Introduction to CS for Data Science

HW2 – Recursion & Files

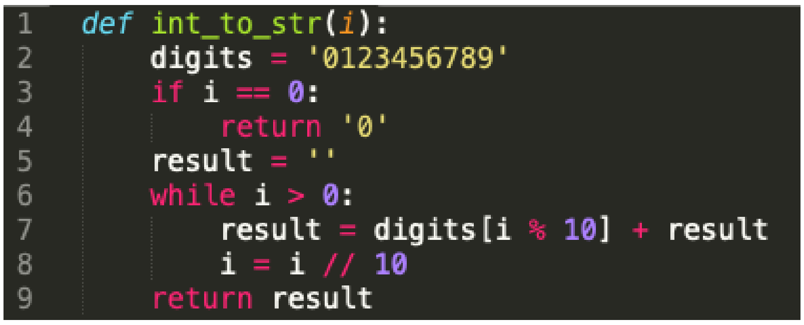
Recursion and Time Complexity

In this exercise, you will not be provided with premade scripts. For the time complexity analysis, you may write your answers in a .txt or .docx or .pdf file zipped with your python file or in the comments of your python file. In your python file, please put your name(s) and ID(s) in a comment at the beginning of the file.

In conducting running time analysis, you are not expected to produce a rigorous proof. You are expected to provide a reasonable explanation similar to the recitations (i.e., explain how many iterations and why, how many operations per iteration and why, and express using big-O notation).

For the following coding problems, use recursion. You are free to define your recursive functions as you see fit (you may define helper functions). Make sure you test your code on interesting cases and make sure it works well. You are encouraged to write tests.

1. Consider the following function that converts and int into a string and analyze its time complexity. The function assumes *i* is a nonnegative int and returns the string representation of *i*.



```python
def int_to_str(i):
    digits = '0123456789'
    if i == 0:
        return '0'
    result = ''
    while i > 0:
        result = digits[i % 10] + result
        i = i // 10
    return result
```

2. The power set of any set S is the set of all subsets of S, including the empty set and S itself. Write a recursive function that takes as input a list (you can assume the list does not have repeats of any element), and returns its power set, and analyze your algorithm's time complexity.
   For example, power_set([1,2,3]) will return [[],[1],[2],[3],[1,2],[1,3],[2,3],[1,2,3]].

3. Write a recursive function that takes as input a non-negative integer n and returns the nth row of Pascal's triangle (assuming that the counting of rows starts at zero). For example, pascal(0) will return [1], pascal(3) will return [1,3,3,1], pascal(4) will return [1,4,6,4,1].

Files

In this part you will be reading from files. Your functions should support taking paths as input and should not hard-code the given file paths. During grading, new files will be passed to your functions.

4. In this question you will be reading from the files ‘cipher.txt’ and ‘key.txt’.

   Write a function that reads the paragraph from the file ‘cipher.txt’, decodes it using the key in the file ‘key.txt’, and prints the decoding. Decoding is performed by substituting every instance of a letter in the cipher-text with the corresponding clear-text letter in the key.

   You should check that the key is valid (every letter in the alphabet appears exactly once).

5. In this question you will be reading from the files ‘database.csv’ and ‘ids.csv’, and writing to a new file that you need to create.

   Write a function that takes as input two comma separated files. The first file, ‘database.csv’, includes a small dataset of popular movies and their ratings. The column ‘tconst’ is a unique column (meaning no two instances share the same value for this feature).
   The second file, ‘ids.csv’, contains a list of ‘tconst’ values. Write a function that creates a new file and writes to it a list of tuples that includes all three features from the file ‘database.csv’ for each of the “tconst’ values that appear in the file ‘ids.csv’.

Good luck :)