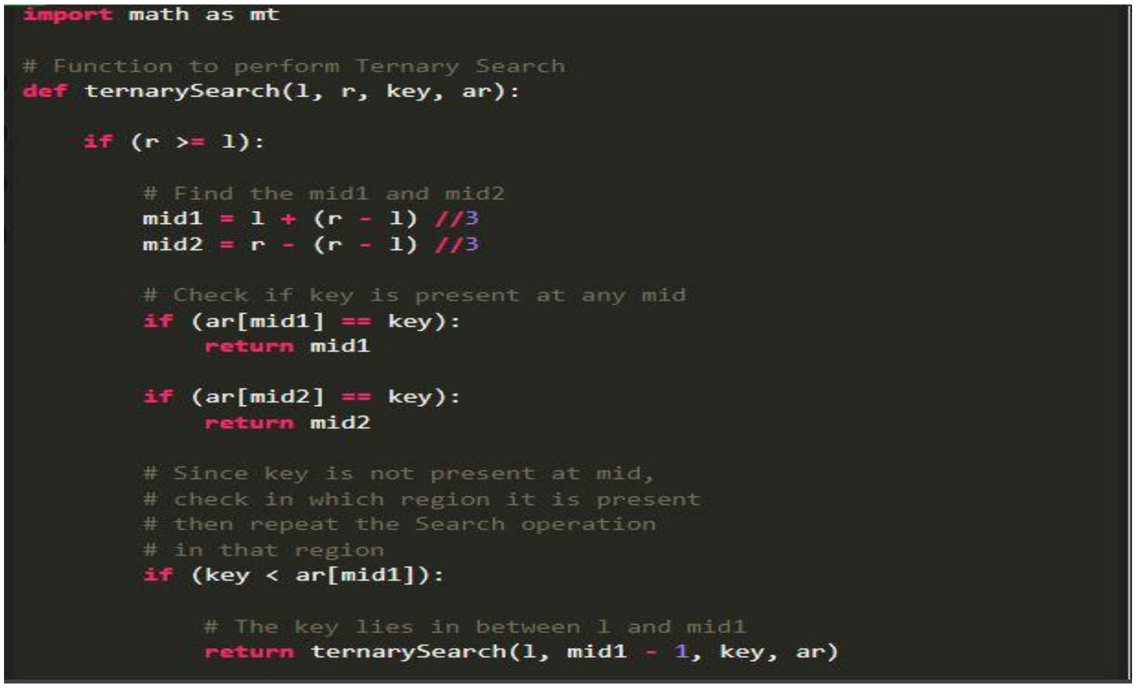
Ternary search is a decrease(by constant) and conquer algorithm that can be used to find an element in an array. It is similar to binary search where we divide the array into two parts but in this algorithm, we divide the given array into three parts and determine which has the key (searched element). We can divide the array into three parts by taking mid1 and mid2 which can be calculated as shown below. Initially, l and r will be equal to 0 and n-1 respectively, where n is the length of the array.

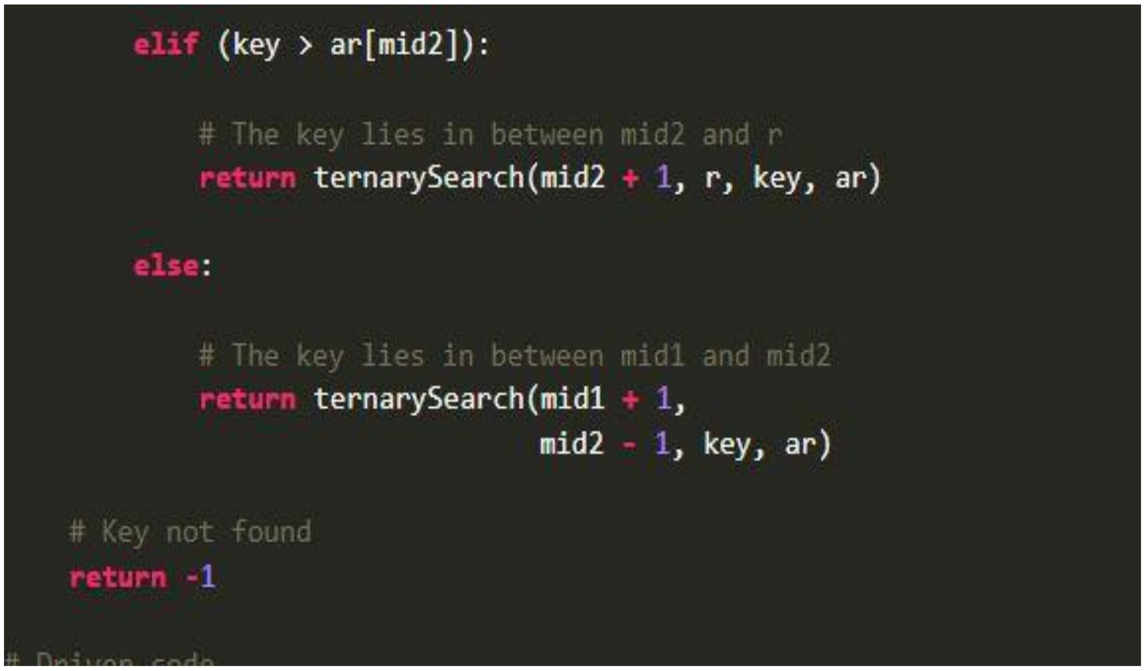
Steps to perform Ternary Search:

1. First, we compare the key with the element at mid1. If found equal, we return mid1.
2. If not, then we compare the key with the element at mid2. If found equal, we return mid2.
3. If not, then we check whether the key is less than the element at mid1. If yes, then recur to the first part.
4. If not, then we check whether the key is greater than the element at mid2. If yes, then recur to the third part.
5. If not, then we recur to the second (middle) part.

Recursive Implementation of Ternary Search

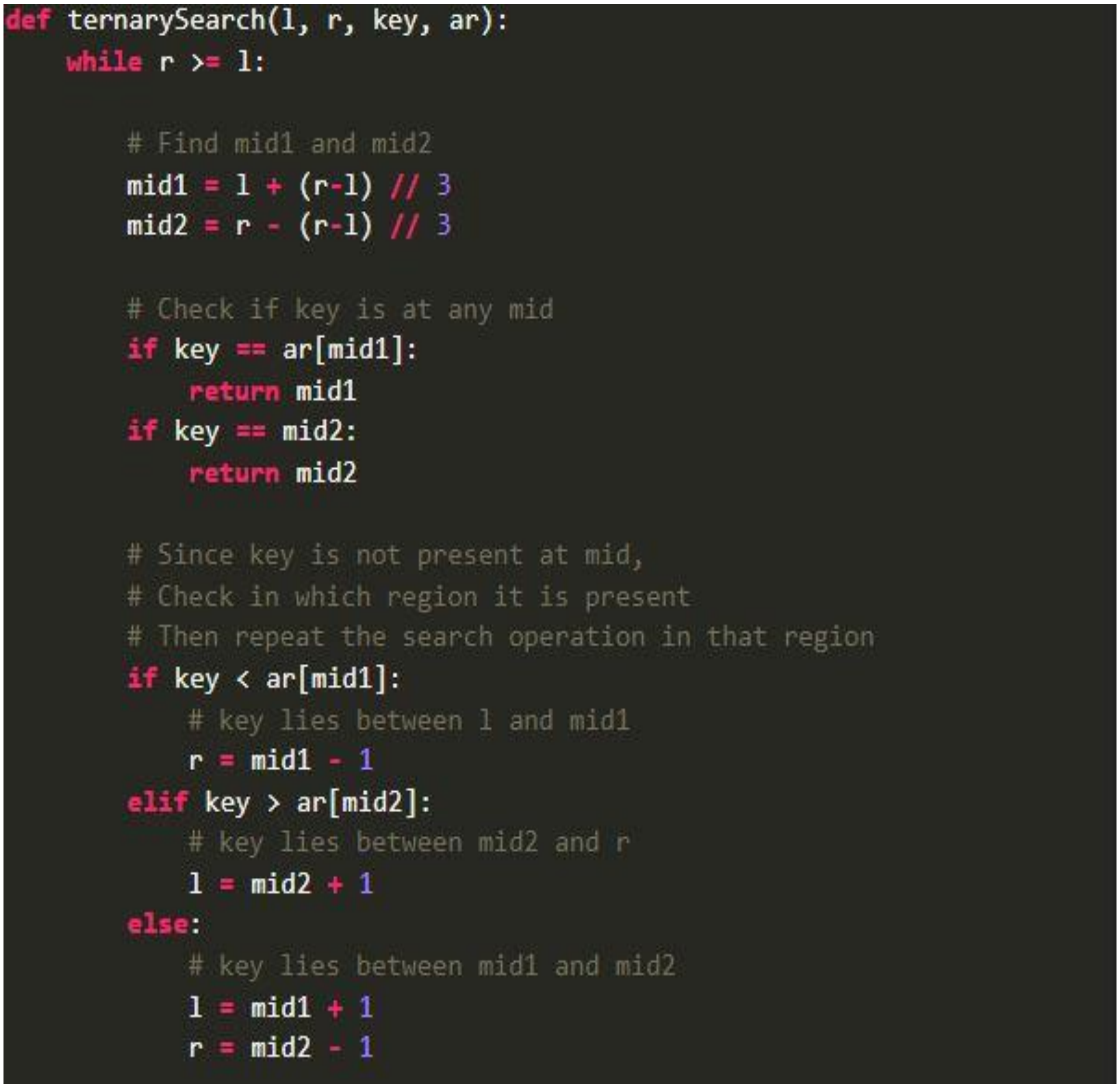


```python
import math as mt

# Function to perform Ternary Search
def ternarySearch(l, r, key, ar):

    if (r >= l):

        # Find the mid1 and mid2
        mid1 = l + (r - l) //3
        mid2 = r - (r - l) //3

        # Check if key is present at any mid
        if (ar[mid1] == key):
            return mid1

        if (ar[mid2] == key):
            return mid2

        # Since key is not present at mid,
        # check in which region it is present
        # then repeat the Search operation
        # in that region
        if (key < ar[mid1]):

            # The key lies in between l and mid1
            return ternarySearch(l, mid1 - 1, key, ar)
```



```python
        elif (key > ar[mid2]):

            # The key lies in between mid2 and r
            return ternarySearch(mid2 + 1, r, key, ar)

        else:

            # The key lies in between mid1 and mid2
            return ternarySearch(mid1 + 1,
                                 mid2 - 1, key, ar)

    # Key not found
    return -1

# Driven code
```

Iterative Approach of Ternary Search



```python
def ternarySearch(l, r, key, ar):
    while r >= l:

        # Find mid1 and mid2
        mid1 = l + (r-l) // 3
        mid2 = r - (r-l) // 3

        # Check if key is at any mid
        if key == ar[mid1]:
            return mid1
        if key == mid2:
            return mid2

        # Since key is not present at mid,
        # Check in which region it is present
        # Then repeat the search operation in that region
        if key < ar[mid1]:
            # key lies between l and mid1
            r = mid1 - 1
        elif key > ar[mid2]:
            # key lies between mid2 and r
            l = mid2 + 1
        else:
            # key lies between mid1 and mid2
            l = mid1 + 1
            r = mid2 - 1
```