MA305 – Classwork #5
Reading (from a file) and Printing Lists

In this classwork you will learn how to use Python functions `f.readline()` and `f.readlines()` to read data from a file, do some calculations, and print the results neatly using `format` statement.

1. First, let us read a "self-terminating" input file. Create a directory `~/MA305/CW5`, and in this directory create a data file (`dat5.txt`) containing the following six lines.

```
  date    index   x         y
20050212    8   15.30     5.40
20060212   12   14.50     9.52
20070212    8   16.30     6.40
20080212   12   13.50     8.52
20090212   12   12.50    -7.52
0    0     0.00     0.00     : date=0 to terminate input
```

Such a file is a typical example of how measurements are recorded. Each line represents a measurement of two quantities x and y, and we may be interested in their averages, say. The first value of our data is an integer (date), recording the date of the measurement. The second integer (index) may encode some other information. The next two (reals) are the measured values of x and y, which we view as components of two arrays x_i and y_i, whose averages we want to find.

2. The Python code `cw5.py` posted in your course Canvas, reads this file, stores the values in appropriate variables, prints out how many lines it read from the data file, then prints the x_i values in one column, and the y_i values in a second column neatly lined up with appropriate labels, and finally computes (and print out!) the average value for the x and y.

$$\bar{x} = \frac{1}{n}\sum_{i=0}^{n-1} x_i, \qquad \bar{y} = \frac{1}{n}\sum_{i=0}^{n-1} y_i$$

a. First open a data-file to read data from.

```
f = open('dat5.txt','r')
```

b. The first line of the `dat5.txt` file is just a label, but it still must be read! A plain `readline()` statement will do it. Read the line as a string (`line`) and print it.

```
line=f.readline()
print(line,end='')  # end='', forces not to print an extra blank line
```

c. To read the remaining lines, use a `while` loop. The last line in `dat5.txt` has 0's, as signal to terminate further input. So the value of 'date' should be checked after each `readline()`. For this, you need to `split` the string `line` read into a list `Line`, and `break` the `while` loop if `int(Line[0])==0`. Once you are done reading the date file, close it.

```python
while True:
    line = f.readline()          # line is a string
    Line = line.split()          # Line is a list
    if int(Line[0])==0:
        break
    print(Line)
f.close()
```

d. To print out all the x_i values in one column, and all the y_i values in another, side-by-side, neatly lined up, place the following statement inside the `while` loop.

```python
print('\t {0:-5.2f} \t {1:-5.2f}'.format(float(Line[2]),float(Line[3])))
```

e. Use a counter `n +=1` inside the `while` loop to get the number of lines read and print it after the loop is terminated.

```python
print(n,"lines from 'dat5.txt' are read for calculations!")
```

f. To calculate average values of x and y, store `float(Line[2])` and `float(Line[3])` values in the lists `x` and `y`. For this, define two empty lists `x` and `y` before the `while` loop.

```python
x=[]; y=[]
```

and `append` the values to the lists `x` and `y`, inside the `while` loop, as they are read.

```python
x.append(float(Line[2]))
y.append(float(Line[3]))
```

Note that 'date' (`int(Line[0])`) and 'index' (`int(Line[1])`) have no bearing on the averages, so we don't need to keep their values.

g. The functions `calc_average(x)` and `calc_max(x)` defined at the beginning of the program (before the main) are called in the main to calculate the average and maximum values of `x` and `y`.

```python
print('==============================')
# Compute and print the average of x and y
xave=calc_average(x)
yave=calc_average(y)
print('Average: {0:-5.2f} \t {1:-5.2f}'.format(xave,yave))

# Evaluate the maximum values in x and y
xmax,xi = calc_max(x)
ymax,yi = calc_max(y)
#print('x[', xi, ']=',xmax)
#print('y[', yi, ']=',ymax)
print('Max: x[{0:1d}]={1:0.2f}    y[{2:1d}]={3:0.2f}'.format(xi,xmax,yi,ymax))
print()
```

DO NOT exit yet... try out these !!! (P.T.O.)

4. Now, you will explore another method of reading the data file. You can also read the whole file with a single command `readlines()`. Do the following steps:

a. Copy the data file `dat5.txt` to `dat5a.txt` and delete the last line of the file `dat5a.txt`. That is, we will work on the following data file.

```
  date    index   X        Y
20050212   8    15.30    5.40
20060212  12    14.50    9.52
20070212   8    16.30    6.40
20080212  12    13.50    8.52
20090212  12    12.50   -7.52
```

b. Open the code `cw5a.py` and see (lines 36-40) how it reads the data from the file `dat5a.txt`.

```python
datafile=input('Enter the name of the file to read data from:')
f=open(datafile,'r')

f.readline()
data=f.readlines()
print(data)

for line in data:
    Line = line.split()
    print(Line)
f.close()
```

c. Run the code `cw5a.py`. You need to enter the name of the data-file (`dat5a.txt`) from the keyborad. You should get the same results as in your `cw5.py` code.

5. Adapt your code `cw5a.py` to compute and print minimum values of x and y.

```python
def calc_min(x):
    """ calculates maximum value and its location in the list x """
    n=len(x)
    minval=x[0]
    minloc=0
    for i in range(1,n):
        if x[i] < minval:
            minval=x[i]
            minloc=i
    return minval, minloc
```

6. Now try **I/O Redirection.** Import the `sys` module (`import sys`) at the beginning, comment the lines 32-33 and uncomment line 34 and run the code by typing the following commands.

```python
    #datafile=input('Enter the name of the file to read data from:')
    #f=open(datafile,'r')
    f = sys.stdin
```

```
$ chmod u+x cw5a.py
$ ./cw5a.py < dat5a.txt > out.txt
```

---

7. Submit your modified code `cw5a.py` and the output file `out.txt` through Canvas. No typescript file, no Email !