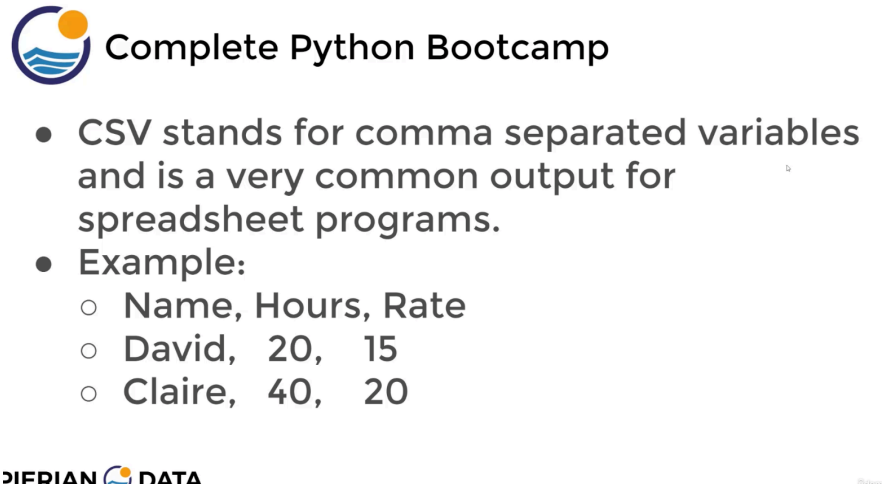
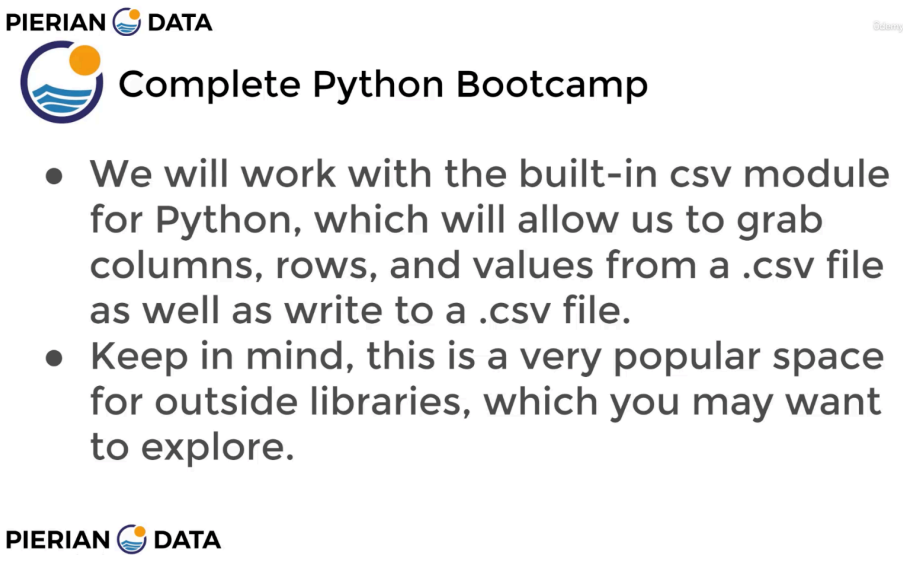
SECTION 17: WORKING WITH PDFS AND SPREADSHEET CSV FILES - 45 minutes, 5 parts

2/5 Working with CSV Files in Python

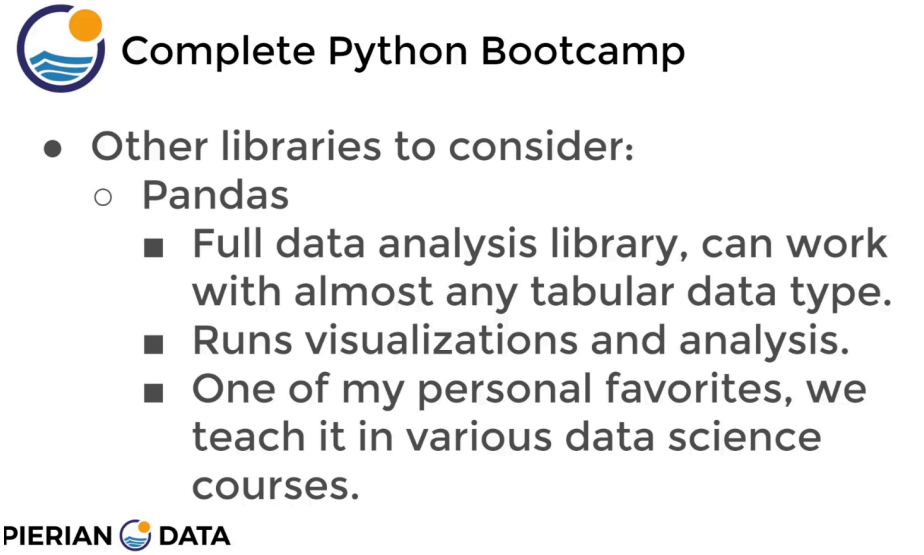
- -> working with CSV files in Python
- -> Comma Separated Variables
- -> these are spreadsheets
- -> the values are separated by commas
- -> see the image for this syntax
- -> information is only exported
- -> the CSV file only contains the raw information from the spreadsheet

- -> there is a built-in csv module
- -> this is a space for outside libraries

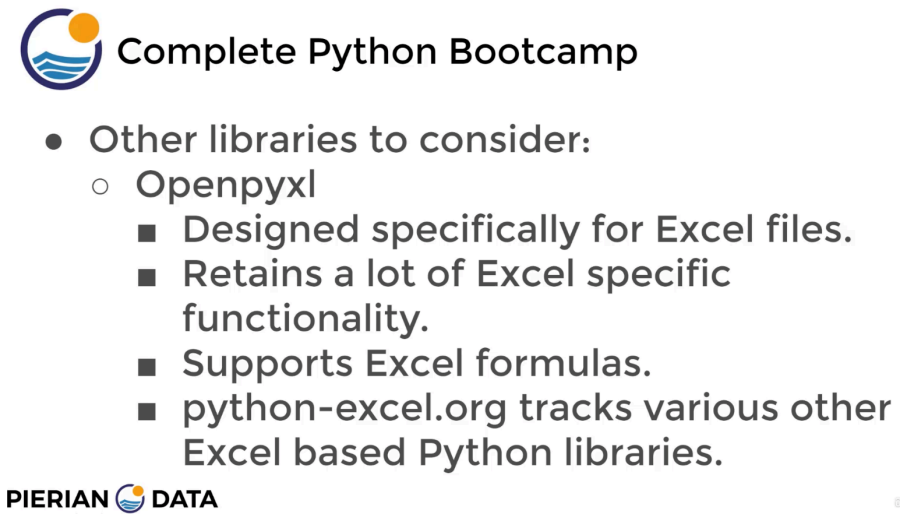




- **-> other libraries**
  - -> Pandas
  - -> this can be used for other data types <- SQL
  - -> this is a larger library and can also be used for data visualisation



  - **-> Openpyxl**
    - -> this is for Excel functionality with Python
    - -> it can do Excel libraries
    - -> we can also track other libraries



  - **-> Google Sheets Python API**
    - -> this allows us to make changes to the spreadsheets which are hosted online
    - -> making changes to the spreadsheets which are hosted online, from a Python environment
    - -> this is available in other programming languages

- -> we need these modules to be able to export to CSV files

Complete Python Bootcamp

- The common factor between all of these spreadsheet programs is that they can always export to .csv.
- Let's explore Python's built-in capabilities with the csv module!

PIERIAN DATA

- **-> in a Jupyter notebook**
  - -> pwd <- for the current working directory of the ipynb file
  - **-> to load in a CSV file**
    - **-> import modules**
      - -> import csv
    - **-> then open the file**
      - -> he sets a variable equal to open('file_name.csv')
    - **-> then call csv.reader on it**
      - -> this converts it into a CSV file
      - -> csv.reader(the variable which the data was loaded into)
      - -> this creates an error
      - -> it's a UnicodeDecodeError
      - -> the encodings which the file can have
      - -> the file can read or can't read certain types of special characters
      - -> in this case there are characters in the file which are stopping it from loading (@)
      - -> there are some characters which accents over certain files
      - -> he's set one of the

```
In [40]: import PyPDF2

In [41]: f = open('Working_Business_Proposal.pdf','rb')

In [42]: pdf_reader = PyPDF2.PdfFileReader(f)

In [ ]: pdf_reader.numPages

In [43]: pdf_reader.numPages

Out[43]: 5

In [44]: page_one = pdf_reader.getPage(0)

In [45]: page_one_text = page_one.extractText()

In [46]: page_one_text

Out[46]: 'Business Proposal\n The Revolution is Coming\n Leverage agil
e frameworks to provide a robust synopsis for high level \nov
erviews. Iterative approaches to corporate strategy foster co
llaborative \nthinking to further the overall value propositi
on. Organically grow the \nholistic world view of disruptive
innovation via workplace diversity and \nempowerment. \nBring
to the table win-win survival strategies to ensure proactive
\ndomination. At the end of the day, going forward, a new nor

In [47]: f.close()

In [48]: f = open('Working_Business_Proposal.pdf','rb')
         pdf_reader = PyPDF2.PdfFileReader(f)

In [ ]: first_page = pdf_reader.getPage(0)

In [51]: type(first_page)

Out[51]: PyPDF2.pdf.PageObject

In [ ]: pdf_writer.addPage(first_page)
```

arguments to the open method to utf-8
- -> in this case there are different emails (for example) which we have in the files

```
In [53]: pdf_output = open('Some_BrandNew_Doc.pdf','wb')

In [54]: pdf_writer.write(pdf_output)

In [55]: f.close()
```

- **-> then reformat it into a Python object**
  - -> a list of lists
  - -> he's loaded in the data and he's looking at the syntax of it
  - -> this is dummy data
  - -> looping through the lists of data
  - -> he's checking the length of the data

```
In [56]: pdf_output.close()

In [ ]: f = open('Working_Business_Proposal.pdf','rb')


        pdf_text = []

        pdf_reader = PyPDF2.PdfFileReader(f)

        for num in range(pdf_reader.numPages):

            page = pdf_reader.getPage(num)
            pdf_text.append(page.extractText())
```

```
In [59]: pdf_text[0]

Out[59]: 'Business Proposal\n The Revolution is Coming\n Leverage agil
         e frameworks to provide a robust synopsis for high level \nov
         erviews. Iterative approaches to corporate strategy foster co
         llaborative \nthinking to further the overall value propositi
         on. Organically grow the \nholistic world view of disruptive
         innovation via workplace diversity and \nempowerment. \nBring
         to the table win-win survival strategies to ensure proactive
         \ndomination. At the end of the day, going forward, a new nor
         mal that has \nevolved from generation X is on the runway hea
```

```
In [61]: print(pdf_text[1])

         Completely synergize resource taxing relationships via premie
         r niche
         markets. Professionally cultivate one-to-one customer service
         with robust
         ideas. Dynamically innovate resource-leveling customer servic
         e for state of
```

  - **-> inspecting the data**
    - -> the first row is the column lines
    - -> he's printed out the length of the data <- the number of datapoints / rows
    - -> he's then iterating through the lines in the set and printed out the first 5 of them
    - -> we can extract certain rows from the set by doing this
    - -> [0] here is the column name, for example
    - -> the first item in this example corresponds to the column header
    - -> to extract a single value, we target a row and then one of its elements [ ][ ], vs just [ ]

  - **-> an example of data extraction**
    - -> he's defined a variable with an empty list
    - -> and then iterated through the lines from 1-15
    - -> we are adding information from those lines to the lists in the variables
    - -> then printed out a list of emails in the set
    - -> we extract information if we want to automate it

  - **-> then in another example**
    - -> he's printed out the first 10 elements in the set
    - -> then is iterating through and adding in logic
    - -> he first defines an empty list to populate
    - -> concatenating the list with the second line and an empty string (a space)

- **-> writing to a CSV file**
  - -> he opens a CSV file and stores it in a variable

- -> we are over writing files with the existing names
- -> the second argument to the open method in this case is for writing the file
- -> the third argument is for a newline
- -> this enables us to write to a file
- -> first opens the CSV file, and then is calling the csv.writer method on it in another cell

- **-> the arguments of this are the variable which stores name of the opened file and a tab separator (\t), or comma**
  - -> so it has three arguments
  - -> both times, he sets them equal to variables
  - -> \t <- tab separated file
  - -> csv.writer

- **-> writing a row**
  - -> this can be done using the .writerow method
  - -> there is also the .writerows method
  - -> the rows which we want added on are passed in as its arguments
  - -> this is used to write multiple rows

- **-> then closing the file**
  - -> we need to close the file
  - -> this is done using the .close() method
  - -> we can add more rows to the file
  - -> we can see the saved file in the same directory as the ipynb file

- **-> to append to a file**
  - -> he opens the file using the .open method
  - -> then the .writer method
  - -> then writes a single row to it using the .writerow method
  - -> then he closes the file using the .close() method

- **-> to generalise this**
  - -> we read in the information
  - -> then we process it
  - -> then we write the information back or save it as a Python object