

Topic 5: Repetition

Are you saying that I am redundant?

That I repeat myself? That I say the
same thing over and over again?

Textbook

- Strongly Recommended Exercises
 - The Python Workbook, 3rd Edition: 72, 77, 84, and 92; or
 - The Python Workbook, 2nd Edition: 66, 71, 77, and 83
- Recommended Exercises
 - The Python Workbook, 3rd Edition: 70, 76, 87, 89, 90, and 93; or
 - The Python Workbook, 2nd Edition: 64, 67, 79, 81, 82, and 84
 - Course Website: Loop Exercises
- Recommended Readings
 - The Python Workbook, 2nd or 3rd Edition: Chapter 3

Repetition

- So far, we have learned...
 - How to use variables
 - Read values from the user
 - Make decisions
 - Compute a result
 - Output a result
- What if we want to perform a task several times?

Types of Loops

- Python includes two types of loops
 - While Loops
 - For Loops

While Loops

- A while loop executes a statement as long as a condition is true
 - `while condition:`
 `statement(s)`
 - Statement may be simple or compound
 - Typically compound
 - Needs to change one of the values being tested in the condition

Example

- How do we compute the average of several numbers?

Example

While Loop Review

- Executes as long as some condition is true
- A pre-tested loop
 - loop condition is tested before the loop executes the first time
- General form:

```
while condition :  
    statement(s)
```


Loop Terminology

- Body of the Loop:
 - simple or compound statement that is repeated
- Loop Condition:
 - a Boolean expression
 - tested to determine if the loop will continue executing

Loop Terminology

- Pre-tested Loop:
 - Any loop where the loop condition is checked before the loop executes the first time
- Post-tested Loop:
 - Any loop where the condition is not checked until the loop has executed once
- Infinite Loop:
 - A loop that never terminates

Another Example

- Using a while loop, compute n factorial

Common Errors

- Initialization Errors
- Termination Errors
- Other Logic Errors

Tracing

- Tracing code:
 - Examine each statement in sequence
 - Perform whatever tasks the statement requires, recording values of interest
 - Usually requires that the value of each variable is recorded
 - Result of tracing could be the value of one or more variables, or the output generated

Another Factorial?

```
n = int(input("Enter a value for n: "))

result = 1
term = 0

while term <= n:
    term = term + 1
    result = result * term

print("n! is", result)
```

While Loop Review

- Executes as long as some condition is True
 - Pre-tested
 - Executes zero or more times
 - Generally
 - need to initialize variables used in conditions before the loop
 - need to change the value of at least one of these variables in the loop body

For Loop

- A counting loop
 - Typically used when we know how many times we need to perform a task in advance
 - A pre-tested loop
 - General form:

```
for variable in collection:  
    body
```


Example

- Use a for loop to display the values from 3 up to and including 10
 - For loop assigns a value from a list into a variable at the beginning of each loop iteration
 - Construct a list with the range function

How Does a For Loop Work?

- Collection is examined
 - If every value has already been processed
 - the loop body does not execute
 - control passes to the first statement after the loop's body
 - If unprocessed values remain
 - variable is assigned the next item in the list
 - the body of the loop executes
 - control returns to the top of the loop
 - list is examined to see if the body should run again

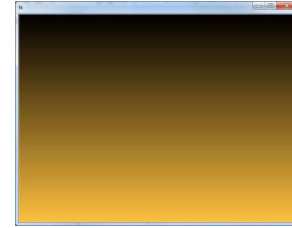
Example

- Rewrite the factorial program using a for loop

Step Values

- Range is flexible
 - With one parameter
 - Counts from 0 to the number provided - 1
 - With two parameters
 - Counts from the first number to the second number (exclusive), increasing by one each time
 - Returns an empty range if the second number is less than or equal to the first
 - With three parameters
 - Counts from the first number to the second (exclusive), increasing by the third

Example



- Create a program that generates a smooth color gradient from black to 255 192 64

For Loops vs. While Loops

- What kind of loop would you use if:
 - You know how many times the loop will execute before the loop's body runs for the first time
 - You want to loop until some event occurs
- Is it possible that the body of a for loop will never execute?
- Is it possible that the body of a while loop will never execute?

Nested Loops

- The body of a loop can be
 - A simple statement
 - A compound statement
- The body of the loop can contain another loop

Nested Loops

- Trace the output from the following program:

```
for i in range(1, 6):  
    print(i)  
    j = i  
    while j < 5:  
        print(j)  
        j = j + 1
```


Nested Loop Example

- Convert an image from color to grayscale

Break and Continue

- Allow a loop iteration to end prematurely
- `break`
 - Entire loop ends immediately
 - Execution continues at the first statement after the loop body
- `continue`
 - Current iteration ends immediately
 - Execution returns to the top of the loop
 - `while` loop: The loop's condition is evaluated
 - `for` loop: The next item in the list is processed

Bringing It All Together

- Write a simple number guessing game
 - The computer will randomly choose a number between 1 and 100
 - The user will be asked to guess a number
 - The computer will let the user know if the guess was too high or too low
 - Goal: guess the correct number in as few guesses as possible

Bringing It All Together

- Improving our program:
 - Should try and protect the user from themselves
 - Don't let them guess a number smaller than the lowest remaining value
 - Don't let them guess a number larger than the largest remaining value
 - Don't count an out of range value as a guess

Wrapping Up

- Two types of loops available
 - While loops
 - For loops
- Both types are pre-tested
 - Will execute zero or more times
- Loops can be nested and combined with other types of statements

Where Are We Going?

- Our number guessing game had a problem
 - Many lines of code in one place
 - Starting to become more difficult to enhance and debug
 - Solution?
 - Use functions to break our solution into pieces that each perform a specific task