**Week 3 Extra In-Class Exercises (Conditions and Tuples)**

**Q1: Recursive Functions [ \*\*\* ]** (This question is out of the scope of the syllabus, i.e., the topic “recursion” covered here won’t appear in any quiz, lab test or the exam.)

A function can call itself. This kind of functions are called **recursive** functions. Recursive functions can be useful in some scenarios.

For example, take a look at the following function. You can assume that the function always takes in **a positive integer** as its parameter. Can you figure out what this function can do?

def count\_down(n):  
 print(n)  
 if n > 1:  
 count\_down(n - 1)

You can try to call count\_down(5), count\_down(10), etc. and observe the output.

**Note:** For all the questions below, you’re NOT allowed to use loops (for-loops or while-loops) to solve the problem.

1. Define a recursive function called compute\_factorial(n). The function takes in a positive integer n as its parameter. The function **returns** the factorial of n, i.e., , which is defined as . (See <https://en.wikipedia.org/wiki/Factorial> .)

For example, compute\_factorial(3) returns 6, compute\_factorial(5) returns 120.

1. Define a recursive function called get\_num\_digits(n). The function takes in a positive integer n. It returns how many digits this number has.

For example, get\_num\_digits(146) should return 3, get\_num\_digits(354928502) should return 9, etc. You are NOT allowed to convert the number n into a string first.

1. Define a recursive function called display\_fibonacci\_numbers(n). The function prints the first n Fibonacci numbers in the Fibonacci sequence. See <https://en.wikipedia.org/wiki/Fibonacci_number> for the definition of Fibonacci sequence, which looks like the following:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, …

For example, display\_fibonacci\_numbers(4) will display the following

1 1 2 3

display\_fibonacci\_numbers(10) will display the following

1 1 2 3 5 8 13 21 34 55