

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)
```

Programiz
Python Online Compiler

Born between 1956 to 1996? You can...
No previous experience needed
sponsored by: Hatal Gat

[LEARN MORE](#)

main.py

```

1 class TreeNode:
2     def __init__(self, val=0, left=None,
      right=None):
3         self.val = val
4         self.left = left
5         self.right = right
6
7 def sortedArrayToBST(nums):
8     if not nums:
9         return None
10
11     mid = len(nums) // 2
12
13     root = TreeNode(nums[mid])
14     root.left = sortedArrayToBST(nums[:mid])
15     root.right = sortedArrayToBST(nums[mid
      + 1:])
16
17     return root
18 nums = [-10, -3, 0, 5, 9]
19 result = sortedArrayToBST(nums)
20

```

Output

=== Code Execution Successful ===

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]

The screenshot displays the Programiz Online Python Compiler interface. The browser address bar shows the URL `programiz.com/python-programming/online-com...`. The page header includes the Programiz logo, a banner for "Born between 1956 to 1996? You can..." with a "LEARN MORE" button, and a "Python Online Compiler" label. The main workspace is divided into three sections: a file explorer on the left showing "main.py", a code editor in the center, and an output panel on the right. The code editor contains the following Python code:

```
1 def missing_number(nums):
2     n = len(nums)
3     total_sum = n * (n + 1) // 2
4     actual_sum = sum(nums)
5     return total_sum - actual_sum
6
7 # Example
8 nums = [3, 0, 1]
9 print(missing_number(nums)) # Output: 2
10
```

The output panel on the right shows the result of the code execution:

```
2
=== Code Execution Successful ===
```

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 n th 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
```

```
1 def findNthDigit(n):
2     length, size, start = 1, 9, 1
3
4     while n > length * size:
5         n -= length * size
6         length += 1
7         size *= 10
8         start *= 10
9
10    start += (n - 1) // length
11    return int(str(start)[(n - 1) % length])
12
13 # Example
14 n = 3
15 print(findNthDigit(n)) # Output: 3
16
```

Output: 3

=== Code Execution Successful ===

5. Given an array `nums` of size `n`, return the majority element. The majority element is the element that appears more than $\lfloor n / 2 \rfloor$ 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
```

The screenshot displays the Programiz Online Python Compiler interface. At the top, there's a navigation bar with the Programiz logo and a banner for a course titled "For people aged 20 to 67: how to start..." with a "LEARN MORE" button. Below the banner, the interface is divided into two main sections: a code editor on the left and an output panel on the right. The code editor shows a file named "main.py" with the following Python code:

```
1 from collections import Counter  
2  
3 def majority_element(nums):  
4     counts = Counter(nums)  
5     return max(counts, key=counts.get)  
6  
7 # Example  
8 nums = [3, 2, 3]  
9 print(majority_element(nums)) # Output: 3  
10
```

The output panel on the right shows the result of the code execution:

```
3  
  
=== Code Execution Successful ===
```

The interface also includes a sidebar on the left with various programming language icons (Python, JavaScript, PHP, etc.) and buttons for "Save", "Run", and "Clear".