CSV

CSV **(comma-separated values):** files contain rows of data, with each value in a row separated by a comma

All of the data for a single record is on one line: each new line is a new record

The comma, in this case, is called the **'delimiter'** as it shows the difference (or limit) between one value and the next

Other common delimiters are semi-colons and tabs (also called **tsv/tab-separated values**)

We must be careful to check what exactly the delimiter is, as a common error is reading in a file with the wrong delimiter, and so getting a weird representation in your data

Usually, if you are using data from mainland European countries (France/Spain etc) they will use semi-colons, hence some people prefer *character*-separated values for CSV

Open CSV Files

```
import csv
with open('Salaries.csv', mode= 'r', newline='') as csvfile:
  reader = csv.reader(csvfile, delimiter=',')
  for n, row in enumerate(reader):
    print(','.join(row))
    if n == 5: # Read the first 5 entries
      break
```

Python counts with a library called `csv` that has the needed functionalities to read and write CSV files.

We open an existing file `Salaries.csv` using a context manager, and the mode in the context manager is set to read `r`. Then, use the reader class from CSV, which will take the values in the CSV and store them into a variable that becomes an iterable.

Create CSV Files

```
my_list = [['Sparky', 7, 'Brown', 'Corgi'], ['Fido', 4, 'White', 'Husky']]

# Create a new file with each row as the characteristic of each dog
import csv
```

The same library can be used to generate CSV files. The only thing you need to change is the mode argument in the context manager is write (`w`). If you

```
with open('Dogs.csv', 'w', newline='') as f:
  writer = csv.writer(f)
  writer.writerow(['Name', 'Age', 'Colour', 'Breed'])
  writer.writerows(my_list)
```

want to append things to the CSV, you can use the mode append `a`

Instead of reading the file, we write the file. If the file already exists it will overwrite it

The `writer` object has methods to create new files. Most common is `writerows` which accepts iterables and parses them into a comma-separated row

Notice the difference between `writerow` and `writerows`

Pandas

```
from xmlrpc.server import XMLRPCDocGenerator
df = pd.read_csv('data/file.csv')
df.head()
df.to_csv('data/modified.csv')
```