

Outline for November 2, 2023

Reading: §14

Assignments: Homework 3, due November 9, 2023

1. Example: memos
 - (a) Remember how slowly the recursive Fibonacci number program *rfib.py* ran? Here is a faster recursive version that uses memos [*rfibmemo.py*]
2. Recursion: permutations of characters in a string [*perm.py*]
3. 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: `*`, `+`, `{m,n}`; greedy matching; put `?` after and they match as few characters as possible
 - (f) Match start, end of string: `^`, `$`; `$` matches end of line, also
 - (g) Grouping: `(,)`
 - (h) Escape metacharacters: `\`
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