Lab Sheet 3

Functions and loops

1. Write a function that takes the lengths of the two shorter sides of a right-angled triangle as parameters. Return the length of the hypotenuse of the triangle, computed using the Pythogorean theorem, as the function's result. Test your function on a range of inputs to check its correctness.

2. Write a function that takes three numbers as parameters and that returns the median of those parameters as the result. Do not use Python's sorting capabilities.

3. Write a function `doughnut(n)` that displays an $n \times n$ "doughnut" pattern as illustrated below (for $n = 9$):

```
=========
=========
=========
===   ===
===   ===
===   ===
=========
=========
=========
```

   The border around the perimiter should be a third of n (integer part thereof) in width.

4. Write a function that takes a string of characters as its first parameter and the width of the screen as its second parameter. Your function should return a new string that consists of the original string and the correct number of leading spaces so that the the original string will appear centered within the provided width when it is printed. Do not add characters at the end of the string.

5. Write a function that simulates a sequence of coin flips and returns the number of flips performed until the first occurrence of two consecutive heads. Use the `randint` function from Python's `random` module to simulate a single coin flip. Perform a maximum of 1000 flips; if two consecutive heads have not appeared by sthat stage return 1000 as the result. Note a return statement can appear anywhere with the body of a function. It will cause the function's execution to cease (even from the middle of a loop say) and for the indicated value to be returned. [1]

6. Write a function that takes a non-negative real number as its parameter and that returns an approximation of the square root of that number using the approach sketched below. (Do no use the $**$ or any of Python's functions such as `sqrt` or `pow` etc.).

   Calculate the root by a generating a sequence of successively better approximations ("guesses") for the true value. For parameter x, take $x/2$ as your initial guess. At each step update guess by replacing it by the average of the current value of guess and $x/$guess. Terminate the process after 100 iterations and return the value of guess. [2]

[1] A better approach to this would to employ a "conditional" form of looping structure that iterates until such times as a particular situation arises (in this case the appearance of two consecutive heads. We will see such a constructe later on, but for now we will rely on the for loop.

[2] Ditto.