CSC 280 Homework Assignment 1 (60pts)

Topics covered: basic data types, print statement, while/for loops.

Due Date: 12:59pm, September 22 (Monday)
Please read the instructions careful!
Also please read the late submission policy on the syllabus (course website).

Hand-In Procedure
1. Save
Save your solution to Problem 1 as ps11.py, Problem 2 as ps12.py, Problem 3 as ps13.py and Problem 4 as ps14.py. Do not ignore this step or save your files with a different name.

2. Header info
At the start of each file, in a comment, write down your name, and the number of hours (roughly) you spent on the problems in that part. You can discuss with other. No copying over each other's code, only high-level discussions are allowed.

For example:

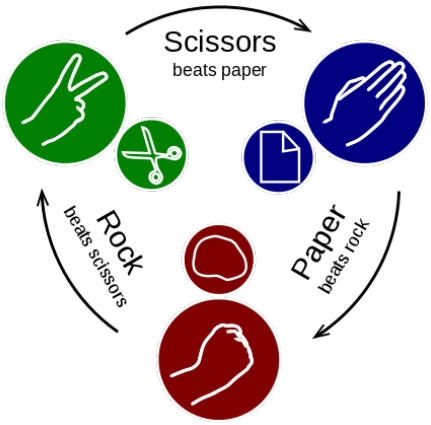
```
# Problem Set 1
# Name: Emily Brown
# Time Spent: 3:30
# Last Modified: Sep 19, 2014

… your code goes here …
```

3. Upload the three files into a .zip folder and upload onto blackboard. The .zipped file should have such a format: **firstname_lastname_ps1.zip.** Notice your grader have to download your files so it is critical you name your folder with your name.

Problem 1: Rock, Scissors, Paper game (10pts)

In this question, you are going to practice using (if, elif, else). You will write a small program that determines 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:



Notice when both players have the same choice, they tie.

Hints:

1. First, print out a truth table for all possible choices of Player 1 and Player 2.
   Player 1 Player 2 Result
   Rock Rock Tie
   Paper Scissor Player 2
   ……
2. Save your code into a file (see above for saving instructions) to generate the result. You can use the following "if" statement:

   ```
   if (player1 == 'rock' and player2 == 'scissors'):
       print 'Player 1 wins.'
   ```

Problem 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.

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.

Problem 4 (10pts): Write a program that asks the user to enter an integer and prints two integers, root and power, such that 0<pwr <6 and root** pwr is equal to the integer entered by the user. If no such pair exists, it should print a message to that effect.

Hint: This program demonstrates another example of exhausted enumeration. That means you can consider incrementing pwr and root using a loop. The output of a test case should be something like this:

```
>>>
Enter a positive integer: 16
the root is  4 the power is  2
>>>
Enter a positive integer: 14
No such pair of integers exists
```