This problem set is due on **Sunday, October 11 at 11:55pm**.

- This problem set assumes that you have installed a Python programming environment such as Canopy on your computer.
- Once you have installed Canopy, you can use it to create programs using the editor that comes with it. If you want to use other text editor, that is fine as well.
- Your program should run without syntax errors. Please do not submit programs that have any syntax error! It is better to submit a partial implementation that runs as opposed to a full implementation that does not even run.
- Add your name and email address as a comment at the top of each file you submit.

Submit the following file(s) to the **Assignments** area on Sakai as an attachment. Multiple submissions are allowed within the given time limit. Please do **not** submit other files that I did not ask for.

`ps3.py`

**Naming Conventions In Python And Programming Style In General**

Please use these conventions as you establish your *programming style*.

- **Names**: Choose informative names, e.g., `hourlyRate` rather than `hr` if it is to be a variable name. If you like `hourly_rate` better than `hourlyRate`, use that form. Whichever style you choose, stay with that style consistently.
- **Variable and function names**: We start a variable or function name with a lower-case letter, e.g., `count`, `hourlyRate`, etc.
- **Use of white space**: Adopt a good indentation style using white spaces and be consistent throughout to make your program more readable.

**Your Solution Format**

See `sample.py` in the [given] folder of Problem Set 2 and put your solution in that format. It is a good format to use and it will also make grading much easier.
Problem 1

In ps3.py write a function called heartRisk that has two parameters (age and bmi in that order) and returns a string that indicates what they mean according to the chart given on page 8 of Lec7.pdf (9/29 lecture).

For example, heartRisk(20, 25) should return ‘Medium’.

Your test should call the function and prints the returned value. Include enough test cases in main so that at least one of all possible outcomes is returned in your tests.

In your implementation use nested conditionals.

Problem 2

In ps3.py include a function named encodeMajor that encodes (converts) the major information of a student into an integer and returns it. Use the following encoding scheme (assuming only these majors are offered in a college somewhere):

<table>
<thead>
<tr>
<th>Major</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>0</td>
</tr>
<tr>
<td>Biology</td>
<td>1</td>
</tr>
<tr>
<td>Chemistry</td>
<td>2</td>
</tr>
<tr>
<td>Computer Science</td>
<td>3</td>
</tr>
<tr>
<td>Economics</td>
<td>4</td>
</tr>
<tr>
<td>English</td>
<td>5</td>
</tr>
<tr>
<td>History</td>
<td>6</td>
</tr>
<tr>
<td>Literature</td>
<td>7</td>
</tr>
<tr>
<td>Mathematics</td>
<td>8</td>
</tr>
<tr>
<td>Philosophy</td>
<td>9</td>
</tr>
<tr>
<td>Political Science</td>
<td>10</td>
</tr>
<tr>
<td>Psychology</td>
<td>11</td>
</tr>
</tbody>
</table>

If an invalid argument is given to the function, it should return -1. For example, encodeMajor(’History’) would return 6. Print the value returned by this call as part of your tests in main. Include at least two more tests including an invalid case.

This time write a function named decodeMajor that decodes (converts) an encoded major value back to the original string and returns it using the encoding scheme given above. For example, decodeMajor(5) would return ‘English’. Let’s make decodeMajor return ‘Invalid’ for any invalid parameter value. Include several test cases in main.

As the final step in this problem, include this call:

```
print decodeMajor(encodeMajor(’Economics’)) == ’Economics’
```

which should print True.

Problem 3

The problems 3 through 6 are designed to have you practice with loops. So, if you are not using at least one loop in each of those problems, you are not doing it right. I am mentioning it once here but it applies to all the problems below.

In ps3.py define a function named first_digit that accepts a positive integer as an argument and returns the first digit in that number as an integer. For example, first_digit(43572) should return the number 4, not a string ’4’ and first_digit(912398) should return 9. Note that you are not allowed to convert the integer argument into a string and deal with a string even if you know how to do it that way. Use a loop and an integer division operator instead since we are trying to practice with loops. Include at least one more test in addition to these two in main like this:
print first_digit(43572)
print first_digit(912398)

Problem 4
In ps3.py add the following functions:

- one named power_of with two parameters: base and exponent, both integers. This function should calculate and return the base raised to the power of the exponent (base^exponent) as an integer. You may assume that both base and exponent are positive. You may not use a function from the math module in Python to perform this calculation. That is, you must define one of your own (again using a loop).

- another named print_powers_of with two parameters: base and max, both integers. Utilizing the power_of function defined above, print_powers_of should print each power of the base from base^0 up to the maximum power, base^max (inclusive). For example:
  - print_powers_of(4, 3) should print: 1 4 16 64
  - print_powers_of(5, 6) should print: 1 5 25 125 625 3125 15625
  - print_powers_of(-2, 8) should print: 1 -2 4 -8 16 -32 64 -128 256

Include at least two tests for power_of and at least these three tests for print_powers_of in main.

Problem 5
In ps3.py add a function named series with two parameters: x and n, where −1.0 < x < 1.0 and n as a positive integer is the number of terms to be added. This function computes the sum of specified number of terms in the following series.

\[ 1 - 2x + 3x^2 - 4x^3 + 5x^4 - \ldots \]

For example,

\[
\begin{align*}
\text{series}(0.5, 1) &= 1 \\
\text{series}(0.5, 2) &= 0 \\
\text{series}(0.5, 3) &= 0.75 \\
\text{series}(1.5, 3) &= -1 \\
\text{series}(-1.5, 3) &= -1 \\
\text{series}(0.5, 0) &= -1
\end{align*}
\]

Use a while loop. This function (series) should contain no print statements. That is, output should be generated by a function other than series, e.g., main. Include the test calls given above in your main. If an invalid value is tried for x or n, the function should return -1.

Problem 6
This problem is designed to give you a chance to practice with nested loops (one or more loops nested inside another loop). Write your solutions using nested loops even if you can alternatively write them without using nested loops.
In ps3.py add the following functions:

- one named print_rectangle with two parameters: height and width, both integers. This function should print the following pattern to the computer screen if it is called as print_rectangle(4, 10):
• another named `print_triangle` with one parameter, `n`, an integer. This function should print the following pattern to the computer screen if it is called as `print_triangle(5)`:  

```
  *
  **
  ***
  ****
  *****
```

• **OPTIONALLY, (i.e., not required)** yet another named `print_triangle2` with one parameter, `n`, an integer. This function should print the following pattern to the computer screen if it is called as `print_triangle2(5)`:  

```
  **
  ****
  *****
  ********
  **********
```

Be sure to include at least one test each for each of the functions above in `main`.  
Hand in `ps3.py` as an attachment on Sakai.