

CSC108H Lecture 13

Dan Zingaro

October 10, 2012

Motivating While Loops

- ▶ So far, we know about one type of loop: `for` loop
 - ▶ It requires a sequence (e.g. a `range` sequence or a string) to loop over
- ▶ Another type of loop is the `while` loop: it repeatedly tests a condition, executing the entire body of the loop if it is `True`, and terminating the loop if it is `False`
 - ▶ Useful when there is no sequence to loop over
 - ▶ Commonly used when we don't know how many times the loop will run

User-Input Loop (inputloop.py)

How many times will this loop run? We don't know — so it's a while loop!

```
def get_yn ():  
    response = ''  
    while response != 'y' and response != 'n':  
        response = input('Type y or n to continue: ')  
    return response
```

ConcepTest

What is printed by the following code? (Output is on one line to save space.)

```
x = 6
while x > 4:
    print(x)
    x = x - 1
```

- ▶ A. 6 5
- ▶ B. 6 5 4
- ▶ C. 5 4
- ▶ D. 5 4 3
- ▶ E. 6 5 4 3

ConcepTest

What is printed by the following code? (Output is on one line to save space.)

```
x = 6
while x > 4:
    x = x - 1
    print(x)
```

- ▶ A. 6 5
- ▶ B. 6 5 4
- ▶ C. 5 4
- ▶ D. 5 4 3
- ▶ E. 6 5 4 3

For vs. While

Use for when:

- ▶ You want to loop through an entire sequence without stopping
- ▶ The number of iterations does not depend on user input
- ▶ The increment to the loop variable is the same on every iteration

```
s = 'abc'
for count in range(len(s)):
    print('Index {0} is {1}'.format(count, s[count]))
```

```
count = 0
while count < len(s):
    print('Index {0} is {1}'.format(count, s[count]))
    count += 1
```

ConceptTest

```
valid = False
while not valid:
    s = input ("Enter a password: ")
    valid = len(s) == 5 and s[:2] == 'xy'
```

Which of the following passwords gets us out of the loop?

- ▶ A. xyz
- ▶ B. abcxy
- ▶ C. xyabc
- ▶ D. More than one of the above passwords get us out of the loop
- ▶ E. None; the loop never executes and no passwords are obtained

True and break

- ▶ There are several ways to write a loop whose body is required to run at least once
 1. Artificially make the condition true before the loop starts (like `inputloop.py`)
 2. Copy some loop code above the loop to make the condition true
 3. Use `True` as the condition and `break` to exit the loop
- ▶ `break` causes immediate termination of the loop
- ▶ `break` can make code difficult to read if used improperly
- ▶ We frequently do not allow `break` on exams or assignments

ConceptTest

A valid password is one that is length 5 and starts with xy. Such passwords should get us out of the loop. Which of these does this?

► A.

```
while True:
    s = input ("Enter a password: ")
    if len(s) == 5 and s[:2] == 'xy':
        break
```

► B.

```
s = input ("Enter a password: ")
while len(s) == 5 and s[:2] == 'xy':
    s = input ("Enter a password: ")
```

► C. Both are correct

► D. None is correct

ConcepTest

What is the output of this code? (Output is on one line here to save space.)

```
n = 3
while n > 0:
    if n == 5:
        n = -99
    print(n)
    n = n + 1
```

- ▶ A. 3 4
- ▶ B. 3 4 5
- ▶ C. 3 4 -99
- ▶ D. 3 4 5 -99