

Week-9

Step1: We'll create a frequency table to keep track of the counts. Here's the frequency table for the given input: [1, 1, 1, 2, 2, 3]

Element	Frequency
1	3
2	2
3	1

Step 2: Sort the elements by frequency.

Sorting the elements in the frequency table:

Element	Frequency
1	3
2	2
3	1

Step 3: Take the top k elements, where k = 2 in this case.

The top 2 elements are [1, 2].

Code:

```
import heapq;
from collections import Counter;

def topKFrequent(nums, k):

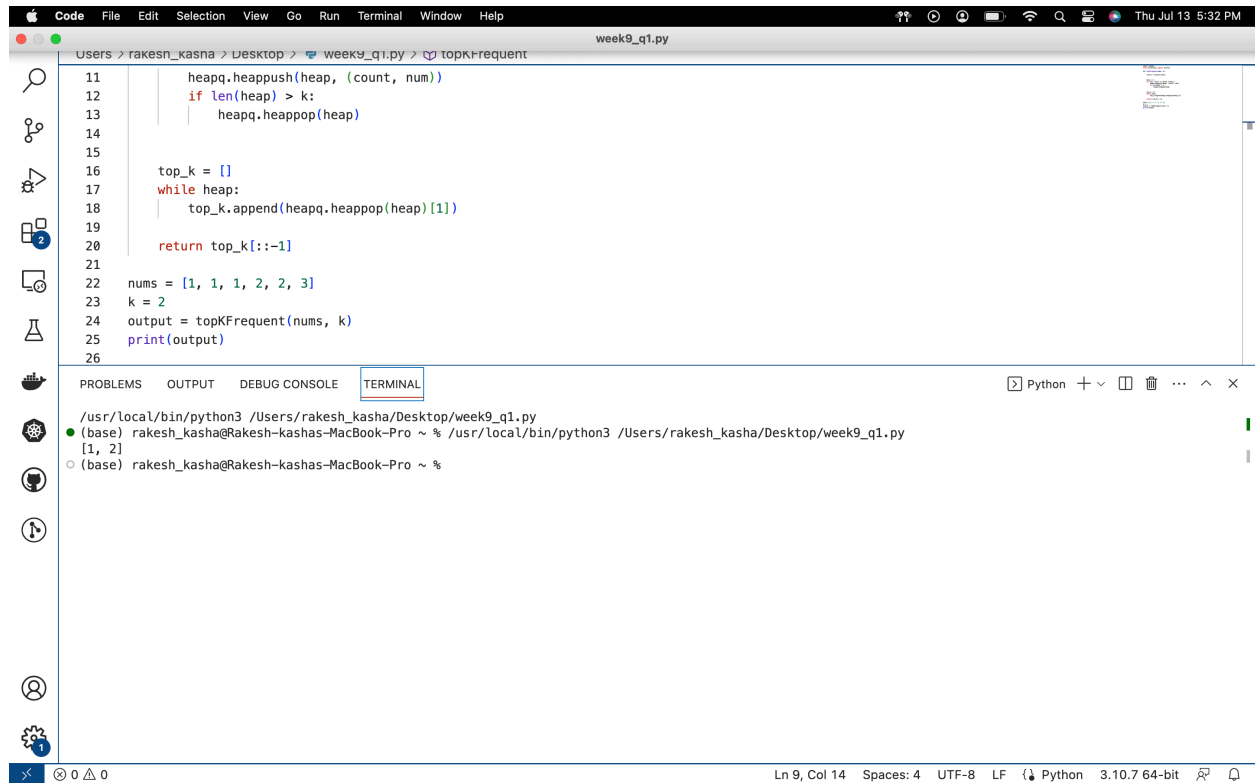
    counts = Counter(nums)

    heap = []
    for num, count in counts.items():
        heapq.heappush(heap, (count, num))
        if len(heap) > k:
            heapq.heappop(heap)

    top_k = []
    while heap:
        top_k.append(heapq.heappop(heap)[1])
```

```
        return top_k[::-1]

nums = [1, 1, 1, 2, 2, 3]
k = 2
output = topKFrequent(nums, k)
print(output)
```



The screenshot shows a Visual Studio Code editor window with a Python file named `week9_q1.py`. The code implements a function `topKFrequent` that uses a heap to find the top `k` frequent elements in a list of numbers. The script is executed, and the output is displayed in the terminal.

```
11         heapq.heappush(heap, (count, num))
12         if len(heap) > k:
13             heapq.heappop(heap)
14
15
16     top_k = []
17     while heap:
18         top_k.append(heapq.heappop(heap)[1])
19
20     return top_k[::-1]
21
22 nums = [1, 1, 1, 2, 2, 3]
23 k = 2
24 output = topKFrequent(nums, k)
25 print(output)
26
```

The terminal output shows the execution of the script, resulting in the output `[1, 2]`.

```
/usr/local/bin/python3 /Users/rakesh_kasha/Desktop/week9_q1.py
(base) rakesh_kasha@Rakesh-kashas-MacBook-Pro ~ % /usr/local/bin/python3 /Users/rakesh_kasha/Desktop/week9_q1.py
[1, 2]
(base) rakesh_kasha@Rakesh-kashas-MacBook-Pro ~ %
```