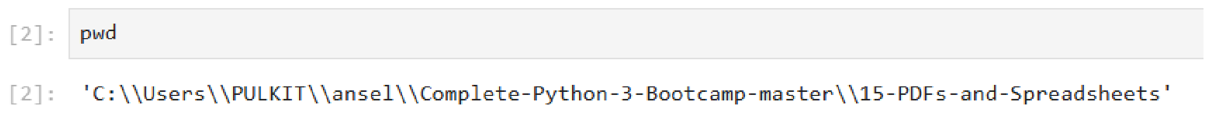
😊 Working with CSV Files

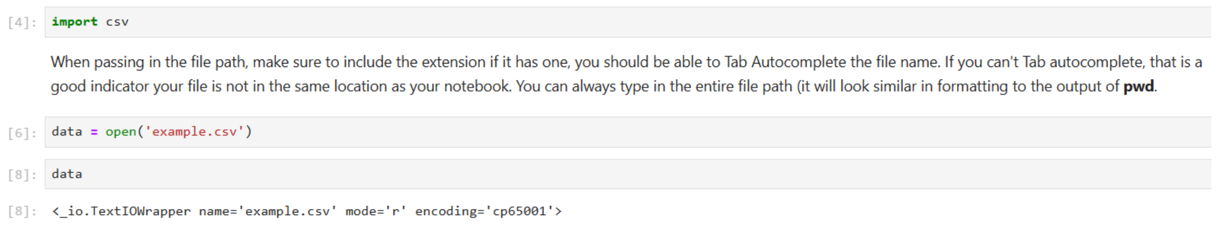
1. Now, in this lab, we are going to work with CSV files in Jupyter Notebook. With this lab, you are going to get some files to work with.



```
[2]: pwd

[2]: 'C:\\Users\\PULKIT\\ansel\\Complete-Python-3-Bootcamp-master\\15-PDFs-and-Spreadsheets'
```

2. The code opens a CSV file named **"example.csv"** using Python's built-in open() function and stores the file object in the variable data. However, at this point, the file is only opened, and no operations (like reading or parsing) have been performed on it yet. To process the CSV data, additional steps such as reading the file with the csv.reader() or csv.DictReader() functions would be needed.



```
[4]: import csv
```

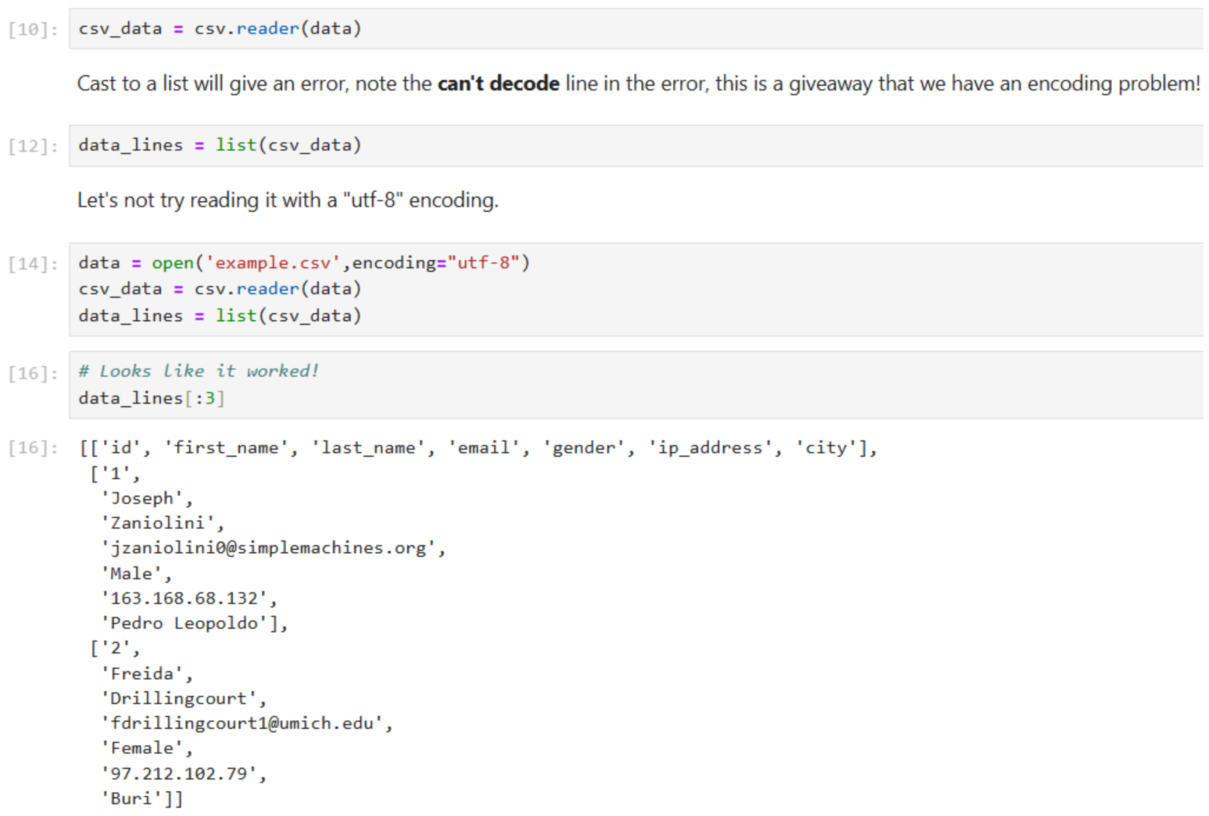
When passing in the file path, make sure to include the extension if it has one, you should be able to Tab Autocomplete the file name. If you can't Tab autocomplete, that is a good indicator your file is not in the same location as your notebook. You can always type in the entire file path (it will look similar in formatting to the output of **pwd**.

```
[6]: data = open('example.csv')

[8]: data

[8]: <_io.TextIOWrapper name='example.csv' mode='r' encoding='cp65001'>
```

3. Often, CSV files may contain characters that you can't interpret with standard Python; this could be something like an @ symbol, or even foreign characters.
4. The code reads a CSV file named **"example.csv"** using Python's csv.reader(). Initially, the file is opened and passed to csv.reader(data), which converts the raw file into an iterable object containing rows of data. The list(csv_data) function stores the rows as a list of lists (data_lines). The last line prints the first three rows to verify that the data has been successfully loaded. The second attempt explicitly sets encoding="utf-8" to handle special characters properly.



```
[10]: csv_data = csv.reader(data)
```

Cast to a list will give an error, note the **can't decode** line in the error, this is a giveaway that we have an encoding problem!

```
[12]: data_lines = list(csv_data)
```

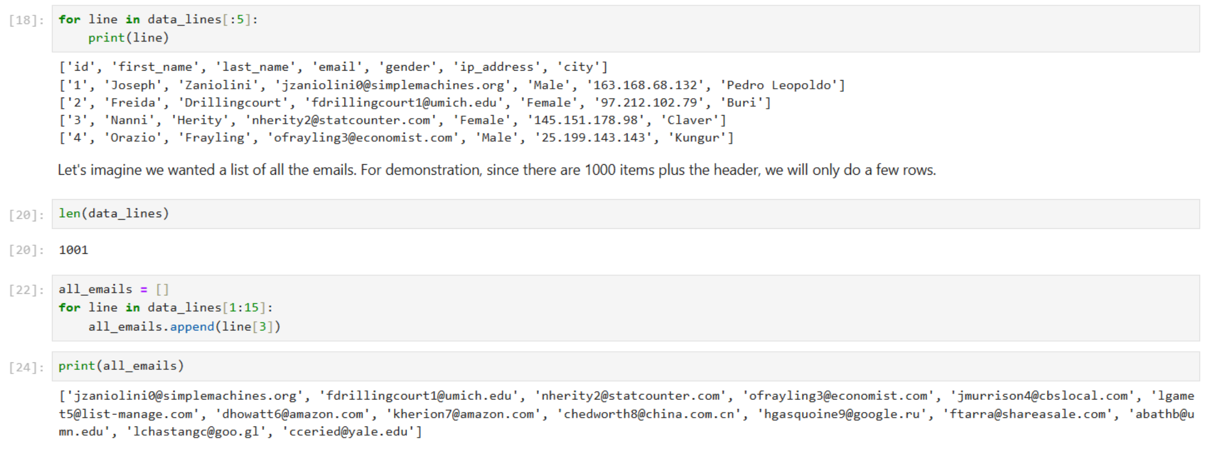
Let's not try reading it with a "utf-8" encoding.

```
[14]: data = open('example.csv',encoding="utf-8")
      csv_data = csv.reader(data)
      data_lines = list(csv_data)
```

```
[16]: # Looks like it worked!
      data_lines[:3]
```

```
[16]: [['id', 'first_name', 'last_name', 'email', 'gender', 'ip_address', 'city'],
       ['1',
        'Joseph',
        'Zaniolini',
        'jzaniolini0@simplemachines.org',
        'Male',
        '163.168.68.132',
        'Pedro Leopoldo'],
       ['2',
        'Freida',
        'Drillingcourt',
        'fdrillingcourt1@umich.edu',
        'Female',
        '97.212.102.79',
        'Buri']]
```

5. The code below first prints the first five rows of the CSV file to inspect its structure. Then, it calculates and prints the total number of rows in data_lines. Next, it extracts email addresses from rows 2 to 15 (excluding the header) by appending the value in the fourth column (line[3]) to the all_emails list. Finally, it prints the collected email addresses.



```
[18]: for line in data_lines[:5]:
          print(line)

      ['id', 'first_name', 'last_name', 'email', 'gender', 'ip_address', 'city']
      ['1', 'Joseph', 'Zaniolini', 'jzaniolini0@simplemachines.org', 'Male', '163.168.68.132', 'Pedro Leopoldo']
      ['2', 'Freida', 'Drillingcourt', 'fdrillingcourt1@umich.edu', 'Female', '97.212.102.79', 'Buri']
      ['3', 'Nanni', 'Herity', 'nherity2@statcounter.com', 'Female', '145.151.178.98', 'Claver']
      ['4', 'Orazio', 'Frayling', 'ofrayling3@economist.com', 'Male', '25.199.143.143', 'Kungur']
```

Let's imagine we wanted a list of all the emails. For demonstration, since there are 1000 items plus the header, we will only do a few rows.

```
[20]: len(data_lines)

[20]: 1001
```

```
[22]: all_emails = []
      for line in data_lines[1:15]:
          all_emails.append(line[3])
```

```
[24]: print(all_emails)

      ['jzaniolini0@simplemachines.org', 'fdrillingcourt1@umich.edu', 'nherity2@statcounter.com', 'ofrayling3@economist.com', 'jmurrison4@cbslocal.com', 'lgame
      t5@list-manage.com', 'dhowatt6@amazon.com', 'kherion7@amazon.com', 'chedworth8@china.com.cn', 'hgasquoine9@google.ru', 'ftarra@shareasale.com', 'abathb@u
      mn.edu', 'lchastangc@goo.gl', 'cceried@yale.edu']
```

6. Now using the below code we can get the full names also.

```
[26]: full_names = []

      for line in data_lines[1:15]:
          full_names.append(line[1]+' '+line[2])
```

```
[28]: full_names
```

```
[28]: ['Joseph Zaniolini',
       'Freida Drillingcourt',
       'Nanni Herity',
       'Orazio Frayling',
       'Julianne Murrison',
       'Lucy Gamet',
       'Dyana Howatt',
       'Kassey Herion',
       'Chrissy Hedworth',
       'Hyatt Gasquoine',
       'Felicdad Tarr',
       'Andrew Bath',
       'Lucais Chastang',
       'Car Cerie']
```

7. Now we can also create a new CSV file; if there is any file with the same name, it will overwrite it.
8. Using the code below, we will write a new file.

```
[30]: # newline controls how universal newlines works (it only applies to text
      # mode). It can be None, '', '\n', '\r', and '\r\n'.
      file_to_output = open('to_save_file.csv','w',newline='')
```

```
[32]: csv_writer = csv.writer(file_to_output,delimiter=',')
```

```
[34]: csv_writer.writerow(['a','b','c'])
```

```
[34]: 7
```

```
[36]: csv_writer.writerows([['1','2','3'],['4','5','6']])
```

```
[38]: file_to_output.close()
```

9. We can also write to an existing file using the code below.

```
[40]: f = open('to_save_file.csv','a',newline='')
```

```
[42]: csv_writer = csv.writer(f)
```

```
[44]: csv_writer.writerow(['new','new','new'])
```

```
[44]: 13
```

```
[46]: f.close()
```