CS 97 Discussion Section

Lun Liu, 1D

Important Exam Tidbits And Tips!

1) You're **not** responsible for "forall". Wash the word(s) away from your mind.
2) Loops: you are **not** responsible for nested loops. Any loop problem where you have to read or write code will not require you to use nested loops.
   - Why?
   - Iterative removeDuplicates is an upper bound on the difficulty you'd see on the exam
   - Also won't need to know anything more complicated than range(n)!
3) Don't try to memorize solutions--it rarely helps.
   - Two similar sounding problems may or may not have very different solutions.
   - Every example we give illustrates specific points: map can never see more than one item in a list or string, only the contents of a list are mutable, etc.
   - Understand the reasoning behind each solution and you can work your way back to the solution.

1. Use this when you only need to touch every element in a list or string once.

```
for x in my_list:
    doSomething(x)
```

2. Use it when you need to iterate a known number of times or when you need to touch every element in a list or string and also need the indexes:

```
for i in range(len(my_list)):
    doSomething(my_list[i])
```

3. Use it when you need to loop an unknown number of times, until a certain condition becomes true, or as a hail mary, which never happens to YOU because YOU studied

```
i = 0
while(i < len(my_list)):
    doSomething(my_list[i])
    i++
```

Loops Practice Problem

- How many times does the following loop execute on the inputs? What value does "i" have just before the function returns?
  - "bierhall "
  - "wonderbubbles"
  - "weirdo"
- There's a bug in this code; on certain inputs it will crash! Identify the bug and give an example of an input that will reveal the buggy behavior.

```
def contains_ie(str):
        found = False
        i = 0
        while not found and i < len(str):
                if str[i] == 'i' and str[i + 1] == 'e'
                        found =  True
                i += 1
        return found
```

More Practice!

- Now, rewrite the previous code *correctly* but you MUST use a for-range loop.
- True or False--if true, write it!
    - It is possible to write the previous function using map.
    - It is possible to write the previous function using a recursive function that does not require a helper function.
    - It is possible to write the previous function using some form of recursion (true if the previous one was true.)

```
def contains_ie(str):
        for i in range(len(str)):
                if str[i] == 'i':
                        if i < len(str)- 1 and str[i+1] == 'e':
                                return True
        return False
```

More Practice!

- True or False--if true, write it!
    - It is possible to write the previous function using a recursive function that does not require a helper function.

```
def contains_ie(s):
        l = list(s)
        if len(l) < 2:
                return False
        else:
                if l[0] == 'i' and l[1] == 'e':
                        return True
                else:
                        return contains_ie(s[1:])
```

Multiple Recursion and Auxiliary Functions

- Multiple recursion is when a function can call itself more than once during execution. There's no difference between this and normal recursion
- Same with helper functions--they don't do anything special, they just add an extra argument that would otherwise be destroyed in the recursive call.
- ex: use recursion to find out how many times the first letter of a string appears in that string?
- Ex: find all divisors of n

Mutation

First, questions?

- Assignment to an item (or += operator) in a list is the only way to update some value “**in place**”

```
>> a = "abcd"
>> a = "bcd"           # points to a new string
>> l1 = [1,2,3,4]
>> l2 = l1             # l1 and l2 points to the same list
# In your own words, how would you describe what happens
here?
>> l2[1] = 3
```

You cannot assign a character in string

```
>> s = "abcd"
>> s[2] = 'a'          #TypeError: 'str' object does not
support item assignment
```

Wrap Up

- Practice exams and practice questions
- Instructor Evaluations are due tomorrow morning!
    - You can fill one out for your TA anonymously as well
- Good job :)
- Thank you!
- bye :(