In [1]:
```
from root_numpy import root2rec, root2array
```

In [2]:
```
#root2rec
ar = root2rec('test/test.root','tree')
print ar.i
print ar.f
#ipython autocomplete columnname patch is available with this numpy patch
#https://github.com/pitti118/numpy/commit/a996292238ab98dcf53f2d48476d637eab9f1a
ar.i[0] #ar[0].i won't work
ar[0][0]
```

```
[ 0  2  4  6  8 10 12 14 16 18]
[  0.           1.20000005   2.4000001    3.5999999    4.80000019   6.
   7.19999981   8.39999962   9.60000038  10.80000019]
```

Out[2]: 0

In [3]:
```
ar.f[ar.i>5]
```

Out[3]:
```
array([  3.5999999 ,   4.80000019,   6.        ,   7.19999981,
         8.39999962,   9.60000038,  10.80000019], dtype=float32)
```

In [4]:
```
#root2array is available if you don't like recarray
a=root2array('test/test.root','tree')
#this tree has two column i and integer and f as float
a #you will see that a is a structure array
```

Out[4]:
```
array([(0, 0.0), (2, 1.2000000476837158), (4, 2.4000000953674316),
       (6, 3.5999999046325684), (8, 4.800000190734863), (10, 6.0),
       (12, 7.199999809265137), (14, 8.399999618530273),
       (16, 9.600000381469727), (18, 10.800000190734863)],
      dtype=[('i', '<i4'), ('f', '<f4')])
```

In [5]:
```
#access whole column
print a['i']
print a['f']
```

```
[ 0  2  4  6  8 10 12 14 16 18]
[  0.           1.20000005   2.4000001    3.5999999    4.80000019   6.
   7.19999981   8.39999962   9.60000038  10.80000019]
```

In [6]:
```
#access 0th record
print a[0]
#and the first record
print a[1]
```

```
(0, 0.0)
(2, 1.2000000476837158)
```

In [7]:
```
#access 1st record column i
print a[1]['i']
#and this may confuse you but
print a['i'][1]
#there is a tiny different here a[some string] will return numpy array of that
#column which you can index it again while a[integer] will return the that stru
```

```
#which you can index it again
print a[1][0] #this one works too
```

```
2
2
2
```