**Applications of Regular Expressions**

1. **Regular Expression in UNIX:** Basically Regular Expression are used in search tools. In UNIX we use tool **egrep** to search a text file. This command searches for a text pattern in the file and list the file names containing the pattern.
2. **Regular Expressions in String processing:** Regular expressions are useful in a wide variety of text string processing, common applications include data validation, data scraping, data wrangling, simple parsing.
3. **Regular Expressions in String replacement:** Regular expressions are useful in replacing a particular matching pattern with our pattern.
4. **Regular Expressions in Lexical Analysis:** The Lexical Analyzer reads source code and generates a stream of tokens. Tokens can be described using regular expressions.
5. **Regular Expressions on Search Engines:** While regexps would be useful on Internet search engines, processing them across the entire database could consume excessive computer resources depending on the complexity and design of the regex.

**Python Packages for accessing PDF documents**

1. **PyPDF2:** A Pure-Python library built as a PDF toolkit. It is capable of:

|  |  |  |
| --- | --- | --- |
| Function | Description | Example |
| The PdfFileReader Class | Initializes a PdfFileReader object. This operation can take some time, as the PDF stream’s cross-reference tables are read into memory. | # importing required modules  import PyPDF2    # creating a pdf file object  pdfFileObj = open('example.pdf', 'rb')    # creating a pdf reader object  pdfReader=PyPDF2.PdfFileReader(pdfFileObj) |
| The PdfFileMerger Class | Initializes a PdfFileMerger object. PdfFileMerger merges multiple PDFs into a single PDF. It can concatenate, slice, insert, or any combination of the above. | import PyPDF2  def PDFmerge(pdfs, output):   pdfMerger = PyPDF2.PdfFileMerger()     # appending pdfs one by one   for pdf in pdfs:   with open(pdf, 'rb') as f:   pdfMerger.append(f)     # writing combined pdf to output pdf file   with open(output, 'wb') as f:   pdfMerger.write(f) |
| The PdfFileWriter Class | This class supports writing PDF files out, given pages produced by another class (typically PdfFileReader). | # creating a pdf writer object for new pdf  pdfWriter = PyPDF2.PdfFileWriter()  #writing modified pages to new file  pdfWriter.write(newFile) |

1. **Textract:** This package provides a single interface for extracting content from any type of file, without any irrelevant markup

|  |  |  |
| --- | --- | --- |
| Function | Description | Example |
| textract.process() | to obtain text from a document. You can also pass keyword arguments to textract.process, for example, to use a particular method for parsing a pdf | import textract text = textract.process('path/to/a.pdf', method='pdfminer') |

1. **Slate:** Slate is a Python package that simplifies the process of extracting text from PDF files. It depends on the PDFMiner package.

|  |  |  |
| --- | --- | --- |
| Function | Description | Example |
| Slate provides one class, PDF | PDF takes a file-like object and will extract all text from the document, presenting each page as a string of text | with open('example.pdf') as f: doc = slate.PDF(f) |

**Applying RE on PDF Document**





