CMSC 201 Fall 2015
Lab 04 – Conditionals

Assignment: Lab 04 – Conditionals
Due Date: During discussion, September 21st through September 24th
Value: 1% of final grade

Part 1: Logic

Mastery of logic is essential to understanding *conditional statements*. It is used in pretty much any program that you will ever write. *Comparisons* are the heart of logical statements. When we write programs we often want to compare variables and/or raw data, testing to see if that comparison holds true or false. Then, we alter the way our program operates depending on the result of that comparison.

We can make those comparisons using any of the following *relational operators*, which compare two variables and/or raw data:

- `<` (less than)
- `>` (greater than)
- `<=` (less than or equal to)
- `>=` (greater than or equal to)
- `==` (equal to)
- `!=` (not equal to)

For example:

```
num = 500                 # Set the value of num
num < 1000                # This evaluates to True
1456 >= num               # This evaluates to True
300 != 300                # This evaluates to False
"hello" == "goodbye"      # This evaluates to False
```

Notice how you can mix variables and raw data and still make valid comparisons. There are a few more operators, but we will learn about them later in the course.

You can also chain two comparison statements together using:

- **`and`**
  - Both comparisons must be **`True`** for this to evaluate to **`True`**
- **`or`**
  - At least one comparison must be **`True`** for this to evaluate to **`True`**

For example:

```
num = 500
(500 <= num) and (num <= 1000)                # True
("hello" == "hello") and ("dog" == "cat")     # False
num > 487 or num <= 342                       # True
"hello" == "hello" or  "dog" == "cat"         # True
```

You do not have to use parentheses around a comparison statement, but it can have the benefit of making your code clearer and easier to read.

A third logical operator available to you is called **`not.`** This can operate on one logical statement, and it flips the truth value of that statement. So, a logical statement that is **`True`** will be flipped to **`False`**, and a logical statement that is **`False`** will be flipped to **`True`**.

For example:

```
isDog = True
not isDog               # False
(4 > 5)                 # False
not (4 > 5)             # True
5 > 4                   # True
not (5 > 4)             # False
not ("dog" == "cat")    # True
```

Part 2: Conditional Statements

Being able to make comparisons is only the first part of conditional statements. We also need a structure to execute different code based on the value of a comparison. There are three such structures available: "**`if`**", "**`if-else`**", and "**`if-elif-else`**". These structures combine with one or more logical statements to form a *conditional statement*.

A basic "if" statement looks like this:

```
if num >= 0:
    print("The number", num, "is positive.")
```

The print statement is only executed if the value of the variable **`num`** is larger than 0. Whatever is "inside" the **`if`** statement (meaning one indentation level in) will be executed if the logical statement used evaluates to **`True`**.

What if you want something different to happen if the logical statement is not **`True`**? To do this, just use an "**`else`**" statement right after an "**`if`**" like so:

```
if num >= 0:
    print("The number", num, "is positive.")
else:
    print("The number", num, "is negative.")
```

What if there are several related logical statements you need to test? Simply use an "el**`if`**" in sequence with an "**`if`**."

Important: The very first logical statement that evaluates to **`True`** will have its associated code executed, and *everything else will be skipped over*. Also, you must have an "**`if`**" statement before you use any "**`elif`**" statements or an "**`else`**" statement.

```
if num > 0:
    print("The number", num, "is positive.")
elif num == 0:
    print("The number is zero.")
else:
    print("The number", num, "is negative.")
```

Part 3A: How cold is it outside?

This is the first of two programs that you will write for this lab.
See Part 3C for instructions on creating this program.

To practice using **`if`** statements, we are going to make a program where the user enters the temperature in Fahrenheit, and the program prints out a description of the weather for them.

Ask the user to input the temperature and store it to a variable. (Don't forget to *cast* it to an **`int`**!)

Using if statements, check the value of the input and print these sentences to the screen:

- If the number is less than 25, print: "It's freezing outside."
- If the number is between 25 and 49, print: "It's a bit chilly, remember to bundle up."
- If the number is between 50 and 79, print: "The weather is wonderful!"
- If the number is between 80 and 100, print: "It's pretty hot outside."
- Otherwise, print: "It is way too hot."

With input 10, the output should looks something like this:

```
bash-4.1$ python temperature.py
Please enter a temperature in Fahrenheit: 10
It's freezing outside.
```

Part 3B: Pets

This is the second of two programs you will write for this lab.
See Part 3C for instructions on creating this program.

Next we will practice comparing strings. First, we will request an input from the user. If the input is "dog" or "cat" exactly, we want to tell the user that it is a pet.

Using if statements, check if the input says "dog" or "cat" in lowercase.

- If the input is "dog" or "cat" print: "This is a pet."
- Otherwise, print: "This is not a pet."

(Python is case-sensitive, so "cat" is not the same as "Cat" or "CAT" to it.)
Hint: Don't forget that the Boolean operators "**`and`**" and "**`or`**" exist!

With input "frog", the output should looks something like this:

```
bash-4.1$ python pets.py
Please enter the animal you have: frog
This is not a pet.
```

Part 3C: Writing the programs

After logging into GL, navigate to the **`Labs`** folder inside your **`201`** folder. Create a folder there called **`lab4`**, and go inside the newly created **`lab4`** directory.

```
linux2[1]% cd 201
linux2[2]% cd Labs
linux2[3]% pwd
/afs/umbc.edu/users/k/k/k38/home/201/Labs
linux2[4]% mkdir lab4
linux2[5]% cd lab4
linux2[6]% pwd
/afs/umbc.edu/users/k/k/k38/home/201/Labs/lab4
linux2[7]%
```

Now you are going to create **two python files**, one for each part of this assignment.

To open the first file for editing, type

`emacs temperature.py &`

and hit enter. (The ampersand at the end of the line is important – without it, your terminal will "freeze" until you close the emacs window. **Do not include the ampersand if you are not on a lab computer.**)

The first thing you should do in your new file is create and fill out the comment header block at the top of your file. Here is a template:

```
# File:          temperature.py
# Author:        YOUR NAME
# Date:          TODAY'S DATE
# Section:       YOUR SECTION NUMBER
# E-mail:        USERNAME@umbc.edu
# Description:   YOUR DESCRIPTION GOES HERE AND HERE
#                YOUR DESCRIPTION CONTINUED SOME MORE
```

Once you've completed the comment header block, you can write the code for the program described in Part 3A.

Once you are done with temperature.py, follow the same instructions to create a file called pets.py, in which you will code the program described in Part 3B. (Don't forget the comment header block!)

To check your programs, first enable Python 3, then run and test each of them with the **`python`** command:

```
linux2[7]% /usr/bin/scl enable python33 bash
bash-4.1$ python temperature.py
Please enter a temperature: 40
It's a bit chilly, remember to bundle up.
bash-4.1$ python pets.py
Please enter the animal you have: dog
This is a pet.
bash-4.1$
```

Part 4: Completing Your Lab

Since this is an in-person lab, you do not need to use the **`submit`** command to complete your lab. Instead, raise your hand to let your TA know that you are finished.

They will come over and check your work – they may ask you to run your program for them, and they may also want to see your code. Once they've checked your work, they'll give you a score for the lab, and you are free to leave.

IMPORTANT: If you leave the lab without the TA checking your work, you will receive a **zero** for this week's lab. Make sure you have been given a grade before you leave!