Lecture 9

Lists

```
my_list = [2, 3, 7.1, "hi"]
my_list
my_list[0]
my_list[3]
my_list[4]
my_list[100] # Guesses???
my_list[-2]  # Guesses???

# Lists are mutable!!
my_list[2] = -45.1
my_list
my_list.append(9) # we can add more stuff to the end!

list2 = [8, 0]
my_list + list2
my_list
list2
my_list = my_list + list2
my_list
my_list * 2
my_list
my_list *= 2

# Listception!
my_list[2] = ["dog", -8]
my_list
my_list[2]
my_list[2][0]

# Empty Lists!
p = []
p
p.append(4)
p

# We can compare lists!
l = [1, 2, 3, 4]
m = [1, 2, 3, 4]
l == m
n = [3, 2, 1, 4]
m == n
```

This means that we can use `assertEqual`! **But**, not `assertAlmostEqual`..., *so* for Lab 5, an `assertListAlmostEqual` is provided!

```
# More things!
n = [5, 10, 15, 20]
5 in n
8 in n
n.pop()
n.remove(5)
len(n)
```

1 List Reference Passing

However, lists are really odd/counter-intuitive in some ways when dealing with functions.

```
def main():
   lst = [1, 2, 3]
   y = func(lst)

   print("Last element of 'lst' is:", lst[-1])
   print("y is:", y)

def func(my_list):
   my_list.append(10)
   return 0

if __name__ == '__main__':
   main()
```

Tuples

```
# Tuples are a lot like lists
my_tuple = (3, 4)
my_tuple
my_tuple[0]

# BUT, they're immutable
my_tuple[0] = 7
my_tuple.append(12)
```

2 Maps

Let's write a function dealing with lists! Given a list, write a function that computes (and returns) a new list with double the entries in the original list. (Do first with while loop, then for, then list comprehension. Mention that list comprehensions are inspired by mathematical notation. Don't forget to write tests!) We call this kind of function a map. It's mapping each value in the old list to a corresponding value in the new list via some kind of function. We're "mapping" the doubling function onto the list.

3 Filters

Let's write more functions dealing with lists! Given a list, write a function that computes (and returns) a new list with only the even values from the original list. (Do first with while loop, then for, then list comprehension. Don't forget to write tests!) We call this kind of function a filter. It's filtering out all the undesired elements.