COSC 120: HW#2
Dr. Fred Park, Whittier College

1. **Comparison of two program outputs:** Write a loop using 100,000 iterations. In the loop, for each iteration, create a 1×10 numpy array and use the 'randint' function from the random module and input random numbers from 0 to 9 into each array location. Call it x. Run your bubble sort code from class to sort this length 10 array x into ascending order and store it each iteration. In this same iteration, use the 'sorted' function from numpy to sort the x array . Calculate the difference between these arrays (which will give you another length 10 array), sum that difference array and store that value in a difference vector of length 100,000. At the termination of the loop, sum the difference vector. If your bubble sort is correct, the sum should equal zero.

2. **The Monty Hall Problem:** Here we will investigate this famous probability phenomenon. Suppose you're on a game show, and you're given the choice of three doors: Behind one door is a car; behind the others, goats. You pick a door, say No. 1, and the host, who knows what's behind the doors, opens another door, say No. 3, which has a goat. He then says to you,"Do you want to pick door No. 2?" Is it to your advantage to switch your choice?"
   Write a simulation in python of the Monty Hall problem based on two strategies. One where you always switch and one where you always stay at your first choice door. Do this for 10,000,000 (10 million) trials. What is the experimental probability in each case? Does the outcome agree with your calculation of the theoretical probability?

3. **Cleaning Data:** Let s be a string that contains a sequence of decimal numbers separated by commas, but somehow the data is corrupted and a random * shows up either right after a comma, before a comma, or before the first term of an integer number. e.g. (that's exempli gratia, latin abbreviation of 'for example') s = '1.23,*2.4*, 3.123,*5.1, 8.7*, *7.23, *3.23*'. Write a program that prints the sum of the numbers in the string s. Note, the whitespace is unevenly distributed in the string as well. You cannot use any built in functions or methods associated to the string object.

4. **Prime Number Generation:** A prime number is a number that is only divisible by itself and 1. e.g the number 2 is prime, so is 3, but $4 = 2 \times 2$ is not. In general, if a number looks prime, then it usually ends up being prime. For example, 1997 is prime and it looks prime. However, this is not a rigorous theorem and the distribution of prime numbers is a concept beyond the scope of this course.

   (a) Write code that prints out all the prime numbers from 1 to 100. Can you do the same for any number N that is input by a user? Write code that does this as well.

   (b) Write code that prompts a user to enter an integer N and print out that many primes.

   (c) Write code that prompts a user to enter an integer and print out that many twin primes. Twin primes are primes that differ by 2. e.g. (3, 5) are twin primes.

5. **Goldbach's Conjecture** Goldbach's Conjecture states that any even number larger than 2 can be written as the sum of two primes. e.g. $12 = 5 + 7$. Write code that prompts a user to enter an even integer and outputs the two primes that sum to it. Format your output so that you get the decomposition on the screen. e.g. you should print out "$12 = 5 + 7$" on the screen in one enters 12. You should also build in error checking to make sure a positive even integer is entered, otherwise ask the user to re-enter.

6. **Lists:**
   a. Generate a random list of numbers of length N where N is input by the user. Calculate the mean and median of the numbers from this list. Note, we are using the data structure 'list.'
   b. Write a bubble sort algorithm that operates on a list of arbitrary numbers.
   c. Write code that sorts a list of names into alphabetical order.

7. **Guess the Number Game:**
   a. Write code that generates and stores a random number. Prompt the user to guess what the number is.
   b. If the number is wrong, indicate whether the guess was too high or low and prompt the user to guess again. If it was guessed correctly, you should output a message of that sort. You may give the user multiple tries. Make sure to print an error message if the value entered by the user is not a number and should prompt the user to re-enter a number.

8. **Newton's Method** Implement Newton's method to find a zero of a polynomial.
   (a) Show that it converges in fewer steps than the bisection algorithm from class by testing on $f(x) = x^2 - 2$ for $x_0 = 1$.
   (b) Can you use it to find the root of $f(x) = x^2 - 2x + 1$ for $x_0 = 0$? How iterations until you get $|f(x^*)| < 10^{-6}$ where x^* is the approximated root?