Outline for November 4, 2024

Reading: §11 **Due:** Homework 3, due November 13, 2024

1. Pattern matching, continued
   (a) Match any of a set of characters: `[0123456789]`, `[^0123456789]`, `[0-9]`
   (b) Repetition:
      i. `*` — match 0 or more of the preceding regular expression
      ii. `?` — match 0 or 1 of the preceding regular expression
      iii. `+` — match 1 or more of the preceding regular expression
      iv. $\{m,n\}$ — match between m and n (inclusive) of the preceding regular expression
      v. greedy matching; each matches as many characters as possible
      vi. put `?` after and they will match as few characters as possible
   (c) `^` — match start of string or line
   (d) `$` — match end of string or line
   (e) `(`, `)` — used to group regular expressions
   (f) `|` — used to indicate one of the regular expressions must be matched
   (g) `\` — used to escape metacharacters
2. Special sequences
   (a) `\b` — match beginning or end of word
3. Useful abbreviations in patterns
   (a) `\`n — match n^{th} group
   (b) `\d` — match any digit; same as `[0-9]`
   (c) `\s` — match any space character; same as `[ \t\n\r\f\v]` (usually)
   (d) `\w` — match any alphanumeric character and underscore; same as `[a-zA-Z0-9_]`
   (e) `\D` — match any character *except* a digit; inverse of `\d`
   (f) `\S` — match any character *except* a space character; inverse of `\s`
   (g) `\W` — match any character *except* an alphanumeric character or underscore; inverse of `\w`
   (h) `\b` — match a word boundary; a word is a sequence of alphanumeric characters
4. Useful functions/methods [*recomp.py*, *renocomp.py*, *regroup.py*]
   (a) `re.compile(`*`str`*`)` compiles the pattern into *pc* (that is, `pc = re.compile(str)`)
   (b) *`pc`*`.match(`*`str`*`)` returns None if compiled pattern *pc* does not match beginning of string *str*
   (c) *`pc`*`.search(`*`str`*`)` returns None if pattern *pc* does not match any part of string *str*
   (d) *`pc`*`.findall(`*`str`*`)` returns a list of substrings of the string*str* that match the pattern *pc*
   (e) *`pc`*`.group(`*`str`*`)` returns the substring of the string *str* that the pattern *pc* matches
   (f) *`pc`*`.start(`*`str`*`)` returns the starting position of the match
   (g) *`pc`*`.end(`*`str`*`)` returns the ending position of the match
   (h) *`pc`*`.span(`*`str`*`)` returns tuple (start, end) positions of match
5. "Raw" string notation: backslash not handled specially; put "r" before string
6. Reading a URL [*geturl.py*, *geturl2.py*, *geturl3.py*]

(a) Opening a URL

(b) Reading the page as a string

(c) The role of `decode()` [*geturl-nd.py*]

7. A program to print links in web pages [*urlpat.py*, *urlpat2.py*]