I. CONDITIONALS

A. Boolean Expressions

A **Boolean expression** can be true or false. The operator == compares two values and returns **True** if they are equal and **False** if they or not:

```
>>> 4==4
True
>>> 8==3
False
>>> 'hello'=='hello'
True
>>> 'hello'=='goodbye'
False
>>> int(3.14)==3
True
>>> 3.00==3
True
>>>
```

As the following IDLE conversation illustrates, **True** and **False** are NOT STRINGS, but rather a special type called **bool**.

```
>>> type(True)
<type 'bool'>
>>> type(False)
<type 'bool'>
>>> type(false)

Traceback (most recent call last):
  File "<pyshell#131>", line 1, in <module>
    type(false)
NameError: name 'false' is not defined
```

B. Relational Operators

There are several relational operators. Note that >= (<=) is acceptable for greater (less) than or equal to , but => (or =<) are not.

```
>>> x=3.0
>>> y=4.0
>>> x!=y
True
>>> x!=y # x is not equal to y
True
>>> x>y # x is greater than y
False
>>> x<y # x is less than y
True
>>> x>=y # x is greater than or equal to y
False
>>> x<=y # x is less than or equal to y
True
```

C. Logical Operators

There are three logical operators: **and, or** and **not** that act as shown in the following IDLE conversation:

```
>>> x=5.0
>>> y=50.0
>>> z=100.0
>>> x>1.0 and y>40.0 # note that BOTH relations are true
True
>>> x==5.0 and y<40.0 # note that ONLY ONE relation is true
False
>>> x==5.0 or y<40.0 # note that ONLY ONE relation is true
True
>>> x==4.0 or z==200.0 # note that NO relations are true
False
>>> not(x==5.0) # not negates a boolean expression
False
>>> not(z>200.0) # NOT negates a boolean expression
True
>>>
```

D. Modulus Operator

The **modulus operator**, which in python is the percent sign(%) yields the remainder when dividing integers. You can check whether one number is divisible by another number (is the remainder zero?) or you can determine the rightmost digit or two digits of a number.

```
>>> quotient=14/3
>>> print quotient
4
>>> remainder=14%3
>>> print remainder
2
>>> last_digit=734%10
>>> print last_digit
4
>>> last_two_digits=734%100
>>> print last_two_digits
34
```

E. Conditional Execution: if, elif, else

Sometimes you want the computer to do one thing or another thing depending on whether some condition is true. This " forking" (execute one of two or more lines of code) in the order of execution of a set of statements is achieved by the use of **if, else** and or **elif** statements. The first example shows a situation where there are two possibilities. Note the similarity to a function definition (use of comma and indentation). Try writing the following program using **gedit** and running it in script mode. The program asks you to input an integer and then executes one of two statement lists depending on the boolean **if(x%2==0)** is **True** (if option) or **False** (else option).

```
x=int(raw_input('enter an integer \n'))

if x%2==0:
  print x
  print 'x is even'
else:
  print x
  print 'x is odd'
```

A second example - called a **chained conditional** - has more than two possibilities. Again try writing the following program using **gedit** and running it in script mode. Note that there is no limit on the number of **elif** - which stands for *else if* statements

```
x=float(raw_input('enter a number (x) \n'))
y=float(raw_input('enter another number (y) \n'))

if x<y:
  print 'x  is less than y'
elif x>y:
  print 'x is greater than y'
else:
  print 'x and y are equal'
```

II. EXERCISES

1. *Source: Think Python by Allen B. Downey*

   (a) If you are given three sticks, you may or not be able to arrange them in a triangle. For example, if one stick is 12 cm long and the other two are 1 cm long, it is clear that you cannot form a triangle. A simple test to see if it is possible to form a triangle is as follows: *If any of the three lengths is greater than the sum of the other two, then you cannot form a triangle. Otherwise, you can.* Write a function named "is_triangle" that takes three floats as arguments, and prints either, "Yes, it is possible to form a triangle with these sticks.", or "No, it is not possible to form a triangle with these sticks."

   (b) Write a function that prompts the user to input three stick lengths, converts them to floats, and uses "is_triangle" to check whether the sticks with the given lengths can form a triangle.

2. Write a program that allows a user to input the real coefficients a, b, c of a general parabola $y(x) = ax^2 + bx + c$. The program will

   (a) determine whether there is one root, two real roots or two complex roots and then print out the roots in a statement

      i. 'There is one real root. y=0 when x= ' (whatever the answer is)

      ii. 'There are real roots. y=0 when x= ' (whatever the answer is) 'and x ='(whatever the answer is)

iii. ' There are two complex roots. y=0 when x= ' (whatever the answer is in the form $x + iy$) 'and x ='(whatever the answer is)