CS 322 Assignment 1

This coding assignment is designed to help you get up to speed with coding in Python. There are three problems. For each problem, use a dedicated c code file. **You must work on this by yourself.** You can discuss questions with your classmates or are very welcome to come to office hours. However, if you and classmates submit an identical solution, it will be treated as cheating and will receive zero credit for the problem —-unless you have proof that the incidence of identical solutions is an extraordinary coincidence.

Work format: For each problem, use a dedicated code file.

Academic honesty: You must complete the work on your own. You may discuss questions with classmates or visit office hours. However, if you and a classmate submit identical solutions, it will be treated as cheating and will receive zero credit—unless you can prove the similarity is an extraordinary coincidence.

Future assignments: For more complex assignments or projects, collaboration may be allowed and even encouraged.

File format: Save each solution in a file ending with .py so it is a proper Python file.

Folder submission: Place all files in one folder. Name the folder exactly as your name appears in NEIUPORT. For example, if your name is Jessica Fatima, submit the folder as Jessica_Fatima.

Due date: Thursday (09/11).

Grace period: You may submit up to three days late without a penalty of 20%.

Late policy: Submissions more than one week past the due date will not be accepted or graded.

Problem 1 (30 points)

The following sequence is formed using only numbers:

1. The first number is 1.
2. The first number contains one 1, so the second number is 11.
3. The second number contains two 1s, so the third number is 21.
4. The third number contains one 2 and one 1, so the fourth number is 1211.
5. The fourth number contains one 1, one 2, and two 1s, so the fifth number is 111221.

The starting number of this sequence is always 1. The sequence continues indefinitely.

Write a function that, given a starting number *m* and an integer *n*, returns the next *n* numbers in order.

- Example 1: for *m* = 1 and *n* = 4, return [11, 21, 1211, 111221].
- Example 2: for *m* = 11 and *n* = 3, return [21, 1211, 111221].

Problem 2 (20 points)

Write a function that, given a list of elements, removes **consecutive duplicate** elements while keeping the first occurrence. The order of elements must be preserved.

Examples

• remove_consecutive_duplicates([1, 2, 2, 3, 3, 3, 2, 2, 4]) → [1, 2, 3, 2, 4]

• remove_consecutive_duplicates(["a", "a", "b", "b", "a", "c", "c"]) → ["a", "b", "a", "c"]

• remove_consecutive_duplicates([7, 7, 7, 7]) → [7]

• remove_consecutive_duplicates([]) → []

• remove_consecutive_duplicates([1, 2, 3, 4]) → [1, 2, 3, 4]

Problem 3 – 3 × 3 Matrix Class in Python (50 pts)

Purpose
Implement a minimal 3 × 3 matrix class using only built-in Python. Focus on constructor validation, a multiply method, and readable string output. No external libraries such as NumPy which we will learn soon.

Requirements
Class MyNumMatrix must:

• represent a 3 × 3 matrix;

• store the numbers in values (a list of three lists);

• keep shape = (3, 3);

• validate in __init__: three rows, three items per row, all items int or float. Raise ValueError on violation.

Constructor skeleton (complete the TODOs)

```
class MyNumMatrix:
    def __init__(self, rows):
        """rows should look like [[a, b, c], [d, e, f], [g, h, i]]"""
        # TODO step 1: verify rows has length 3
        # TODO step 2: verify each row has length 3
        # TODO step 3: verify each entry is int or float
        # TODO step 4: assign self.values and self.shape
        You should challenge yourself and learn “raise Value/TypeError” by yourself.
```

multiply(self, other)
Return a new MyNumMatrix that equals the product of self and other (both 3 × 3).

Matrix-multiplication math. How does it work in a case of two (3 × 3) matrices?
Let A = $[a_{ij}]$ and B = $[b_{ij}]$ be two 3 × 3 matrices, where index i denotes the row (1 to 3) and index j denotes the column (1 to 3). The product C = AB is also a 3 × 3 matrix whose entries are

$C_{ij} = a_{i1} \cdot b_{1j} + a_{i2} \cdot b_{2j} + a_{i3} \cdot b_{3j}$ for $i, j \in \{1, 2, 3\}$.

In words: to compute entry (row i, column j) of C, take row i of A, column j of B, multiply the corresponding numbers, then add them.

Or check this out: 3*3 matrix multiplication

__str__ output format (similar to toString() in Java)
When printed, a matrix must appear exactly as [[a, b, c], [d, e, f], [g, h, i]].

Here a = self.values[0][0], b = self.values[0][1], etc.

Minimal test harness (keep unchanged). Again, use this in your solution.

```
if __name__ == "__main__":


    m1 = MyNumMatrix([[1, 2, 3],
                      [4, 5, 6],
                      [7, 8, 9]])
    m2 = MyNumMatrix([[9, 8, 7],
                      [6, 5, 4],
                      [3, 2, 1]])
    print("Product:", m1.multiply(m2))  # expected [[30, 24, 18], [84, 69, 54], [138, 114, 90]]
```

Grading (50 pts)

- proper __init__ validation: 15 pts
- correct multiply(): 20 pts
- accurate __str__: 15 pts