Regular Expressions Outline

<https://github.com/zeeshanu/learn-regex>

Select-String -Path jennysinfo.txt

Select-String –Path PhoneNumbers.txt

1. History
   1. Created in
2. Different Engines
3. Explanation of Demos
   1. Waldo
      1. Explain English Name is Wally
4. Match Name: Waldo
   1. **Waldo**
   2. **Explain .**
      1. **W.ldo**
   3. **We don’t want to include Sir Waldo so**
      1. Beginning: ^
   4. **Explain + and \***
      1. **Show the WWaldo: ^W+aldo**
      2. **Show the Aldo: ^W\*aldo**
   5. What if we know that the number we’re looking for ends with an 8
      1. End: $
      2. 8$
   6. Now what if we know that it starts with Waldo and ends with an 8
      1. ^Waldo.\*8$
   7. Then Match Name Wally or Waldo ()
      1. **W(aldo|ally)**
   8. Match WAldo
      1. W[@Aa]ldo
      2. W[aeiou]ldo
      3. It allows Ranges (full and partial)
         1. W[a-z]ldo
         2. W[a-ck-w]ldo
      4. It ignores special characters except for the backslash
         1. W[\*+.^ \\]ldo
      5. Negated Character put the ^ character at the beginning and it reverses the pattern (it looks for not those things)
         1. W[^\*+.^ \\]ldo
   9. Space meta character
      1. \s -> this matches any whitespace character
      2. \S -> this matches any non whitespace character
   10. Word meta character
       1. \w -> Matches against any chacter a-z A-Z 0-9 \_ : ^W\w.+
       2. \W -> Matches the reverse
   11. Digit meta character
       1. \d -> matches any digit: ^W\d.+
       2. \D -> matches the opposite: ^W\D.+
   12. The Question Mark ?
       1. Matches 0 or 1 instances of a thing
       2. ^Wa@?ldo: It will match Wa@ldo but not wa@@ldo
       3. ^Wa@?@?ldo
   13. Certain Length
       1. {3,15}
       2. Look for movie number (starting with 555)
          1. \s[5-]{3,7}
   14. Flags
       1. By default PowerShell is case insensitive
       2. So we’re going to jump to egrep for a moment
          1. Case insensitive: –i
          2. Print Line Number: -n
          3. Match the Regex Compliment: -v
          4. Count: -c
          5. Which files it found a match in: -l
5. Look for Girlfriend’s Name by Looking for Phone Number:
   1. Jenny: .\*
   2. Greediness
      1. "This is a <EM>first</EM> test." -match "<.+>" | %{$matches[0]}
      2. "This is a <EM>first</EM> test." -match "<[A-Za-z]>" | %{$matches[0]}
   3. Genny, Jenny, Ginny
      1. (G|J)\w+\s.\*
      2. Filter out Great Old One
         1. (G|J)\w+\s\w+
   4. Jenny : Look for 867+
   5. Look for 867/d
   6. Look for Literal character
      1. Maybe you stored the phone number as
         1. 867/5309
            1. so look for it as 867\/5309
6. Look Arounds
   1. <http://www.rexegg.com/regex-disambiguation.html#confusingcouples>
   2. Look ahead
      1. After: '[\d-]+(?= : Home)'
      2. Before: (?=\d : Home)[\d-]+
      3. Not After: [\d-]+(?! : Home)
7. Conditionals
   1. <http://www.rexegg.com/regex-conditionals.html>
   2. Select-String -Path jennysinfo.txt `  
      -Pattern '(555-).\*|5309' `
8. Matching on multiple lines
9. Capturing Data
   1. <http://www.rexegg.com/regex-disambiguation.html#namedcapture>
   2. In VS Code Replace is $0-$9
   3. "<phoneNumber>867-5309</phoneNumber>"`  
      -match "([\d-]+)" | `  
      %{Write-Host "$($matches[1])" }
   4. Show 0 – 9
      1. Talk about how $0 is the whole matched string
      2. $1 is the first match
   5. Talk about more than 9
   6. Show Named Variables
      1. "<phoneNumber>867-5309</phoneNumber>"`  
         -match "(?<open><)(?<tag>.+)(?<close>>)" |`  
         %{Write-Host `  
         "  
         $($matches["open"])   
         $($matches["tag"])   
         $($matches["close"])" }
   7. Recursion
   8. <http://www.rexegg.com/regex-disambiguation.html#recursion>
10. Validating Data
    1. First and Last Name
    2. Phone Number
    3. Password Validation
    4. URL
11. Cheatsheet
    1. <https://www.cheatography.com/davechild/cheat-sheets/regular-expressions/>