Lecture 18: Writing CSV files

```
In [1]: # So far we have only read CSV files, let's start writing CSV files to store and share data
        import csv

In [49]: # Let's first define a simple helper function to inspect the content of a file
         #
         def display_file_content(file_path):
             try:
                 f_check = open(file_path, mode='r')
             except:
                 print('File', file_path, "doesn't exist!")
             else:
                 for r in f_check:
                     print(r, end='')
             finally:
                 f_check.close()

In [50]: # Let's start by using the plain csv syntax
         # We want to create a file with a few records organized as a comma-separated csv
         #
         # At this aim, we need to open a new csv file for writing and define field names, delimiter,
         # if we want to write fields that include the delimiter character, we have to specify
         # how to quote fields that include the delimiter, and what is the quoting policy
         #
         file_path = 'csv/univ.csv'
         f_csv = open(file_path, mode='w+')

         csv_writer = csv.writer(f_csv, delimiter=',', quotechar='"', quoting=csv.QUOTE_MINIMAL)

         print(type(csv_writer))
         # csv_writer is a writer object handle for csv files
```

```
<class '_csv.writer'>
```

```
In [44]: # The quotechar optional parameter tells the writer which character to use to
         # quote fields when writing. Whether quoting is used or not, however, is determined
         # by the quoting optional parameter:
         #
         # If quoting is set to csv.QUOTE_MINIMAL, then .writerow() will quote fields
         #    only if they contain the delimiter or the quotechar. This is the default case.
         # If quoting is set to csv.QUOTE_ALL,
         #   then .writerow() will quote all fields.
         # If quoting is set to csv.QUOTE_NONNUMERIC,
         #   then .writerow() will quote all fields containing text data and
         #   convert all numeric fields to the float data type.
         # If quoting is set to csv.QUOTE_NONE, then .writerow() will escape delimiters instead of
         #   quoting them. In this case, a value for the escapechar optional parameter must be defined.
         #
```

```
In [51]: # Let's throw out some data (no need to use quoting for now)
         #
         name = 'John Smith'
         dept = 'Eng'
         since = 2004
         score = 4.72/7

         # Which are the fields here? name, dept, since, score
         #
         # How do we write the fields into a record? writerow(what_we_want_to_write)

         csv_writer.writerow([name, dept, since, score])
```

```
Out[51]: 40
```

```
In [58]: # Files are "streams" (e.g., the standard output on the screen is a stream), and streams
         # might be "buffered": data written on the stream is not sent directly to the stream support,
         # instead, it is temporarily held in memory (a "buffer") until, at least, a certain amount
         # of data is written in the buffer. Whenthis happens, all the data in the memory buffer
         # is "flushed" to the stream and reaches the desired physical medium
         # (e.g., a file on a hard disk, or the screen)
         # Since here we are writing step by step small amount of data, and inspecting the file after
         # that and before closing the file, let's force the flushing of the file buffer. This ensures
         # that data will be written in the file following each writing operation.
         # Note the a close() operation automatically flushes the buffer.
         #
         f_csv.flush()
```

```
In [52]: # Let's check the file ...
         display_file_content(file_path)
```

```
John Smith,Eng,2004,0.6742857142857143
```

```
In [56]: # Let's add another data record
         name = 'Ann White'
         dept = 'CS'
         since = 2012
         score = 3.81/7
         score = float('{:4.2f}'.format(score))

         csv_writer.writerow([name, dept, since, score])
```

```
In [59]: # Let's check the file again, flushing the stream first
         #
         f_csv.flush()
         display_file_content(file_path)
```

```
John Smith,Eng,2004,0.6742857142857143
Ann White,CS,2012,0.54
Ann White,CS,2012,0.54
Ann White,CS,2012,0.54
```

```
In [29]: # What if we already have data in "tabular" format?, such as a list of lists?
         # do we have to write them row by row, or can we just dump the data with one instruction?
         # writerows(tabular_data) does do job for us!
         #
         univ_employees = [ [1, 'John Smith', 'Eng', 2004, 4.72/7], [2, 'Ann White','CS', 2012, 3.81/7]]
         csv_writer.writerows(univ_employees)
         f_csv.close()
```

```
In [30]: # Our file dosn't include an header that specifies what the fields are about.
         # It's good practice to add it (for sharing and clarity purposes).
         # Let's reopen the file (checking for its existence first) and let's add the header.
         #

In [67]: try:
             # We need to open the file for writing, preserving the contents, and be positioned at
             # the beginning: r+ mode
             f_csv = open(file_path, mode='r+')

         except:
             print('File ', file_path.split('/')[-1], 'does not exist!')
         else:
             csv_writer = csv.writer(f_csv, delimiter=',')
             #print(list(csv_data))  #if we'd like to see what's in the file so far

             header = ['name','dept','since','score']
             csv_writer.writerow(header)
         finally:
             f_csv.close()

In [68]: # Let's check the file:
         display_file_content(file_path)

         # Something wrong has happened there: the first record
         # [John Smith,Eng,2004,0.6742857142857143] has been overwritten!
         # We can't really 'squeeze' new data into existing data.
```

```
name,dept,since,score
742857142857143
Ann White,CS,2012,0.54
Ann White,CS,2012,0.54
Ann White,CS,2012,0.54
name,dept,since,score
name,dept,since,score
```

```
In [69]: # The header must be written first!
         # Or, we read the entire file, write the header, and then dump previous contents.
         # Let's do it!
         #
         try:
             f_csv = open(file_path, mode='r+')
         except:
             print('File ', file_path.split('/')[-1], "does not exist, we can write the header")
             header = ['name','dept','since','score']
             csv_writer = csv.writer(f_csv, delimiter=',')
             csv_writer.writerow(header)
         else:
             # File does exist. We need both a csv writer and csv reader object in this case
             csv_writer = csv.writer(f_csv, delimiter=',')
             csv_reader = csv.reader(f_csv)

             # Let's read and save all data first
             current_data = list(csv_reader)

             # After the previous read instruction, we are at the end of the file,
             # let's go to the very beginning to add the header
```

```
            f_csv.seek(0)

            # First, let's add the header
            header = ['name','dept','since','score']
            csv_writer.writerow(header)

            # Now, let's dump back all the data that was previously in the file
            csv_writer.writerows(current_data)
        finally:
            f_csv.close()
```

```
In [70]: # Let's check the file:
         display_file_content(file_path)
```

```
name,dept,since,score
name,dept,since,score
742857142857143
Ann White,CS,2012,0.54
Ann White,CS,2012,0.54
Ann White,CS,2012,0.54
name,dept,since,score
name,dept,since,score
```

```
In [71]: # Why did we use a list to write our data?
         # Couldn't we just use a single string: 'name,dept,since,score'?
         #
         f_csv = open(file_path, mode='w')
         csv_writer = csv.writer(f_csv, delimiter=',')
         header = 'name,dept,since,score'
         csv_writer.writerow(header)
         f_csv.close()

         display_file_content(file_path)
         #
         # ...what's going on there?? Individual characters get separated by commas,
         # and each field gets quotes using the standard quoting string "
         # Okay, let's not mess up further with these things ...
```

```
n,a,m,e,",",d,e,p,t,",",s,i,n,c,e,",",s,c,o,r,e
```

```
In [72]: # Since we can read csv data into a dictionary, it is expected that
         # we can write it out from a dictionary as well

         # Let's create a simple dictionary list out of our data. Keys must be strings
         #
         employees_list = []
         employees_list.append({'name': 'John Smith', 'dept': 'Eng', 'since': 2004, 'score': 4.72/7})
         employees_list.append({'name': 'Ann White', 'dept': 'CS', 'since': 2012, 'score': 3.81/7})
```

```
In [73]: f_csv = open(file_path, mode='w')

         # The csv file is treated as a dictionary, therefore in this case we must provide
         # the keys/fieldnames to the csv writer, which are passed as a list using the
         # fieldnames argument
         #
         field_names = ['name','dept','since','score']
         csv_writer = csv.DictWriter(f_csv, delimiter=',', fieldnames=field_names)
```

```
In [74]: # The first row must contain the keys, since this will be used for writing further data
         # the .writeheader() methods does the job for us
         #
         csv_writer.writeheader()

         for d in employees_list:
             csv_writer.writerow(d)

         f_csv.close()
```

```
In [75]: # Let's check the file:
         display_file_content(file_path)
```

```
name,dept,since,score
John Smith,Eng,2004,0.6742857142857143
Ann White,CS,2012,0.5442857142857143
```

```
In [77]: # Let's read the file using csv dictionary methods, just to be sure that everything went well
         # We need to get a dict reader object this time
         #
         f_csv = open(file_path, mode='r')
         csv_data = csv.DictReader(f_csv)
```

```
In [78]: # csv_data is already positioned after the header
         #
         keys = field_names
         for row in csv_data:
             print('{:<12s} is in department {:4s} since {:4d} and has score {:4.2f}'.format(
                     row[keys[0]], row[keys[1]], int(row[keys[2]]), float(row[keys[3]])))
         f_csv.close()
         # everything looks good!
```

```
John Smith   is in department Eng  since 2004 and has score 0.67
Ann White    is in department CS   since 2012 and has score 0.54
```