Even more on functions
Announcements

• Lab 5 is due Saturday
  - Ninja session tonight, 7-10pm
Today’s Plan

• Go over quiz
• Review Monday
• More mutability vs. reassignment examples
• functionWorksheet.py
def main():
    text = "abcde"
    space = " 
    for i in range(len(text)):
        spacing = i*space
        print(spacing + text[i])
main()
def main():
    text = "abcde"
    space = " "
    repeats = 3
    for i in range(len(text)):
        for j in range(repeats):
            num_spaces = i*repeats + j
            spacing = space*num_spaces
            print(spacing + text[i])

main()}
```python
def main():
    text = "abcde"
    space = " "
    repeats = 3
    for i in range(len(text)):
        for j in range(repeats):
            num_spaces = i*repeats + j
            spacing = space*num_spaces
            print(spacing + text[i])

main()
```

```
$ python q2.py
a
  a
  a
b
  b
  b
  b
  b
  b
c
  c
  c
  c
  c
d
  d
  d
  d
  d
e
  e
  e
  e
  e
  e
```
gpa = float(input("GPA: "))
act = int(input("ACT: "))
eligibility_score = gpa*10 + act

if eligibility_score >= 68:
scholarship = "Presidential"
elif eligibility_score >= 60:
scholarship = "Chancellor's"
elif eligibility_score >= 50:
scholarship = "Dean's"
else:
scholarship = "not eligible"

print("%.1f - %s" % (eligibility_score, scholarship))
def main():
    text = "car"
    accumulator = "e"
    for ch in text:
        accumulator = ch + accumulator + ch
    print(accumulator)

main()
• Lists and objects are mutable, strings are immutable.

• Some methods will mutate the lists and objects they are called on. Lists can also be mutated with item assignment. Item assignment won’t work for strings.
>>> S = "sharples"
>>> L = ["rock", "paper", "scissors"]
>>> S[1]
'h'
>>> L[2]
'scissors'
>>> L[2] = "shears"
>>> L
['rock', 'paper', 'shears']
>>> S[1] = 'm'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
• Lists and objects can be mutated when passed as arguments to a function.

• It's easy to mistake reassignment for mutation!
Reassigning vs. Mutating

```python
def foo3(L):
    L = L[1:]

def main():
    L = [1, 2, 3]
    foo3(L)
    print("The list is: %s" % L)

main()
```
Reassigning vs. Mutating

```python
def foo4(L):
    L = L[1:]
    return L

def main():
    L = [1, 2, 3]
    L = foo4(L)
    print("The list is: %s" % L)

main()
```
Reassigning vs. Mutating

def foo5(L):
    M = L
    M[0] = "one"

def main():
    L = [1, 2, 3]
    foo5(L)
    print("The list is: \%s\" % L)

main()
Objects are also mutable

```python
from graphics import *

def movePoint(p):
    p.move(5, 0)

def main():
    q = Point(0, 0)
    movePoint(q)
    print("q's x coordinate is: %d" % q.getX())

main()
```
Creating a copy

```python
from graphics import *

def movePoint2(p):
    q = p.clone()
    q.move(5, 0)

def main():
    q = Point(0, 0)
    movePoint2(q)
    print("q's x coordinate is: \%d" \ % q.getX())

main()
```