Comma Separated Value

CSV can be used to store the data contained in a spread sheet. The following example file contains pressure gauge calibration data. In addition to commas, CSV files also use quotes (") to delimit strings. Quotes are escaped by using two quotes together.

g5000.csv

```
"Gauge's calibration, for 5000 psi",
"weight(psi)","gauge(psi)"
200,200
400,400
600,600
800,800
1000,990
1200,1185
1400,1390
1500,1490
2000,1995
2500,2485
3000,2995
3500,3500
4000,3995
4500,4490
4900,4900
```

Comma Separated Value Reader

The python csv module can read and write CSV formatted files. The gauge data can be read and a formatted report generated with:

read_gauge.py

```python
import csv

# read the calibration data
# first row contains title
# second row contains headings
with open( "g5000.csv" ) as f:
    # return an interator
    reader = csv.reader( f )
    # get first row and first column
    title = next(reader)[0]
    # get next row
    header = next(reader)
    weight = []
    gauge = []
    # get the rest
    for row in reader :
        weight.append( float(row[0]) )
        gauge.append( float(row[1]) )

# generate a report
print(title.center(24))
print("  %12s%12s" % tuple( header )) # two spaces for formatting
for w,g in zip(weight, gauge) :
    print("%12.1f%12.1f" % (w,g))
```

Its execution produces:

Command:

```
python3 read_gauge.py
```

Standard output:

```
Gauge's calibration, for 5000 psi
   weight(psi)  gauge(psi)
       200.0       200.0
       400.0       400.0
       600.0       600.0
       800.0       800.0
      1000.0       990.0
      1200.0      1185.0
      1400.0      1390.0
      1500.0      1490.0
      2000.0      1995.0
      2500.0      2485.0
      3000.0      2995.0
      3500.0      3500.0
      4000.0      3995.0
      4500.0      4490.0
      4900.0      4900.0
```

Comma Separated Value Writer

CSV can be written with:

```
>>> import csv
>>> import sys
>>> writer = csv.writer(sys.stdout)
>>> # notice the double " for embedded (escaping) quotes
>>> writer.writerow( ("'first'", 'second,"2nd"', "third") )
'first',"second,""2nd""",third
>>> t = [ ("a column","b column"), (3, 3.14), (5,8,9) ]
>>> writer.writerows( t )
a column,b column
3,3.14
5,8,9
>>> # the comma is hidden in side quotes
>>> # a row with only two columns can be written
>>> writer.writerow( ('hello, world', 3.14) )
"hello, world",3.14
```

Can `split` be used to parse CSV files?

Weather data in CSV

Environment Canada publishes weather data in CSV format on their web site. A sample of the data for October 2007 is:

oct-20.csv

```
"Station Name","ST JOHN'S A"
"Province","NEWFOUNDLAND"
"Latitude","47.62"
"Longitude","-52.74"
"Elevation","140.50"
"Climate Identifier","8403506"
"WMO Identifier","71801"
"TC Identifier","YYT"

"All times are specified in Local Standard Time (LST). Add 1 hour to adjust for Daylight Saving Time where and when it is observed."

"Legend"
"M","Missing"
"E","Estimated"
"NA","Not Available"

"Date/Time","Year","Month","Day","Time","Temp (C)","Temp Flag","Dew Point Temp (C)","Dew Point Temp Flag","Rel Hum (%)","Rel Hum Flag","Wind Dir (10's deg)","Wind Dir Flag","Wind Spd (km/h)","Wind Spd Flag","Visibility (km)","Visibility Flag","Stn Press (kPa)","Stn Press Flag","Hmdx","Hmdx Flag","Wind Chill","Wind Chill Flag","Weather"
"2007-10-1 0:00","2007","10","1","0:00","","","","","","","","","","","","","","","","","","",""
"2007-10-1 0:30","2007","10","1","0:30","2.10","","0.60","","90.00","","28.00","","13.00","","24.10","","101.22","","","","","","Clear"
"2007-10-1 1:00","2007","10","1","1:00","","","","","","","","","","","","","","","","","","",""
```

The first 16 lines identify the location and provide a legend. The 17th line gives the headers for all the data columns. The remaining lines are the data values for the values described on the 17th line.

Removing columns in original csv file

The `csv` module can be used to read the original csv file, and then write out a reduced csv file with only some of the data columns.

reduce.py

```python
import csv
import datetime

def extract_time( row ) :
    y = int(row[1])
    mn = int(row[2])
    d = int(row[3] )
    h,m = row[4].split(':')
    h = int( h )
    m = int( m )
    return datetime.datetime(y, mn, d, h, m )

def extract_temp( row ) :
    t = row[5]
    try :
        return float( t )
    except :
        return None

def extract_rel_humidity( row ) :
    rh = row[9]
    try :
        return float( rh )
    except :
        return None

def extract_wind_dir( row ) :
    wd = row[11]
    try :
        # adjust to degs instead of 10s of degs
        return float( wd ) * 10.0
    except :
        return None

def extract_wind_speed( row ) :
    sp = row[13]
    try :
        return float( sp )
    except :
        return None

def extract_pressure( row ) :
    p = row[17]
    try :
        return float( p )
    except :
        return None

def extract_description( row ) :
    return row[23]

f = open( "oct.csv" )
reader = csv.reader( f )
name = next(reader)[0]  # get the first column
prov = next(reader)[0]
latitude = next(reader)[0]
longitude = next(reader)[0]
elevation = next(reader)[0]

# skip to line 16
# the line number of the input file is maintained in line_num
while reader.line_num < 16 :
    next(reader)

headers = next(reader)
# print headers to ensure skip worked
print(headers)

fout = open("reduced.csv", "w")
writer = csv.writer( fout )

writer.writerow( ('time', 'temp', 'humidity', 'wind dir',
                'wind speed', 'pressure',  'desc' ) )

# skip bad data rows
for row in reader :
    t = extract_time( row )
    if not t : continue
    temp = extract_temp( row )
    if not temp : continue
    h = extract_rel_humidity( row )
    if not h : continue
    wd = extract_wind_dir( row )
    if not wd : continue
    ws = extract_wind_speed( row )
    if not ws : continue
    press = extract_pressure( row )
    if not press : continue
    description = extract_description( row )
    if not description : continue
    writer.writerow( ( t, temp, h, wd, ws, press, description) )

fout.close()
f.close()
```

Reducing data in original csv file

A sample of the reduced data is:

reduced.csv

```
time,temp,humidity,wind dir,wind speed,pressure,desc
2007-10-01 00:30:00,2.1,90.0,280.0,13.0,101.22,Clear
2007-10-01 01:30:00,2.7,91.0,280.0,17.0,101.25,Mainly Clear
2007-10-01 02:30:00,3.3,91.0,280.0,13.0,101.27,Mainly Clear
2007-10-01 03:30:00,2.3,92.0,290.0,15.0,101.31,Mostly Cloudy
```

The time is output in YYYY-MM-DD HH:MM:SS. The wind direction is in degrees (not 10s of degrees).

CSV, numpy, statistics

Statistics of the weather data can be calculated with the `numpy` module. Once parsed with `csv`, the data list can be converted into arrays.

stats_report_np.py

```
import numpy as np
import csv

with open( "reduced.csv" ) as f:
    reader = csv.reader( f )
    #skip header
    header = next(reader)
    # get all the rows
    rows = [r for r in reader]

# create numpy arrays
n = len( rows )
temp = np.zeros( n, float)
humidity = np.zeros( n, float)
wd = np.zeros( n, float)
ws = np.zeros( n, float)
pressure = np.zeros( n, float)

# fill in the elements
# this approach avoids unecessary memory allocation
for i,r in enumerate(rows) :
    temp[i] = float(r[1])
    humidity[i] = float(r[2])
    wd[i] = float(r[3])
    ws[i] = float(r[4])
    pressure[i] = float( r[5] )

# print out statistics
print('temp:', temp.mean(), temp.min(), temp.max(), temp.std())
print('humidity:', humidity.mean(), humidity.min(), humidity.max(), humidity.std() )
print('wind dir:', wd.mean(), wd.min(), wd.max(), wd.std())
print('wind speed:', ws.mean(), ws.min(), ws.max(), ws.std())
print('pressure:', pressure.mean(), pressure.min(), pressure.max(), pressure.std())
```

Command:

```
python3 stats_report_np.py
```

Standard output:

```
temp: 7.38106591865 -0.5 19.0 4.18724008256
humidity: 82.1893408135 46.0 98.0 11.5549446008
wind dir: 235.203366059 10.0 360.0 91.9432985211
wind speed: 20.8934081346 4.0 46.0 10.0117672013
pressure: 99.4569845722 96.96 101.59 1.05180591165
```

CSV, standard statistics

The same calculations with standard python is:

stats_report_std.py

```
import numpy as np
import statistics as st
import csv

with open( "reduced.csv" ) as f:
    reader = csv.reader( f )
    #skip header
    header = next(reader)
    # get all the rows
    rows = [r for r in reader]

# create lists
n = len( rows )
temp = [0.0] * n
humidity = [0.0] * n
wd = [0.0] * n
ws = [0.0] * n
pressure = [0.0] * n

# fill in the elements
# this approach avoids unecessary memory allocation
for i,r in enumerate(rows) :
    temp[i] = float(r[1])
    humidity[i] = float(r[2])
    wd[i] = float(r[3])
    ws[i] = float(r[4])
    pressure[i] = float( r[5] )

# print out statistics
print('temp:', st.mean(temp), min(temp), max(temp), st.pstdev(temp))
print('humidity:', st.mean(humidity), min(humidity), max(humidity), st.pstdev(humidity))
print('wind dir:', st.mean(wd), min(wd), max(wd), st.pstdev(wd))
print('wind speed:', st.mean(ws), min(ws), max(ws), st.pstdev(ws))
print('pressure:', st.mean(pressure), min(pressure), max(pressure), st.pstdev(pressure))
```

Command:

```
python3 stats_report_std.py
```

Standard output:

```
temp: 7.381065918653577 -0.5 19.0 4.187240082557808
humidity: 82.18934081346424 46.0 98.0 11.554944600765657
wind dir: 235.20336605890603 10.0 360.0 91.94329852113935
wind speed: 20.893408134642357 4.0 46.0 10.011767201349379
pressure: 99.45698457223001 96.96 101.59 1.051805911648539
```

numpy and statistics

The same report can be generated with only numpy.

stats_report1.py

```
import numpy

# simple csv files can be read with numpy.loadtxt
# skip the date and description columns with usecols=()
# skip the first row with skiprows=1
# unpack=True transposes the table, places the columns
# in their own vector
temp,humidity,wd,ws,pressure = numpy.loadtxt("reduced.csv",
    delimiter=",", usecols=(1,2,3,4,5), unpack=True, skiprows=1)

def report( name, arr ) :
    print("%-12s" % name, end=' ')
    print("%6.2f" % arr.mean(), end=' ')
    print("%6.2f" % arr.min(), end=' ')
    print("%6.2f" % arr.max(), end=' ')
    print("%6.2f" % arr.std() )

# print out statistics
report( 'temp', temp )
report( 'humidity', humidity )
report( 'wind dir', wd )
report( 'wind speed', ws )
report( 'pressure', pressure )
```

Command:

```
python3 stats_report1.py
```

Standard output:

```
temp           7.38  -0.50  19.00   4.19
humidity      82.19  46.00  98.00  11.55
wind dir     235.20  10.00 360.00  91.94
wind speed    20.89   4.00  46.00  10.01
pressure      99.46  96.96 101.59   1.05
```

numpy and statistics and headings

The headings in the csv file can be used to label the data.

stats_report2.py

```
import numpy

with open( "reduced.csv" ) as f :
    # are there any problems with not using the csv module
    headings = f.readline().split(',')
headings = headings[1:-1] # skip first and last

data = numpy.loadtxt("reduced.csv",
    delimiter=",", usecols=(1,2,3,4,5), unpack=True, skiprows=1)

def report( name, arr ) :
    print("%-12s" % name, end=' ')
    print("%6.2f" % arr.mean(), end=' ')
    print("%6.2f" % arr.min(), end=' ')
    print("%6.2f" % arr.max(), end=' ')
    print("%6.2f" % arr.std() )

# print out statistics
for i,h in enumerate( headings ):
    report( h, data[i] )
```

Command:

```
python3 stats_report2.py
```

Standard output:

```
temp           7.38  -0.50  19.00   4.19
humidity      82.19  46.00  98.00  11.55
wind dir     235.20  10.00 360.00  91.94
wind speed    20.89   4.00  46.00  10.01
pressure      99.46  96.96 101.59   1.05
```

Is `stats_report2.py` better than `stats_report1.py`? Why?

Plotting October's Weather

Only slight changes are necessary to `stats_report2.py` to plot the weather data.

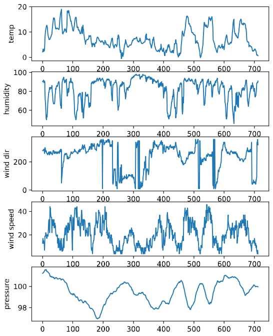
```
import matplotlib.pyplot as plt
import numpy as np
import csv

# user csv to extract the headings
with open( "reduced.csv" ) as f:
    reader = csv.reader( f )
    headings = next(reader)
headings = headings[1:-1] # skip first and last

# simple csv files can be read with np.loadtxt
# skip the date and description columns
data = np.loadtxt("reduced.csv",
    delimiter=",", usecols=(1,2,3,4,5), unpack=True, skiprows=1)

plt.figure(1, figsize=(6.0,8.0), dpi=100)
# print out statistics
for i,h in enumerate( headings ):
    plt.subplot(len(data), 1, i+1)
    plt.plot( data[i] )
    plt.ylabel( h, rotation="90" )

plt.savefig('october.svg')
```



Plotting Histograms of October's Weather

Only slight changes are necessary to `stats_plot.py` to plot the histograms of the weather data.

hist_plot.py

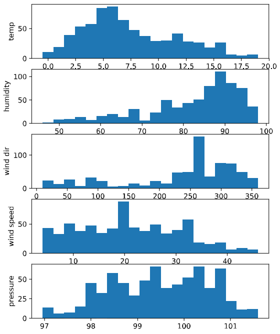
```
import matplotlib.pyplot as plt
import numpy as np
import csv

# user csv to extract the headings
with open( "reduced.csv" ) as f:
    reader = csv.reader( f )
    headings = next(reader)
headings = headings[1:-1] # skip first and last

# simple csv files can be read with np.loadtxt
# skip the date and description columns
data = np.loadtxt("reduced.csv",
    delimiter=",", usecols=(1,2,3,4,5), unpack=True, skiprows=1)

plt.figure(1, figsize=(6.0,8.0), dpi=100)
# print out statistics
for i,h in enumerate( headings ):
    plt.subplot(len(data), 1, i+1)
    plt.hist( data[i], 20 )
    plt.ylabel( h, rotation="90" )

plt.savefig('hist_october.svg')
```


temp
humidity
wind dir
wind speed
pressure