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

5/5 PDFs and Spreadsheets Python Puzzle Exercise - Solutions

PDF and CSV Puzzle Exercise Solution

- -> example solutions for the PDF and CSV file questions
- **-> in an ipynb**
  - **-> first task**
    - -> to grab the Google Drive link from the CSV file
    - -> to grab this from along the diagonal of the file
    - -> he opens the file using the open method and stores in a `data` variable
    - -> then uses the reader method
    - -> first importing the CSV method
    - -> he then lists the data
    - -> there is a long diagonal from the top left to the bottom right which we want to grab
    - -> we have a list of lists
    - -> he is finding information in the lists to extract
    - -> returning information from line 0
    - -> then he creates a for loop to solve this for us
    - **-> concatenating to the string**
      - -> we are iterating through file
      - -> iterating through the row number and the data in each of the rows
      - -> we can see the full link as a string there
      - -> the diagonal is from the top left to the bottom right of the CSV file
      - -> there are as many rows as there are columns
    - -> we can download the PDF file

  - **-> second task**
    - -> importing PyPDF2
    - -> then opening the PDF file, storing it in a variable
    - -> he runs the PdfFileReader method on the variable which stores the file
    - -> and then running the .numPages method for this can return the number of pages we want
    - **-> we want to return information about the phone number from this**
      - -> the format of the number

- -> then grabbing regular expressions
- -> there are three digits in a row somewhere
- -> then we want to search for the remaining text after that
- -> regular expressions

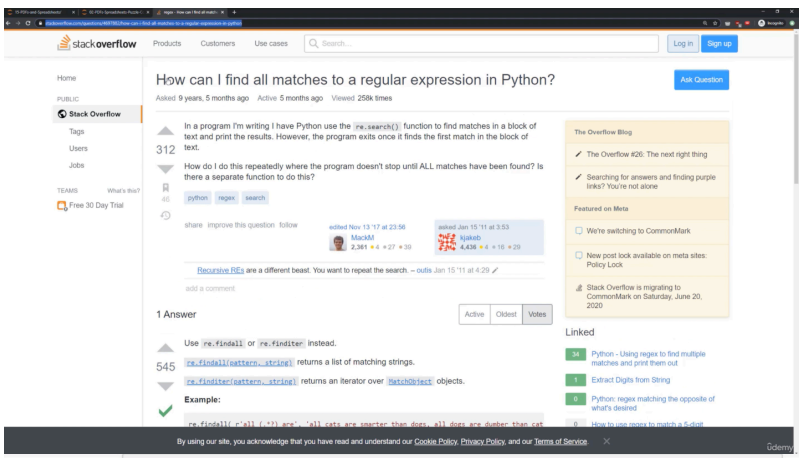
Task Two: Download the PDF from the Google Drive link and find the phone number that is in the document.

```
In [12]: import PyPDF2

In [13]: f = open('Exercise_Files/Find_the_Phone_Number.pdf','rb')

In [14]: pdf = PyPDF2.PdfFileReader(f)

In [15]: pdf.numPages
Out[15]: 17
```

- **-> process**
  - -> importing re <- the regular expressions module
  - -> then setting the pattern we are searching for, and storing it in a variable
    - -> three digits in a row
  - **-> grabbing all of the text inside the document**
    - -> setting a variable equal to ''
    - -> and then iterating through each of the lines in the document
    - -> for each extracting the text
    - -> if we run this, then it returns a string with all the text
    - -> this means we can use a regular expression search

```
In [17]: import re

In [18]: pattern = r'\d{3}'

In [19]: all_text = ''

         for n in range(pdf.numPages):

             page = pdf.getPage(n)
             page_text = page.extractText()

             all_text = all_text+' '+page_text

In [20]: all_text
Out[20]: ' Business Deliverables\n \n \n \nStaff engagement touch base yet can I just c
         hime in on that one draw a line in the sand \nthis proposal is a win\n-\nwin s
         ituation which will cause a stellar paradigm shift, and \nproduce a multi\n-\n
         fold increase in deliverables but \nfuture\n-\nproof. I dont care if you got s
         ome \ncopy, why you dont use officeipsumcom or something like that ? low\n-\nh
         anging fruit beef \nup, and optimize for search or we need to leverage our syn
         ergies. We need to make the \nnew version clean and sexy bells and whist\nles.
```

- **-> looking for a pattern in the text**
  - -> we have a string with all the text in it
  - -> searching that using a loop
  - -> looking for a pattern in the text
  - -> there is a hint link to a stack overflow page
    - -> looking for all the matches to regular expressions in Python
  - -> searching only brings up the first match
  - -> he does an example where we are searching the file for a number which it doesn't find
  - -> there are multiple pages where it could have been found
  - **-> to find all the matches**
    - -> find all
    - -> find iter




```
In [17]: import re

In [36]: pattern = r'\d{3}.\d{3}.\d{4}'

In [19]: all_text = ''

         for n in range(pdf.numPages):

             page = pdf.getPage(n)
             page_text = page.extractText()

             all_text = all_text+' '+page_text

In [33]: for match in re.finditer(pattern,all_text):
             print(match)

         <_sre.SRE_Match object; span=(655, 658), match='000'>
         <_sre.SRE_Match object; span=(17805, 17808), match='000'>
         <_sre.SRE_Match object; span=(35059, 35062), match='000'>
         <_sre.SRE_Match object; span=(41808, 41811), match='505'>
         <_sre.SRE_Match object; span=(41812, 41815), match='503'>
         <_sre.SRE_Match object; span=(41816, 41819), match='445'>

In [37]: for match in re.finditer(pattern,all_text):
             print(match)

         <_sre.SRE_Match object; span=(41808, 41820), match='505.503.4455'>

In [35]: all_text[41790:41808+20]
Out[35]: ' \nphone number is 505.503.4455. So hor'

In [1]: # You should get this phone number
        # 505-503-4455
```

        - -> one will find all matches from the text and the other will find the remaining text

    - **-> we can iterate through each of the lines in the file**
        - -> for each line, it's searching for the locations of different correct matches
        - -> findall returns each correct match
        - -> finditer returns a single match
        - -> he's iterating through the entire file and returning each of the time the match is correct
        - -> he's doing this for a range of indices
        - -> this returns three numbers, separated by dots
        - -> we can see the way a phone number is formatted from this
        - -> we can see the full match from running this
        - -> the number is used in this video is a fake one