In [1]:
```
import root_numpy
# there is only one method read(fnames,treename,branches=None)
# you can specify branch to extract as list of string if memory usage is an issue
```

In [2]:
```
a=root_numpy.read('test/test.root','tree')
#this tree has two column i and integer and f as float
print a #you will see that a is a structure array
```

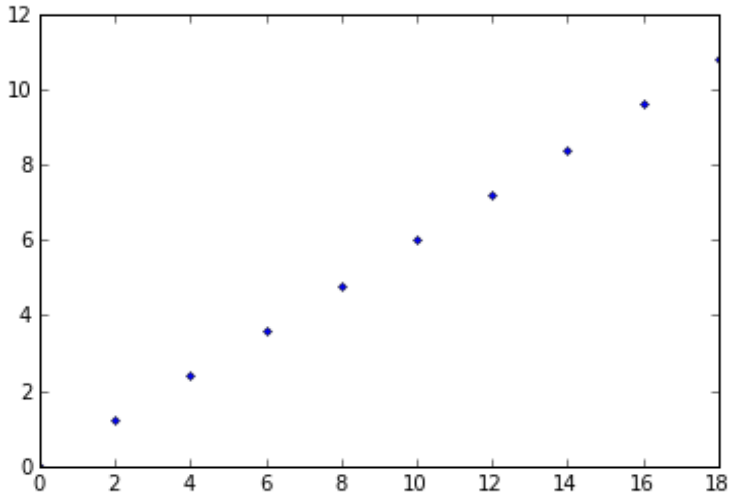
```
[(0, 0.0) (2, 1.2000000476837158) (4, 2.4000000953674316)
 (6, 3.5999999046325684) (8, 4.800000190734863) (10, 6.0)
 (12, 7.199999809265137) (14, 8.399999618530273) (16, 9.600000381469727)
 (18, 10.800000190734863)]
```

In [3]:
```
#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]
```

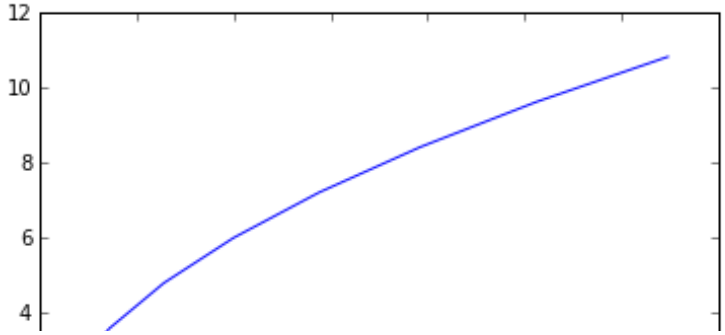
In [4]:
```
#plot something
plot(a['i'],a['f'],'.')
```

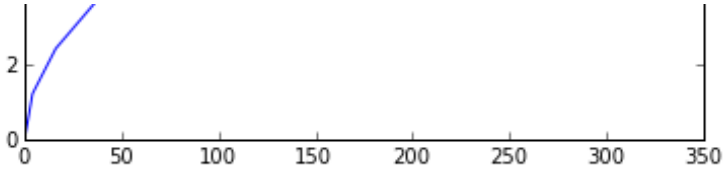
Out[4]: [<matplotlib.lines.Line2D at 0x103b1f250>]



In [5]:
```
#some numpy magic
plot(a['i']**2,a['f'])
```

Out[5]: [<matplotlib.lines.Line2D at 0x106287e10>]





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']
```

```
2
```

In [8]:
```
#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 co
#while a[integer] will return the that structure which you can index it again
```

```
2
```

In [9]:
```
#this one works too
print a[1][0]
```

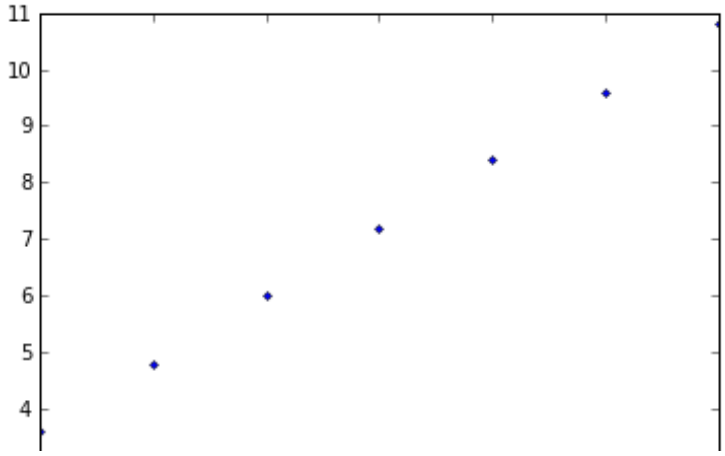
```
2
```

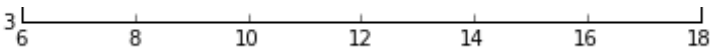
In [10]:
```
#if you don't like typing ['somecol']
ar = a.view(np.recarray)
print ar.i
print ar.f
```

```
[ 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 [11]:
```
#make some cut
plot(ar.i[ar.i>5],ar.f[ar.i>5],'.')
```

Out[11]:
```
[<matplotlib.lines.Line2D at 0x1068f07d0>]
```





In [31]:

In [ ]: