Modules: Leveraging Internet Data

Python 3

home (../handouts.html)

Importing Python Modules for Versatility and Power

A Module is Python code (a *code library*) that we can *import* and use in our own code -- to do specific types of tasks.

```
import datetime                       # make datetime (a library module) p

dt = datetime.date.today()            # generate a new date object (dt)
print(dt)                              # prints today's date in YYYY-MM-DD
dt = dt + datetime.timedelta(days=1)
print(dt)                              # prints tomorrow's date
```

Once a module is imported, its Python code is made available to our code. We can then call specialized functions and use objects to accomplish specialized tasks.

Python's module support is profound and extensive. Modules can do powerful things, like manipulate image or sound files, munge and process huge blocks of data, do statistical modeling and visualization (charts) and much, much, much more.

CSV

The CSV module parses CSV files, splitting the lines for us. We read the CSV object in the same way we would a file object.

```
import csv
fh = open('../python_data/students.txt', 'r')  # second argument: default
reader = csv.reader(fh)

for record in reader:     # loop through each row

    print('id:{};  fname:{}; lname: {}'.format(record[0], record[1], recor
```

This module takes into account more advanced CSV formatting, such as quotation marks (which are used to allow commas within data.)

The second argument to **open()** ('rB') is sometimes necessary when the csv file comes from Excel, which output newlines in the Windows format (**\r\n**), and can confuse the **csv** reader.

Writing is similarly easy:

```
import csv
wfh = open('some.csv', 'w')
writer = csv.writer(wfh)
writer.writerow(['some', 'values', "boy, don't you like long field values?
writer.writerows([['a', 'b', 'c'], ['d', 'e', 'f'], ['g', 'h', 'i']])
wfh.close()
```