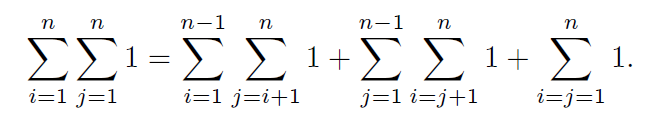
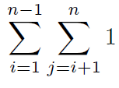
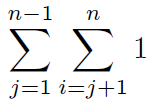
**2.10 Exercises**

2.1)The question asks us to show that the number of pairs (i,j) such that i<j for i,j∈{1,2,…,n} is given by n(n−1)2. We begin by analyzing the sum:

Each of these parts counts pairs in the set {(i,j)∣1≤i,j≤n}\{(i,j) | 1 \leq i, j \leq n\}{(i,j)∣1≤i,j≤n} with specific conditions:



1. **First term:** counts the pairs where i<j.

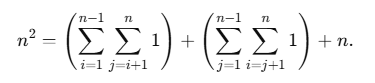


1. **Second term:** counts the pairs where j<i.

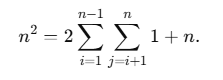


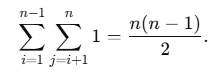
1. **Third term:** counts the pairs where i=j.

The total number of pairs (i,j) with no restrictions on the relation between i and j is n×n, or n^2. This leads to:



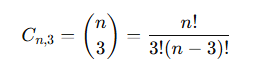
Since each of the first two terms counts the pairs where i<j and j<i respectively, these two terms are equal. Therefore:





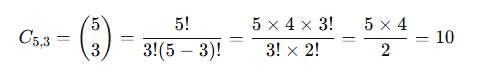
This represents the total number of pairs (i,j) such that i<j and j<i, proving the result.

2.2.a) The formula for Cn,3 the number of ways to choose 3 items from nnn items without replication, is given by the combination formula:

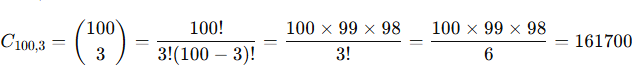


Here, () is the number of combinations (or subsets) of size 3 that can be selected from a total of n items, where order does not matter and repetition is not allowed.

1. **For C5,3:**



1. **For C100,3:**



Thus:

* C5,3=10
* C100,3=161700C

These values represent the number of unique sets of 3 items that can be chosen from 5 items and 100 items, respectively.

2.2.b)

Python:

triplets = [(i, j, k) for i in range(5) for j in range(i+1, 5) for k in range(j+1, 5)]

print(triplets)

output:

[(0, 1, 2), (0, 1, 3), (0, 1, 4), (0, 2, 3), (0, 2, 4), (0, 3, 4), (1, 2, 3), (1, 2, 4), (1, 3, 4), (2, 3, 4)]

2.2.c)

Python:

triplets = {frozenset([i, j, k]) for i in range(5) for j in range(i+1, 5) for k in range(j+1, 5)}

print(triplets)

output:

frozenset({0, 1, 2})

frozenset({0, 1, 3})

frozenset({0, 1, 4})

frozenset({0, 2, 3})

frozenset({0, 2, 4})

frozenset({0, 3, 4})

frozenset({1, 2, 3})

frozenset({1, 2, 4})

frozenset({1, 3, 4})

frozenset({2, 3, 4})

2.3)

Output lst1:

[{(0, 0)},

{(0, 1), (1, 0)},

{(0, 2), (2, 0)},

{(0, 3), (3, 0)},

{(1, 1)},

{(1, 2), (2, 1)},

{(1, 3), (3, 1)},

{(2, 2)},

{(2, 3), (3, 2)},

{(3, 3)}]

Output lst2:

[{(0, 1), (1, 0)},

{(0, 2), (2, 0)},

{(0, 3), (3, 0)},

{(1, 2), (2, 1)},

{(1, 3), (3, 1)},

{(2, 3), (3, 2)}]

2.4)

Sort according to the first coordinate:

Python:

L.sort(key=lambda x: x[0])

print(L)

Output:

[(0, 0, 6), (1, 2, 3), (4, 1, 5)]

Sort according to the third coordinate:

Python:

L.sort(key=lambda x: x[2])

print(L)

Output:

[(1, 2, 3), (4, 1, 5), (0, 0, 6)]

Sort in descending order using the second coordinate:

Python:

L.sort(key=lambda x: x[1], reverse=True)

print(L)

Output:

[(1, 2, 3), (0, 0, 6), (4, 1, 5)]