CSCI 141 Computational Problem Solving
Project 1: Some Really Random Things
(The projects get more entertaining, we promise.)

In this project, you provide four Python functions that solve particular problems as outlined below. The problem-solving aspects of these are the same as the exercises at the ends of the labs so far. The only difference is that instead of typing your code in the web browser and executing it with shift-enter, you will type the code in your text editor application, save it to a file ~~ending in~~called random_functions.py, and execute it in the terminal application by typing "`python ./~~<filename>~~random_functions.py`" without quotes. All of your functions should be in ~~a single file called "<your_id>l1.py" without quotes. Jim's file would, for example, be called 93011226711.py. Notice that after your id number is the lower-case letter 'L', as in lab~~that single file~~1~~.

1. Write a function that takes as a parameter a positive integer `h` and prints an isosceles triangle of asterisks. Your function must be called `print_isosceles(h)`, and must not return a value. For example, `print_isosceles(5)` should print

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

2. Write a function that takes as parameters three integers and computes and returns the median of the three. That is, for the three provided integers, return b such that $a \leq b \leq c$. Your function must be called `median_of_three(a,b,c)`, and must not print anything to screen. For example,

   `median_of_three(7, -3, 0)` should return `0`.

3. One way to define a circle is by the coordinates of the upper left hand corner of the smallest square that contains the circle, and providing the width of that square, which is equivalent to the diameter of the circle. Assume that coordinate 0,0 is the upper-left hand corner of the universe, and our coordinate system is relative to that point so that x coordinates increase to the right and y coordinates increase down.

   Provide a function that takes as parameters `origin_x` and `origin_y` representing the coordinates of the upper left hand corner ~~origin~~ of the circle's bounding square ~~rectangle~~, the `diameter` of the circle, and the `x` and `y` coordinates of a point somewhere in the same coordinate system. Your function should return `True` if the point is inside of the circle, and `False` otherwise. Your function must be called `point_is_in_circle(origin_x,`

`origin_y, diameter, x, y)`, and must not print anything to the screen. For example,

`point_is_in_circle(5, 15, 100, 52, 61)` should return `True`.

4. To use a function that is provided by Python, but not in the `__main__` namespace, we use the `import` keyword. The statement

   ```
   from random import randint
   ```

   imports the function `randint` from the `random` library for use in your program. Typically (and in this project), we place all import lines at the top of the file, at the left margin.

   The `randint` function takes two parameters a and b and returns an integer r between a and b, inclusive so $a \leq r \leq b$. Using the `randint` function, write a function that simulates a Pokéstop by returning items (represented as these **exact** strings) with the following probabilities:

`'Poke Ball'`	40%
`'Great Ball'`	22%
`'Razz Berry'`	12%
`'Potion'`	12%
`'Super Potion'`	7%
`'Hyper Potion'`	4%
`'Revive'`	2%
`'Egg'`	1%

   Your function must be called `pokestop()`, and must not print anything to the screen. For example,
   `pokestop()` should return the string `'Razz Berry'` around 12 out of every 100 calls.

10% Bonus Opportunity

Successful, well-presented implementations of the above functions will receive full credit. For a bonus, replace the test invocations of your function with a loop that implements a menu system. Each time the loop runs, it should prompt the user with the choices of functions, and numbers or letters to select each. Upon selection of a function, the loop should prompt the user for the values of each parameter (if any are needed) and then call the specified function with the provided parameters, print any returned results, and cycle to the menu again. The last option on the menu should be to exit the program. If the user selects that option, change the value of the control variable in your `while` loop so that it is `False`. This will terminate the loop and your program will exit.

SUBMISSION EXPECTATIONS

`random_functions.py`: A single Python file containing your four implementations and optionally the while-loop driven menu system. The four functions must be called

`print_isosceles(h)`,
`median_of_three(a,b,c)`,
`point_is_in_circle(origin_x, origin_y, diameter, x, y)`, and
`pokestop()`.

Each function should either return a value or print to screen as specified, but never both.