1.Counting Elements

```
def count_elements(arr):
    elements = set(arr)
    count = 0
    for x in arr:
        if x + 1 in elements:
            count += 1
    return count
arr = [1, 2, 3]
print(count_elements(arr))
```

```
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:/DAA/Couting Elements.py
2
>>>
```

2.Perform String Shifts

```
def stringShift(s, shift):
        val = 0
        for i in range(len(shift)):
                val += -shift[i][1] if shift[i][0] == 0 else shift[i][1]
        Len = len(s)
        val = val % Len
        result = ""
        if (val > 0):
                result = s[Len - val:Len] + s[0: Len - val]
        else:
                result = s[-val: Len] + s[0: -val]
        print(result)
s = "abc"
shift = [
        [ 0, 1 ],
        [ 1, 2 ]
]
stringShift(s, shift)
```

```
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:/DAA/Perform String Shifts.py
cab
>>>
```

3.Leftmost Column With at Least a One

```
import sys
N = 3
def search(mat, n, m):
        a = sys.maxsize
        for i in range (n):
                low = 0
                high = m - 1
                ans = sys.maxsize
                while (low <= high):
                        mid = (low + high) // 2
                        if (mat[i][mid] == 1):
                                if (mid == 0):
                                        ans = 0
                                        break
                                elif (mat[i][mid - 1] == 0):
                                        ans = mid
                                        break
                        if (mat[i][mid] == 1):
                                high = mid - 1
                        else:
                                low = mid + 1
                if (ans < a):
                        a = ans
        if (a == sys.maxsize):
                return -1
        return a + 1
mat = [[0, 0, 0],
           [0, 0, 1],
           [0, 1, 1]]
print(search(mat, 3, 3))
```

```
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:/DAA/Leftmost Column With at Least a One.py
2
>>>
```

4.First Unique Number

```
def firstNonRepeating(arr, n):
        for i in range(n):
                j = 0
                while(j < n):
                        if (i != j and arr[i] == arr[j]):
                                break
                        j += 1
                if (j == n):
                        return arr[i]
        return -1
arr = [9, 4, 9, 6, 7, 4]
n = len(arr)
print(firstNonRepeating(arr, n))
```

```
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:/DAA/First Unique Number.py
6
>>>
```

5. Check If a String Is a Valid Sequence from Root to Leaves Path in a Binary Tree Given a binary tree where each path going from the root to any leaf form a valid sequence, check if a given string is a valid sequence in such binary tree. We get the given string from the concatenation of an array of integers arr and the concatenation of all values of the nodes along a path results in a sequence in the given binary tree.

```
File  Edit  Format  Run  Options  Window  Help
class Node:
        def __init__(self, val):
                self.val = val
                self.left = None
                self.right = None
def existPathUtil(root, arr, n, index):
        if not root or index == n:
                return False

        # If current node is leaf
        if not root.left and not root.right:
                if root.val == arr[index] and index == n-1:
                        return True
                return False

        return ((index < n) and (root.val == arr[index]) and
                        (existPathUtil(root.left, arr, n, index+1) or
                        existPathUtil(root.right, arr, n, index+1)))
def existPath(root, arr, n, index):
        if not root:
                return (n == 0)
        return existPathUtil(root, arr, n, 0)
arr = [5, 8, 6, 7]
n = len(arr)
root = Node(5)
root.left = Node(3)
root.right = Node(8)
root.left.left = Node(2)
root.left.right = Node(4)
root.left.left.left = Node(1)
root.right.left = Node(6)
root.right.left.right = Node(7)
if existPath(root, arr, n, 0):
    print("Path Exists")
else:
    print("Path does not Exist")
```

```
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/Vinot/AppData/Local/Programs/Python/Python312/valid path tre
e.py
Path Exists
>>>
Ln: 6  Col: 0
```

6.Kids With the Greatest Number of Candies

```python
def kidsWithCandies(candies, extraCandies):
    result = []
    for i in range(len(candies)):
        if candies[i] + extraCandies >= max(candies):
            result.append(True)
        else:
            result.append(False)
    return result
candies = [2,3,5,1,3]
extraCandies = 3
print(kidsWithCandies(candies, extraCandies))
```

```
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:/DAA/Candies.py
[True, True, True, False, True]
>>>
```

7.Max Difference You Can Get from Changing an Integer

```python
def maxDifference(num: int) -> int:
    str_num = str(num)
    def replace_digit(s, x, y):
        return ''.join([y if char == x else char for char in s])
    min_val = float('inf')
    max_val = float('-inf')
    for x in '0123456789':
        for y in '0123456789':
            if x == y:
                continue
            new_num_str = replace_digit(str_num, x, y)
            if new_num_str[0] != '0':
                new_num = int(new_num_str)
                min_val = min(min_val, new_num)
                max_val = max(max_val, new_num)
    return max_val - min_val
num = 555
print("Max Difference:", maxDifference(num))
```

```
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:/DAA/Max Difference You can Get From Changing an Integer.py
Max Difference: 888
>>>
```

8.Check If a string Can Break Another String

```python
def can_break(s1, s2):
    s1_sorted = sorted(s1)
    s2_sorted = sorted(s2)
    s1_breaks_s2 = all(c1 >= c2 for c1, c2 in zip(s1_sorted, s2_sorted))
    s2_breaks_s1 = all(c2 >= c1 for c1, c2 in zip(s1_sorted, s2_sorted))
    return s1_breaks_s2 or s2_breaks_s1
s1 = "leetcodee"
s2 = "interview"
print(can_break(s1, s2))
```

```
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:/DAA/Check If a String Can Break Another String.py
True
>>>
```

9.Number of Ways to Wear Different Hats To Each Other

```python
def numberWays(hats):
    MOD = 10**9 + 7
    n = len(hats)
    hat_to_people = [[] for _ in range(41)]
    for i, hat_list in enumerate(hats):
        for hat in hat_list:
            hat_to_people[hat].append(i)
    dp = [0] * (1 << n)
    dp[0] = 1
    for hat in range(1, 41):
        for mask in range((1 << n) - 1, -1, -1):
            for person in hat_to_people[hat]:
                if mask & (1 << person) == 0:
                    dp[mask | (1 << person)] = (dp[mask | (1 << person)] + dp[mask]) % MOD
    return dp[(1 << n) - 1]
hats = [[3, 4], [4, 5], [5]]
print(numberWays(hats))
```

```
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:/DAA/Number of Ways To Wear Different Hats to Each Other.py
1
>>>
```

10.Next Permutation

```
def next_permutation(nums):
    i = len(nums) - 2
    while i >= 0 and nums[i] >= nums[i + 1]:
        i -= 1
    if i >= 0:
        j = len(nums) - 1
        while nums[j] <= nums[i]:
            j -= 1
        nums[i], nums[j] = nums[j], nums[i]
    nums[i + 1:] = reversed(nums[i + 1:])
nums = [3, 2, 1]
next_permutation(nums)
print(nums)
```

```
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:/DAA/Next Permutation.py ==========
[1, 2, 3]
>>>
```