10.11. Writing data to a CSV File

The typical pattern for writing data to a CSV file will be to write a header row and loop through the items in a list, outputting one row for each. Here we a have a list of tuples, each representing one Olympian, a subset of the rows and columns from the file we have been reading from.

Save & Run Original - 1 of 1

```
olympians = [("John Aalberg", 31, "Cross Country Skiing"),
             ("Minna Maarit Aalto", 30, "Sailing"),
             ("Win Valdemar Aaltonen", 54, "Art Competitions"),
             ("Wakako Abe", 18, "Cycling")]

outfile = open("reduced_olympics.csv", "w")
# output the header row
outfile.write('Name,Age,Sport')
outfile.write('\n')
# output each of the rows:
for olympian in olympians:
    row_string = '{},{},{}'.format(olympian[0], olympian[1], olympian[2])
    outfile.write(row_string)
    outfile.write('\n')
outfile.close()
```

Activity: 1 -- ActiveCode (ac9_14_1)

Data file: `reduced_olympics.csv`

```
Name,Age,Sport
John Aalberg,31,Cross Country Skiing
Minna Maarit Aalto,30,Sailing
Win Valdemar Aaltonen,54,Art Competitions
Wakako Abe,18,Cycling
```

There are a few things worth noting in the code above.

First, using .format() makes it really clear what we're doing when we create the variable row_string. We are making a comma separated set of values; the {} curly braces indicated where to substitute in the actual values. The equivalent string concatenation would be very hard to read. An alternative, also clear way to do it would be with the .join method: `row_string = ','.join(olympian[0], olympian[1], olympian[2])` .

Second, unlike the print statement, remember that the .write() method on a file object does not automatically insert a newline. Instead, we have to explicitly add the character `\n` at the end of each line.

Third, we have to explicitly refer to each of the elements of olympian when building the string to write. Note that just putting `.format(olympian)` wouldn't work because the interpreter would see only one value (a tuple) when it was expecting three values to try to substitute into the string template. Later in the book we will see that python provides an advanced technique for automatically unpacking the three values from the tuple, with `.format(*olympian)` .

As described previously, if one or more columns contain text, and that text could contain commas, we need to do something to distinguish a comma in the text from a comma that is separating different values (cells in the table). If we want to enclose each value in double quotes, it can start to get a little tricky, because we will need to have the double quote character inside the string output. But it is doable. Indeed, one reason Python allows strings to be delimited with either single quotes or double quotes is so that one can be used to delimit the string and the other can be a character in the string. If you get to the point where you need to quote all of the values, we recommend learning to use python's csv module.

Save & Run Original - 1 of 1 Show in CodeLens

```
olympians = [("John Aalberg", 31, "Cross Country Skiing, 15KM"),
             ("Minna Maarit Aalto", 30, "Sailing"),
             ("Win Valdemar Aaltonen", 54, "Art Competitions"),
             ("Wakako Abe", 18, "Cycling")]

outfile = open("reduced_olympics2.csv", "w")
# output the header row
outfile.write('"Name","Age","Sport"')
outfile.write('\n')
# output each of the rows:
for olympian in olympians:
    row_string = '"{}", "{}", "{}"'.format(olympian[0], olympian[1], olympian[2])
    outfile.write(row_string)
    outfile.write('\n')
outfile.close()
```

Activity: 2 -- ActiveCode (ac9_14_2)

Data file: `reduced_olympics2.csv`

```
"Name","Age","Sport"
"John Aalberg", "31", "Cross Country Skiing, 15KM"
"Minna Maarit Aalto", "30", "Sailing"
"Win Valdemar Aaltonen", "54", "Art Competitions"
```

"Wakako Abe", "18", "Cycling"

You have attempted 3 of 2 activities on this page

10.10. Reading in data from a CSV File">
10.12. Tips on Handling Files">
eading in data from a CSV File">
✔ Completed. Well Done!
10.12. Tips on Handling Files">Next Section - 10.12. Tips on Handling Files

| Back to top