Syntax

```
if condition:
  block

if condition1:
  block
elif condition2:
  block
elif ...

if condition1:
  block
elif condition2:
  block
elif ...
else:
  block
```

Lab 4: Decision Statements in Python

Exercise 1: Fix your calculator

Use `calc.py` (Lab 2, not Lab 3 where you put it in a function) as a base. Save it as `bettercalc.py`.

Make two additions to this program: a menu to choose a mathematical operation and a division-by-zero check.

Menu

When the program starts, print a menu allowing the user to type in a number 1 through 4 (matching the following table) to choose the operation they wish to use:

Number	Operation
1	Addition (+)
2	Subtraction (-)
3	Multiplication (*)
4	Division (/)

Based on what number the user chooses, perform *that* operation. If the user chooses 2, only subtract the two numbers. Do not add, multiply, and divide the numbers.

Division-by-Zero Check

In normal arithmetic, division by zero is undefined. If you try to divide by 0 in Python, you receive an error that looks like this:

```
ZeroDivisionError: integer division or modulo by zero
```

A program crashing on a user with a `ZeroDivisionError` is obviously not a good thing. What would be better is catching that situation before doing the division and stopping it. Add a decision statement to your program to not allow division by 0. If the user chooses 0 as their second number, simply display an error message and don't perform any mathematical operation.

Exercise 2: Guessing Game

Create a number guessing game. Start with the following code:

```
import random

guessed = False
number = random.randrange(1, 10)
```

Note: You will probably have to type this in manually as copying and pasting from these PDFs rarely works as planned.

This code will use a random number library, a set of functions, types, and constants. The `randrange()` function will generate a random number between two numbers, in this case 1 and 10 (inclusive).

Continue on with that code to write a program that gives a user *up to* three chances to guess the number.

1. The *guessed* variable is to specify whether or not the user has guessed the correct value. You do not want to give them the second and third chances if they guessed it on the first try.
2. After the user has guessed, if the guess is wrong, give the user a hint. Tell them the number is higher than their guess or is lower than their guess. (Use the relational operators to check that.) For this, you will need to do nested if statements.

This problem is not at all difficult, but it does require more thought than previous programs you have written. Feel free to work in groups of 2 or 3 on this problem (do Exercise 1 alone). **If you do work with others**, place a comment near the top of the file listing the names of people who worked on it and their login name. Each person should submit the file. Example:

```
#!/usr/bin/env python
#
# Lab 4, Exercise 2 (Guessing Game)
#
# Ross Nelson <rnelson>, Chris McMacken <cmcmacke>, Lisa
# Hemingson <lhemingso>

import random

guessed = False
number = random.randrange(1, 10)
.
.
.
```

Handing in

Submit your two Python scripts on the `cse101l` handin. They are due by 23:59 on Friday, February 13, 2009.

Grading

Area	Points
Programs work and meet requirements:	15
Correct filenames:	5
Total:	20