CSC280 Homework 1 (Total points 60+20= 80)
Topic covered: Basic data types, assignments, functions, Control flow, While/For loops, Nested Loops.

Due Date: September 28th, Monday. 12:59pm.
No late assignments are acceptable!
The homework is not easy! Please start early!!!!! There is no way you can finish this the night before!

Hand-in Procedure (Read carefully, fail to deliver your code in the correct way might result in credit loss) .

1. Save
   Save your solution as YourLastName_YourFirstName_Problem1.py, as YourLastName_YourFirstName_Problem2.py, as YourLastName_YourFirstName_Problem3.py…etc

2. Header Information
   Homework is expected to be completed by yourself. But you can discuss high-level details with **anyone you like**. No copy of lines of code is allowed.

   At the start of each file, in a comment, write down your name, the number of hours you roughly spend on the assignment, and the person you have discussed your code with.

   For example:
   # Problem Set 1
   # Name: Zach Brown
   # Time Spent: 3:30
   # Last Modified: September 17, 2015
3. Upload all of you files as a .zip file to blackboard (no .rar file can be read by your instructor!
4. Make sure all of your code can be run in Command Line Terminal by typing Python yourcode.py

Grading scheme:

To receive full credits, your code MUST generate correct results (I/O) for any test case the grader will use. Please make sure your code compile and run. Try to run ask your classmate to run your code if you want to for a test. Broken code (that results in error) will receive at least 40% credit reduction. Remember when you define a function, it outputs nothing if you don't call the function.

Here is some tips of how to write functions:

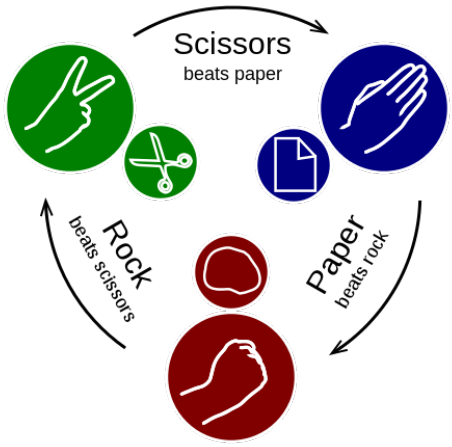
1. Functions are each separated by two blank lines.
2. Lines are short enough (80 chars) that horizontal scrolling is not necessary.
3. The specifications for all of the functions are complete **and are docstrings**.

4. Specifications are immediately after the function header and indented.

Some of these problems have solutions (not always correct) in the Internet. YOU ABSOLUTELY CAN NOT COPY the solution. If caught, you will be reported to violations of academic integrity. This is serious! You cannot learn a programming language without doing the exercises and this is your chance to improve and learn!

Problem 1: Rock, Scissors, Paper game (20pts)

In this question, you are going to practicing using (if, elif, and else). You will write a function that determine the results of a rock, paper, scissors game, given Player 1 and Player 2's choices. Your program will print out the results. Here is the rule of the game: https://en.wikipedia.org/wiki/Rock-paper-scissors



Hints: 1 First, print out a truth table for all possible choices of Player1 and Player2.

Player 1	Player 2	Result
Rock	Rock	Tie
Paper	Scissor	Payer 2

2. **Paying off credit card debts (20pts)**

Each month, a credit card statement will come with the option for you to pay a minimum amount of your charge, usually 2% of the balance due. However, the credit card company earns money by charging interest on the balance that you don't pay. So even if you pay credit card payments on time, interest is still accruing on the outstanding balance.

Say you've made a $5,000 purchase on a credit card with 18% annual interest rate and 2% minimum monthly payment rate. After a year, how much is the remaining balance? Use the following equations.

Minimum monthly payment = Minimum monthly payment rate x Balance
(Minimum monthly payment gets split into interest paid and principal paid)
Interest Paid = Annual interest rate / 12 months x Balance
Principal paid = Minimum monthly payment – Interest paid
Remaining balance = Balance – Principal paid

For month 1, we can compute the minimum monthly payment by taking 2% of the balance: Minimum monthly payment = .02 x $5000.0 = $100.0

We can't simply deduct this from the balance because there is compounding interest. Of this $100 monthly payment, compute how much will go to paying off interest and how much will go to paying off the principal. Remember that it's the annual interest rate that is given, so we need to divide it by 12 to get the monthly interest rate.

Interest paid = .18/12.0 x $5000.0 = $75.0
Principal paid = $100.0 – $75.0 = $25

The remaining balance at the end of the first month will be the principal paid this month subtracted from the balance at the start of the month.

Remaining balance = $5000.0 – $25.0 = $4975.0

For month 2, we repeat the same steps:
Minimum monthly payment = .02 x $4975.0 = $99.50 # notice $4975.0 is last month's balance
Interest Paid = .18/12.0 x $4975.0 = $74.63
Principal Paid = $99.50 – $74.63 = $24.87
Remaining Balance = $4975.0 – $24.87 = $4950.13

After 12 months, the total amount paid is $1167.55, leaving an outstanding balance of $4708.10. Pretty depressing! Let's hope after doing this exercise, you will never want to own the credit card company any money!

Paying the minimum:

Test case 1:

Write a program to calculate the credit card balance after one year if a person only pays the minimum monthly payment required by the credit card company each month.

Use raw_input() to ask for the following three floating point numbers:

1. the outstanding balance on the credit card
2. annual interest rate

3. minimum monthly payment rate

For each month, print the minimum monthly payment (for 12 months), remaining balance, principle paid in the format shown in the test cases below. All numbers should be rounded to the nearest penny. Finally, print the result, which should include the total amount paid that year and the remaining balance.

>>>

Enter the outstanding balance on your credit card: 4800
Enter the annual credit card interest rate as a decimal: .2
Enter the minimum monthly payment rate as a decimal: .02
Month: 1
Minimum monthly payment: $96.0
Principle paid: $16.0
Remaining balance: $4784.0
Month: 2
Minimum monthly payment: $95.68
Principle paid: $15.95
Remaining balance: $4768.05

Test case 2:

How about a credit card that requires a minimum of 4%? Demonstrate that people will pay less interest over the years with the new minimum by printing out the test case. Ask the same questions and print the same information as Test case 1 for this case.

Hint: use *round()* function.
The outline of your code should be:

1. Retrieve user input.
2. Initialize some state variables. Remember to find the monthly interest rate from the annual interest rate taken in as input.
3. For each month: Compute the new balance. This requires computing the minimum monthly payment and figuring out how much will be paid to interest and how much will be paid to the principal.
4. Update the outstanding balance according to how much principal was paid off.
5. Output the minimum monthly payment and the remaining balance.
6. Keep track of the total amount of paid over all the past months so far.
7. Print out the result statement with the total amount paid and the remaining balance.

3. **Problem 3 Nims and Stone (20pts)**

In this game, two players sit in front of a pile of 100 stones. They take turns, each removing between 1 and 5 stones (assuming there are at least 5 stones left in the pile). The person who removes the last stone(s) wins.

Write a program to play this game. This may seem tricky, so break it down into parts. Like many programs, we have to use nested loops (one loop inside another).
In the outermost loop, we want to keep playing until we are out of stones.

Inside that, we want to keep alternating players. You have the option of either writing two blocks of code, or keeping a variable that tracks the current player. The second way is slightly trickier since we haven't learned lists yet, but it's definitely do-able!

Finally, we might want to have an innermost loop that checks if the user's input is valid. Is it a number? Is it a valid number (e.g. between 1 and 5)? Are there enough stones in the pile to take off this many? If any of these answers are no, we should tell the user and re-ask them the question.

So, the basic outline of the program should be something like this:

```
totalStones = 100
maxStones = 5
pile = TOTAL # all stones are in the pile to start
while [pile is not empty]:
    while [player 1's answer is not valid]:
        [ask player 1]
    [execute player1's move]
    Do the same for player 2…. (this can be achieved by a for loop)
```

Note how the important numbers 100 and 5 are stored in a single variable at the top. This is good practice -- it allows you to easily change the constants of a program. For example, for testing, you may want to start with only 15 or 20 stones.

Be careful with the validity checks. Specifically, we want to keep asking player 1 for their choice as long as their answer is not valid, BUT we want to make sure we ask them at least ONCE. So, for example, we will want to keep a variable which tracks whether their answer is valid, and set it to False initially.

4. **Latin Square** (**extra 20 pts**) This problem is slightly hard!

A Latin Square is an n*n table filled with n different symbols in such a way that each symbol occurs exactly once in each row and exactly once in each column
(see http://en.wikipedia.org/wiki/Latin_square).

http://mathworld.wolfram.com/LatinSquare.html

There is even a youtube video about this game:
https://www.youtube.com/watch?v=L0YLxEE4Oxk

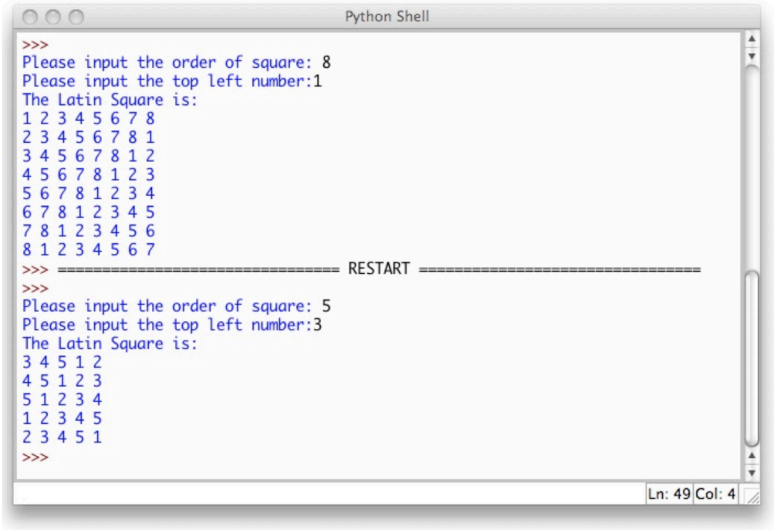
For example, two possible Latin Squares of order 6:

```
123456
234561
345612
456123
561234
612345
```

```
345612
456123
561234
612345
123456
```

Obviously, the top-left numbers are 1 and 3 respectively.

Your program will ask user to input two numbers. The first number is the **order** of square; the second one is the **top-left number** of the square. Note that the second number should be between 1 and the first number, so your program should check this situation. Then, your program will print the corresponding Latin Square. Here is some example output:



Getting Started

Break the problem down into smaller parts. For example:

1. The range function and the % (modulus) operator are both useful for this problem.
2. Can you generate a sequence beginning with 1 of the appropriate order? A sequence of order 5 would be 1 2 3 4 5 3. Can you generate a sequence of the appropriate order beginning with a number other than 1? For a sequence of order 5 starting with 3 would be 3 4 5 1 2
4. Can you generate the second in the Latin sequence? For example, starting with 1 2 3 4 the next sequence would be 2 3 4 1

You need to use **nested For loop** for this question and you can write a couple of functions.