1.Given an integer array num where the elements are sorted in ascending order, convert it to a height-balanced binary search tree.

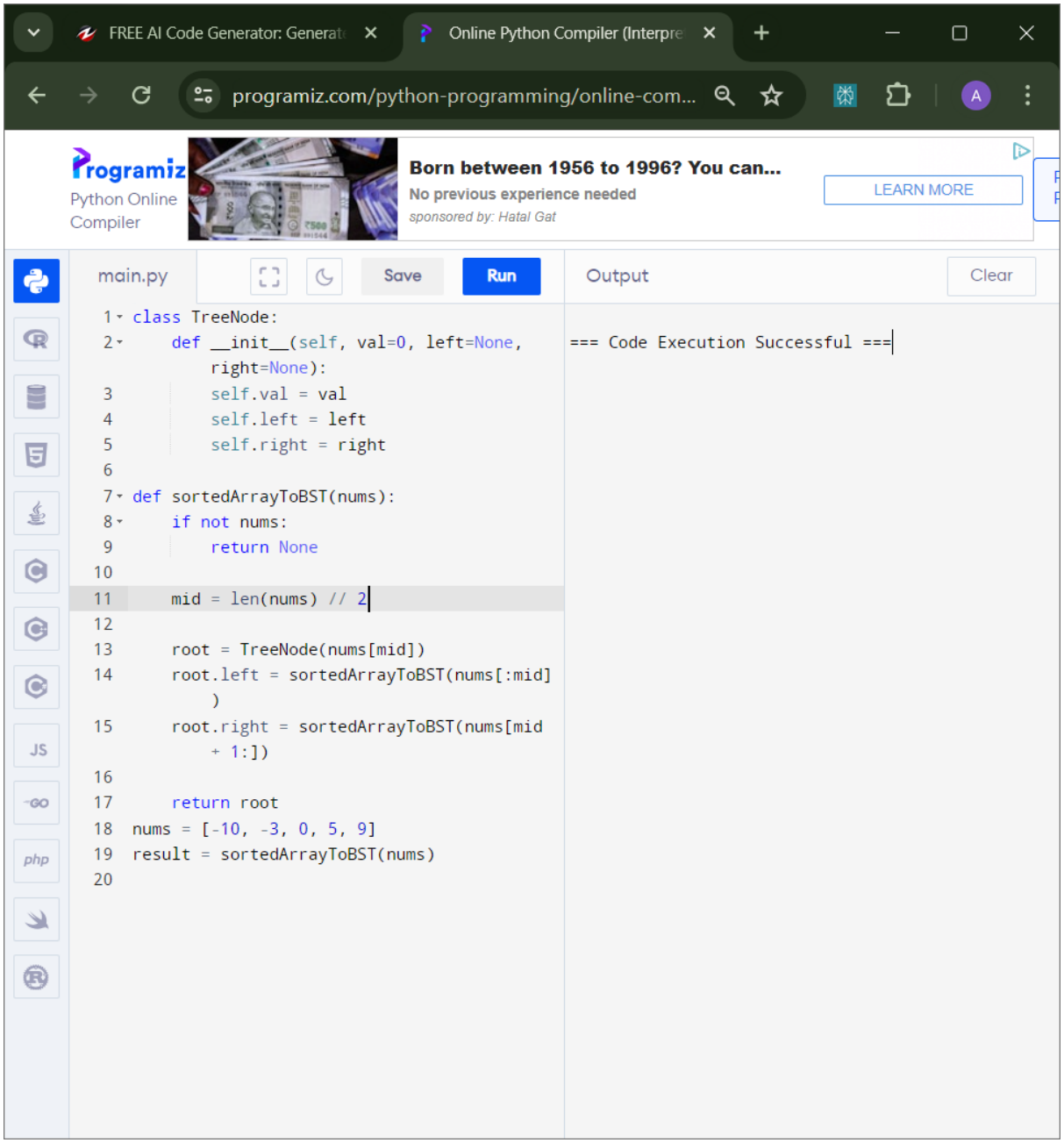
Example 1:

Input num = [-10, -3,0,5,9]

Output: [0 ,-3,9,-10,null,5]

Code:-

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

def sortedArrayToBST(nums):
    if not nums:
        return None

    mid = len(nums) // 2

    root = TreeNode(nums[mid])
    root.left = sortedArrayToBST(nums[:mid])
    root.right = sortedArrayToBST(nums[mid + 1:])

    return root

# Example
nums = [-10, -3, 0, 5, 9]
result = sortedArrayToBST(nums)
```



2. Given an array nums containing n distinct numbers in the range [0, n], return the only number in the range that is missing from the array.

Example 1:

Input: nums = [3,0,1]Output: 2

Code:-

```
def missing_number(nums):
    n = len(nums)
    total_sum = n * (n + 1) // 2
    actual_sum = sum(nums)
```

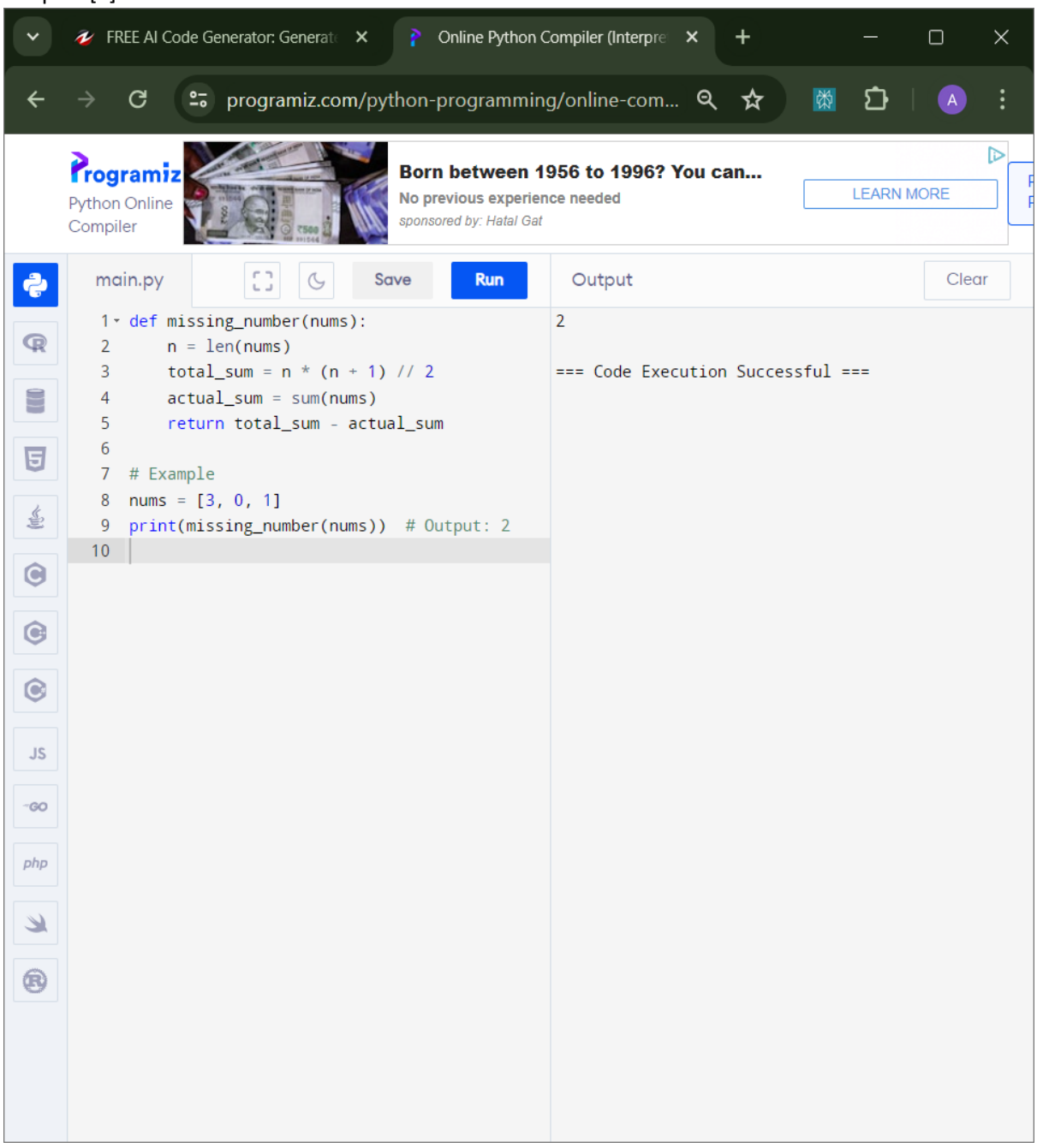
return total_sum - actual_sum

Example

nums = [3, 0, 1]

print(missing_number(nums)) # Output: 2

Output: [2]



3. Given two integer arrays nums1 and nums2, return an array of their intersection . Each element in the result must be unique and you may return the result in any order.

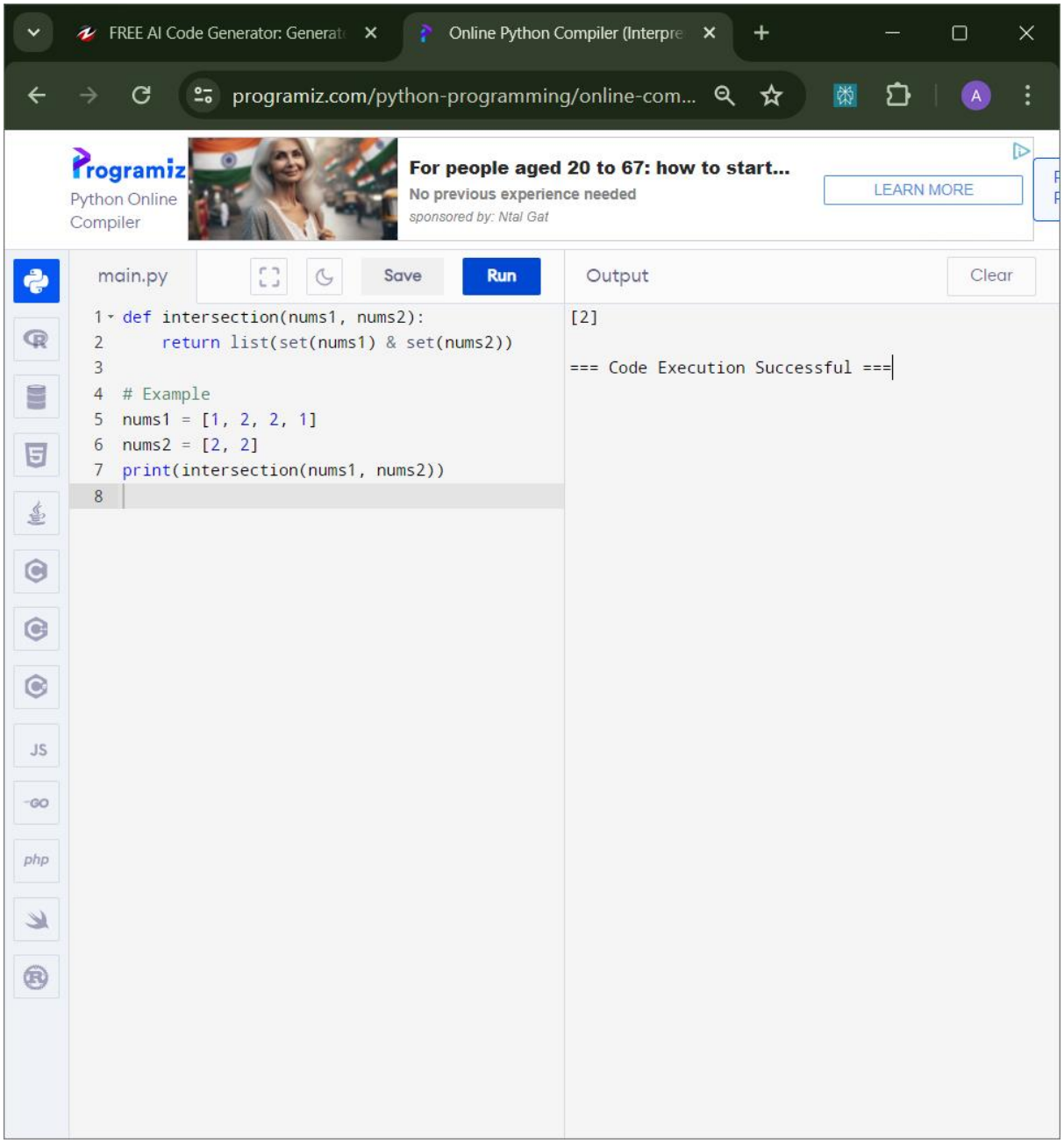
Example 1:

Input: nums1 = [1,2,2,1], nums2 = [2,2]

Output: [2]

CODE:-

```
def intersection(nums1, nums2):
    return list(set(nums1) & set(nums2))


# Example
nums1 = [1, 2, 2, 1]
nums2 = [2, 2]
print(intersection(nums1, nums2))
```



4. Given an integer n, return the nth digit of the infinite integer sequence [1, 2, 3,

4, 5, 6, 7, 8, 9, 10, 11, ...].

Example 1:

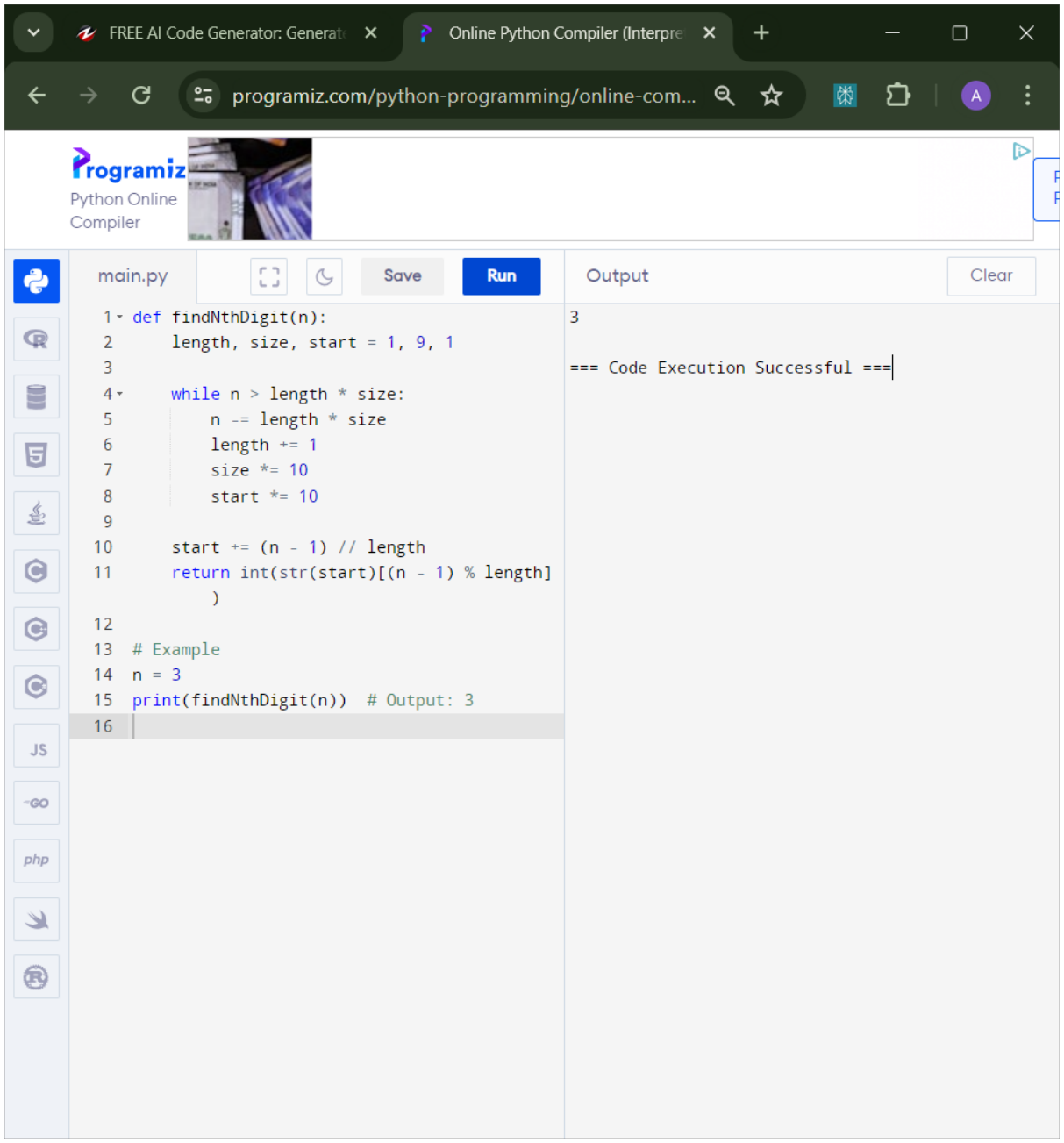
Input: n = 3

Output: 3

Code:-

```
def findNthDigit(n):
```

```
    length, size, start = 1, 9, 1

    while n > length * size:
        n -= length * size
        length += 1
        size *= 10
        start *= 10

    start += (n - 1) // length
    return int(str(start)[(n - 1) % length])

# Example
n = 3
print(findNthDigit(n))  # Output: 3
```



5. Given an array nums of size n, return the majority element.The majority element is the element that appears more than ⌊n / 2⌋ times. You may assume that the majority element always exists in the array.

Example 1:

Input: nums = [3,2,3]

Output: 3

Code:-

from collections import Counter

```
def majority_element(nums):
    counts = Counter(nums)
    return max(counts, key=counts.get)

# Example
nums = [3, 2, 3]
print(majority_element(nums))  # Output: 3
```