Exercises: Functions

For all the exercises you should write a test program as well as the program itself. NB The test program may well be bigger and more complex than the program itself!

Part A

Write a Python module that provides an iterative and a recursive implementation of the factorial function. Factorial is defined:

$$\text{factorial}(x) = \begin{cases} 1 & x = 0,1 \\ x \times \text{factorial}(x-1) & x > 1 \end{cases}$$

NB The factorial function is not defined for negative integer values or for float values. Is there a situation where the recursive version is unable to work?

Part B

Write a Python module that provides a function for creating random UUIDs (universally unique identifiers), for example:

f81d4fae-7dec-11d0-a765-00a0c91e6bf6

A UUID is formally defined by the grammar:

```
UUID                        = time-low "-" time-mid "-"
                              time-high-and-version "-"
                              clock-seq-and-reserved
                              clock-seq-low "-" node
      time-low              = 4hexOctet
      time-mid              = 2hexOctet
      time-high-and-version = 2hexOctet
      clock-seq-and-reserved = hexOctet
      clock-seq-low         = hexOctet
      node                  = 6hexOctet
      hexOctet              = hexDigit hexDigit
      hexDigit              =
            "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" |
            "a" | "b" | "c" | "d" | "e" | "f" |
            "A" | "B" | "C" | "D" | "E" | "F"
```

Part C

Write a Python program that takes a string and, optionally, the name of a file and searches the file for instances of any and all permutations of the string. On Debian, Ubuntu and Mac OS X machines /usr/share/dict/words is a list of words that is useful for testing. It is believed that this file also exists on Fedora. On Solaris 10 the file is at /usr/dict/words or /usr/share/lib/dict/words. It is not known where the file is on Windows – nor indeed whether there is, in fact, such a file on this platform.

Part D

Write a Python module that provides methods for calculating the *arithmetic mean*, *median*, and *mode* of a sequence (list or tuple) of values. For the data:

3.4 , 5.6 , 2.0 , 4.5 , 7.8 , 4.666666 , 2.0

then:

mean = 4.28095228571428571428
median = 4.5
mode = 2.0

and for the data:

1.0 , 1.0 , 1.0 , 2.0 , 2.0 , 2.0 , 3.0 , 3.0 , 3.0 , 3.0

then:

mean = 2.1
median = 2.0
mode = 3.0

The *arithmetic mean* of a data set is defined by: $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$

The *median* of a data set is the central value if there is an odd number of items, or the arithmetic mean of the central two items if there is an even number of items of the data as a sorted sequence.

The *mode* is the most frequently occurring value in the data set. There can be more than one mode for a data set.

Hint: Comparison-based sorting algorithms include Bubble Sort, Quicksort and Merge Sort. There is no need to contemplate programming any of these algorithms as the Python library already has a sort function available.

Hint: Dictionary (aka map, associative array, or lookup table) might be a useful data structure when creating counts.

Part E

Write a Python script to draw the von Koch Snowflake using the Turtle module. This is a fractal created by drawing a triangle but where each side that is longer that some lower limit is not actually a straight line but has a triangle on it.