

Introduction to Computing Loops

Andrew Block

Boolean Statements

- ▶ Booleans have two values: True or False
- ▶ Boolean statements test whether a condition is true or false
- ▶ Testing Equality
 - == - Equals
 - != - Not Equals
 - > - Greater Than
 - < - Less Than
 - >= - Greater Than or Equals
 - <= - Less Than or Equals



Python *if* Statement

- ▶ Syntax:

```
if <expression>:  
    <body>
```
- ▶ Example:

```
if 5 > 4:  
    print("5 is greater than 4")
```
- ▶ Code within the body of the *if* statement will only be executed if the condition is True



For Loops

- ▶ The Python *for* loop is its most powerful
 - The most commonly used Python loop
 - It traverses an "iterable", setting the loop variable to each successive value
- ▶ Syntax

for variable in iterable:
 <body of loop>



What is an Iterable?

- ▶ Definition: Any data structure that supports simple element traversal
 - Sequences (strings, lists, tuples, range)
 - Sequence like objects
 - Iterators, lines of a file, keys of a dictionary, generators, sets
- ▶ We will be only using the range() function



The range() function

- ▶ range(): A helper for Python *for* loops
 - Iterating rather than simple counting
 - Will count up to but not include the end index
- ▶ Syntax: will use Integer arguments
 - `range(start, end[, step])`
 - `range([start,] end)`
- ▶ Examples
 - `range(0, 10)`
 - `range(0, 10, 2)`
 - `range(10)` ← Assumes the starting value is 0



Simple For loop range() Example

▶ for i in range(0, 10):
 print(i) Declare i to hold the current value of
 the iteration of range

▶ Will print out:

```
0
1
2
3
4
5
6
7
8
9
```



Stepping range() Example

▶ for i in range(0, 10, 2):
 print(i)

▶ Will print out:

```
0
2
4
6
8
```



Python *while* Loops

- ▶ *While* loops are conditionals, like looping *if*
 - Body of while loops will continue to run until the condition is no longer satisfied
 - The test for condition occurs before each loop iteration

▶ Uses

- Counting loops
- Infinite Loops

▶ Syntax

```
while <conditional_expression>:  
    <Body>
```



Simple while Loop Example

```
i = 0
while i < 5:
    print(i)
    i += 1
```

▶ Will print out:

```
0
1
2
3
4
```



Sentinel Loops

- ▶ Used for repetitively reading data
 - Loop will continue to run until a specified value is entered which will stop the loop
- ▶ Common uses
 - Grade books
 - Creating lists of names



Sentinel Loop Example

```
grade = input("Enter grade, -1 to end: ")
grade = int(grade)
while grade != -1:
    grade = input("Enter grade, -1 to end: ")
    grade = int(grade)
```

- ▶ Program will continue to ask for grades to be inputted until a grade of -1 is entered



In Class Exercise (Part 1)

- ▶ You are creating a program that will take in scores based on a recent exam (out of 100 points) and print out the average score.
 - Prompt the user for input for the number of scores they wish to enter.
 - Keep track of the total number of inputted scores and total value of inputted score
 - Create a for loop using the range() function that will loop through the list of scores prompting for each individual students' score
 - Add each students inputted score to the total value of inputted scores
 - After the for loop, compute the average of the scores and print the value out



In Class Exercise (Part 2)

- ▶ You are creating a multiplication table generator (from 0-10). The program is intended to assist elementary school children so the program will run continuously until they are ready to exit.
 - Prompt the user for the multiplication table they would like to generate. If they input -1, exit the program.
 - You will need to use int(variable) to convert the text from the user from a string to an integer
 - Create a sentinel while loop that checks the value of the multiplication table inputted by the user and if it does not equal -1, will enter the loop



In Class Exercise (Part 2 cont.)

- ▶ Within the while loop, once again create a for loop using the range() function that will loop through the table from 0-10 .
- ▶ Print out each value
- ▶ When all the values have printed out, ask the user for the next table they would like to generate.
 - The loop will restart unless they entered -1
 - You must once again make sure to convert your inputted value to a string to an integer



In Class Exercise (Part 2 Output)

Enter Multiplication Table (-1 to exit): 5

0
5
10
15
20
25
30
35
40
45
50

Enter Multiplication Table (-1 to exit):



Submission

- ▶ One member of the group must submit both Python code files (.py files) to the Assignment on BlackBoard
- ▶ Make sure all names of the group appear as comments in the program


