Before you turn this problem in, make sure everything runs as expected. First, **restart the kernel** (in the menubar, select Kernel→Restart) and then **run all cells** (in the menubar, select Cell→Run All).

Make sure you fill in any place that says `YOUR CODE HERE` or "YOUR ANSWER HERE", as well as your name and collaborators below:

In [ ]:
```
NAME = "Sharjil Mohsin"
COLLABORATORS = "N/A"
```

Before you turn this problem in, make sure everything runs as expected. First, **restart the kernel** (in the menubar, select Kernel→Restart) and then **run all cells** (in the menubar, select Cell→Run All).

Make sure you fill in any place that says `YOUR CODE HERE` or "YOUR ANSWER HERE", as well as your name and collaborators below:

In [ ]:
```
NAME = "Sharjil Mohsin"
COLLABORATORS = "N/A"
```

Intro Lab

Intro Lab Assignment Instructions:

Your task today will be to convert the provided (broken) code to a working copy by adding instructions and adjusting values throughout the provided code. Please read through this entire document carefully for specific instructions. As this is probably your first time using Jupyter, you can go to Help toolbar at the top of the page and click on the first option, User Interface Tour. This will give you a quick introduction to some of the features available to you on Jupyter.

Requirements:

The requirements section of any software documention or assignment will tell you exactly the desired behaviour of your program. This should be very specific and very precise.

Today's Requirements:

You will create a minimalistic flash card program. Upon running the program, the following will be printed to the Python shell in the Python cell below: **

```
* Flash Cards *
********************************
The user will be prompted with two integers and an operator of +, - or * (
where * is multiplication). For example:
4
+ 3
---------
=
The user then enters their guess for the solution. This guess will be compared
to the actual solution. The computer then responds by printing “Correct!” to
the screen if the solutions match, otherwise “Incorrect”:
4
+ 3
---------
= 7
Correct!
The user will again be prompted with another expression to solve. This
happens a total of five times. After all five expressions have been solved by the
user the following summary screen is printed:
2
********************************
* Game Over *
********************************
Your score:
4 / 5
In 8.4353252 seconds.
********************************
```

Design Instructions:

Design instructions are meant to fill in the gaps that are left by the requirements section. They sometimes provide definitions for things referred to in the requirements section. They may also provide hints or examples of similar programs which should help you in your apprach to your specific program. Thus, you should always read the entire document provided before beginning for a full understanding of your program.

Today’s Design Instructions:

In your provided code below, look for comments corresponding to the following numbers for appropriate changes to make to the file. Comments will look something like this:

1. Comments are regions of code that python ignores, but these are useful for developers to communicate about the code. Comment lines typically start with a #. Include here at the top of the file, in comments: your firstName, lastName, macID (NOT your student number) and the date.
2. The import statement will take code that has been already been built by others so that you can use it. Here you can see the time library has already been imported. In addition to the time library we also need the random library. Please import the random library.
3. runs is a variable that stores the amount of times an expression will be presented to the user and tests if it is correct. Currently it is set to hold a value of 1. Change this value so that the user will be presented with an expression five times.
4. The startup message for your program was mentioned above in the requirements section. Use the print statement (or many such statements) and replace the currently existing string with another to match the one specified in the requirements section of this document.
5. The line arg1 = random.randint(1,5) generates a random number between 1 and 5. Alter this line so that it generates a random number between 1 and 10. This number is used later for our equations.
6. arg1 stores the value we intend to add, multiply or subtract in our expression. Write a (VERY) similar line to store another random number between 1 through 10 in a variable called arg2.
7. This next piece of code always assigns the equation to use a plus sign in our expression. But we already have a function above that generates an operator for us. The function is called makeOp(). Replace the string "+" (including its quotes) with makeOp(), the function call. This will store a multiplication, addition or subtraction sign in the variable op.
8. It looks like the developer of this program has a function to print the expression for the user, but they accidentally left it in a commented region of code. Remove the comment symbol so the program prints the expression and the user knows what expression to solve for!
9. The stopTimer() function takes a variable name as input that holds the start time. The function returns the totalTime taken since the start time, this will give us the total time it takes for the user to complete all the flash cards. In the following line, startVariableNameGoesHere is not a valid variable name because it doesn't have a value: totalTime = stopTimer(startVariableNameGoesHere) Replace this variable with the variable that is storing the starting time of our program.
10. It looks like the developer forgot to include some information about the game being over. Using the print statement (or many such statements), tell the user:

    -The game is over, according to the requirements from above. -What their score was -How long it took for them to complete the game run.

Make sure the the output looks exactly like what is specified in the requirements section of this document.

In [8]:
```
# 1.
# Sharjil Mohsin
# mohsis2
# January 4, 2018

# 2.
```

```
import time, random


#############################################################################
# The following are function defintions -- These have already been built for you
# You do not need to update anything inside these functions
# You can look at comments to understand what each of these functions do
#############################################################################

def solveExpression( e ):
        ''' This function returns the solution of a string expression '''
        return eval( e )

def makeExpression( i, j, op ):
        ''' This function returns a string representing the expression '''
        return str(i) + op + str(j)

def makeOp():
        ''' This function returns a character representing math operators: + (addition), - (subtraction) o
r x (multiplication) '''
        index = random.randint(0,2)
        x = ['+', '-', '*']
        return x[index]

def checkGuess( guess, soln, score ):
        ''' This function returns "True" if guess == soln; and false otherwise '''
        if guess == soln:
                score +=1
                print("Correct!\n")
        else:
                print("Incorrect.\n")
        return score


def printExpression( arg1, arg2, op):
        ''' This function prints the expression where arg1 & arg2 are integers and op is a string operato
r: "+", "-" or "*" '''
        print ("\t", arg1)
        print (op, "\t", arg2)
        print ("---------")

def startTimer():
        ''' This function returns the current time '''
```

```
        return time.time()

def stopTimer( startTime ):
        ''' This function takes a start time and returns the current time minus the start time'''
        return time.time() - startTime


#############################################################################
# The next non-comment line is where our program actually begins executing
# You are only required to change the following lines for the program to work.
#############################################################################

# score is a vaiable that will store number of correct guesses
# we initialize this to zero
score = 0

# 3.
# variable "runs" holds the # of times the expression will be presented to user
runs = 5

# The startTimer() function returns the current time in seconds
# we store that value in the variable startTime
startTime = startTimer()

# 4.
print("""**********************************
* Flash Cards *
**********************************""")

# Everything inside this for statement ( we can tell by the indentation )
# will be executed "runs" times
for i in range( runs ):
    # 5.
    arg1 = random.randint(1,10)

    # 6.
    arg2 = random.randint(1,10)

    # 7.
    op = makeOp()

    #makes expression of type: string
    exp = makeExpression( arg1, arg2, op)
```

```
    # 8.
    printExpression(arg1, arg2, op)

    #stores solution in variable: soln
    soln = solveExpression( exp )

    #gets guess from user
    guess = int(input("=\t"))

    # the checkGuess() function checks the user's guess and
    # updates the score
    # the score holds the amount of correct guesses
    score = checkGuess( guess, soln, score )


# note that indentation is different here. This is normal.
# this signifies what is done after the for loop has completed.

# 9.
totalTime = stopTimer(startTime)
# totalTime variable now holds the amount of time since the game started

# 10.
print("""*********************************
* Game Over *
*********************************""")
print("Your score:")
print(score, "/ 5")
print("In", totalTime, "seconds.")
print("*********************************")
```

```
**********************************
* Flash Cards *
**********************************
         6
*        7
---------
=       5
Incorrect.

         5
*        10
---------
=       5
Incorrect.

         3
*        8
---------
=       5
Incorrect.

         5
+        1
---------
=       5
Incorrect.

         8
+        2
---------
=       5
Incorrect.

**********************************
* Game Over *
**********************************
Your score:
0 / 5
In 4.218592405319214 seconds.
**********************************
```

Design Questions:

This section of your assignment will be filled with questions that you must answer in the markdown cell below. In order to edit a markdown cell, double-click on the cell and then you should be able to type text normally. If you want to see some of the formatting options available in markdown, you can look up a Jupyter Markdown Cheat Sheet.

Today's Question:

Debugging is a process where you take code that doesn't work in some way and alter it so that it behaves as you want it to! Describe how what you've changed in the flash.py program today is debugging. What kinds of problems did you encounter and what was your solution?

Today, we changed some of the codes that was already in the program using the instructions given above. Some of the changes we made in the program would be considered debugging because the codes that was already there was giving us wrong results earlier and it was our task to fix this. An example of this is when we did step 9, where the code totalTime = stopTimer(startVariableNameGoesHere) is an invalid code because the variable startVariableNameGoesHere doesn't cotain the start time. In order for the variable totalTime to hold the amount of time since the game started, I had to change the variable startVariableNameGoesHere to startTime since the variable startTime contained the amount of time since the game had started. Since we altered the code so that it now contains the total elapsed time of the game, this is an example of debugging.

Practice:

The practice section is something you should attempt in preparation for your upcoming labs. They are unmarked, but you should definitely attempt to try accomplish these tasks. Just thinking about it is not enough! Practice, Practice, Practice. For this week's practice, take some time to play around with all the concepts on Jupyter to get familiar with it, since you will be using it for the rest of the term. These are very simple things, but there are many things to remember when programming which will make debugging and python programming much easier if you are already familiar with them. There is a Python cell below for you to play around with as well, if you want to get more familiar with some of the concepts introduced in today's lab.

In [ ]:
```
# YOUR CODE HERE
raise NotImplementedError()
```

In [ ]: