Administrivia
Homework #5 is due Friday Sep. 30.
Vote!
Midterm #1 will be Monday Oct. 3. (evening)
No class on Monday,
Labs WILL be held all week.
Extra credit available on website this week (feedback).
Fancy Slicing
Fancy slicing for containers

```python
a = list(range(10))  # also for strings
```
a = list(range(10))  # also for strings
# [ 0,1,2,3,4,5,6,7,8,9 ]

# from beginning to index 4 (exc.)
# a[:4]
# from index 6 to end
# a[6::]
# copy a list
# a[:]
# from index 1 to -1 by twos
# a[1:-1:2]
# odd indices only
# a[1::2]
# even indices only
# a[::-1]
# reverse a list (!)
Fancy slicing for containers

```python
a = list(range(10))  # also for strings
# [ 0,1,2,3,4,5,6,7,8,9 ]

a[:4]  

a[:4]
```
a = list(range(10))  # also for strings %
#  [ 0,1,2,3,4,5,6,7,8,9 ] %
a[:4]  # from beginning to index 4 (exc.)
a[6:]  #
Fancy slicing for containers

```python
a = list(range(10))  # also for strings
# [ 0,1,2,3,4,5,6,7,8,9 ]

a[:4]  # from beginning to index 4 (exc.)
a[6:]  # from index 6 to end
a[:]  # copy a list
```

Fancy Slicing
Fancy slicing for containers

```python
a = list(range(10))  # also for strings
#    [ 0,1,2,3,4,5,6,7,8,9 ]

a[:4]  # from beginning to index 4 (exc.)
a[6:]   # from index 6 to end
a[:]    # copy a list
a[1:-1:2]  # reverse a list (!)
```
Fancy slicing for containers

```python
a = list(range(10))  # also for strings %
#  [ 0,1,2,3,4,5,6,7,8,9 ] %

a[:4]  # from beginning to index 4 (exc.)
a[6:]  # from index 6 to end
a[:]  # copy a list
a[1:-1:2]  # from index 1 to -1 by twos
a[1::2]  #
```
Fancy slicing for containers

```python
a = list(range(10)) # also for strings
# [ 0,1,2,3,4,5,6,7,8,9 ]

a[:4]  # from beginning to index 4 (exc.)
a[6:]   # from index 6 to end
a[:]    # copy a list
a[1:-1:2]  # from index 1 to -1 by twos
a[1::2]  # odd indices only
a[::2]   #
```
Fancy slicing for containers

```python
a = list(range(10))  # also for strings
#    [ 0,1,2,3,4,5,6,7,8,9 ]

a[:4]  # from beginning to index 4 (exc.)
a[6:]  # from index 6 to end
a[:]  # copy a list
a[1:-1:2]  # from index 1 to -1 by twos
a[1::2]  # odd indices only
a[::2]  # even indices only
a[:::-1]  #
```
a = list(range(10))  # also for strings 
#   [ 0,1,2,3,4,5,6,7,8,9 ]  

a[:4]  # from beginning to index 4 (exc.)
a[6:]  # from index 6 to end
a[:]  # copy a list
a[1:-1:2]  # from index 1 to -1 by twos
a[1::2]  # odd indices only
a[::2]  # even indices only
a[:::-1]  # reverse a list (!)
Loop Management
i = 0
while i < 10:
    i += 1
    if i == 4:
        break  # terminate the loop
print(i)
i = 0
while i < 10:
    i += 1
    if i == 4:
        continue  # skip ONLY this loop
print(i)
Warmup Quiz
# given a string ‘s’
t = 
for c in s:
    if c not in 'aeiou':
        t += c

What does this program do with string s?
A Counts the vowels.
B Removes the vowels. ⋆
C Counts the consonants.
D Removes the consonants.
```python
x = 0
for i in [1, 4950, 99, 100][0:-1]:
    x = i

What is the final value of x?
A 0
B 99 ★
C 100
D 4950
```
Multidimensional Indexing
Just as we can nest control structures, we can nest container values.

\[ a = \left[ \left[ 1, 2 \right], \left[ 3, 4 \right] \right] \]

What does this look like to you?
Multidimensional indexing

- Access member values of a nested container by coordinates:

```python
a = [ [ 1, 2 ], [ 3, 4 ] ]
a[0] #? %
```
Access member values of a nested container by coordinates:

```python
a = [[1, 2], [3, 4]]
a[0]  #? %
a[0][0]  #? %
```
Access member values of a nested container by coordinates:

```python
a = [ [ 1, 2 ], [ 3, 4 ] ]
a[0]  #? %
a[0][0]  #? %
```

- Python orders by \( (row, column) \)–that is, the first number selects the row and the second selects the column in that row.
- Side effect: easy to select “rows”, hard to select “columns”!
a = [ [1,2,3], [4,5,6], [7,8,9] ]

How would you refer to the value 6?
A  $a[2][3]$  
B  $a[1][2]$  
C  $a[2,3]$  
D  $a[2][1]$
What is the difference between `list.extend` and `list.append`?

```python
a = [1, 2]
b = [3, 4]
a.extend(b)  #?
a.append(b)  #?
```
File Operations
- It is uncommon to generate the source data in the same program as one uses it.
- What is a file?
Punch card deck—5 MB
Secondary storage
File structure
File structure
File structure

- 0011011010110110
- '69' (str)
- 14006 (int)
- [54, 182] (list)

...
File structure

File Operations
File structure

0011011010110110

'6¶' (str)
14006 (int)
[54,182] (list)

001101101011011011001110000000010...
- file is an iterable type created by the function `open`.
- `open` accepts one option: the file name as a string (for now).
- Each item in the iterable is a string representing one line in the file.

```python
myfile = open( 'wordlist.txt' )
for line in myfile:
    print( line )
```
total = 0
for line in open('numbers.txt '):
    total += int(line)
print(total)
```python
for w in open( 'words.txt' ):
    vowels = 0
    for c in w.lower():
        if c in 'aeiou':
            vowels += 1
    print( w.strip() + ' %i' % vowels )
```
If we open a file, we should close it as well.

- close protects the file against data loss.

```python
myfile = open( 'words.txt' )
for line in myfile:
    print( line )
myfile.close() # process responsibly
```
The default way of opening a file is to 'r'ead it.

We can extract all of the data from the file at once with:
- `read`, which returns a string

```python
myfile = open( 'words.txt' )
mydata = myfile.read()
myfile.close()
print( mydata )
```
The default way of opening a file is to read it.

We can extract all of the data from the file at once with:
- `read`, which returns a string
- `readlines`, which returns a list of strings

```python
myfile = open( 'words.txt' )
mydata = myfile.readlines()
myfile.close()
for line in mydata:
    print( line )
```
File modes

- We can also write to a file, but we need to open it differently.
- We can specify a file mode when we open a file:
  - 'r' to read a file's data (default)
  - 'w' to write data to a file

```python
myfile = open( 'words.txt','w' )
myfile.write( 'Hello, this is a test.' )
myfile.close() # ultra-important now!
```

- Other modes available but not important for 101.
assert

- Python keyword requiring that a boolean expression evaluate to True.
- Useful in testing code.

```
assert 2+2=5  # test math 
```
Python keyword requiring that a boolean expression evaluate to True.
Useful in testing code.

```python
assert 2+2==5  # test math
from math import sin
assert sin(0.0) == 0.0
```
Python keyword requiring that a boolean expression evaluate to True.
Useful in testing code.

```python
assert 2+2==5  # test math
from math import sin
assert sin(0.0) == 0.0

(More on this later, won’t be on test.)
```
Reminders
Homework #5 is due Friday Sep. 30.
Midterm #1 will be Monday Oct. 3. (evening)
No class on Monday,
Labs WILL be held all week.
Extra credit available on website this week (feedback).
Write a loop that reads positive integers from standard input, printing out those values that are greater than 100, each on a separate line. The loop terminates when it reads an integer that is not positive.