Homework #5 is due Friday Sep. 30.
Midterm #1 will be Monday Oct. 3. (7 p.m.)
No class on Monday,
Labs WILL be held all week.
Contact cs101admin@cs.illinois.edu for conflict exam.
<table>
<thead>
<tr>
<th>Code</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>AYA</td>
<td>114 David Kinley Hall</td>
</tr>
<tr>
<td>AYB</td>
<td>1404 Siebel Center</td>
</tr>
<tr>
<td>AYC</td>
<td>66 Main Library</td>
</tr>
<tr>
<td>AYD</td>
<td>141 Wohlers Hall</td>
</tr>
<tr>
<td>AYE</td>
<td>112 Gregory Hall</td>
</tr>
<tr>
<td>AYF</td>
<td></td>
</tr>
<tr>
<td>AYG</td>
<td></td>
</tr>
<tr>
<td>AYH</td>
<td></td>
</tr>
<tr>
<td>AYI</td>
<td></td>
</tr>
<tr>
<td>AYJ</td>
<td></td>
</tr>
<tr>
<td>AYK</td>
<td></td>
</tr>
<tr>
<td>AYL</td>
<td></td>
</tr>
<tr>
<td>AYM</td>
<td></td>
</tr>
<tr>
<td>AYN</td>
<td></td>
</tr>
<tr>
<td>AYO</td>
<td></td>
</tr>
<tr>
<td>AYP</td>
<td></td>
</tr>
<tr>
<td>AYQ</td>
<td></td>
</tr>
<tr>
<td>AYR</td>
<td></td>
</tr>
</tbody>
</table>
Midterm Instructions

- 30 multiple-choice questions
- 60 minutes
- Requires NetID and University I-Card.
- Exams are unique—omitting the exam code will dock one letter grade (10%).
- Will cover all material except dictionaries and file operations. Practice midterm is a good guide.
Given a variable \( t \) that is associated with a tuple whose elements are numbers, write some statements that use a while loop to count the number of times the first element of the tuple appears in the rest of the tuple, and associate that number with the variable repeats. Thus if the tuple contains \((1, 6, 5, 7, 1, 3, 4, 1)\), then repeats would be assigned the value 2 because after the first 1 there are two more 1s.
Given the lists, lst1 and lst2, create a new sorted list consisting of all the elements of lst1 that also appears in lst2. For example, if lst1 is [4, 3, 2, 6, 2] and lst2 is [1, 2, 4], then the new list would be [2, 2, 4]. Note that duplicate elements in lst1 that appear in lst2 are also duplicated in the new list. Associate the new list with the variable new_list, and don’t forget to sort the new list.
Warmup Quiz
Most scientists think of programming as a tax they have to pay in order to do science. (Hannay et al., 2009)

Do you agree?
   A  Yes
   B  No
a = [ [1,2,3],
     [4,5,6],
     [7,8,9] ]

How would you refer to the value 8?
A a[2][3]
B a[1][2]
C a[2,3]
D a[2][1] ★
x = [ 'a', 'b' ]
y = [ 'c', 'd' ]
def add_it( x,y ):
    y.append( x )
add_it( y,x )

What is the final value of x?
A [ 'a', 'b', 'c', 'd' ]
B [ 'a', 'b' ]
C [ 'a', 'b', [ 'c', 'd' ] ] *
D [ 'c', 'd', [ 'a', 'b' ] ]
E None
x = [ 'a', 'b' ]
y = [ 'c', 'd' ]
def add_it( x,y ):
    y.append( x )
add_it( y,x )

What is the final value of x?
A [ 'a', 'b', 'c', 'd' ]
B [ 'a', 'b' ]
C [ 'a', 'b', [ 'c', 'd' ] ]
D None
myfile = open('words.txt')
myfileasstring = myfile.read()
myfileaslist = myfile.readlines()
myfile.close()

What is the final value of myfileaslist?
A [ ]
B The contents of the file as a list of strings.
C None
Library Functions
Python has built-in functions:
  - abs, type, len
There are also specialized libraries:
  - math, string, itertools

# These are basically equivalent:
import math
math.sin( 5.4 * math.pi )

from math import sin, pi
sin( 5.4 * pi )
How do we index a list?
- lists and tuples are ordered.
- What else may make sense—how else could you organize data?
Example

- Models: Mustang, Viper, Corvette, Charger, 911
- Makes: Ford, Dodge, Chevrolet, Porsche

Dictionaries
The `dict` indexes data by any value (unordered).

Easy to think of as dictionary, but can use lots besides strings.

This container maps keys to values.

```
key           value
'911'          'Porsche'
cars['911'] =  'Porsche'
```
Dictionaries

cars = {}
cars[ 'Mustang' ] = 'Ford'
cars[ 'Viper' ] = 'Dodge'
cars[ 'Corvette' ] = 'Chevrolet'
cars[ 'Charger' ] = 'Dodge'
cars[ '911' ] = 'Porsche'
We create a dict as follows:

- opening brace {
- key : value pairs, separated by commas
- closing brace }

```python
model = {
    'Civic': 'Honda',
    'Mustang': 'Ford',
    'Model S': 'Tesla',
    'Model T': 'Ford'
}
```
d = { 'one':1, 'two':2, 'three':3 }
print( d['one'] )
d[ 'four' ] = 4
del d[ 'four' ]
'five' in d
for key in d: # no guarantee on order
    print( key, d[key] )
d.keys()
d.values()
d = { 'a':2, 'c':3, 'b':1 }

x = d[ 'a' ] + d[ 'c' ]

What is the final value of x?
A 4
B 'ac'
C '5'
D 5
Example

d = { }
words = [ 'red', 'orange', 'yellow' ]
for word in words:
    d[ word ] = words.index( word )

What is the final value of d?
A { 'red':3, 'orange':6, 'yellow':6 }
B { 'red':0, 'orange':2, 'yellow':2 }
C None
D {'orange': 1, 'red': 0, 'yellow': 2} ✭
Dictionaries can encode/decode data, or translate from one representation to another.

```python
x = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
y = 'BCDEFGHIJKLMNOPQRSTUVWXYZA'
e = {}
for i in range(len(x)):
    e[x[i]] = y[i]
encoded = 
for c in 'HELLO':
    encoded += e[c]
```

How would you reverse (decode) this?
```python
x = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
y = 'BCDEFGHIJKLMNOPQRSTUVWXYZA'
d = {}
for i in range(len(x)):
    d[y[i]] = x[i]
decoded = ''
for c in encoded:
    decoded += d[c]
```
Exercise

- Encode all of the words in a file using a Caesar cipher.
- Decode all of the words in the file.
Dictionaries can also function as accumulators.

```python
x = 'ABBACAB'
d = {}
for c in x:
    if c not in d:
        d[c] = 0
        d[c] += 1
```

How would you reverse (decode) this?
Exercise

- Count category frequencies in Jeopardy questions.
- Count bigram frequencies in Jeopardy clues.
We can link data based on a common field.

```python
zipcode = { 'Bill': 60644,
            'Jill': 41073,
            'Tony': 63103 }

city = { 60644: 'Chicago',
         41073: 'Cincinnati',
         63103: 'St. Louis' }

for name in zipcode:
    print( name, city[ zipcode[ name ] ] )
```
Mutable Arguments
Exercise: mutability

```
x = [ 3,2,1 ]
y = x
y.sort()
x.append( 0 )
```

What is the final value of `x`?

A [ 3,2,1 ]
B [ 1,2,3 ]
C [ 1,2,3,0 ] *
D [ 0,1,2,3 ]
Mutable arguments

- Mutability causes lists to work differently in functions.
- Lists used as arguments can be changed by the function.
- This is very useful!

```python
def fun(q):
    q.append(3)

a = []
for i in range(3):
    fun(a)
print(a)
```
def readfile(fname,a):
    for line in open(fname):
        a.append(line)

all_lines = []
readfile( 'file1.txt', all_lines )
readfile( 'file2.txt', all_lines )
def readfile(fname,a):
    for line in open(fname):
        a.append(line)

all_lines = []
for f in open("filenames.txt"): readfile(f,all_lines)
What if we want a copy of a list (not an alias)?
Slice everything!

```python
x = [ 3, 2, 1 ]
y = x[ : ]
y.sort()
print( x )
```
x = [ 1,2,3 ]
y = x[ : ]
y.append( 4 )
print( x == y )
is tests identity

```python
a = [1,2,3]
b = a
c = a[:]

b is a  # True
c is a  # False
```
Reminders
Homework #5 is due Friday Sep. 30.

Midterm #1 will be Monday Oct. 3. (7 p.m.)
No class on Monday,
Labs WILL be held all week.
Contact cs101admin@cs.illinois.edu for conflict exam.

Extra credit up through Friday Sep. 30.