
```

Out[4]:
```
8
```

In [5]:
```
#Get values in a range
arr[1:5]
```

Out[5]:
```
array([1, 2, 3, 4])
```

In [6]:
```
#Get values in a range
arr[0:5]
```

Out[6]:
```
array([0, 1, 2, 3, 4])
```

Broadcasting

NumPy arrays differ from normal Python lists because of their ability to broadcast. With lists, you can only reassign parts of a list with new parts of the same size and shape. That is, if you wanted to replace the first 5 elements in a list with a new value, you would have to pass in a new 5 element list. With NumPy arrays, you can broadcast a single value across a larger set of values:

In [7]:
```
#Setting a value with index range (Broadcasting)
arr[0:5]=100

#Show
arr
```

Out[7]:
```
array([100, 100, 100, 100, 100,   5,   6,   7,   8,   9,  10])
```

In [8]:
```
# Reset array, we'll see why I had to reset in  a moment
arr = np.arange(0,11)

#Show
arr
```

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

In [9]:
```
#Important notes on Slices
slice_of_arr = arr[0:6]

#Show slice
slice_of_arr
```

Out[9]:
```
array([0, 1, 2, 3, 4, 5])
```

In [10]:
```
#Change Slice
slice_of_arr[:]=99

#Show Slice again
slice_of_arr
```

Out[10]:
```
array([99, 99, 99, 99, 99, 99])
```

Now note the changes also occur in our original array!

In [11]:
```
arr
```

Out[11]:
```
array([99, 99, 99, 99, 99, 99,  6,  7,  8,  9, 10])
```

Data is not copied, it's a view of the original array! This avoids memory problems!

In [12]:
```
#To get a copy, need to be explicit
arr_copy = arr.copy()

arr_copy
```

Out[12]:
```
array([99, 99, 99, 99, 99, 99,  6,  7,  8,  9, 10])
```

Indexing a 2D array (matrices)

The general format is **arr_2d[row][col]** or **arr_2d[row,col]**. I recommend using the comma notation for clarity.

In [13]:
```
arr_2d = np.array(([5,10,15],[20,25,30],[35,40,45]))

#Show
arr_2d
```

Out[13]:
```
array([[ 5, 10, 15],
       [20, 25, 30],
       [35, 40, 45]])
```

In [14]:
```
#Indexing row
arr_2d[1]
```

Out[14]:
```
array([20, 25, 30])
```

In [15]:
```
# Format is arr_2d[row][col] or arr_2d[row,col]

# Getting individual element value
arr_2d[1][0]
```

Out[15]:
```
20
```

In [16]:
```
# Getting individual element value
arr_2d[1,0]
```

Out[16]:
```
20
```

In [17]:
```
# 2D array slicing

#Shape (2,2) from top right corner
arr_2d[:2,1:]
```

Out[17]:
```
array([[10, 15],
       [25, 30]])
```

In [18]:
```
#Shape bottom row
arr_2d[2]
```

Out[18]:
```
array([35, 40, 45])
```

In [19]:
```
#Shape bottom row
arr_2d[2,:]
```

Out[19]:
```
array([35, 40, 45])
```

More Indexing Help

Indexing a 2D matrix can be a bit confusing at first, especially when you start to add in step size. Try google image searching *NumPy indexing* to find useful images, like this one:

```
>>> a[0,3:5]
array([3,4])

>>> a[4:,4:]
array([[44, 45],
       [54, 55]])

>>> a[:,2]
array([2,12,22,32,42,52])

>>> a[2::2,::2]
array([[20,22,24]
       [40,42,44]])
```

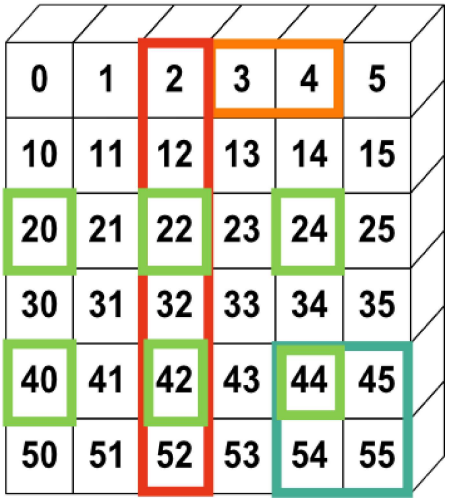

Image source: http://www.scipy-lectures.org/intro/numpy/numpy.html (http://www.scipy-lectures.org/intro/numpy/numpy.html)

Conditional Selection

This is a very fundamental concept that will directly translate to pandas later on, make sure you understand this part!

Let's briefly go over how to use brackets for selection based off of comparison operators.

In [20]:
```
arr = np.arange(1,11)
arr
```

Out[20]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

In [21]:
```
arr > 4
```

Out[21]:
```
array([False, False, False, False,  True,  True,  True,  True,  True,
        True])
```

In [22]:
```
bool_arr = arr>4
```

In [23]:
```
bool_arr
```

Out[23]:
```
array([False, False, False, False,  True,  True,  True,  True,  True,
        True])
```

In [24]:
```
arr[bool_arr]
```

Out[24]:
```
array([ 5,  6,  7,  8,  9, 10])
```

In [25]:
```
arr[arr>2]
```

Out[25]:
```
array([ 3,  4,  5,  6,  7,  8,  9, 10])
```

In [26]:
```
x = 2
arr[arr>x]
```

Out[26]:
```
array([ 3,  4,  5,  6,  7,  8,  9, 10])
```

Great Job!