Low-level I/O

In [87]:
```
# Matlab: fopen, fread, fwrite, fclose  -->  Python: open, .read, .close
#Let's open a file and write some things..
f=open('ioformatlabbers_temp.txt' ,'w');
# f is now a file object.
f.write("apple,5\n" );
f.write("orange,6\n" );
f.write("banana,7\n" );
f.close();


#Let's now read the file back...
f=open('ioformatlabbers_temp.txt' ,'r');
s=f.read(15);
print('--- Read 15 characters: "' +s+'"');

#Let's now read line by line...
#First move back to the beginning of the file:
f.seek(0,0);
s=f.readline();
print('--- Read the first line: "' +s+'"');

#Note that f.readline() does not strip the newline character (whereas Matlab's
# fgetl() does.)
#If you want to strip any "space" characters from the end (including newline),
# use str.rstrip()
s=f.readline();
print('--- Read the second line: "' +s.rstrip()+'"');

f.close();
```

```
--- Read 15 characters: "apple,5
orange,"
--- Read the first line: "apple,5
"
--- Read the second line: "orange,6"
```

Reading a file line by line, the easy way

In [88]:
```
# A python file object is "iterable" in a for loop, it acts as if it is a
# list of strings (each string being a line from the file.)
f=open('ioformatlabbers_temp.txt' ,'r');
for aline in f:
    print('--- Read a line: "' +aline+'"');

f.close();
```

```
--- Read a line: "apple,5
"
--- Read a line: "orange,6
"
--- Read a line: "banana,7
"
```

Reading the entire file

In [89]:
```
# To read the entire file at once, just use .open() without any input arguments.
```

In [90]:
```
f=open('ioformatlabbers_temp.txt' ,'r');
s=f.read();
print('--- Read the entire file: "' +s+'"');

f.close();
```

```
--- Read the entire file: "apple,5
orange,6
banana,7
"
```

In [91]:
```
# If you think having a function that reads an entire file is a useful thing to ha
# then define one. (this would be similar to Matlab's fileread())

# We can even let user define the number of bytes to read. So, if bytes is given
# we'll read that many characters, otherwise we read the entire file.
def myfileread(filename,bytes=None):
    f=open(filename,'r');
    s=f.read(bytes);
    f.close();
    return s;

# Let's now make use of this function.
s=myfileread('ioformatlabbers_temp.txt' )
print('--- Read the entire file using myfileread(): "'  +s+'"')

s=myfileread('ioformatlabbers_temp.txt' ,3)
print('--- Read the first 3 characters using myfileread(): "'  +s+'"')
```

```
--- Read the entire file using myfileread(): "apple,5
orange,6
banana,7
"
--- Read the first 3 characters using myfileread(): "app"
```

Downloading a file from the web

In [92]:
```
# The url library has changed a bit from python 2 to python 3.
# You may use the following function, so your code works regardless of which
#  python version it is being run in.

import sys

def mydownloadfile(url, filename):
 if (sys.version_info > (3, 0)):
    import urllib.request
    urllib.request.urlretrieve(url, filename)
 else:
    import urllib
    urllib.urlretrieve(url,filename)


# Let's test it:
mydownloadfile('http://httpbin.org/','mydownload_temp.html');
s=myfileread('mydownload_temp.html',300)
print('--- Downloaded file contents (first 300bytes): "'  +s+'"')
```

```
--- Downloaded file contents (first 300bytes): "<!DOCTYPE html>
<html>
<head>
  <meta http-equiv='content-type' value='text/html;charset=utf8'>
  <meta name='generator' value='Ronn/v0.7.3 (http://github.com/rtomayko/ronn/
tree/0.7.3)'>
  <title>httpbin(1): HTTP Client Testing Service</title>
  <style type='text/css' media='all'>
  /* style: man */"
```

In [93]:
```
# When you are working with file downloads, you'd frequently want to download
# a file only if it hasn't been downloaded before. To accomplish that, you can
# just check if the file is present from before.
import os
if os.path.isfile('mydownload_temp.html'):
    print('file "mydownload_temp.html" exists.' )
```

```
file "mydownload_temp.html" exists.
```

Reading csv files

Python has its own csv module. But you may find the pandas third-party module easier to use. Once you read a file as a pandas object, you can index/select rows or columns. See http://pandas.pydata.org/pandas-docs/stable/indexing.html (http://pandas.pydata.org/pandas-docs/stable/indexing.html)

In [94]:
```
import pandas as pd

es = pd.read_csv('ioformatlabbers_temp.txt' ,names=['fruit','count']);
es
```

Out[94]:

	fruit	count
0	apple	5
1	orange	6
2	banana	7

In [95]:
```
# Use [columname] to extract a specific column
es['fruit']
```

Out[95]:
```
0     apple
1    orange
2    banana
Name: fruit, dtype: object
```

In [96]:
```
# and [columname][rowindex] to extract a single cell
es['fruit'][0]
```

Out[96]:
```
'apple'
```

In [97]:
```
# Use iloc() to extract a specific row
es.iloc[0]
```

Out[97]:
```
fruit    apple
count        5
Name: 0, dtype: object
```

In [98]:
```
# Use row-column indexing with iloc to extract specific rows & columns
es.iloc[0:2,0:2]
```

Out[98]:

	fruit	count
0	apple	5
1	orange	6

In [99]:
```
es.iloc[1,1]
```

Out[99]:
```
6
```

Reading Excel files

In [102]:
```
#Pandas can read Excel files, too.
url='http://sacan.biomed.drexel.edu/ftp/bmeprog/crps_data.xlsx'   ;
mydownloadfile (url,'crps_data.xlsx' );
es = pd.read_excel ('crps_data.xlsx' );

#data too large. let's just show first 5 rows and first 7 columns.
es.iloc[0:5,0:7]
```

Out[102]:

	Group	Age	Gender	Weight	Height	BMI	Pain
0	CRPS	50	M	224	73.533514	31.404267	9.377812
1	CRPS	59	F	180	62.625479	28.676935	8.647270
2	CRPS	22	F	167	62.253894	32.966045	8.666216
3	CRPS	48	F	113	61.476092	19.600357	7.131505
4	CRPS	53	M	166	70.076665	25.661336	5.791540

In [ ]: