Lab 3 Tasks

Task 1.

Download the comma-separated file *scores.csv* from the module Moodle page and save it to the same directory as your notebooks.

Write a function *display_numbers()* which takes one parameter - a file path. The function should read floating point numbers from each line in the specified file, and compute the total for the values on each line. Print each total to 2 decimal places. Use exception handling to deal with the potential case where the input file does not exist.

Apply the function *display_numbers()* to *scores.csv*.

In [45]:
```
#Setting global variable for practice
#How to get 2 decimal places: "{:1.2f}".format(x)
total = 0;

#Function which takes in csv file path and sums up each row
def getTotal(file_path):

    try:
        f = open(file_path, "r")
        lines = f.readlines();
        f.close();
    except IOError:
         print("Unable to read from file", file_path);
    #file path is correct
    lineCounter = 1;
    for line in lines:
        #Using global variable in this scope
        global total;
        line = line.strip();
        #line is the whole line
        #parts is a list of strings, need to convert each
        parts = line.split(",");
        parts = (list(map(float, parts)));

        total += sum(parts);
        response = '''\
        This is a {lineCounter}
        Here is a {total}
        '''.format(lineCounter=lineCounter, total="{:1.2f}".format(total));
        print(response)
        lineCounter += 1;
        total = 0;
        #end of for loop
    return total;


print(getTotal("scores.csv"));
```

```
        This is a 1
        Here is a 2.84

        This is a 2
        Here is a 3.57

        This is a 3
        Here is a 1.57

        This is a 4
        Here is a 2.41

        This is a 5
        Here is a 2.47

        This is a 6
        Here is a 3.02

0
```

Code for multi-line String with interpolated variables

In [22]:
```
s = '''\
... This is a {length} example.
... Here is a {ordinal} line.\
... '''.format(length='multi-line', ordinal='second')
>>> print(s)
```

```
This is a multi-line example.
Here is a second line.
```

Task 2.

Write a function *reverse_numbers()* which takes two parameters - an input file path and an output file path. The function should read floating point numbers from each line in the specified input file. The order of the values from in line should then be reversed, and these lines should be written to the specifed output file. Include exception handling code.

Apply the function *reverse_numbers()* to *scores.csv* to create a new file *reversed.csv*.

In [84]:
```
def reverse_numbers(input_file_path, output_file_path):
    try:
        fin = open(input_file_path,"r")
        fout = open(output_file_path,"w")
        lineCounter = 1
        for line in fin.readlines():
            fout.write("Line Number ")
            fout.write(str(lineCounter))
            fout.write("\n")
            parts = line.split(",")
            parts = (list(map(float, parts)))
            #Returns None, but acts on the original list parts
            parts.reverse()
            #parts is now in reverse order
            fout.write(str(parts))
            fout.write("\n")
            lineCounter += 1
        #Close connections
        fin.close()
        fout.close()
    except IOError:
         print("Unable to read from files given", input_file_path, " ", output_file_path)
#end of function
reverse_numbers("scores.csv", "reverse_numbers.csv");
```

```
Line Number 1
[0.89, 0.58, 0.63, 0.74]
Line Number 2
[0.99, 0.78, 0.89, 0.91]
Line Number 3
[0.45, 0.34, 0.35, 0.43]
Line Number 4
[0.58, 0.66, 0.61, 0.56]
Line Number 5
[0.72, 0.76, 0.49, 0.5]
Line Number 6
[0.78, 0.61, 0.75, 0.88]

```

```
0.74,0.63,0.58,0.89
0.91,0.89,0.78,0.99
0.43,0.35,0.34,0.45
0.56,0.61,0.66,0.58
0.50,0.49,0.76,0.72
0.88,0.75,0.61,0.78
```