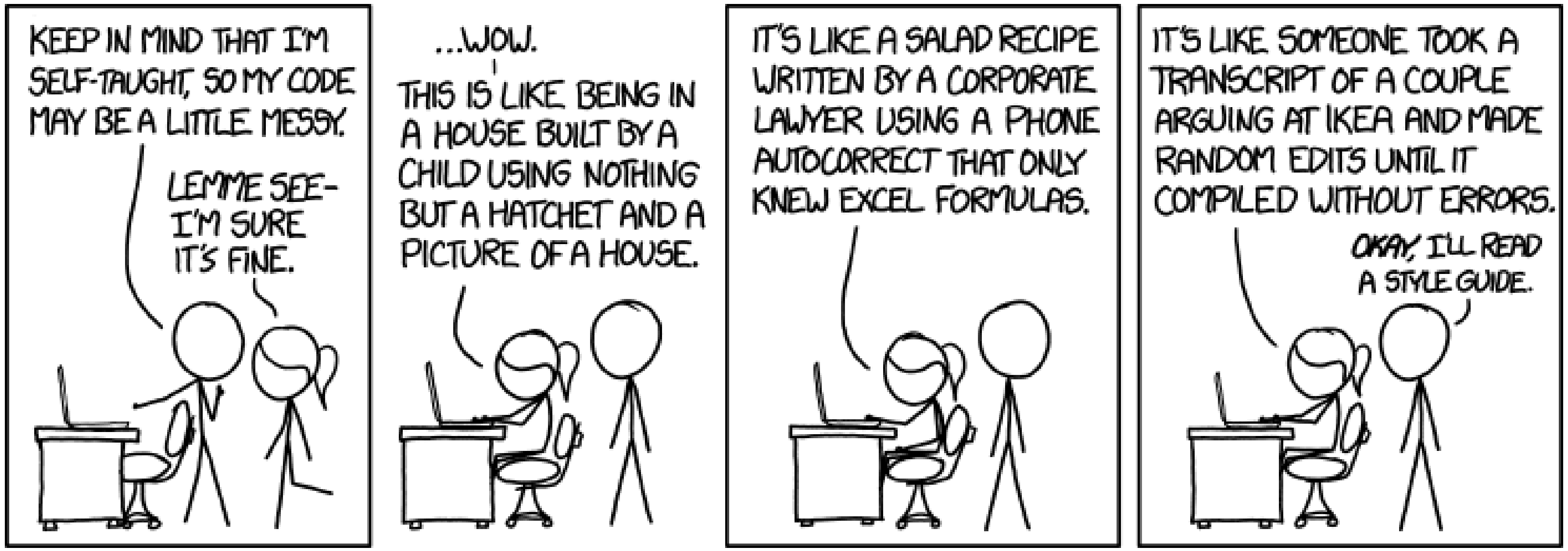
CSc 110, Spring 2018

Lecture 27: Lists that change size and File Output

Adapted from slides by Marty Stepp and Stuart Reges


KEEP IN MIND THAT I'M SELF-TAUGHT, SO MY CODE MAY BE A LITTLE MESSY.
LEMME SEE– I'M SURE IT'S FINE.
...WOW.
THIS IS LIKE BEING IN A HOUSE BUILT BY A CHILD USING NOTHING BUT A HATCHET AND A PICTURE OF A HOUSE.
IT'S LIKE A SALAD RECIPE WRITTEN BY A CORPORATE LAWYER USING A PHONE AUTOCORRECT THAT ONLY KNEW EXCEL FORMULAS.
IT'S LIKE SOMEONE TOOK A TRANSCRIPT OF A COUPLE ARGUING AT IKEA AND MADE RANDOM EDITS UNTIL IT COMPILED WITHOUT ERRORS.
OKAY, I'LL READ A STYLE GUIDE.

Assertion example

```
# Assumes y >= 0, and returns x^y
def pow(x, y):
    prod = 1

    # Point A
    while y > 0:
        # Point B
        if y % 2 == 0:
            # Point C
            x = x * x
            y = y // 2
            # Point D
        else:
            # Point E
            prod = prod * x
            y -= 1
            # Point F
    # Point G
    return prod
```

Which of the following assertions are true at which point(s) in the code? Choose ALWAYS, NEVER, or SOMETIMES.

	`y > 0`	`y % 2 == 0`
Point A	SOMETIMES	SOMETIMES
Point B	ALWAYS	SOMETIMES
Point C	ALWAYS	ALWAYS
Point D	ALWAYS	SOMETIMES
Point E	ALWAYS	NEVER
Point F	SOMETIMES	ALWAYS
Point G	NEVER	ALWAYS

List functions

Function	**Description**
`append(x)`	Add an item to the end of the list. Equivalent to `a[len(a):] = [x]`.
`extend(L)`	Extend the list by appending all the items in the given list. Equivalent to `a[len(a):] = L`
`insert(i, x)`	Inserts an item at a given position. i is the index of the element before which to insert, so `a.insert(0, x)` inserts at the front of the list.
`remove(x)`	Removes the first item from the list whose value is *x*. Errs if there is no such item.
`pop(i)`	Removes the item at the given position in the list, and returns it. `a.pop()` removes and returns the last item in the list.
`clear()`	Remove all items from the list.
`index(x)`	Returns the index in the list of the first item whose value is *x*. Errs if there is no such item.
`count(x)`	Returns the number of times *x* appears in the list.
`sort()`	Sort the items of the list
`reverse()`	Reverses the elements of the list
`copy()`	Return a copy of the list.

Lists that change size

- Sometimes we don't know how big we want our list to be when our program starts
  - It can be useful to create an empty list and fill it up.

```
data = []
data.append("hello")
data.append("world")
print(data)                      # ['hello', 'world']
```

- How would we insert another word in the middle?

Exercise

Write a function called `remove_duplicates` that takes a **sorted** list of numbers and removes any duplicates. For example, if it is called on the following list:

```
    data = [-2, 1, 1, 3, 3, 3, 4, 5, 6, 78, 78, 79]
```

after the call the list should be

```
    data = [-2, 1, 3, 4, 5, 6, 78, 79]
```

Looping and removing

- When you loop through a list and remove elements you change the length of the list. This means you need to change your upper bound as you are looping.
  - **You must use a while loop when removing items from a list**
  - A `for i in range` loop won't work as it can't adjust when the length of the list changes
  - A `for num in data` loop won't work as it cannot alter the list.

Solution

```
def remove_duplicates(data):
    i = 0
    while i < len(data) - 1:
        if data[i] == data[i + 1]:
            data.pop(i)
        else:                # we don't want to move on
            i += 1           # to the next element if we
                             # remove as that will me we
                             # will skip the one that
                             # just moved back into the one
                             # we removed's place
```

Output to files

- Open a file in write or append mode
  - 'w' - write mode – replaces everything in the file
  - 'a' – append mode – adds to the bottom of the file preserving what is already in it

```
 name = open("filename", "w")     # write
name = open("filename", "a")     # append
```

Output to files

name.`write(`**str**`)` – writes the given string to the file

name.`close()` – closes file once writing is done

Example:

```
out = open("output.txt", "w")
out.write("Hello, world!\n")
out.write("How are you?")
out.close()

text = open("output.txt").read()   # Hello, world!\nHow are you?
```