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

3/5 Working with PDF Files in Python

- **working with PDF files with Python**
  - **-> portable document format**
    - -> this was developed by Adobe in the 1990s
    - -> they share the same extension
    - **-> PDFs aren't machine readable in Python**
    - -> they display a fixed layout flat document
    - -> CSV files can be read in Python but PDFs are harder for it to read
    - -> images / tables / font changes in PDFs can be harder to read for Python
    - -> there are external programs which can be used for this

Working with PDF Files

PIERIAN DATA

  - **-> this is using PyPDF2 for this**
    - -> not all PDFs can be read by PyPDF2
    - -> pip install PyPDF2

Complete Python Bootcamp

- PDF stands for Portable Document Format and was developed by Adobe in the 1990s.
- The most important thing to keep in mind is that while PDFs share the same extension and can be viewed in PDF readers, many PDFs are **not** machine readable through Python.

PIERIAN DATA

  - **-> in an ipynb file**
    - -> the PDF s are in the same directory as the ipynb file
    - -> there are links at the top of the boilerplate ipynb file for this
    - -> we can only read from PDF files
    - -> this is reading in from PDF files, adding in pages and working with the text from them

```
In [40]: import PyPDF2

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

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

In [ ]: pdf_reader.numPages
```

    - **-> importing modules**
      - -> pip install PyPDF2
      - -> then importing it <- import PyPDF2
    - -> then attaching a reader object to it

```
In [43]: pdf_reader.numPages

Out[43]: 5
```

    - **-> he defines a variable and sets it equal to the opened PDF file**
      - -> the first argument is the name of the pdf file we are opening
      - -> then the second is read binary ('rb') <- this is not a normal text file

```
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
```

- -> he then defines another variable, to read this PDF file

- **-> returning the number of pages (et al) in the PDF file**
  - -> we can then view how many pages are in this file, by using the .numPages method
  - -> to return the first page
    - -> he uses the .getPage(n) method
  - -> then there is the .extractText() method
  - -> once we have the text, we can search for specific text

- **-> steps**
  - -> read in the file with 'rb' as the mode
  - -> use PdfFileReader on the file
  - -> confirm the number of files
  - -> check if this works with the .getPage method
  - -> then using the extractText method
  - -> then viewing the text
  - -> some pages can be too blury to extract the text
  - -> if there is an empty string as a result of this, then the PDF isn't comparable with the module
  - -> then you close it f.close()

- **-> you can create a for loop for the range in number of pages**
  - -> having a variable call

- **-> to add to PDF files**
  - -> we can write to PDF files

```
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)

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

In [54]: pdf_writer.write(pdf_output)

In [55]: f.close()

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
```

- -> writing new pages
- -> we can't go to the middle of a PDF file and add to the middle of it
- -> this is due to the complexity of the PDF format
- -> text formats are preferable to PDF

- **-> process**
  - **-> open the pdf file**
    - -> using the open method
    - -> the first argument of this is the name of the PDF file which we are opening
  - -> wb is write binary
  - -> then we write to this using the .write argument
  - -> then f.close() to close the file

  - **-> overall**
    - -> we read the file, got a page object then wrote to a new file
    - -> then added pages with a write object
    - -> then wrote binary and used a the write method to write to it
    - -> then closed the file

- **-> to grab all of the text in a PDF file**
  - -> he opens a PDF file, using the open method
  - -> the second argument to this is 'read binary'

  - **-> then defines a new variable**
    - -> this is an empty list
    - -> the index of this corresponds to the page number in the PDF

  - **-> then he creates a reader object**
    - -> this is a reader object with the file we opened as the argument

  - **-> then he creates a for loop**
    - -> we get a page at a particular number position
    - -> appending / extracting the text on the page
    - -> he returns pdf_text, which in this case is a variable which contains all of the text in the PDF
    - -> we can print out different elements of this
    - -> this is text at different elements of the page

  - -> this was reading / writing the PDFs
  - -> next is a puzzle exercise for CSV files and PDF files