Outline for October 13, 2025

Reading: §4.5, 4.9, 6, 8 **Due:** Homework 1, due October 15, 2025

1. Greetings and Felicitations!
   (a) Zoom office hour on Wednesday moved to 4:00pm–4:50pm; same meeting id, password
2. String methods: methods that change, add, or delete characters do *not* alter the string to which they are applied; they return a new string that is a copy of the old string, suitably modified
3. String methods: type of characters in string (return `True` or `False`) [*strtype.py*]
   (a) *S*`.isalpha()` — `True` if only alphabetics (letters) in *S*
   (b) *S*`.isalnum()` — `True` if only alphanumerics (letters or digits) in *S*
   (c) *S*`.isdigit()` — `True` if only digits in *S*
   (d) *S*`.isspace()` — `True` if only white space (blanks, tabs, newlines) in *S*
   (e) *S*`.isupper()` — `True` if all letters in *S* are upper case
   (f) *S*`.islower()` — `True` if all letters in *S* are lower case
4. String methods: changing case of letters in string (return result of applying method) [*strchcase.py*]
   (a) *S*`.capitalize()` — If the first character of *S* is a letter, capitalize it
   (b) *S*`.title()` — Capitalize each word in *S*
   (c) *S*`.lower()` — Change all upper case letters in *S* to lower case
   (d) *S*`.upper()` — Change all lower case letters in *S* to upper case
   (e) *S*`.swapcase()` — Change all upper case letters in *S* to lower case and *vice versa*
5. String methods: stripping blanks from strings (return result of applying method) [*strstrip.py*]
   (a) *S*`.lstrip()` — Delete all leading white spaces from *S*
   (b) *S*`.rstrip()` — Delete all trailing white spaces from *S*
   (c) *S*`.strip()` — Delete all leading and trailing white spaces from *S*
6. String methods: find characters and substrings (return position or cause exception) [*strfind.py*]
   (a) *S*`.find(`*s*`)` — Return the index of the first occurrence of *s* in *S*; -1 if *s* not in *S*
   (b) *S*`.index(`*s*`)` — Return the index of the first occurrence of *s* in *S*; `ValueError` exception if *s* not in *S*
   (c) *S*`.rfind(`*s*`)` — Return the index of the last occurrence of *s* in *S*; -1 if *s* not in *S*
   (d) *S*`.rindex(`*s*`)` — Return the index of the last occurrence of *s* in *S*; `ValueError` exception if *s* not in *S*
7. String methods: miscellaneous [*strmisc.py*]
   (a) *S*`.count(`*s*`)` — Return the number of times *s* occurs in *S*
   (b) *S*`.startswith(`*s*`)` — `True` if *S* starts with *s*
   (c) *S*`.endswith(`*s*`)` — `True` if *S* ends with *s*
   (d) *S*`.replace(`*s*`,`*t*`)` — Replace all occurrences of *s* with *t* in *S*
8. What you can do with lists
   (a) Check membership: `in`, `not in`
   (b) `+`: concatenation
   (c) `*`: repetition
   (d) `list[a:b]`: slice list from *a* to $b-1$
   (e) `del list[i]`: delete element `list[i]`; *i* can be a slice

9. Objects, references, aliasing
   (a) For strings, one copy: assume `a = "banana"`
      i. After `b = a` or `b = a[:]`, then `a is b` is `True`
   (b) For lists, multiple copies: assume `A = [ 1, 2, 3 ]`
      i. After `B = A` then `A is B` is `True`
      ii. After `B = A[:]`, then `A is B` is `False`
10. Lists and strings [*datecvt.py*]
11. Program to print words in a line [*lines.py*]
12. What you can do with lists
   (a) Add elements to, remove elements: L`.append(`x`)`, L`.extend(`ls`)`, L`.insert(`i`, `x`)`, L`.pop()`, L`.remove(`x`)`
   (b) Element ordering: L`.reverse()`, L`.sort()`
   (c) Other: L`.count(`x`)`, L`.index(`x`)`
13. Searching a list
   (a) Example use: linear search [*linsearch.py*]
14. `isinstance(obj,type)` function
   (a) `type` is `bool`, `float`, `int`, `list`, `str`, `tuple`
15. Lists as parameters: can change list elements in function and they are changed in caller [*args2.py*]
16. More on parameters: named arguments and variable number of arguments [*args3.py*]
17. Recursion
   (a) *n* factorial [*nfact.py*]
18. Thinking recursively [*recfun.py*]
   (a) First: think of the recursive case (write the problem in terms of something involving a smaller instance of the problem)
   (b) Next: think of base case (when to stop)
   (c) Example: Find the length of a string
   (d) Example: Does the string only have alphabetic characters in it?
   (e) Example: Find the maximum element of a list
   (f) Example: Construct a string from a list of strings
   (g) Example: Reverse a string
19. Recursion
   (a) Palindromes [*palindrome.py*]
   (b) Fibonacci numbers [*rfib.py*]
   (c) Sum of digits [*sumdigits.py*]
   (d) Nested lists: is an item in a list? [*isinlist.py*]