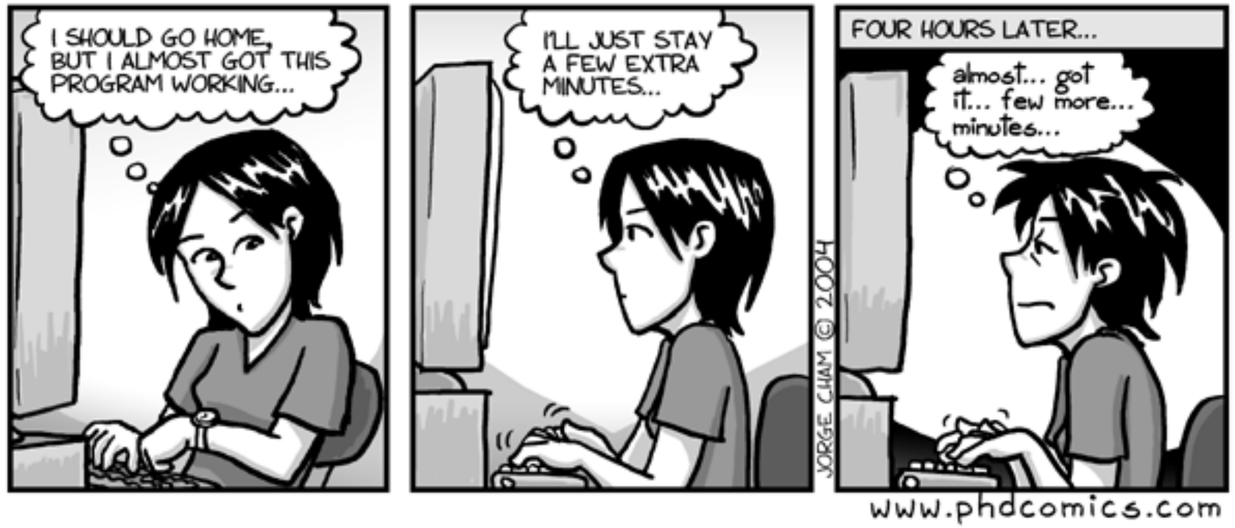
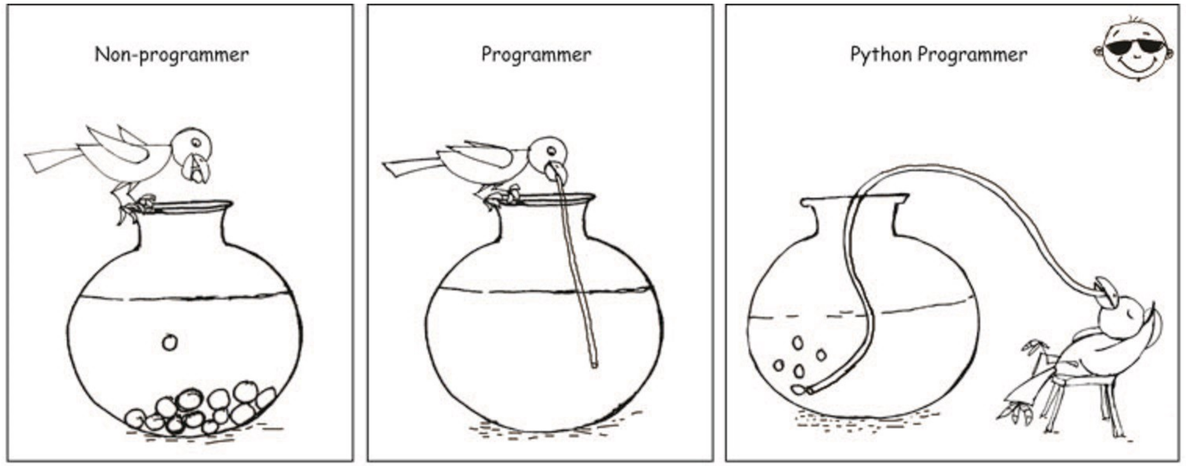
Python Scripting - Part 2

Fall 2018
PCfB Class 5
September 29, 2018





Lists

```
[1, '1', 'one', [1,2]]
```

- Ordered arrays of items
- Mutable
- Square brackets

Lists

```
[1, '1', 'one', [1,2]]
```

- Ordered arrays of items
- Mutable
- Square brackets

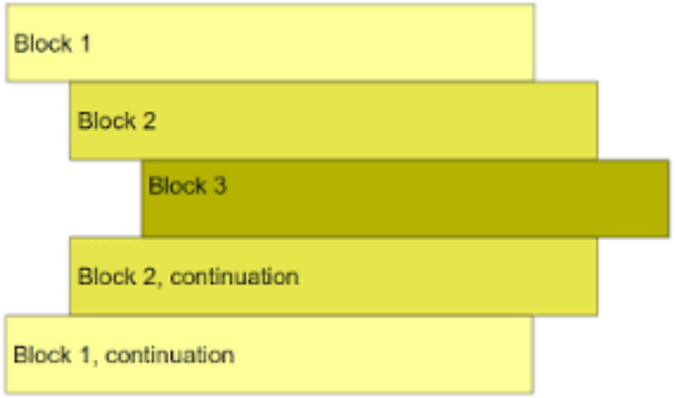
Dictionaries

```
{1:'one',2:'two'}
```

- Unordered key:value pairs
- Mutable
- Keys must be non-mutable and unique
- Curly brackets

Indentation

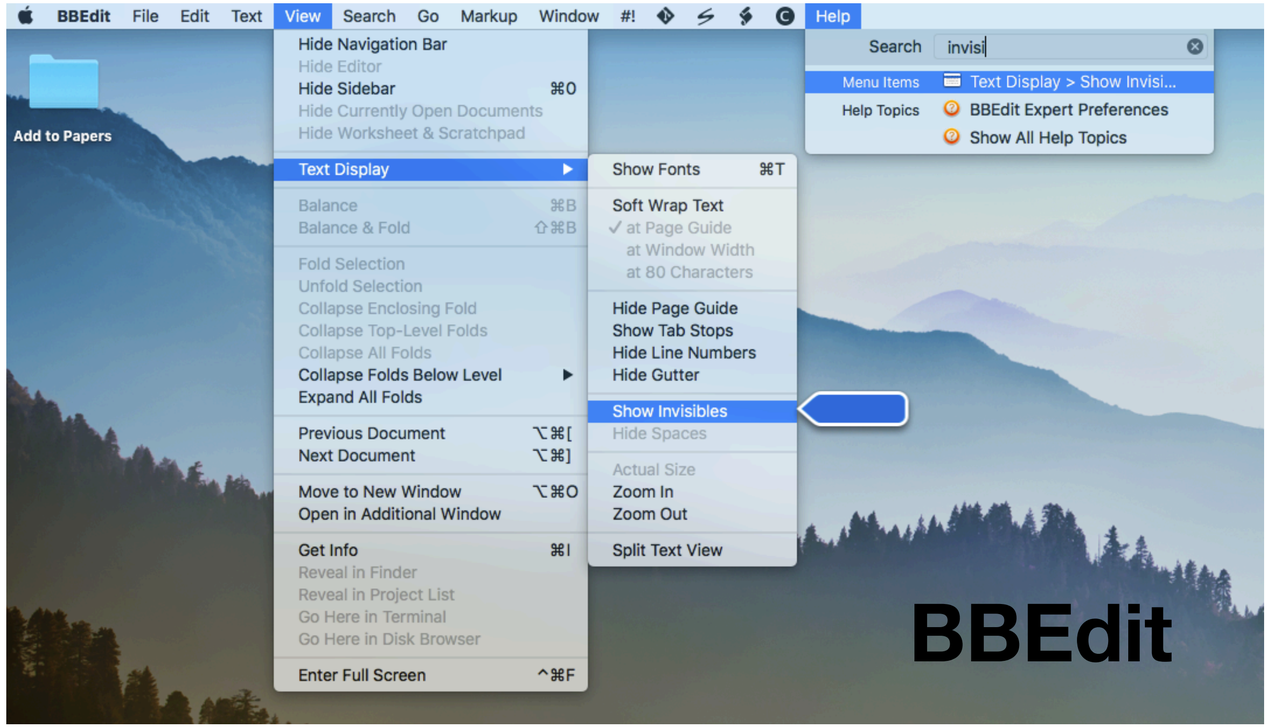
- White space at the beginning of a line is used to define blocks of code
- Can use tabs, spaces or even a combination
- Best practice is to choose one and stick with it




```
2  for item in range(10):
3      print('I')
4      print('am')
5      print('a')
6      if item % 2 == 0:
7          print('funny')
8          print('and')
9          print('silly')
10     else:
11         print('dull')
12         print('and')
13         print('serious')
14     print('block')
15     print('used')
16     print('as')
17     print('example.')
```

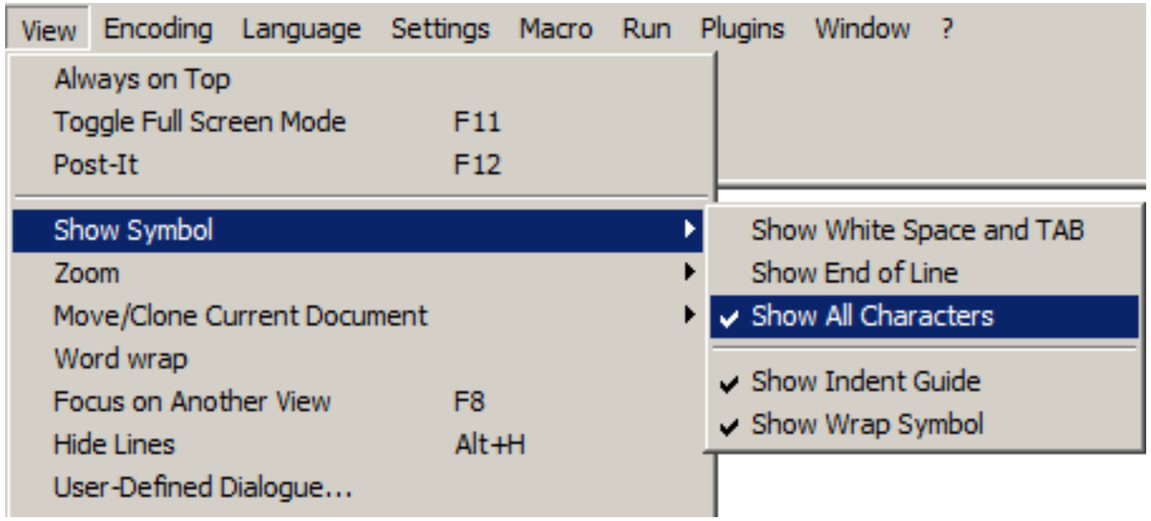
Show invisibles

```
1   ⇥   ⇥   ⇥   ⇥   This·line·starts·with·4·tabs¬
2   ····************This·line·starts·with·16·spaces¬
3   ¬
4   ¬
```



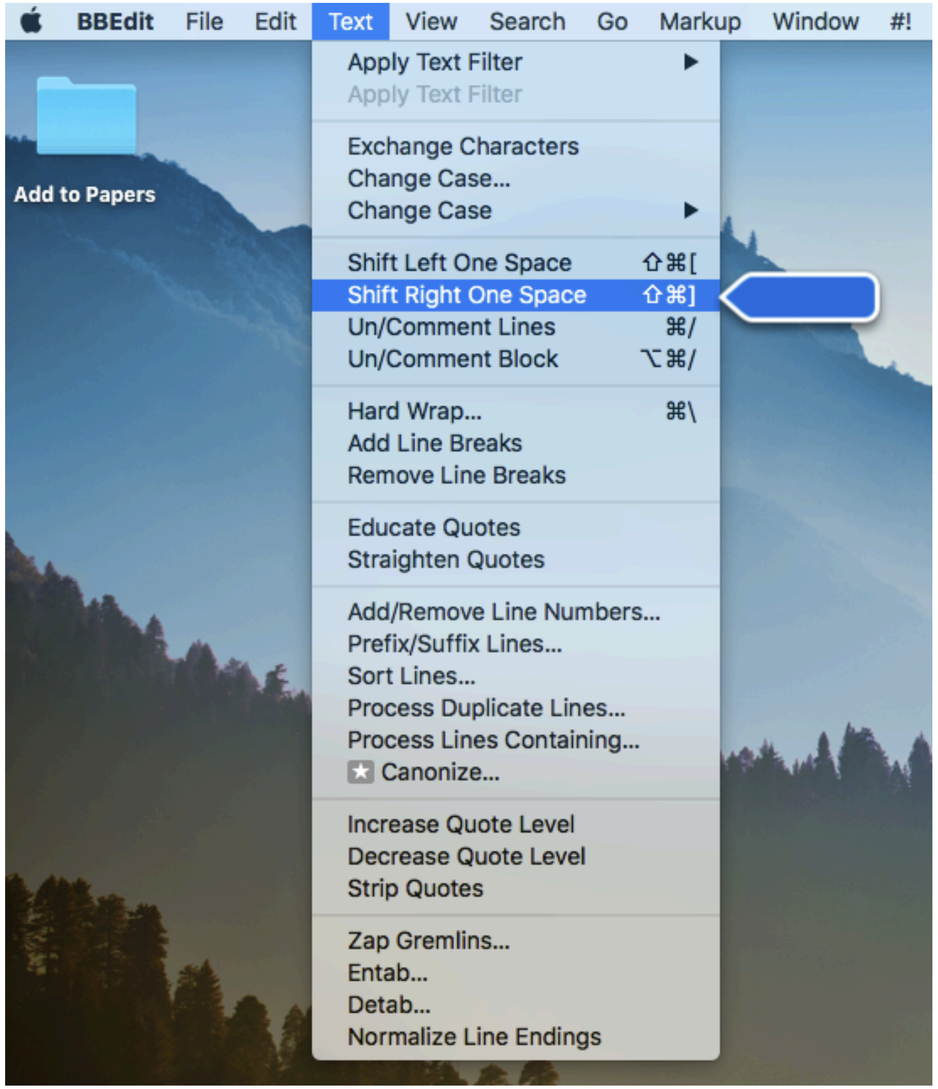
BBEdit

Notepad++



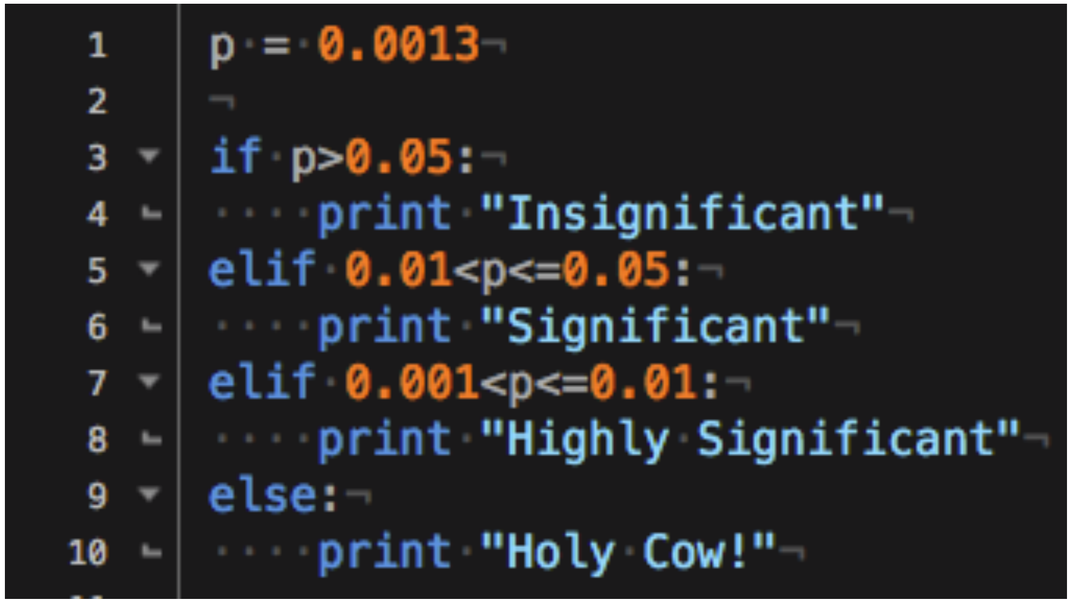
Indentation shortcut

- Many text editors have shortcuts for quickly indenting blocks of code



if, elif, else

- Defines blocks of code that are only executed under certain conditions
- Contents must be indented



`for` loops

- Repeatedly execute the same commands for different files, parameter values, etc.
- Syntax very similar to Bash
- Contents must be indented

```
15  pvalues = [0.67, 0.0003, 0.0013, 0.05, 0.76]
16
17  for p in pvalues:
18      if p>0.05:
19          print "Insignificant"
20      elif 0.01<p<=0.05:
21          print "Significant"
22      elif 0.001<p<=0.01:
23          print "Highly Significant"
24      else:
25          print "Holy Cow!"
```

List comprehension

- Essentially simple for loops contained within a list
- Always returns a list

```
28  numstring="123456789"¬
29  sqroots·=·[float(x)**(0.5)·for·x·in·numstring]¬
```

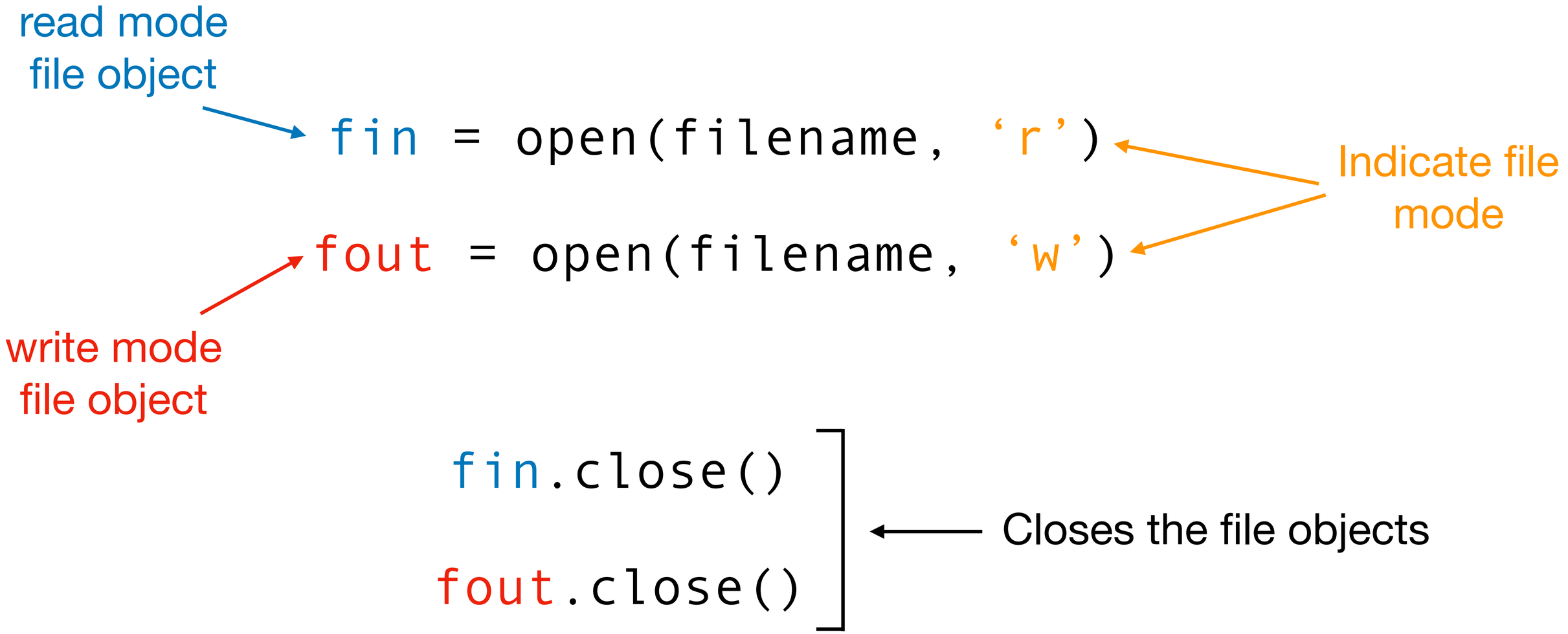
```
>>> numstring="123456789"
>>> sqroots = [float(x)**(0.5) for x in numstring]
>>>
>>> sqroots
[1.0, 1.4142135623730951, 1.7320508075688772, 2.0, 2.236067
97749979, 2.449489742783178, 2.6457513110645907, 2.82842712
47461903, 3.0]
```

Functions

- Easy to write custom functions for repetitive tasks
- Always start with `def`
- Contents must be indented

```
33  def sqrt(num):
34      squareroot = float(num)**(0.5)
35      return squareroot
36
37  sqroots = [sqrt(x) for x in numstring]
```

Reading from/Writing to files



Exercises

- Jupyter Notebook
  - Get familiar with syntax for new tools
- Stand-alone analysis scripts