Outline for October 27, 2025

Reading: §11 **Due:** Homework 2, due October 29, 2025

1. Writing a program to play rock-paper-scissors: top-down design
   (a) Problem statement and general algorithm idea
   (b) Data representation and program structure [*rps-1.py*]
   (c) Figure out who wins [*rps-2.py*]
   (d) Get computer choice [*rps-3.py*]
   (e) Get user input [*rps-4.py*]
   (f) Make it user-friendly [*rps-5.py*]
2. Writing a recursive permutation program [*perm.py*]
3. Character/integer conversions [*chr.py*, *ord.py*]
   (a) Cæsar cipher (shift cipher) [*caesarenc.py*, *caesardec.py*]
4. Pattern matching
   (a) Regular expressions
   (b) Atoms: letters, digits
   (c) Match any character except newline: `.`
   (d) Match any of a set of characters: `[0123456789]`, `[^0123456789]`, `[0-9]`
   (e) 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 pattern matches as many characters as possible
      vi. put `?` after and pattern matches as few characters as possible
   (f) `^` — match start of string or line
   (g) `$` — match end of string or line
   (h) `(`, `)` — used to group regular expressions
   (i) `|` — used to indicate one of the regular expressions must be matched
   (j) `\` — used to escape metacharacters
5. Special sequences
   (a) `\b` — match beginning or end of word
6. 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

7. 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

8. "Raw" string notation: backslash not handled specially; put "r" before string