ain.py

Save Run

```python
def twoSum(nums, target):
    num_dict = {}
    for i, num in enumerate(nums):
        if num in num_dict:
            return [num_dict[num], i]
        else:
            num_dict[target - num] = i
    return []

# Example usage:
nums = [2,7,11,15]
target = 9
result = twoSum(nums, target)
print("Indices of the two numbers that add up to", target, "are:", result)
```

Output

```
Indices of the two numbers that add up to 9 are: [0, 1]

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

main.py

Save Run

```python
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

def addTwoNumbers(l1, l2):
    dummy = ListNode(0)
    current = dummy
    carry = 0

    while l1 or l2 or carry:
        sum_val = carry
        if l1:
            sum_val += l1.val
            l1 = l1.next
        if l2:
            sum_val += l2.val
            l2 = l2.next
        carry = sum_val // 10
        current.next = ListNode(sum_val % 10)
        current = current.next

    return dummy.next

# Example usage:
l1 = ListNode(2, ListNode(4, ListNode(3)))
l2 = ListNode(5, ListNode(6, ListNode(4)))
```

Output

```
Sum of the two numbers is:
7 0 8
=== Code Execution Successful ===
```

```
    return 0

char_index_map = {}
max_length = 0
start = 0

for end in range(len(s)):
    if s[end] in char_index_map:
        start = max(start, char_index_map[s[end]] + 1)

    char_index_map[s[end]] = end
    max_length = max(max_length, end - start + 1)

return max_length
```

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

```
def findMedianSortedArrays(self, nums1, nums2):
    if len(nums1) > len(nums2):
        nums1, nums2 = nums2, nums1

    x, y = len(nums1), len(nums2)
    low, high = 0, x

    while low <= high:
        partition_x = (low + high) // 2
        partition_y = (x + y + 1) // 2 - partition_x

        max_left_x = float('-inf') if partition_x == 0 else nums1[partition_x
            - 1]
        min_right_x = float('inf') if partition_x == x else nums1[partition_x]

        max_left_y = float('-inf') if partition_y == 0 else nums2[partition_y
            - 1]
        min_right_y = float('inf') if partition_y == y else nums2[partition_y]

        if max_left_x <= min_right_y and max_left_y <= min_right_x:
            if (x + y) % 2 == 0:
                return (max(max_left_x, max_left_y) + min(min_right_x,
                    min_right_y)) / 2
            else:
                return max(max_left_x, max_left_y)
```

main.py

Save Run

```python
def longestPalindrome(s):
    n = len(s)
    if n <= 1:
        return s

    start = 0
    maxLength = 1

    def expandAroundCenter(left, right):
        while left >= 0 and right < n and s[left] == s[right]:
            left -= 1
            right += 1
        return right - left - 1

    for i in range(n):
        len1 = expandAroundCenter(i, i)
        len2 = expandAroundCenter(i, i + 1)
        currLen = max(len1, len2)
        if currLen > maxLength:
            maxLength = currLen
            start = i - (currLen - 1) // 2

    return s[start:start + maxLength]

# Example usage:
input_string = "babad"
result = longestPalindrome(input_string)
print("Longest palindromic substring is:", result)
```

Output

```
Longest palindromic substring is: bab

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

main.py

```python
def convert(s, numRows):
    if numRows == 1 or numRows >= len(s):
        return s

    result = [''] * numRows
    index, step = 0, 1

    for char in s:
        result[index] += char
        if index == 0:
            step = 1
        elif index == numRows - 1:
            step = -1
        index += step

    return ''.join(result)

# Example usage:
s = "PAYPALISHIRING"
numRows = 3
result = convert(s, numRows)
print("Zigzag conversion of", s, "with", numRows, "rows is:", result)
```

Output

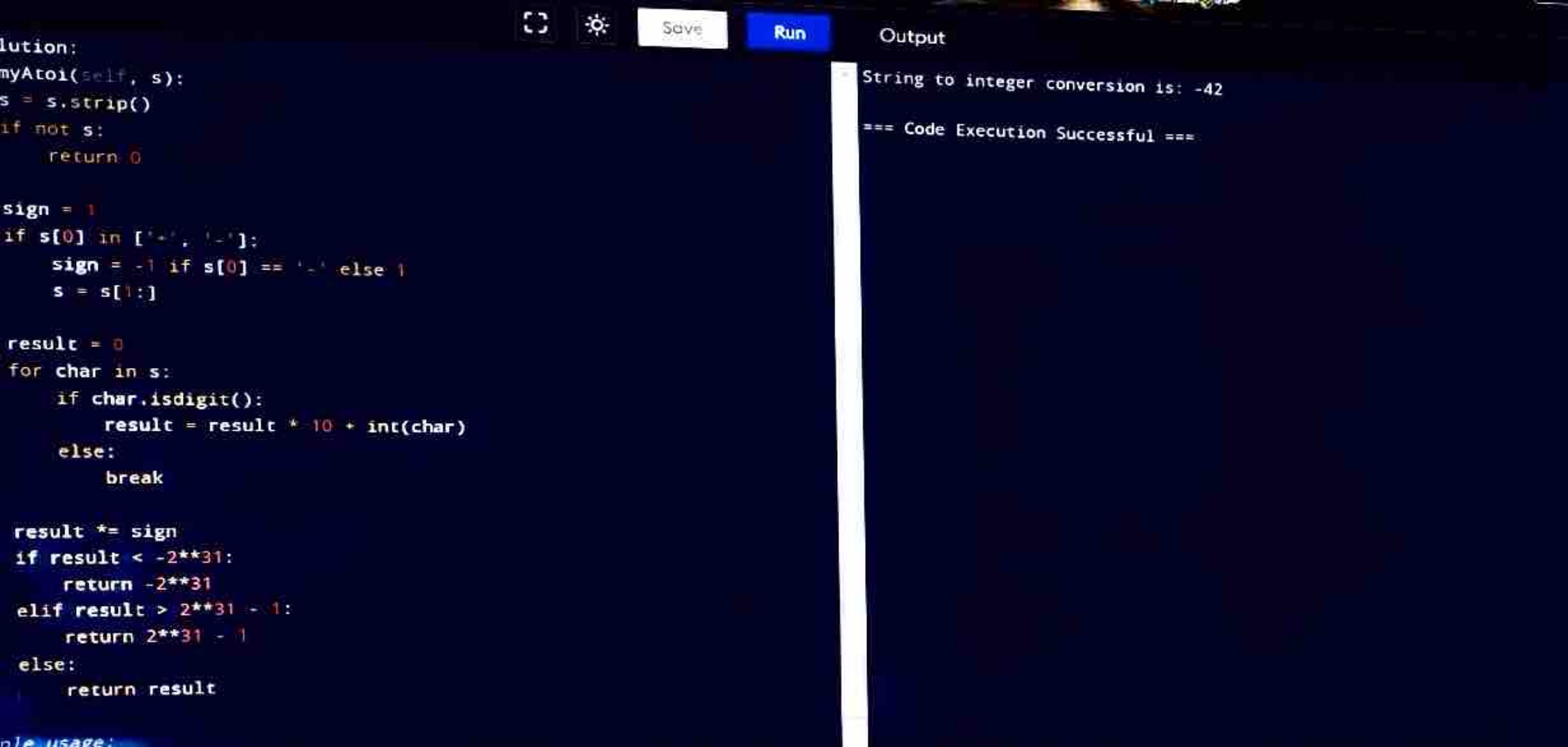
```
Zigzag conversion of PAYPALISHIRING with 3 rows is: PAHNAPLSIIGYIR

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

main.py

```python
class Solution:
    def reverse(self, x):
        negative = False
        if x < 0:
            negative = True
            x = -x

        reversed_num = 0
        while x > 0:
            reversed_num = reversed_num * 10 + x % 10
            x //= 10

        if reversed_num > 2**31 - 1:
            return 0

        return -reversed_num if negative else reversed_num

# Example usage:
x = 123
result = Solution().reverse(x)
print("Reversed integer is:", result)
```

Output

```
Reversed integer is: 321

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


Save
Run
Output
lution:
myAtoi(self, s):
s = s.strip()
if not s:
return 0
sign = 1
if s[0] in ['+', '-']:
sign = -1 if s[0] == '-' else 1
s = s[1:]
result = 0
for char in s:
if char.isdigit():
result = result * 10 + int(char)
else:
break
result *= sign
if result < -2**31:
return -2**31
elif result > 2**31 - 1:
return 2**31 - 1
else:
return result
String to integer conversion is: -42
=== Code Execution Successful ===

```
Solution:
ef isPalindrome(self, x):
    if x < 0:
        return False
    reversed_num = 0
    original = x
    while x > 0:
        reversed_num = reversed_num * 10 + x % 10
        x //= 10
    return original == reversed_num

mple usage:
21
t = Solution().isPalindrome(x)
("Is", x, "a palindrome?", result)
```

Output

```
Is 121 a palindrome? True

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

```python
olution:
 isMatch(self, s: str, p: str) -> bool:
 m, n = len(s), len(p)
 dp = [[False] * (n + 1) for _ in range(m + 1)]
 dp[0][0] = True

 for j in range(1, n + 1):
     if p[j - 1] == '*':
         dp[0][j] = dp[0][j - 2]

 for i in range(1, m + 1):
     for j in range(1, n + 1):
         if p[j - 1] == '*':
             dp[i][j] = dp[i][j - 2] or (dp[i - 1][j] and (s[i - 1] == p[j -
                 2] or p[j - 2] == '.'))
         else:
             dp[i][j] = dp[i - 1][j - 1] and (s[i - 1] == p[j - 1] or p[j -
                 1] == '.')
```