# CSV Module

## GETTING STARTED

```
import csv
with open('name.csv', 'r') as csv_file:
    csv_reader = csv.reader(csv_file)

    for line in csv_reader:
        print(line)
```

- With open is a context manager. 'Names.csv' is the name of the file that we want to parse. The r signigifes that we want to read this file as 'csv_file'.
- Csv_reader = csv.reader(csv_file) : the .reader function within the  csv_reader variable allows us to actually read the csv_file that we have opened.
- The for loop, loops through all the lines in the csv_reader and prints out each line.

```
        print(line[2])
```

- This print function would print out all values that correspond with 2 index column.

```
    next(csv_reader)
    for line in csv_reader:
        print(line)
```

- The next(csv_reader) allows us to step over the first value.

WRITING IN A NEW DELIMITER

```
import csv

with open(‘name.csv’, ‘r’) as csv_file:

    csv_reader = csv.reader(csv_file)


    with open(‘new_names.csv’, ‘w’) as new file:

        csv_writer = csv.writer(new_file, delimiter = ‘-’)


        for line in csv_reader:

            csv_writer.writerow(line)
```

- with open(‘new_names.csv’, ‘w’) as new file:  This opens a new file for writing, ‘w’ tells us that we want to write. It is called on as new file
- Csv.writer method writes in the new_file, and changes the delimiter as a dash
- By indenting the for loop it is now in the context of the new_file.
- The for loop takes the lines within the original csv data and our csv_reader, and writes in the new lines to our csv_writer.
- This creates a new file - new_names.csv that uses dashes instead of a comma.
- If we wanted to change the delimiter to a tab - ‘\t’

USING DICT READER- calling on values

```
import csv
with open('name.csv', 'r') as csv_file:
    csv_reader = csv.DictReader(csv_file)


    for line in csv_reader:
        print(line['email'])
```

- Within the csv_reader variable, csv.DictReader changes the way the lines are printed out
- When printed each value correlates to a dictionary. I.E.  ('first_name' , 'John') ('last_name', 'Doe')
- This makes it a lot easier to parse out the information that we want.
- print(line['email']) with the DictReader we are able to call on data that correlates with the key email.
-

HOW TO USE DICTWRITER

```
import csv

with open(‘name.csv’, ‘r’) as csv_file:

        csv_reader = DictReader(csv_file)


        with open(‘new_names.csv’, ‘w’) as new file:

                fieldnames = [‘first_name’, ‘last_name’ , ‘email’]


                Csv_writer = csv.DictWriter(new_file, fieldnames = fieldnames

        delimiter = ‘\t’)

                csv_writer.writeheader()

                for line in csv_reader:

                        csv_writer.writerow(line)
```

- Fieldnames are input because that is what we want to write using a dictwriter.
- Fieldnames must be passed in within the csv_writer variable.
- In order to keep the header, or field names as the first line we must input csv_writer.writeheader()
- If we wanted to delete a key, we would remove it from the field names, and input **del line[‘email’]** under for line in csv_reader: