Homework #3 is due Wed Oct. 26.
Homework #4 is due Wed Nov. 4.
Midterm #1 will be on the day of the 12th lecture (Nov. 7 Monday), covering through Lecture #11. (evening)
s = 'ABCDEFGH'
t = ''
i = 0
while i < 8:
    t = t + s[i+1]
    i += 2

What is the final value of t?
A  "ACEG"
B  "BDFH"
C  "ABCDEF"
D  "ABEF"
Question #2

s = '0123456789'
t = ''
i = 0
while i < 5:
    if (i%2) == 1:
        t = t + s[i-1]
    if (i%2) == 0:
        t = t + s[i+1]
i = i + 1

What is the final value of t?
A "92143"
B "103254"
C "10325"
D "921436"
E None (loop doesn’t terminate)
Question #3

\[
z = [ 1.2, 0.6, 0.5, 0.3 ]
z = z.sort()
\]

What is the final value of \( z[1] \)?

A 0.6  
B 0.5  
C None  
D None of the above.
What are two changes this code needs to be executable?

```python
if x < 1.5:
    x = x + 1
if x == (1.5 or 2.0):
    x = x - 1
```
What are two changes this code needs to be executable?

```python
if x < 1.5:
    x = x + 1
if x == 1.5 or x == 2.0:
    x = x - 1
```
Example: if statement

```python
ans = input( "Enter a number:" )
if float(ans) < 0:
    print( "The number was negative." )
```
Control flow represents actual sequence of lines executed by processor.

Conditional execution lets you execute (or not) a block of code based on logical comparison.
Branched control flow

- We often need to make decisions with several options.
- *Branched conditional execution* lets you execute one of several blocks of code.
def absolute(x):
    if x >= 0:
        return x
    else:
        return -x
We create an if/else statement as follows:
- the keyword `if`
- a logical comparison (results in `bool`)
- a block of code
- the keyword `else`
- a different block of code
Sequence operators

- These produce Boolean output.
  - `in`  Is one string inside of the other?
  - `not in`  Is one string not inside of the other?
def fun(s):
    return s.isalpha() and 'a' in s

x = fun( "sam" ) and fun( "AS" )

What is the value of x?
A True
B False
Sometimes we need to make more than one decision.

We can *nest* blocks.

```python
word = input( 'Enter a Scrabble word: ' )
if not word.isalpha():
    print( 'There are only letters in Scrabble!' )
else:
    if not word.isupper():  # why not
        #`word.islower()`?
        word = word.upper()
    print( 'You entered %s.' % word )
```
Nesting

Conditional Execution

```
not alphabetical?

- True: "There are only letters in Scrabble."
- False:

  not upper-case?

  - True: .upper()
  - False: "You entered %s."
```
Exercise: Nesting

Conditional Execution
def evenpos(x):
    if x >= 0:
        if (x%2) == 0:
            return x
        else:
            return x + 1
    else:
        if (x%2) == 0:
            return -x
        else:
            return (-x) + 1
Sometimes we need to select among many choices.
Example

```python
if day == 1:
    print("Sunday")
else:
    if day == 2:
        print("Monday")
    else:
        if day == 3:
            print("Tuesday")
        else:
            if day == 4:
                print("Wednesday")
            else:
                if day == 5:
                    print("Thursday")
                else:
                    if day == 6:
                        print("Friday")
                    else:
                        if day == 7:
                            print("Saturday")
```
Example

```python
if day == 1:
    print("Sunday")
elif day == 2:
    print("Monday")
elif day == 3:
    print("Tuesday")
elif day == 4:
    print("Wednesday")
elif day == 5:
    print("Thursday")
elif day == 6:
    print("Friday")
elif day == 7:
    print("Saturday")
else:
    print("That is not a valid day.")
```
We create an if/elif/else statement as follows:
- the keyword if
- a logical comparison (results in bool)
- a block of code
- the keyword elif
- a logical comparison (results in bool)
- a block of code
- the keyword else
- a different block of code
Iteration Redux
Example

colors = [ 'red', 'yellow', 'blue',
          'jale', 'ulfire' ]
for color in colors:
    print( color )
A for loop requires:
- the keyword for
- a loop variable
- the keyword in
- a set of values
- a block of code

for loops iterate over iterable types one at a time.
s = 'abcdefg'
t = ''
for c in s:
    t = c + t

What is the value of t?
A 'abcdefg'
B 'gfedcba'
C 'a'
D 'g'
Write a function to sum all of the digits in a number. 
I.e.,

\[ 12145 \rightarrow 1 + 2 + 1 + 4 + 5 \rightarrow 13 \]
def sum_digits( n ):
    result = 0
    for letter in str( n ) :
        result += int( letter )
    return result
for i in range(10):
    print(i ** 2)
The `range` function returns an iterator containing integers.

- `range` can be cast as a list.

**Two arguments:**
- (optional) the starting value of the range (inclusive)
- the ending value of the range (exclusive)
Reminders
Reminders

- Homework #4 is due Wed Nov. 4.
- Midterm #1 will be on the day of the 12th lecture (Nov. 7 Monday), covering through Lecture #11. (evening)