Assignment 1

- Assignments are to be done individually by each student. You can discuss the problems with each other, but you must write the code yourself.
- The deadline for submission is Fri, 16th Dec (midnight). You have to submit your code on GC - further instructions on this will be posted.
- You should use VS code for developing the code (so you become familiar with VS IDE).
- Each question will be given 0, 1, or 2 marks for incorrect, partially correct, and fully correct answers. (The total marks will be converted to out of 10).
- Solve bonus questions only after you have done all (or at least 80%) of the regular problems correctly. Bonus questions will count only if you have done at least 80% of the regular questions. They together carry only 1 mark.
- The assignment is not yet complete - more questions may be added (it is being released soon, so you can start working on them.)

1. Write a program to take as input a number between 0 and 99, and print the text equivalent of the number. I.e., for inputs 5, 13, 43, and 85, outputs are "five", "thirteen", "forty three", and "eighty five". (Hint: Have a function to write the text for tens digit, and another to write the text for the unit digit - use of elif construct will be helpful in these. In main, take the input, get the decimal and units digits of the number and print their text. The code is simple though it may be long as you have to print for different cases. You may first write a function to print the text for a units digit and test it, and then build the rest)

   Bonus Question. Expand the code above to write in text any number lesser than 100 crore. (FYI we have: units, tens, hundred, thousand, ten-thousand, lac, ten-lac, crore, ten-crore).

2. Two variable optimization. Many problems require optimization of some function – minimizing (e.g., loss) or maximizing (e.g., profit) given some function and some constraints. This is an example of this problem.

   A furniture company manufactures dining room tables and chairs. The relevant manufacturing data are given in the table.

Department	Labor-Hours per Tables	Labor-hrs per Chairs	Maximum Labor-Hours Available/day
Assembly	8	2	400
Finishing	2	1	120
Profit per Unit	$90	$25	

Write a program to determine how many tables and chairs should be manufactured each day to realize a maximum profit. Print the number of chairs, tables, and the maximum profit. Keep the code to compute this simple (even if it is inefficient), and you do not have to optimize the code.

Hint: Let x1 represent the number of tables and x2 represent the number of chairs. Determine the equations for total profit P, and the constraints on the two types of labor hours. Write functions to compute P and each constraint (these will be very simple functions). Write the main program that ranges over the values of x1 and x2 (from min possible value to max possible value) to find at which values max is reached.

3. Distance traveled. A vehicle is to travel on demand. Its initial location is given as x0, y0 (assume the first statement in your program as assigning initial values, say x0,y0 = 5.0, 5.0). Repeatedly take as input the distance the vehicle has to travel. If the input given is 0 or lesser, the travel ends - assume that at least one positive distance will be given. The direction in which the vehicle is to travel is determined as follows: if distance is <= 25 it travels north, if between 26-50 it travels south, between 51 and 75 it travels east, and >= 76 it travels west. Find the final coordinate of the vehicle, the total distance it has traveled, and the straight line distance between the initial location and the final location (use standard formula for distance between two coordinates; note that this distance is not same as total distance traveled).

   Hint: Initialize current coordinates x,y to x0,y0, and distance traveled to 0. For taking the distance use a while loop (as you dont know how many distances will be given). This can be done in two ways - you can initialize a variable (say dist) to some +ve value, and then loop till the value is 0 (or lesser) and in the loop get the input from the user for next distance. Alternatively, you can have an infinite while loop, and take input in the loop and `break` if the input given is 0 (or lesser). For each input distance, determine the new coordinates, and keep track of total distance traveled. After the loop is done, compute the distance also.

4. Integration through computation. Suppose the velocity of a rocket at a time t is given by:
   $f(t) = 2000ln[\frac{140000}{140000 - 2100t}] - 9.8t$
   Take as input starting time (a) and ending time (b), and find the distance covered by a rocket between time a and b. For computationally determining the distance, work with time increments (delta) of 0.25 seconds. You can use the math module of python for this problem.

   Hint: Compute the velocity at time t and t+delta, take the average, and then compute the distance traveled in this delta time duration. Start from a and keep computing in increments of delta till b.

5. A person is standing and is looking at a pole in front of him. Given the angle of view to the top (in degrees – should be between 0 and 90) and the horizontal distance from the person to the base of the pole (in meters), you have to find the height of the pole and the length of the line from the person to the top of the pole.

Your program has to take as input (from the user on the terminal) the angle (in degrees) and another input for distance to the base of the pole. Then it computes and prints the height of the pole and the distance to the tip of the pole.

Note. You must write functions to compute sin(), cos(), … using the series for them and ***cannot use*** the python provided functions (e.g., in the math module). If the value of pi is needed, you can use the standard value (3.14). First, write these functions and test them. Then write the main program to take inputs for angle and distance, and call the functions to compute the height and distance.

6. Write a function that takes a positive integer “n” as a parameter and prints the pattern of the type given below for n=7. Remember that print("*", end="") will not print the newline, and you can print() and can just print a new line. In the main program, take user input as n and call this function for printing.
   User Input: 7

7. You have just entered college and would like to buy a laptop of cost, **cost**. Your monthly allowance is **allowance**, and you can save a fraction **sf** of it every month. Every month, you put your savings in a bank account that gives you a monthly rate of interest **r** (interest is compounded). If your allowance is Rs 20,000, your saving fraction sf is 0.1, and the rate of interest r is 0.5, determine after how many months you will be able to purchase the laptop and the amount of savings that will remain after purchase. For laptop cost, take the input from the user (try Rs 65000, 45000, 125000, etc.)

   Hint. Keep track of your total saving after every month - you have to add the interest you earn on your total savings of last month and the savings of this month. For (i) write a function that takes the cost and other parameters, and keeps increasing the month count from 1 onwards, till your savings become more than cost and returns the number of months and the remaining saving (recall that return can return more than one value). In the main program, you take the cost as input and then call this function.

   Bonus Problem. Suppose you want to buy the laptop in fewer months. As the interest rate is not in your control, you can only change the saving fraction. Determine the savings rate needed for different numbers of months for purchasing the laptop, starting from the number of months determined earlier.

8. The current population of the world is combined together into groups that are growing at different rates, and the population of these groups is given (in millions) in a list. The population of the first group (is Africa) is currently growing at 2.5% per year, and each subsequent group is growing at 0.4% lesser (note: the growth rate can be negative also). For each group, every year, the growth rate reduces by 0.1. With this, eventually, the population of the world will peak out and after that start declining. Write a program that prints: (i) the current total population of the world, (ii) the years after which the maximum population will be reached, and the value of the maximum population.

   Hint: Write a function to compute the population of a group after n years, given the initial population and the current rate of growth. For this, you compute the population after every year till n (note that the growth rate for each year can be determined easily once the initial rate is known). In the main program, you can loop increasing the years (Y) from 1 onwards, and for each Y, you will need to compute the value of each group after Y years using the function and compute the total population of the world as a sum of these. To check if the population is declining, save the population for the previous Y, and every year check if the population has declined – if it has declined, the previous year's population was the highest (otherwise, this year's population will become the previous year's for next iteration).

   The first statement in your program should give the population as an assignment:

   ```
   pop = [50, 1450, 1400, 1700, 1500, 600, 1200]
   ```

   In the program, you can loop over this list, but you cannot use any list functions (and you need not index into the list).

9. The demand for an item in the market (in terms of the number of items) decreases as the price (p) of the items increases. On the other hand, the supply of the item increases as the price increases (as producers expect to make more profit). Suppose the demand and supply changes as follows with price:

   Demand function, D is: $\log D(p) = a - b \log p$ (let a be 5, b be 6)
   Supply function, S is: $\log S(p) = c + d\ p$ (let c be 7 and d be 8)

   An equilibrium is achieved when demand and supply match - this is a basic economics theory. (As we are working with floating point numbers, we cannot check for ==, we should check if the difference between them is less than some value – you can put your own value, say 0.5) Determine at what price an equilibrium will be reached, and output the equilibrium price and the number of items produced/bought at this equilibrium. Take the price range as input (i.e., starting price and end price), and if no equilibrium is found within this range, your program should print that no equilibrium is found.

   Hint: Write a function to compute demand and another to compute supply (for a given price). Call these functions in another function to find the equilibrium - for this loop over the range to find where they are close to each other (do not check for equal with float). In main, you can get the input range and then call the function to find the equilibrium price and production number.

10. Write a program to computationally find a root of a given polynomial in x. Your program should have a function to compute the value of the polynomial for a given value of x (you can assign the coefficients in the first few statements and then compute and return the value). You should have another function that takes as parameters the function to compute the polynomial and an initial value of x (x0), which is used to find the root. For computing the root of the polynomial, use the newton-Raphson method (look it up) - it will start with assuming the initial root as x0, and if the value of the polynomial is not 0 at x0, determine the slope at x0 to find x1 - the point where the straight line with this slope will intersect the x-axis. Then use this x1 and repeat the process. Till the value of the polynomial reaches close to 0 (you can assume that if the polynomial is within +/-0.2, it is a root).

    Your main program should ask for the value of x0 and then compute the root. Try with different values of x0 and see what root it finds – you will see that starting point can lead to different roots. As newton Raphson may not converge, or the polynomial may not have a root, after trying for some number iterations (say 100), your program should print a suitable message and quit.

    For the problem, use the polynomial: x**3 - 10.5*x**2 + 34.5*x - 35 (FYI, this one has 3 different roots)

    Bonus problem: Given that the polynomial has n roots and a range x1 and x2 within which all the roots are, expand the root finding program to find all the roots between x1 and x2.

11. Extra bonus Problem. To be defined later.
12. Extra bonus problem. To be defined later.