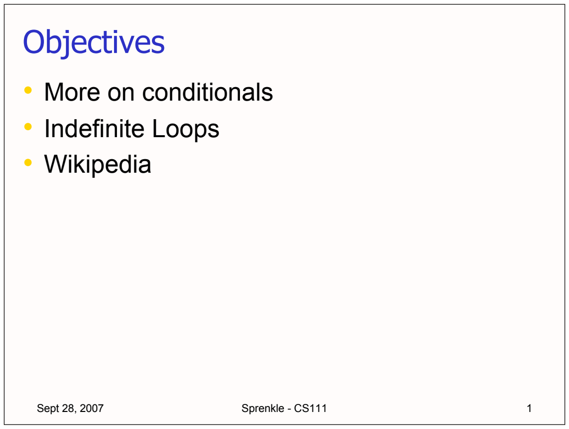
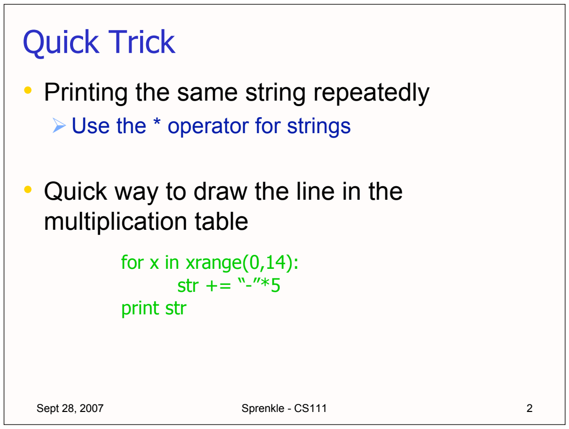


Objectives

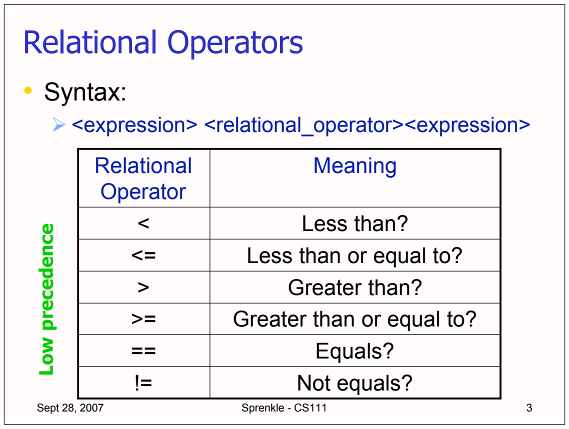
- More on conditionals
- Indefinite Loops
- Wikipedia



Quick Trick

- Printing the same string repeatedly
  - Use the * operator for strings
- Quick way to draw the line in the multiplication table

```
for x in xrange(0,14):
          str += "-"*5
print str
```

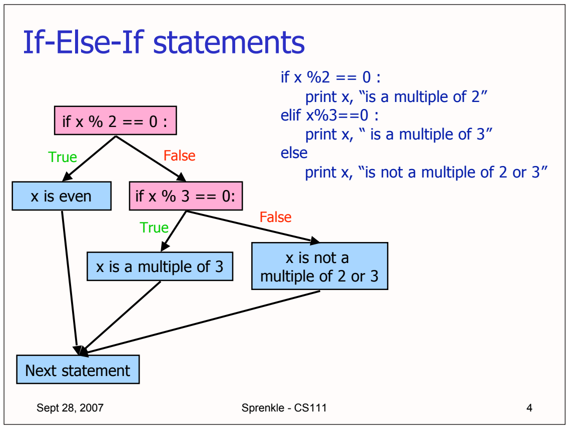


Relational Operators

- Syntax:
  - <expression> <relational_operator><expression>

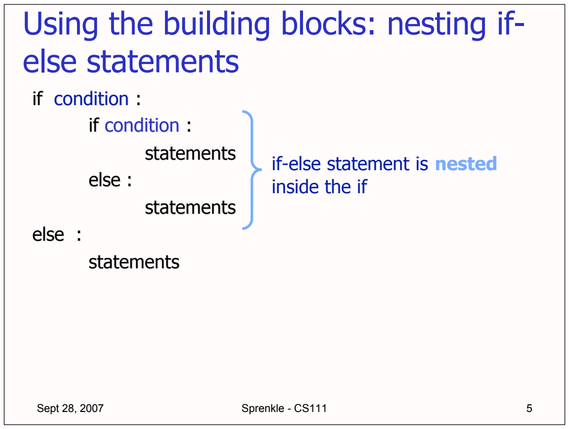
Relational Operator	Meaning
<	Less than?
<=	Less than or equal to?
>	Greater than?
>=	Greater than or equal to?
==	Equals?
!=	Not equals?

Low precedence



If-Else-If statements

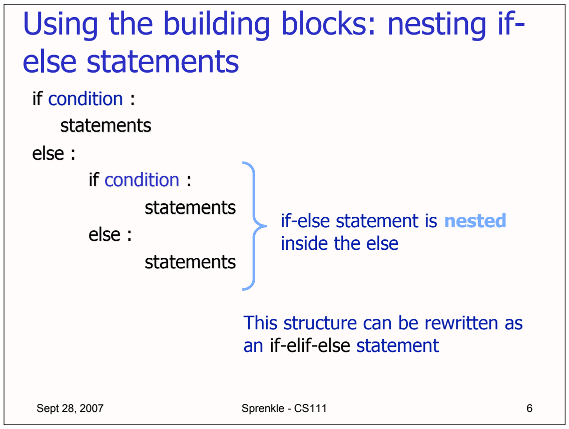
```
if x %2 == 0 :
     print x, "is a multiple of 2"
elif x%3==0 :
     print x, " is a multiple of 3"
else
     print x, "is not a multiple of 2 or 3"
```



Using the building blocks: nesting if-else statements

```
if  condition :
          if condition :
                    statements
          else :
                    statements
else  :
          statements
```

if-else statement is **nested** inside the if



Using the building blocks: nesting if-else statements

```
if condition :
     statements
else :
          if condition :
                    statements
          else :
                    statements
```

if-else statement is **nested** inside the else

This structure can be rewritten as an if-elif-else statement

Modify: Check for Other Bad Input

```
print "This program determines your birth year"
print "given your age and current year"
print
age = input("Enter your age >> ")
if age > 110:
      print "Don't be ridiculous, you can't be that old."
else:
      currentYear = input("Enter the current year >> ")
      birthyear = currentYear - age
      print
      print "You were either born in", birthyear, "or", birthyear-1
```

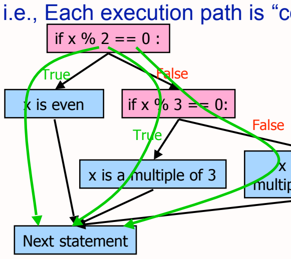
Practice: Numeric to Letter Grade

- Determine if a numeric grade is a letter grade (A, B, C, D, or F)

Numeric Grade	Letter Grade
90 and above	A
80 to below 90	B
70 to below 80	C
60 to below 70	D
Below 60	F

Testing with If Statements

- Make sure have test cases that execute each branch in control flow diagram
  - i.e., Each execution path is “covered”



Three execution paths

More Complex Conditions

- Boolean
  - Two logical values: True and False
- Combine conditions with Boolean operators
  - **and** – True only if **both** operands are True
  - **or** – True if **at least** one operand is True
  - **not** – True if the operand is not True
- English examples
  - If it is raining **and** it is cold
  - If it is Saturday **or** it is Sunday

Truth Tables

operands

A	B	A **and** B	A **or** B	**not** A	**not** B	**not** A **and** B	A **or** **not** B
T	T						
T	F						
F	T						
F	F						

Truth Tables

operands

A	B	A **and** B	A **or** B	**not** A	**not** B	**not** A **and** B	A **or** **not** B
T	T	T	T	F	F	F	T
T	F	F	T	F	T	F	T
F	T	F	T	T	F	T	F
F	F	F	F	T	T	F	T

What is the output?

```
x = 2
y = 3
z = 4

b = x==2
c = not b
d = (y<4) and (z<3)
d = (y<4) or (z<3)
d = not d

print b, c, d
```

eval_cond.py

Practice: Numeric Grade Input Range

- Enforce that user must input a numeric grade between 0 and 100
  - Using **and**
  - Using **or**

Short-circuit Evaluation

- Don't necessarily need to evaluate all expressions in a compound expression
- For A and B
  - If A is false, compound expression is false
- For A or B
  - If A is true, compound expression is true
- No need to evaluate B
  - Put more important/limiting expression first
  - Example:

```
if count > 0 and sum/count > 10:
        do something
```

Indefinite Loops

- For loops are ***definite*** loops
  - Execute a fixed number of times
- Indefinite loops: keeps iterating until certain conditions are met
  - Depending on condition, no guarantee in advance of how many times the loop body will be executed

While Loop Syntax

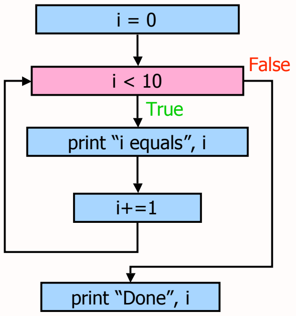
```
while condition :
    statements
```

← body of while loop

- Like a looped **if** statement
  - Execute statements only when condition is true

While Loop

```
i = 0
while i < 10 :
    print "i equals ", i
    i+=1
print "Done", i
```



Questions:

- How many times will "i" get printed out?
- How many times is the condition evaluated?
- What is the value of i after the loop?

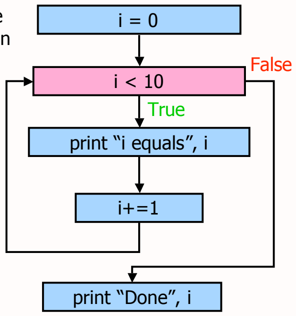
whilevsfor.py

While Loop

Initialize i before using in condition

```
i = 0
while i < 10 :
    print "i equals ", i
    i+=1
print "Done", i
```

Questions:
• How many times will "i" get printed out?
• How many times is the condition evaluated?
• What is the value of i after the loop?



whilevsfor.py

What Will This Loop Do?

```
count = 1
while count > 0:
   print count
   count += 1
```

Infinite Loop

- Condition will never be false so keeps executing
- To stop an executing program in Linux use
  - Control-C

CS Issue: Citing Wikipedia

- What were the major issues that the article brought up?
- How does this article relate to this course?

Jen
Maya
Will R
Matt

Keith
Cathy
Oliver

Alysen
Laura
Will L

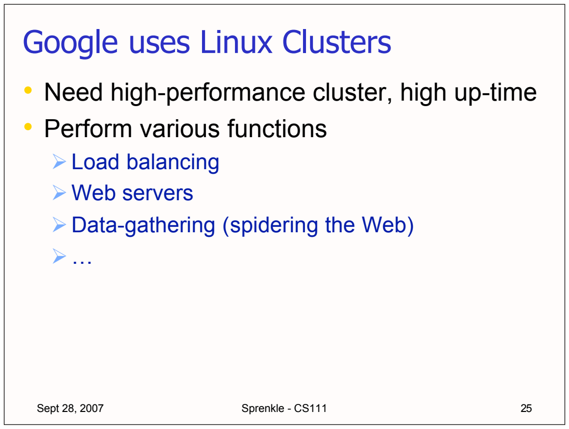
Google Algorithms

- Google founders: Larry Page and Sergey Brin
  - Stanford CS graduate students
  - http://infolab.stanford.edu/~backrub/google.html
- Based on PageRank
  - <a href="webpage">description</a>
  - Pages linked with same description most get higher rank
  - Rank pages linked from highly-ranked pages higher

WikiScanner

- Virgil Griffith
  - Grad student, trouble maker :)
  - Also discovered vulnerabilities in Blackboard
- Combines information from two databases
  - IP addresses of anonymous edits
  - IP addresses belong to what companies

resolve_ips.sh



Google uses Linux Clusters

- Need high-performance cluster, high up-time
- Perform various functions
  - Load balancing
  - Web servers
  - Data-gathering (spidering the Web)
  - …