CSCI 145 · Quiz 7 Reference Sheet (Lectures 13–14)

SETS (NO DUPLICATES, NO ORDER)

`{3, 1, 2}`	A set: curly braces, no colons
`set(nums)`	A set from a list; duplicates disappear
`set(word)`	A set of the letters in a string
`set()`	An empty set; `{}` is an empty **dictionary**
`len(s)`	The number of values in s
`x in s`	`True` if x is in s. Otherwise, `False`
`s.add(x)`	Puts x in s; if already there, nothing happens
`s.remove(x)`	Takes x out; if not there, a **KeyError**
`s.discard(x)`	Takes x out if it's there; no error
`s[0]`	**TypeError**: no order, so no indexing
`sorted(s)`	A sorted list of the values in s

- A set may print its values in any order
- Values must be immutable, like dictionary keys

`a & b`	In both: the intersection	`a.intersection(b)`
`a | b`	In either: the union	`a.union(b)`
`a - b`	In a, not in b: the difference	`a.difference(b)`

WHICH WOULD YOU USE?

	List	Tuple	Dictionary	Set
Looks like	`[3, 7]`	`(3, 7)`	`{"Ana": 92}`	`{3, 7}`
Find an item by	Position	Position	Key	Only `in`
Can change?	Yes	No	Yes	Yes
Duplicates?	Yes	Yes	Not as keys	No

REFERENCES

- A variable holds a **reference** to an object

`b = a`	An **alias**: a second name for the same object
`b.append(5)`	Changes the object; every name for it sees it
`b = [5]`	b names a new object; a doesn't change
`a == b`	The same value?
`a is b`	The same object?
`id(a)`	A number that names the object

- A function gets a reference: if it changes a list, dictionary, or set passed in, the caller sees the change
- A function with no `return` returns `None`
- Numbers, strings, and tuples are immutable, so sharing them is harmless

COPYING A LIST OF LISTS

`b = grid`	**Alias**: the same list
`b = grid.copy()`	**Shallow copy**: rows shared
`b = copy.deepcopy(grid)`	**Deep copy**: nothing shared

- Shallow copy: a new outer list holding the same rows (also `grid[:]`, `list(grid)`). Replacing a row (`grid[0] = [5, 6]`) changes one list only; a change inside a row (`grid[1][0] = 7`) shows up through both
- `copy.deepcopy` needs `import copy`

WRITING

`f.write(text)`	Writes text; adds no newline
`f.write(name + "\n")`	Writes one line
`f.write(str(92))`	`f.write(92)` is a **TypeError**

TEXT FILES

- A text file is one long string, with `\n` (newline) at the end of each line
- Escapes: `\n` newline, `\t` tab, `\"` quote, `\\` backslash
- `print` shows `\n` as a line break, then adds its own newline

```
with open(filename, mode) as f:
    read from or write to f here
the file is closed here
```

Mode	Means	If the file exists	If it doesn't
`"r"`	Read (default)	Reads it	**FileNotFoundError**
`"w"`	Write	Erases it first	Creates it
`"a"`	Append	Adds to the end	Creates it

- Without `with`: `f = open(filename)` … `f.close()`
- A name alone: Python looks in the program's folder

READING

`f.read()`	The whole file as one string
`f.readlines()`	A list of lines; each keeps its `\n`
`for line in f:`	One line per pass; each keeps its `\n`
`line.strip()`	Removes spaces and `\n` at both ends
`line.split(",")`	A list of strings, split at each comma
`line.split()`	A list of the words, split at spaces
`int(text)`	A string to a number; also `float`
`if line.strip():`	Skips a blank line: `int("")` crashes
`lines[1:]`	Skips the header line

Everything read from a file is a **string**

CSV (import csv)

A header line names the columns; then one row per line

`rows = list(csv.reader(f))`	A list of lists of strings
`writer = csv.writer(f)`	A writer for the open file
`writer.writerow(row)`	Writes a list as one line

- Open with `open(filename, "w", newline="")` to write
- `writerow` adds the commas and the newline, and converts numbers
- A field with a comma inside goes in quotes: `csv.reader` handles them, `split(",")` doesn't
- CSV has no types: every field comes back a string

JSON (import json)

`data = json.load(f)`	File to Python data
`json.dump(data, f)`	Python data to file

- Like Python dictionaries and lists; can nest
- Keeps types; JSON's `true` is Python's `True`
- A table becomes a list of dictionaries

CSV OR JSON?

	CSV	JSON
Looks like	A table	Dictionaries and lists
Types	All strings	Kept
Shape	Same fields per row	Any nesting

Lines of words, like a song's: plain text