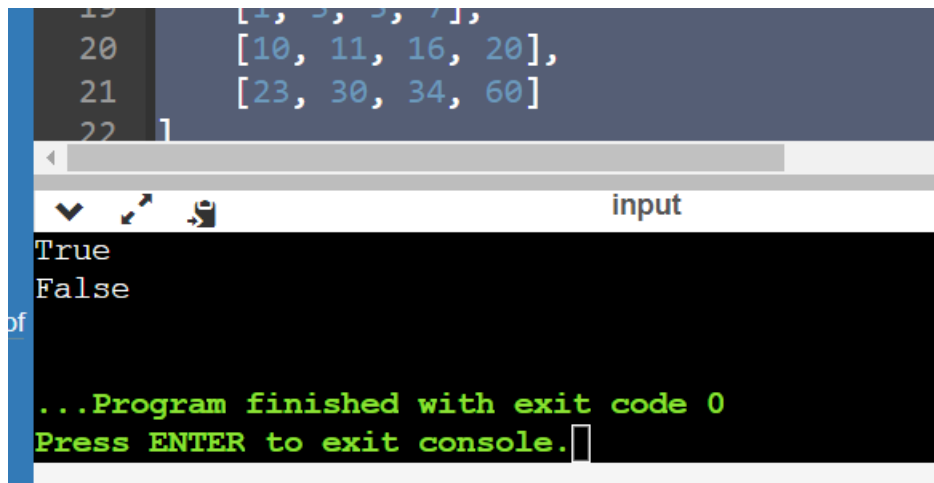


Day – 3 : Arrays- III

Problem 1: Given an m*n 2D matrix and an integer, write a program to find if the given integer exists in the matrix.

```
def search_matrix(matrix, target):  
    if not matrix or not matrix[0]:  
        return False  
  
    rows = len(matrix)  
    cols = len(matrix[0])  
  
    row = 0  
    col = cols - 1  
  
    while row < rows and col >= 0:  
        if matrix[row][col] == target:  
            return True  
        elif matrix[row][col] > target:  
            col -= 1  
        else:  
            row += 1  
  
    return False  
  
matrix1 = [  
    [1, 3, 5, 7],  
    [10, 11, 16, 20],  
    [23, 30, 34, 60]  
]  
  
target1 = 3  
  
print(search_matrix(matrix1, target1))  
  
matrix2 = [  
    [1, 3, 5, 7],  
    [10, 11, 16, 20],
```

```
[23, 30, 34, 60]  
]  
target2 = 13  
print(search_matrix(matrix2,target2))
```



The screenshot shows a code editor with a dark theme. The code in the editor is as follows:

```
19 [1, 3, 5, 7],  
20 [10, 11, 16, 20],  
21 [23, 30, 34, 60]  
22 ]
```

Below the code editor is a terminal window. The terminal shows the output of the program:

```
True  
False
```

At the bottom of the terminal, there is a green message:

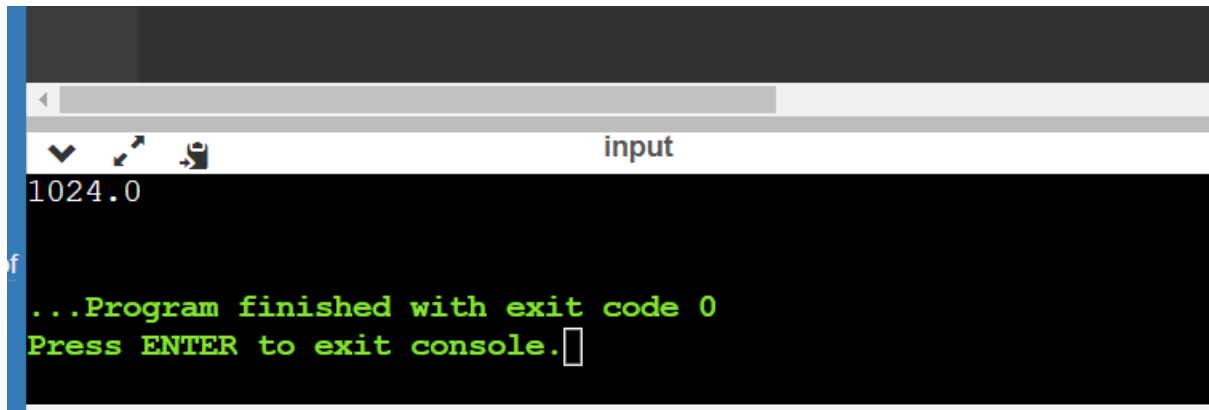
```
...Program finished with exit code 0  
Press ENTER to exit console.
```

Problem -2: Given a double x and integer n, calculate x raised to power n.
Basically Implement pow(x, n).

```
def pow(x, n):  
    if n == 0:  
        return 1.0  
    elif n < 0:  
        x = 1 / x  
        n = -n  
  
    result = 1.0  
    while n > 0:  
        if n % 2 == 1:  
            result *= x  
        x *= x  
        n //= 2
```

```
    return result

x = 2.00000
n = 10
print(pow(x, n))
```

A screenshot of a terminal window. The window has a title bar with the word "input" in the center. Below the title bar, the text "1024.0" is displayed in white on a black background. At the bottom of the terminal, green text reads "...Program finished with exit code 0" and "Press ENTER to exit console." followed by a small white cursor icon. The terminal window is set against a dark background with a blue vertical bar on the left side.

Problem 3: Given an array of **N integers**, write a program to return an element that occurs more than **N/2** times in the given array. You may consider that such an element always exists in the array.

```
def majority_element(nums):
    candidate = None
    count = 0

    for num in nums:
        if count == 0:
            candidate = num
            count = 1
        elif num == candidate:
            count += 1
        else:
            count -= 1

    count = 0
```

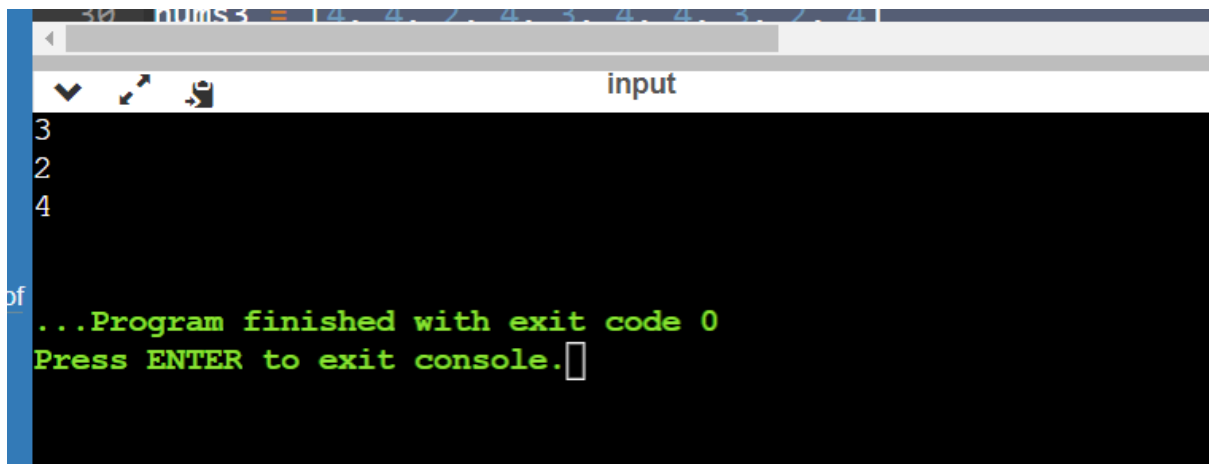
```
for num in nums:
    if num == candidate:
        count += 1
```

```
if count > len(nums) // 2:
    return candidate
```

```
nums1 = [3, 2, 3]
print(majority_element(nums1))
```

```
nums2 = [2, 2, 1, 1, 1, 2, 2]
print(majority_element(nums2))
```

```
nums3 = [4, 4, 2, 4, 3, 4, 4, 3, 2, 4]
print(majority_element(nums3))
```



The screenshot shows a Jupyter Notebook interface. At the top, a code cell contains the definition of the `majority_element` function and three test cases. Below the code cell, the console output is visible. It shows the results of the function calls for `nums1`, `nums2`, and `nums3`. The output for `nums1` is 3, for `nums2` is 2, and for `nums3` is 4. The console also displays the message "...Program finished with exit code 0" and "Press ENTER to exit console."

```
3
2
4
...Program finished with exit code 0
Press ENTER to exit console.
```

Problem 4: Given an array of N integers. Find the elements that appear more than $N/3$ times in the array. If no such element exists, return an empty vector.

```
def majority_element(nums):
```

```
candidate1, candidate2 = None, None
```

```
count1, count2 = 0, 0
```

```
for num in nums:
```

```
    if candidate1 == num:
```

```
        count1 += 1
```

```
    elif candidate2 == num:
```

```
        count2 += 1
```

```
    elif count1 == 0:
```

```
        candidate1, count1 = num, 1
```

```
    elif count2 == 0:
```

```
        candidate2, count2 = num, 1
```

```
    else:
```

```
        count1 -= 1
```

```
        count2 -= 1
```

```
count1, count2 = 0, 0
```

```
for num in nums:
```

```
    if num == candidate1:
```

```
        count1 += 1
```

```
    elif num == candidate2:
```

```
        count2 += 1
```

```
n = len(nums)

result = []

if count1 > n // 3:

    result.append(candidate1)

if count2 > n // 3:

    result.append(candidate2)

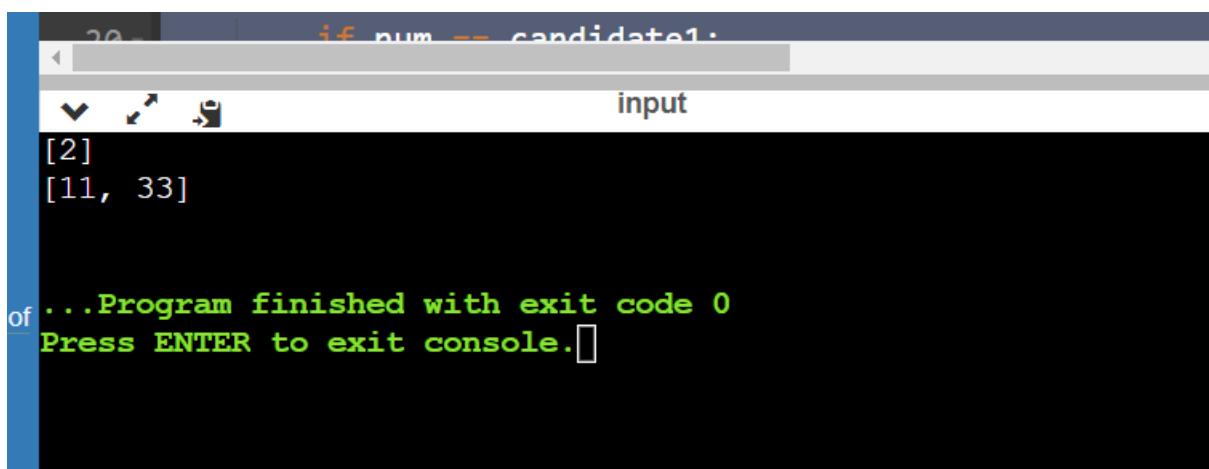

return result
```

```
nums1 = [1, 2, 2, 3, 2]

print(majority_element(nums1))
```

```
nums2 = [11, 33, 33, 11, 33, 11]

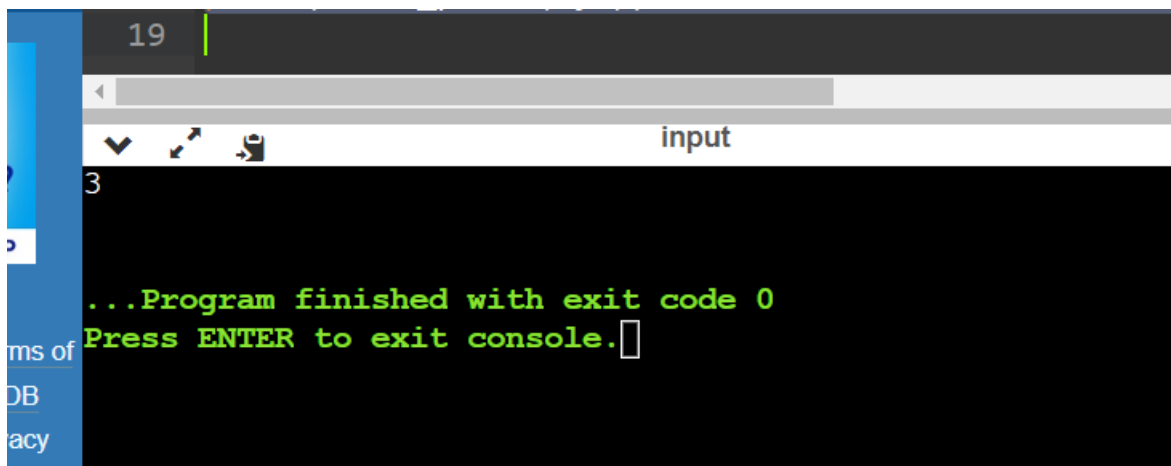
print(majority_element(nums2))
```

A screenshot of a terminal window with a dark background. At the top, there's a header bar with a search icon, a refresh icon, and a file icon, followed by the word "input". Below this, the output of the program is displayed in green text: "[2]" on the first line and "[11, 33]" on the second line. At the bottom, a message reads "...Program finished with exit code 0" followed by "Press ENTER to exit console." and a cursor icon. On the left side of the terminal, there is a blue vertical bar with the word "of" next to it.

```
20 if num == candidate1:
input
[2]
[11, 33]
...Program finished with exit code 0
Press ENTER to exit console.
```

Problem 5: Given a matrix **m X n**, count paths from left-top to the right bottom of a matrix with the constraints that from each cell you can either only move to the rightward direction or the downward direction.

```
def count_paths(m, n):  
    dp = [[0] * n for _ in range(m)]  
    dp[0][0] = 1  
  
    for j in range(1, n):  
        dp[0][j] = 1  
  
    for i in range(1, m):  
        dp[i][0] = 1  
  
    for i in range(1, m):  
        for j in range(1, n):  
            dp[i][j] = dp[i-1][j] + dp[i][j-1]  
  
    return dp[m-1][n-1]  
  
m = 2  
n = 3  
print(count_paths(m,n))
```



```
19  
input  
3  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Problem 6: Given an array of numbers, you need to return the count of reverse pairs. **Reverse Pairs** are those pairs where $i < j$ and $arr[i] > 2 * arr[j]$.

What is an inversion of an array? Definition: for all $i \& j < \text{size of array}$, if $i < j$ then you have to find pair $(A[i], A[j])$ such that $A[j] < A[i]$.

```
def mergeSortAndCount(arr, start, end):
```

```
    if start == end:
```

```
        return 0
```

```
    mid = (start + end) // 2
```

```
    countLeft = mergeSortAndCount(arr, start, mid)
```

```
    countRight = mergeSortAndCount(arr, mid + 1, end)
```

```
    countPairs = 0
```

```
    i = start
```

```
    j = mid + 1
```

```
    while i <= mid and j <= end:
```

```
        if arr[i] > 2 * arr[j]:
```

```
            countPairs += (mid - i + 1)
```

```
            j += 1
```

```
        else:
```

```
            i += 1
```

```
    merged = []
```

```
    i = start
```

```
    j = mid + 1
```

```
    while i <= mid and j <= end:
```

```
        if arr[i] <= arr[j]:
```



```
merged.append(arr[i])
```

```
i += 1
```

```
else:
```

```
merged.append(arr[j])
```

```
j += 1
```

```
while i <= mid:
```

```
merged.append(arr[i])
```

```
i += 1
```

```
while j <= end:
```

```
merged.append(arr[j])
```

```
j += 1
```

```
arr[start:end + 1] = merged
```

```
return countPairs + countLeft + countRight
```

```
arr = [3,2,1,4]
```

```
count = mergeSortAndCount(arr, 0, len(arr) - 1)
```

```
print(count)
```

```
43  
44  
45 arr = [3,2,1,4]  
46 count = mergeSortAndCount(arr, 0, len(arr) - 1)  
47 print(count)  
48
```

input

1

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