

Day – 1 : Arrays – I

Problem 1 – Set Matrix Zeros

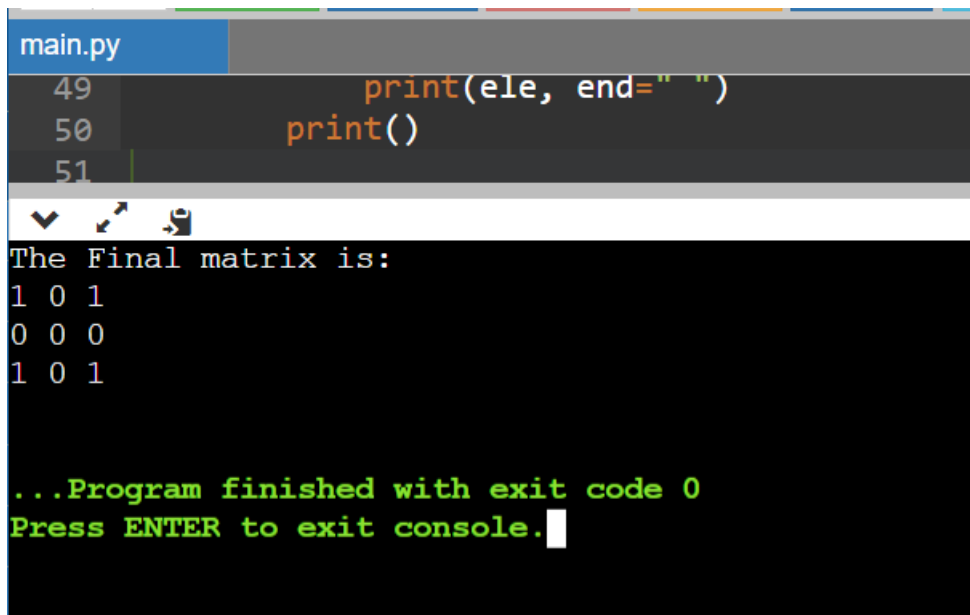
```
def zeroMatrix(matrix, n, m):  
    # int row[n] = {0}; --> matrix[..][0]  
    # int col[m] = {0}; --> matrix[0][..]  
  
    col0 = 1  
    for i in range(n):  
        for j in range(m):  
            if matrix[i][j] == 0:  
                # mark i-th row:  
                matrix[i][0] = 0  
  
                # mark j-th column:  
                if j != 0:  
                    matrix[0][j] = 0  
            else:  
                col0 = 0  
  
    for i in range(1, n):  
        for j in range(1, m):  
            if matrix[i][j] != 0:  
                # check for col & row:  
                if matrix[i][0] == 0 or matrix[0][j] == 0:  
                    matrix[i][j] = 0  
  
    if matrix[0][0] == 0:
```

```
        for j in range(m):
            matrix[0][j] = 0
    if col0 == 0:
        for i in range(n):
            matrix[i][0] = 0

    return matrix
```

```
matrix = [[1, 1, 1], [1, 0, 1], [1, 1, 1]]
n = len(matrix)
m = len(matrix[0])
ans = zeroMatrix(matrix, n, m)
```

```
print("The Final matrix is:")
for row in ans:
    for ele in row:
        print(ele, end=" ")
    print()
```



The screenshot shows a code editor window titled 'main.py' with the following code:

```
49         print(ele, end=" ")
50     print()
51
```

Below the editor is a console window showing the output of the program:

```
The Final matrix is:
1 0 1
0 0 0
1 0 1

...Program finished with exit code 0
Press ENTER to exit console.
```

Problem – 2 : Pascal's Triangle

```
def pascals_triangle_element(r, c):  
    if c == 1 or c == r:  
        return 1  
    else:  
        return pascals_triangle_element(r - 1, c - 1) + pascals_triangle_element(r - 1, c)
```

```
def pascals_triangle_row(n):  
    row = []  
    for i in range(1, n + 1):  
        row.append(pascals_triangle_element(n, i))  
    return row
```

```
def pascals_triangle(n):  
    triangle = []  
    for i in range(1, n + 1):  
        triangle.append(pascals_triangle_row(i))  
    return triangle
```

Variation 1: Print the element at position (r, c) in Pascal's triangle

```
r = 5  
c = 3  
element = pascals_triangle_element(r, c)  
print("Result (Variation 1):", element)
```

Variation 2: Print the n-th row of Pascal's triangle

```
n = 5  
row = pascals_triangle_row(n)  
print("Result (Variation 2):", ' '.join(str(x) for x in row))
```

Variation 3: Print the first n rows of Pascal's triangle

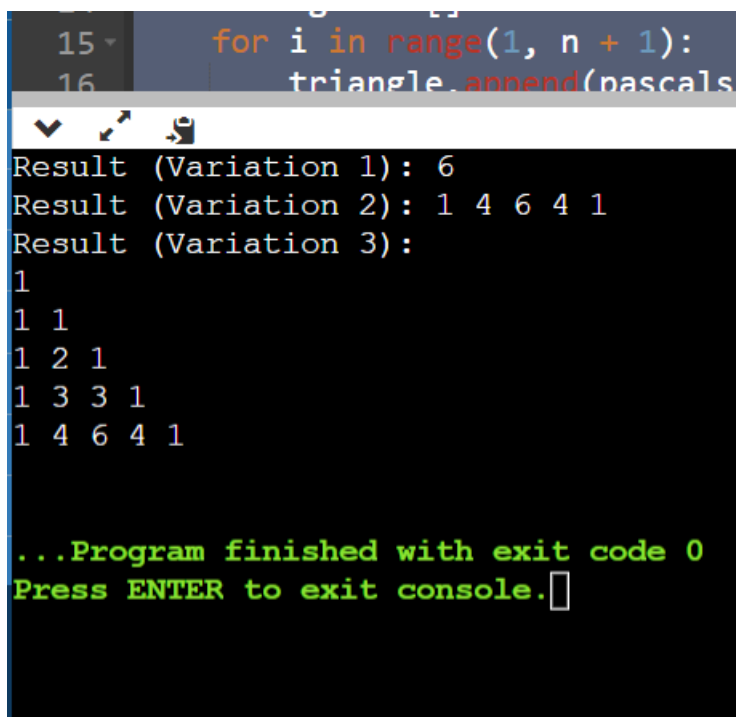
n = 5

triangle = pascals_triangle(n)

print("Result (Variation 3):")

for row in triangle:

print(' '.join(str(x) for x in row))



```
15 for i in range(1, n + 1):
16     triangle.append(pascals

Result (Variation 1): 6
Result (Variation 2): 1 4 6 4 1
Result (Variation 3):
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1

...Program finished with exit code 0
Press ENTER to exit console.
```

Problem – 3 : Next permutation array

def next_permutation(arr):

n = len(arr)

i = n - 2

while i >= 0 and arr[i] >= arr[i+1]:

i -= 1

```
if i >= 0:
    j = n - 1
    while arr[j] <= arr[i]:
        j -= 1
    arr[i], arr[j] = arr[j], arr[i]

    left = i + 1
    right = n - 1
    while left < right:
        arr[left], arr[right] = arr[right], arr[left]
        left += 1
        right -= 1

    return arr
```

```
arr = [1, 3, 2]
next_permuted_arr = next_permutation(arr)
print(next_permuted_arr)
```

```
arr = [3, 2, 1]
next_permuted_arr = next_permutation(arr)
print(next_permuted_arr)
```

```
28
29 arr = [3, 2, 1]
30 next_permuted_arr = next_permutation(arr)
31 print(next_permuted_arr)
32
```

input

```
[2, 1, 3]
[1, 2, 3]
```

...Program finished with exit code 0
Press ENTER to exit console.

Problem – 4 : Kadane's Algorithm

```
def max_subarray_sum(arr):
```

```
    if not arr:
```

```
        return 0
```

```
    currentMax = arr[0]
```

```
    globalMax = arr[0]
```

```
    start, end = 0, 0
```

```
    subarray = [arr[0]]
```

```
    for i in range(1, len(arr)):
```

```
        if arr[i] > arr[i] + currentMax:
```

```
            currentMax = arr[i]
```

```
            start = i
```

```
        else:
```

```
            currentMax += arr[i]
```

```

    if currentMax > globalMax:
        globalMax = currentMax
        end = i
        subarray = arr[start:end+1]

    print("Subarray:", subarray)
    return globalMax

arr = [-2, 1, -3, 4, -1, 2, 1, -5, 4]
max_sum = max_subarray_sum(arr)
print("Maximum subarray sum:", max_sum)

```

```

14 else:
15     currentMax = arr[i]

```

input

Subarray: [4, -1, 2, 1]
Maximum subarray sum: 6

...Program finished with exit code 0
Press ENTER to exit console.

Problem – 5 : Sort an array of 0's, 1's, and 2's

```

def sortColors(nums):
    low = 0
    mid = 0
    high = len(nums) - 1

    while mid <= high:
        if nums[mid] == 0:
            nums[mid], nums[low] = nums[low], nums[mid]
            low += 1

```

```

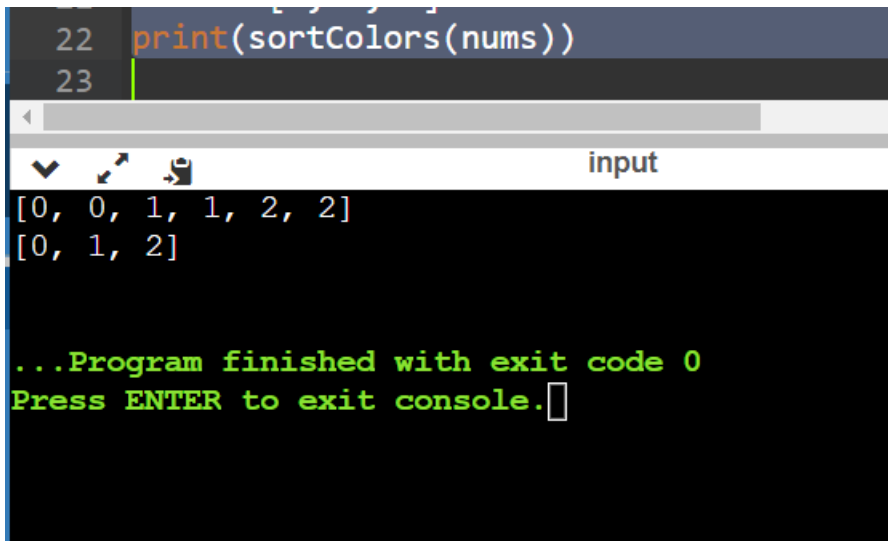
        mid += 1
    elif nums[mid] == 1:
        mid += 1
    else:
        nums[mid], nums[high] = nums[high], nums[mid]
        high -= 1

    return nums

nums = [2, 0, 2, 1, 1, 0]
print(sortColors(nums))

nums = [2, 0, 1]
print(sortColors(nums))

```



```

22 print(sortColors(nums))
23
input
[0, 0, 1, 1, 2, 2]
[0, 1, 2]

...Program finished with exit code 0
Press ENTER to exit console.

```

Problem – 6 : Stock buy & sell

```

def maxProfit(prices):
    minPrice = float('inf')
    maxProfit = 0

    for price in prices:

```



```
if price < minPrice:  
    minPrice = price  
elif price - minPrice > maxProfit:  
    maxProfit = price - minPrice
```

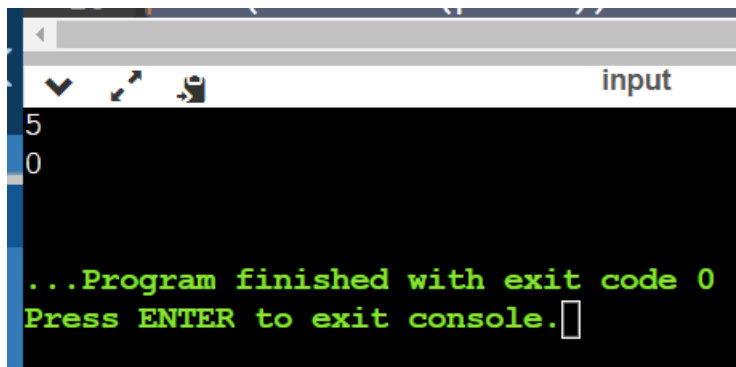
```
return maxProfit
```

```
prices = [7, 1, 5, 3, 6, 4]
```

```
print(maxProfit(prices))
```

```
prices = [7,6,4,3,1]
```

```
print(maxProfit(prices))
```

A screenshot of a terminal window with a dark background. The window title is "input". The output shows the number "5" on the first line and "0" on the second line. Below these, a green message reads "...Program finished with exit code 0" and "Press ENTER to exit console." followed by a small white cursor icon.

```
5  
0  
...Program finished with exit code 0  
Press ENTER to exit console.
```