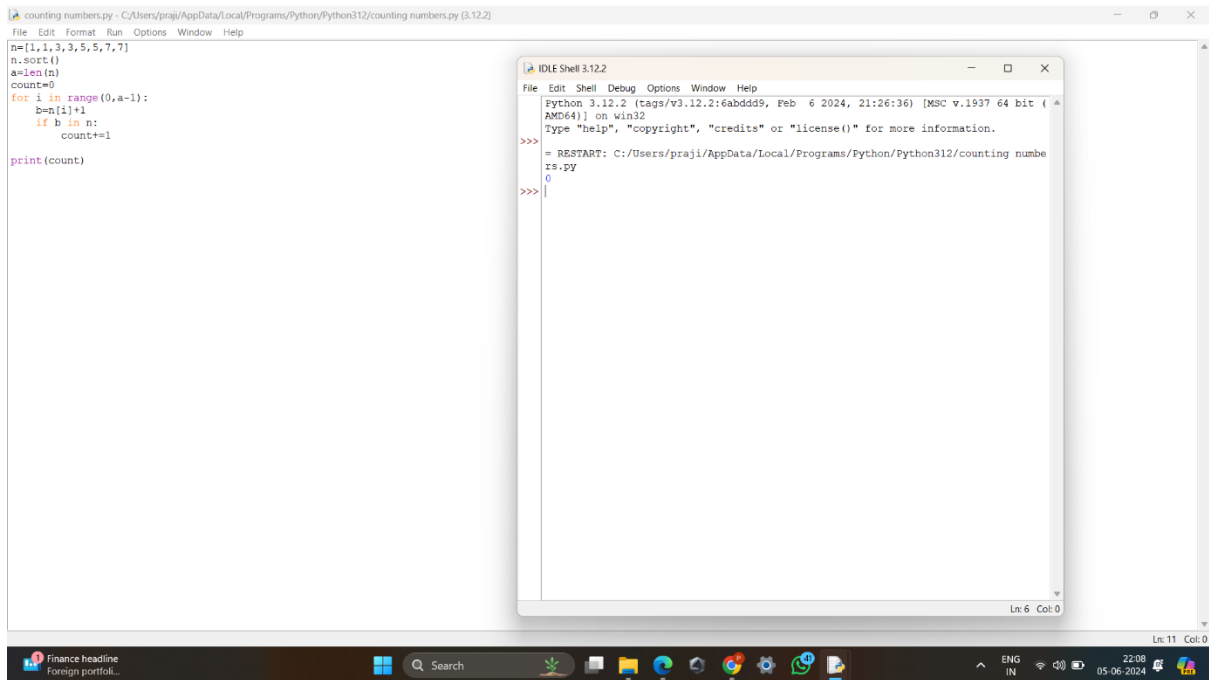


1. Counting Elements Given an integer array arr, count how many elements x there are, such that x + 1 is also in arr. If there are duplicates in arr, count them separately. Example Input: arr = [1,2,3] Output: 2 Explanation: 1 and 2 are counted cause 2 and 3 are in arr. Example 2: Input: arr = [1,1,3,3,5,5,7,7] Output: 0 Explanation: No numbers are counted, cause there is no 2, 4, 6, or 8 in arr.



```
counting numbers.py - C:/Users/praji/AppData/Local/Programs/Python/Python312/counting numbers.py (3.12.2)
File Edit Format Run Options Window Help
n=[1,1,3,3,5,5,7,7]
n.sort()
a=len(n)
count=0
for i in range(0,a-1):
    b=n[i]+1
    if b in n:
        count+=1
print(count)
```

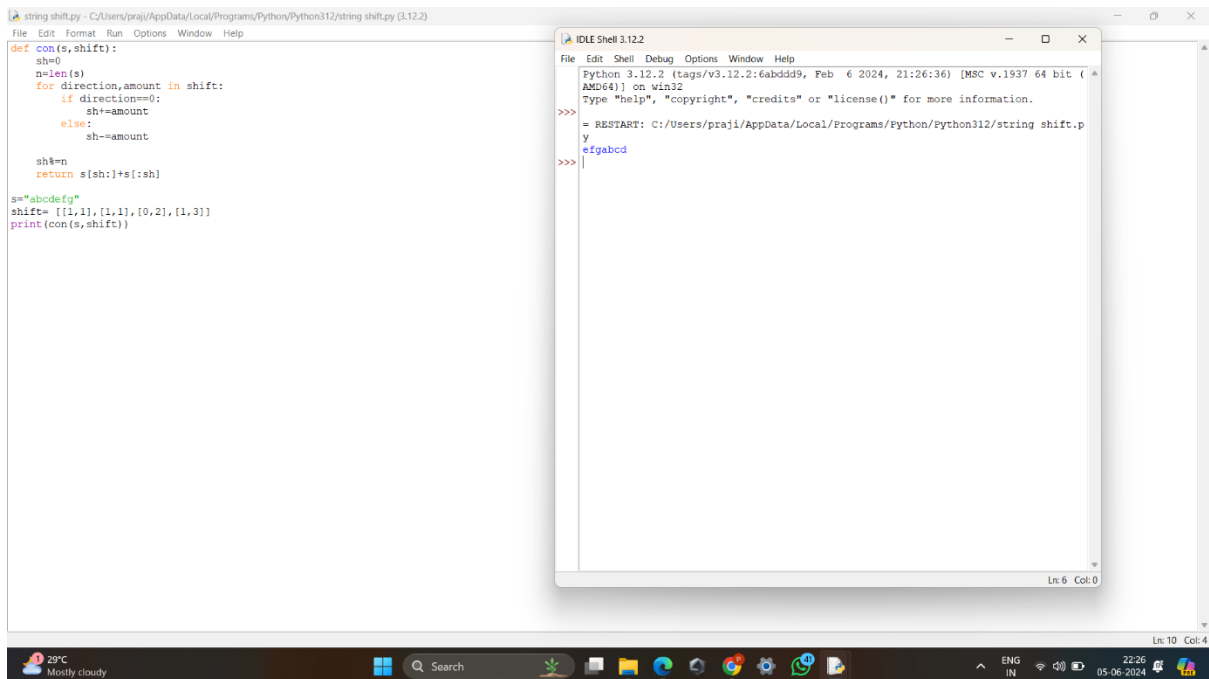
```
IDLE Shell 3.12.2
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/praji/AppData/Local/Programs/Python/Python312/counting numbers.py
0
>>>
```

Time: $O(n)$

2. Perform String Shifts You are given a string *s* containing lowercase English letters, and a matrix *shift*, where *shift*[*i*] = [*direction**i*, *amount**i*]:

- *direction**i* can be 0 (for left shift) or 1 (for right shift).
- *amount**i* is the amount by which string *s* is to be shifted.
- A left shift by 1 means remove the first character of *s* and append it to the end.
- Similarly, a right shift by 1 means remove the last character of *s* and add it to the beginning.

Return the final string after all operations.



The screenshot shows a Python IDE with two windows. The main window displays a Python script for string shifts, and a smaller window shows the execution output.

```
string shift.py - C:/Users/praji/AppData/Local/Programs/Python/Python312/string shift.py (3.12.2)
File Edit Format Run Options Window Help
def con(s, shift):
    sh=0
    n=len(s)
    for direction, amount in shift:
        if direction==0:
            sh+=amount
        else:
            sh-=amount
    sh%=n
    return s[sh:]+s[:sh]
s="abcdefg"
shift= [[1,1],[1,1],[0,2],[1,3]]
print(con(s, shift))
```

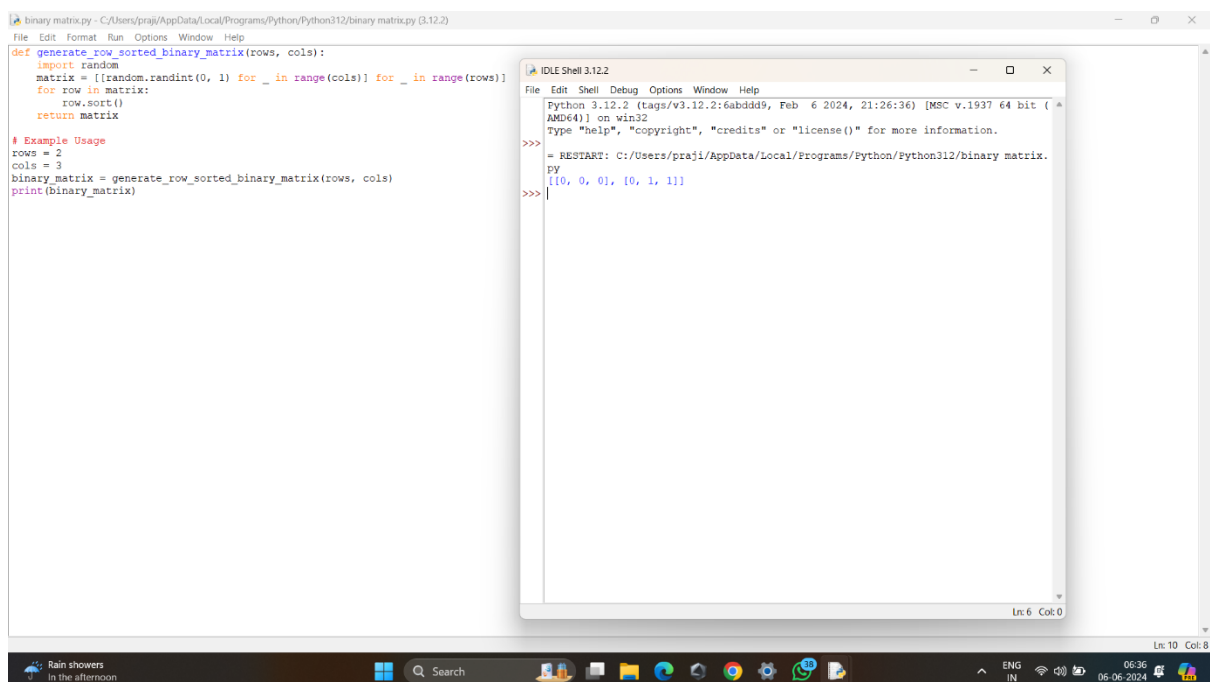
The execution window shows the following output:

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:/Users/praji/AppData/Local/Programs/Python/Python312/string shift.p
y
efgabcd
>>>
```

Time: $O(n)$

3. Leftmost Column with at Least a One A row-sorted binary matrix means that all elements are 0 or 1 and each row of the matrix is sorted in non-decreasing order. Given a row-sorted binary matrix `binaryMatrix`, return the index (0-indexed) of the leftmost column with a 1 in it. If such an index does not exist, return -1. You can't access the Binary Matrix directly. You may only access the matrix using a `BinaryMatrix` interface:

- `BinaryMatrix.get(row, col)` returns the element of the matrix at index `(row, col)` (0-indexed).
- `BinaryMatrix.dimensions()` returns the dimensions of the matrix as a list of 2 elements `[rows, cols]`, which means the matrix is `rows x cols`. Submissions making more than 1000 calls to `BinaryMatrix.get` will be judged Wrong Answer. Also, any solutions that attempt to circumvent the judge will result in disqualification. For custom testing purposes, the input will be the entire binary matrix `mat`. You will not have access to the binary matrix directly.



The screenshot shows a Python IDE with two windows. The main window displays a script named `binary_matrix.py` that defines a function `generate_row_sorted_binary_matrix` to create a row-sorted binary matrix. It includes an example usage section with `rows = 2`, `cols = 3`, and prints the resulting matrix. An interactive shell window titled "IDLE Shell 3.12.2" shows the execution of the script, displaying the output: `[[0, 0, 0], [0, 1, 1]]`. The taskbar at the bottom shows the system clock as 06:36 on 06-06-2024.

```
def generate_row_sorted_binary_matrix(rows, cols):
    import random
    matrix = [[random.randint(0, 1) for _ in range(cols)] for _ in range(rows)]
    for row in matrix:
        row.sort()
    return matrix

# Example Usage
rows = 2
cols = 3
binary_matrix = generate_row_sorted_binary_matrix(rows, cols)
print(binary_matrix)
```

```
>>>
= RESTART: C:/Users/praji/AppData/Local/Programs/Python/Python312/binary matrix.
PY
[[0, 0, 0], [0, 1, 1]]
>>>
```

Time: $O(m \cdot n)$

6. Kids With the Greatest Number of Candies There are n kids with candies. You are given an integer array `candies`, where each `candies[i]` represents the number of candies the i th kid has, and an integer `extraCandies`, denoting the number of extra candies that you have. Return a boolean array `result` of length n , where `result[i]` is `true` if, after giving the i th kid all the `extraCandies`, they will have the greatest number of candies among all the kids, or `false` otherwise. Note that multiple kids can have the greatest number of candies.

```
candy.py - C:/Users/praji/AppData/Local/Programs/Python/Python312/candy.py (3.12.2)
File Edit Format Run Options Window Help
candies=[2,3,5,1,3]
extraCandies=3

n=len(candies)
res=[]

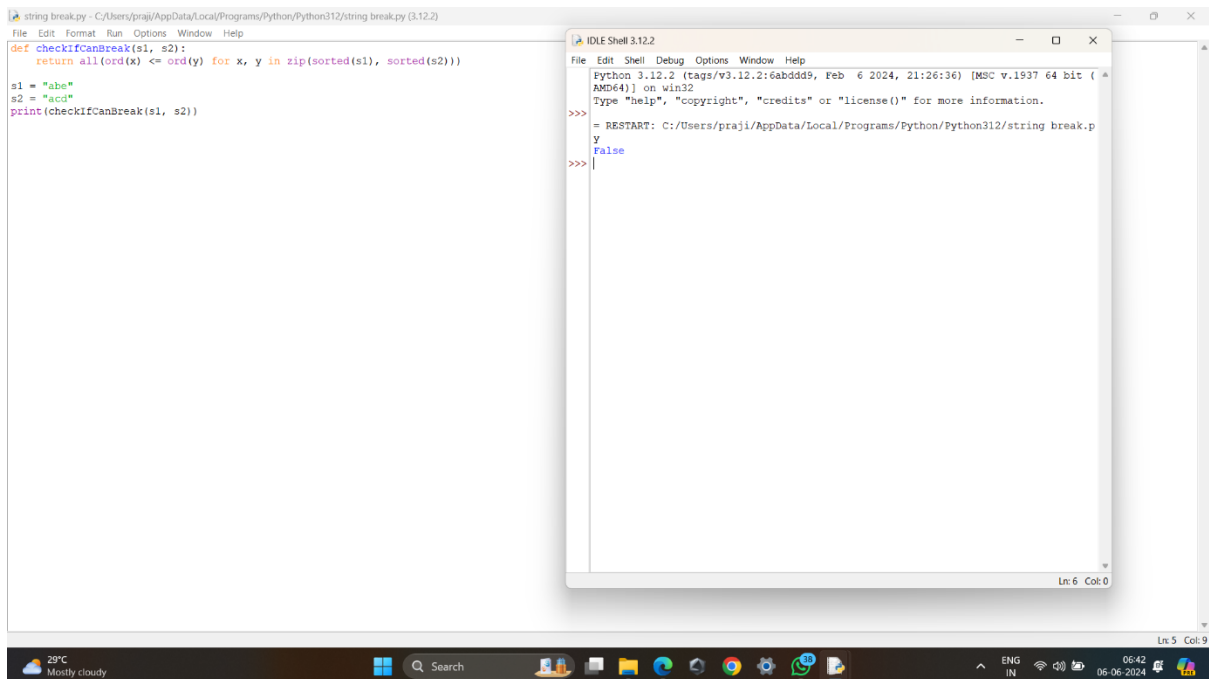
maxcandies=max(candies)

for i in range(0,n):
    a=candies[i]+extraCandies
    if(a>=maxcandies):res.append("True")
    else:res.append("False")
print(res)
```

```
IDLE Shell 3.12.2
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/praji/AppData/Local/Programs/Python/Python312/candy.py
>>> ['True', 'True', 'True', 'False', 'True']
>>>
```

Time: $O(n)$

8. Check If a String Can Break Another String Given two strings: s1 and s2 with the same size, check if some permutation of string s1 can break some permutation of string s2 or vice-versa. In other words s2 can break s1 or vice-versa. A string x can break string y (both of size n) if $x[i] \geq y[i]$ (in alphabetical order) for all i between 0 and n-1.



The screenshot shows a Python IDE with a script titled 'string break.py'. The script defines a function 'checkIfCanBreak(s1, s2)' that returns 'all(ord(x) <= ord(y) for x, y in zip(sorted(s1), sorted(s2)))'. It then sets s1 = 'abe' and s2 = 'acd', and prints the result of 'checkIfCanBreak(s1, s2)'. An IDLE Shell window is open, showing the execution of the script, which outputs 'False'.

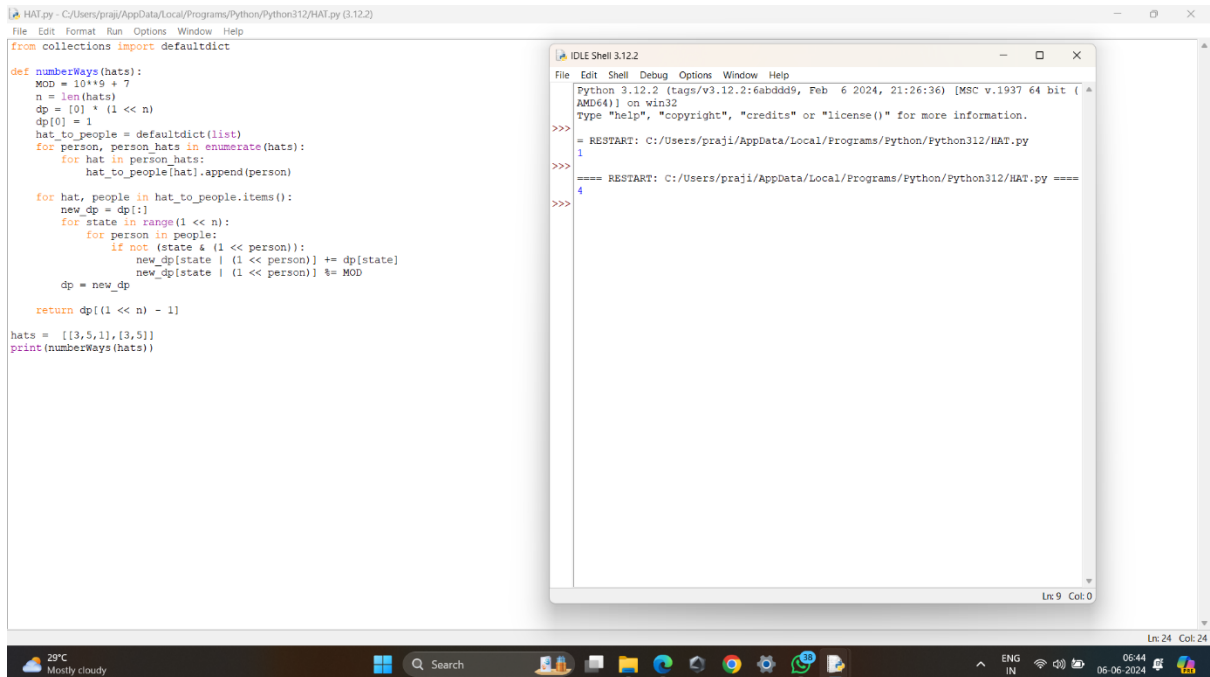
```
string break.py - C:/Users/praji/AppData/Local/Programs/Python/Python312/string break.py (3.12.2)
File Edit Format Run Options Window Help
def checkIfCanBreak(s1, s2):
    return all(ord(x) <= ord(y) for x, y in zip(sorted(s1), sorted(s2)))

s1 = "abe"
s2 = "acd"
print(checkIfCanBreak(s1, s2))

IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:/Users/praji/AppData/Local/Programs/Python/Python312/string break.py
False
>>>
```

Time: $O(n)$

9. Number of Ways to Wear Different Hats to Each Other There are n people and 40 types of hats labeled from 1 to 40. Given a 2D integer array `hats`, where `hats[i]` is a list of all hats preferred by the i th person. Return the number of ways that the n people wear different hats to each other. Since the answer may be too large, return it modulo $10^9 + 7$.



```
HAT.py - C:/Users/praji/AppData/Local/Programs/Python/Python312/HAT.py (3.12.2)
File Edit Format Run Options Window Help
from collections import defaultdict

def numberWays(hats):
    MOD = 10**9 + 7
    n = len(hats)
    dp = [0] * (1 << n)
    dp[0] = 1
    hat_to_people = defaultdict(list)
    for person, person_hats in enumerate(hats):
        for hat in person_hats:
            hat_to_people[hat].append(person)

    for hat, people in hat_to_people.items():
        new_dp = dp[:]
        for state in range(1 << n):
            for person in people:
                if not (state & (1 << person)):
                    new_dp[state | (1 << person)] += dp[state]
                    new_dp[state | (1 << person)] %= MOD
        dp = new_dp

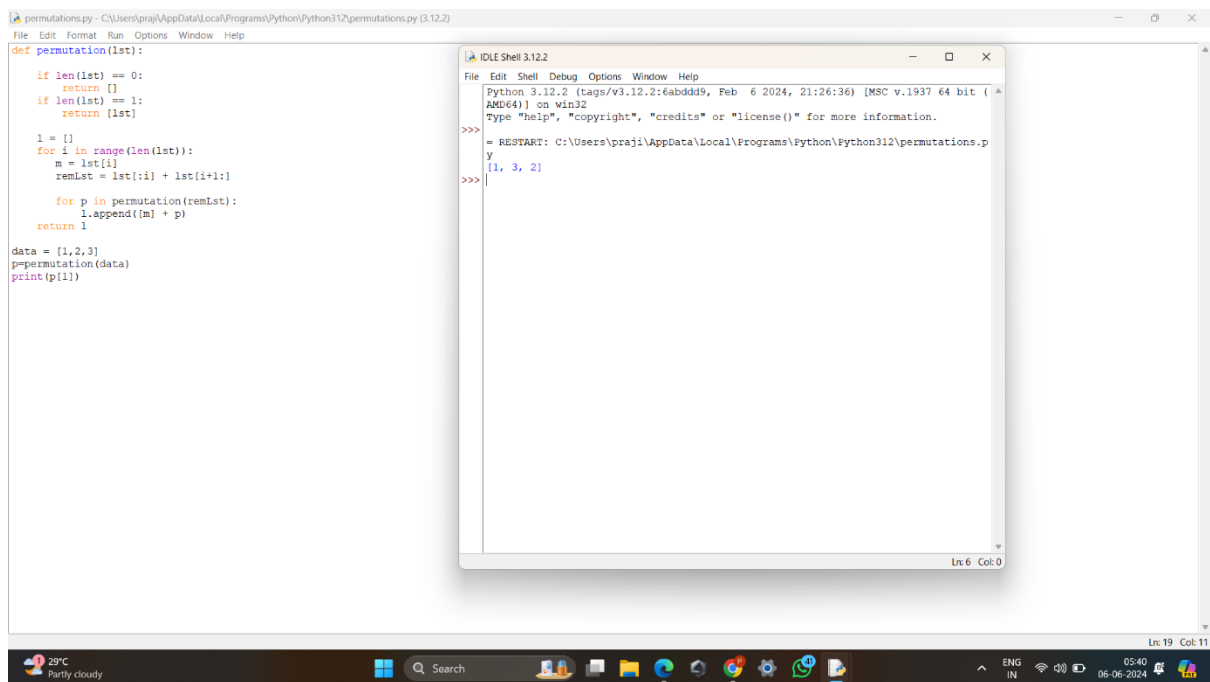
    return dp[(1 << n) - 1]

hats = [[3,5,1],[3,5]]
print(numberWays(hats))

IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/praji/AppData/Local/Programs/Python/Python312/HAT.py
1
>>>
==== RESTART: C:/Users/praji/AppData/Local/Programs/Python/Python312/HAT.py ====
4
>>>
```

Time: $O(n)$

10. Next Permutation A permutation of an array of integers is an arrangement of its members into a sequence or linear order. • For example, for $arr = [1,2,3]$, the following are all the permutations of arr : $[1,2,3]$, $[1,3,2]$, $[2, 1, 3]$, $[2, 3, 1]$, $[3,1,2]$, $[3,2,1]$. The next permutation of an array of integers is the next lexicographically greater permutation of its integer. More formally, if all the permutations of the array are sorted in one container according to their lexicographical order, then the next permutation of that array is the permutation that follows it in the sorted container. If such arrangement is not possible, the array must be rearranged as the lowest possible order (i.e., sorted in ascending order). • For example, the next permutation of $arr = [1,2,3]$ is $[1,3,2]$. • Similarly, the next permutation of $arr = [2,3,1]$ is $[3,1,2]$. • While the next permutation of $arr = [3,2,1]$ is $[1,2,3]$ because $[3,2,1]$ does not have a lexicographical larger rearrangement. Given an array of integers $nums$, find the next permutation of $nums$. The replacement must be in place and use only constant extra memory.



The screenshot shows a Python IDE with two windows. The main window displays a Python script for finding the next permutation. The script defines a recursive function `permutation` that generates all permutations of a list. It then applies this function to the input list `[1,2,3]` and prints the result. The output window shows the execution of the script, displaying the next permutation `[1, 3, 2]`.

```
def permutation(lst):
    if len(lst) == 0:
        return []
    if len(lst) == 1:
        return [lst]

    l = []
    for i in range(len(lst)):
        m = lst[i]
        remList = lst[:i] + lst[i+1:]

        for p in permutation(remList):
            l.append([m] + p)
    return l

data = [1,2,3]
p=permutation(data)
print(p[1])
```

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\praji\AppData\Local\Programs\Python\Python312\permutations.p
Y
[1, 3, 2]
>>>
```

Time: $O(n^2)$