Objectives

- More on conditionals
- Indefinite Loops

Review: if, if-else Statements

```python
if x < 0 :
    x *= -1
    print "abs=", x
else :
    abs = x
    print "abs=", abs
```

If statement
```
if x < 0 :
    x *= -1
    print "abs=", x
```

If-else statement
```
if x < 0 :
    abs = x * -1
else :
    abs = x
    print "abs=", abs
```

Review: Relational Operators

- Syntax:

  ```python
  <expr> <relational_operator> <expr>
  ```

- Low precedence

<table>
<thead>
<tr>
<th>Relational Operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>Less than?</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to?</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than?</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to?</td>
</tr>
<tr>
<td>==</td>
<td>Equals?</td>
</tr>
<tr>
<td>!=</td>
<td>Not equals?</td>
</tr>
</tbody>
</table>

Practice: Draw the Flow Chart

```python
print "This program determines your birth year"
print "given your age and current year"
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 "You were either born in", birthyear, "or", birthyear-1
```

sys module

- Has useful "system" functions
- Use the `exit([status])` function
  - Exits the whole program
  - If status is empty, defaults to 0
  - Status of 0 means success
  - Other values are various failures
- Another example of changing control flow

Example Use of sys module

```python
import sys
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."
    sys.exit(1)
```

# input is reasonable...
currentYear = input("Enter the current year >> ")
birthyear = currentYear - age
print "You were either born in", birthyear, "or", birthyear-1
Practice: Speeding Ticket Fines

- Any speed clocked over the limit results in a fine of at least $50, plus $5 for each mph over the limit, plus a penalty of $200 for any speed over 90 mph.
- Let's write a program that will take as input the speed limit and the clocked speed. It will then print that the clocked speed was under the limit or it will print the appropriate fine.

Enhanced Lottery Game

- Check if user's pick matches the number you generated
- When using strings, needed the user's input to be in the same format
- The debate:
  - User's input as a string or as a number --> create the lottery number a string or number

Syntax of if statement: Multi-Way Decision

```python
if condition:
    <then-body1>
elif condition:
    <then-body2>
elif condition:
    <then-body3>
else:
    <default-body>
```

English Example:

```python
if it is Saturday:
    I wake up at 10 a.m.
elif it is Sunday:
    I wake up at 9 a.m.
else:
    I wake up at 7 a.m.
```

If-Else-If statements

```python
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

```python
if condition:
    if condition:
        statements
    else:
        statements
else:
    statements
```
Using the building blocks: nesting if-else statements

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

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"
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 "You were either born in", birthyear, "or",
    print birthyear-1
```

Practice: Numeric to Letter Grade

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

<table>
<thead>
<tr>
<th>Numeric Grade</th>
<th>Letter Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>90 and above</td>
<td>A</td>
</tr>
<tr>
<td>80 to below 90</td>
<td>B</td>
</tr>
<tr>
<td>70 to below 80</td>
<td>C</td>
</tr>
<tr>
<td>60 to below 70</td>
<td>D</td>
</tr>
<tr>
<td>Below 60</td>
<td>F</td>
</tr>
</tbody>
</table>

Testing with If Statements

• Make sure have test cases that execute each branch in control flow diagram

<table>
<thead>
<tr>
<th>i.e., Each execution path is &quot;covered&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>if $x % 2 == 0$</td>
</tr>
<tr>
<td>x is even</td>
</tr>
<tr>
<td>if $x % 2 == 0$</td>
</tr>
<tr>
<td>x is not a multiple of 2 or 3</td>
</tr>
<tr>
<td>Three execution paths</td>
</tr>
<tr>
<td>Next statement</td>
</tr>
<tr>
<td>x is a multiple of 3</td>
</tr>
</tbody>
</table>

Convert the Code to if-elif-else

```
clockspeed = input("Enter the clocked speed: ")
speedlimit = input("Enter the speed limit: ")

if clockspeed <= speedlimit:
    print "Continue safe driving practices"
else:
    diff = clockspeed - speedlimit
    fine = 50 + 5 * diff
    if clockspeed > 90:
        print "Slow down! You've been fined "+str(fine)+"."
```

Convert the Code to if-elif-else

```
clockspeed = input("Enter the clocked speed: ")
speedlimit = input("Enter the speed limit: ")

if clockspeed <= speedlimit:
    print "Continue safe driving practices"
elif clockspeed <= 90:
    diff = clockspeed - speedlimit
    fine = 50 + 5 * diff
    print "Slow down! You've been fined "+str(fine)+"."
else:
    diff = clockspeed - speedlimit
    fine = 250 + 5 * diff
    print "Slow down! You've been fined "+str(fine)+"."
```
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

<table>
<thead>
<tr>
<th>operands</th>
<th>A</th>
<th>B</th>
<th>A and B</th>
<th>A or B</th>
<th>not A</th>
<th>not B</th>
<th>not A and B</th>
<th>not B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

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
  - In Python, we can’t (always) write a condition like 0 <= num_grade <= 100, so we need to break it into two conditions
  - Write an appropriate condition for this check on the numeric grade
  - Using **and**
    - Using **or**

- Enforce that user must input a numeric grade between 0 and 100
  - Using **and**
    - if num_grade >= 0 and num_grade <= 100:
      - computation
  - Else:
    - print error message
  - Using **or**
    - if num_grade < 0 or num_grade > 100:
      - print error message
    - Else:
      - computation
Short-circuit Evaluation

• Don’t necessarily need to evaluate all expressions in a compound expression

• A and B
  ➢ If A is False, compound expression is false

• 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