**Python for Everyone: Notes**

**Chapter 11: Regex**

* Regular Expressions
  + In computing, a regular expression, also referred to as “regex” or “regexp”, provides a concise and flexible means for matching strings of text, such as particular characters, words, or patterns of characters. A regular expression is written in a formal language that can be interpreted by a regular expression processor
  + Really clever “wild car” expressions for matching and parsing strings
* Understanding regular expressions
  + Very powerful and quite cryptic
  + Fund once you understand them
  + Regular expressions are a language unto themselves
  + A language of “marker characters” – programming with characters
  + It is kind of an “old school” language – compact
* Regular Expression quick guide
  + ^ Matches the beginning of a line
  + $ Matches the end of the line
  + . Matches any character
  + \s Matches whitespace
  + \S Matches any non-whitespace character
  + \* Repeats a character zero or more times
  + \*? Repeats a character zero or more times (non-greedy)
  + + Repeats a character one or more times
  + +? Repeats a character one or more times (non-greedy)
  + [aeiou] Matches a single character in the listed set
  + [^XYZ] Matches a single character not in the listed set
  + [a-z0-9] The set of characters can include a range
  + ( Indicates where string extraction is to start
  + ) Indicates where string extraction is to end
* The regular expression module
  + Before you can use regular expressions in your program, you must import the library using “import re”
  + You can use re.search() to see if a string matches a regular expression, similar to using the find() method for strings
  + You can use re.findall to extract portions of a string that matches your regular expression, similar to a combination of find() and slicing, var[5:10]
* Wilf card characters
  + The dot character matches any character
  + If you add the asterisk character, the character is “any number of times”
    - ^X.\*:
  + ^ match the start of the line
  + . match any character
  + \* many times
* Find tuning your match
  + Depending on how clean your data is and the purpose of your application, you may want to narrow your match down a bit
    - ^X-\S+:
* Matching and extracting data
  + Re.search() returns a True/False depending on whether the string matches the regular expression
  + If we actually want the matching strings to be extracted, we use re.findall()
  + Re.find(‘[0-9]+’,x)
  + When we use re.findall(), it returns a list of zero or more sub-strings that matches the regular expression
* Warning: Greedy matching
  + The repeat characters (\* and +) push outward in both directions (greedy) to match the largest possible string
* Non-greedy matching
  + Not all regular expression repeat codes are greedy
  + If you add a ? character, the + and \* chill out a bit
* Fine-tuning string extraction
  + You can refind the match for re.findall() and separately determine which portion of the match is to be extracted by using parentheses
  + \S+@\S+
  + Parentheses are not part of the match, but they tell where to start and what strings to extract
  + ^From (\S+@\S+)
* The double split pattern
  + Sometimes we split a line one way, and then grab one of the pieces of the line and split that piece again
* Escape character
  + If you want a special regular expression character to just behave normally you prefix it with ‘\’