

Outline for October 20, 2025

Reading: §4.5, 7, 9**Due:** Homework 2, due October 29, 2025

1. Greetings and Felicitations!
 - (a) Thursday's office hours moved to 12:10pm–1:00pm (they are in person)
2. Printing [*printing.py*]
3. Recursion
 - (a) Tower of Hanoi [*hanoi.py*]
4. Using random numbers
 - (a) `import random`
 - (b) Problem: compute π by tossing darts at a unit square
 - (c) First build routine to simulate dart toss at unit square [*mc-1.py*]
 - (d) Then build routine to see if co-ordinates are in unit circle [*mc-2.py*]
 - (e) Then build routine to read in number of tosses [*mc-3.py*]
 - (f) Put it all together [*mc-4.py*]
 - (g) Graphics! [*mc-5.py*]
5. Files
 - (a) What is a file?
 - (b) What can you do with it? (For example, read, write, append)
 - (c) Types of files (text, binary)
6. File Input and Output for text files
 - (a) Opening and closing: `open(filename, mode), close()`
 - (b) Reading: `readline(), readlines(), read(), read(n)`
 - (c) Writing: `write(str), writelines(list)`
7. Exception `EOFError` — input function encounters an end of file
8. Examples
 - (a) Print out a named file [*fileio1.py*]
 - (b) Print out a named file and prepend line numbers [*fileio2.py*]
 - (c) Store the output in *filename.lst* [*fileio3.py*]
9. Examples
 - (a) Put lines in a file in random order [*randlines.py*]
 - (b) Read in a list of words from a file, then search it as requested; similar to linear search program [*search-1.py*]
 - (c) Now see how many words you checked total [*search-1c.py*]
10. Dictionary
 - (a) Collection of key-value pairs
11. Creating dictionaries
 - (a) Using `d = {}`
 - (b) Using `d = dict()`
12. Methods for dictionaries
 - (a) `k in D`: True if dictionary `D` has key `k`; else False

- (b) `D.keys()`: list of keys in `D`
- (c) `D.values()`: list of values in `D`
- (d) `D.items()`: list of tuples (key, value) in `D`
- (e) `D.get(k, d)`: if key `k` in `D`, return associated value; else return `d`
- (f) `del D[k]`: delete tuple with key `k` from `D`
- (g) `D.clear()`: delete all entries in `D`

13. 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*]

14. Example: word frequency count

- (a) Unsorted [*wfc-1.py*]
- (b) Sorted alphabetically [*wfc-2.py*]
- (c) Sorted alphabetically, but dictionary order (note `key=str.lower()` in sorted [*wfc-3.py*])
- (d) Sorted by frequency (treat `lambda x: x[1]` as an idiom to reference the *value* of the dictionary entry, not the *key*—to go from highest to lowest, replace `x[1]` with `-x[1]`) [*wfc-4.py*]
- (e) Sorted by frequency first, then alphabetically—note use of function `alphafreq(x)`; you can use any function here, and the parameter is the item [*wfc-5.py*]

15. Handling exceptions

- (a) `except` [*except0.py*]
- (b) `except exceptcode` [*except1.py*]
- (c) `else` [*except2.py*]
- (d) `except exceptcode as msgvar` [*except3.py*]
- (e) `finally` [*except4.py*]
- (f) Exceptions in a function: who handles them? [*except5.py*, *except6.py*]
- (g) Using global variables as error flags [*except7.py*]
- (h) `raise exceptcode message` [*except8.py*]