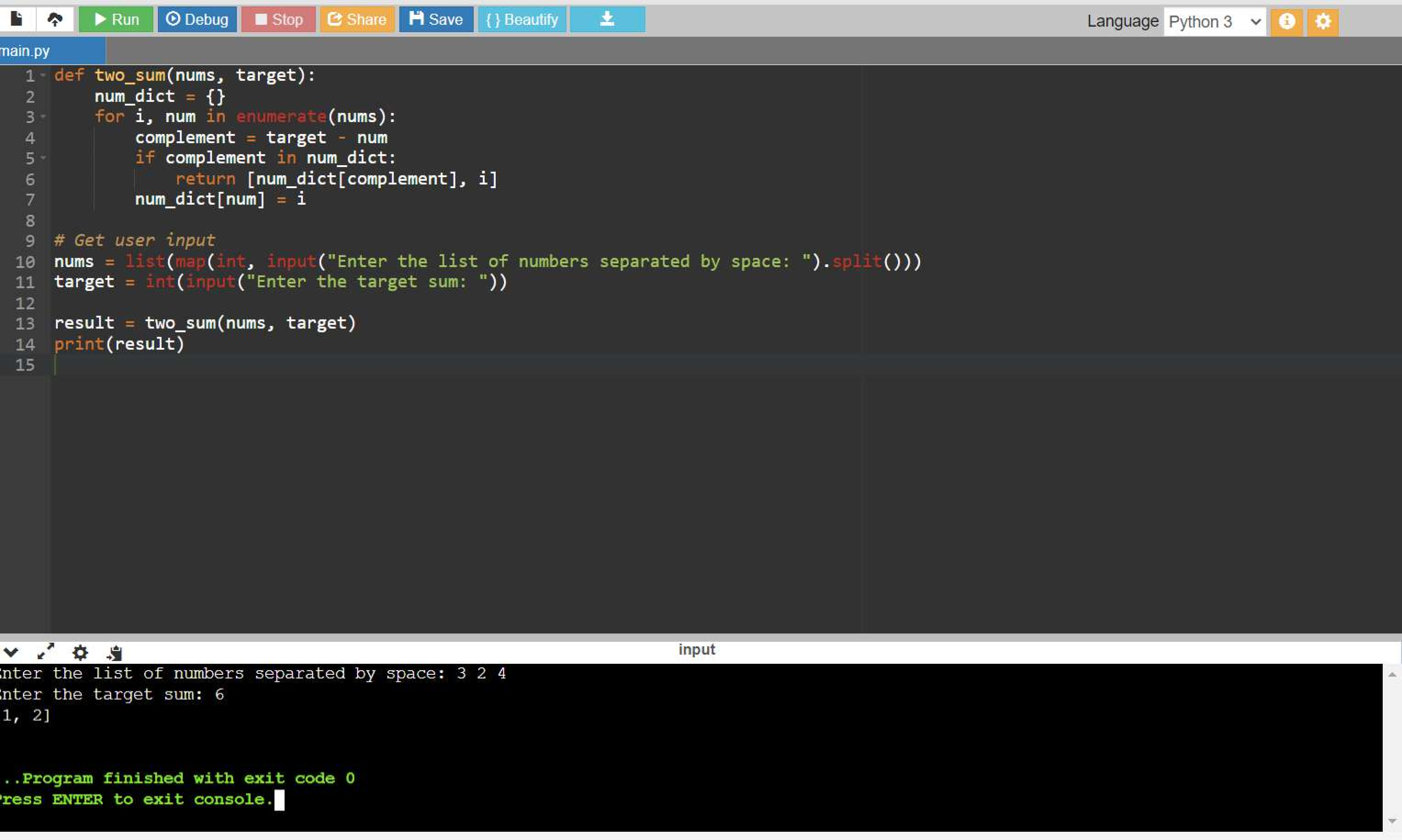
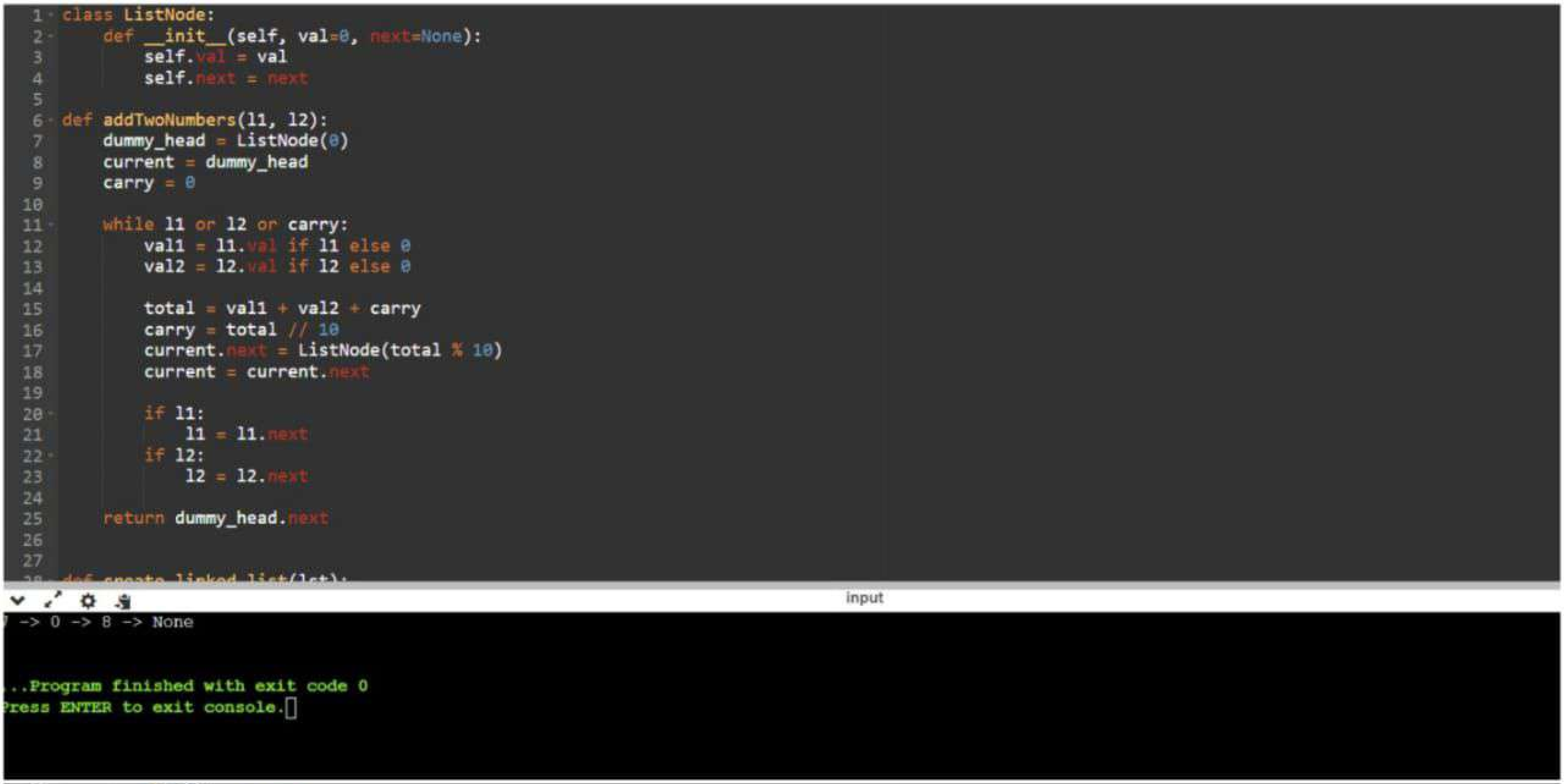


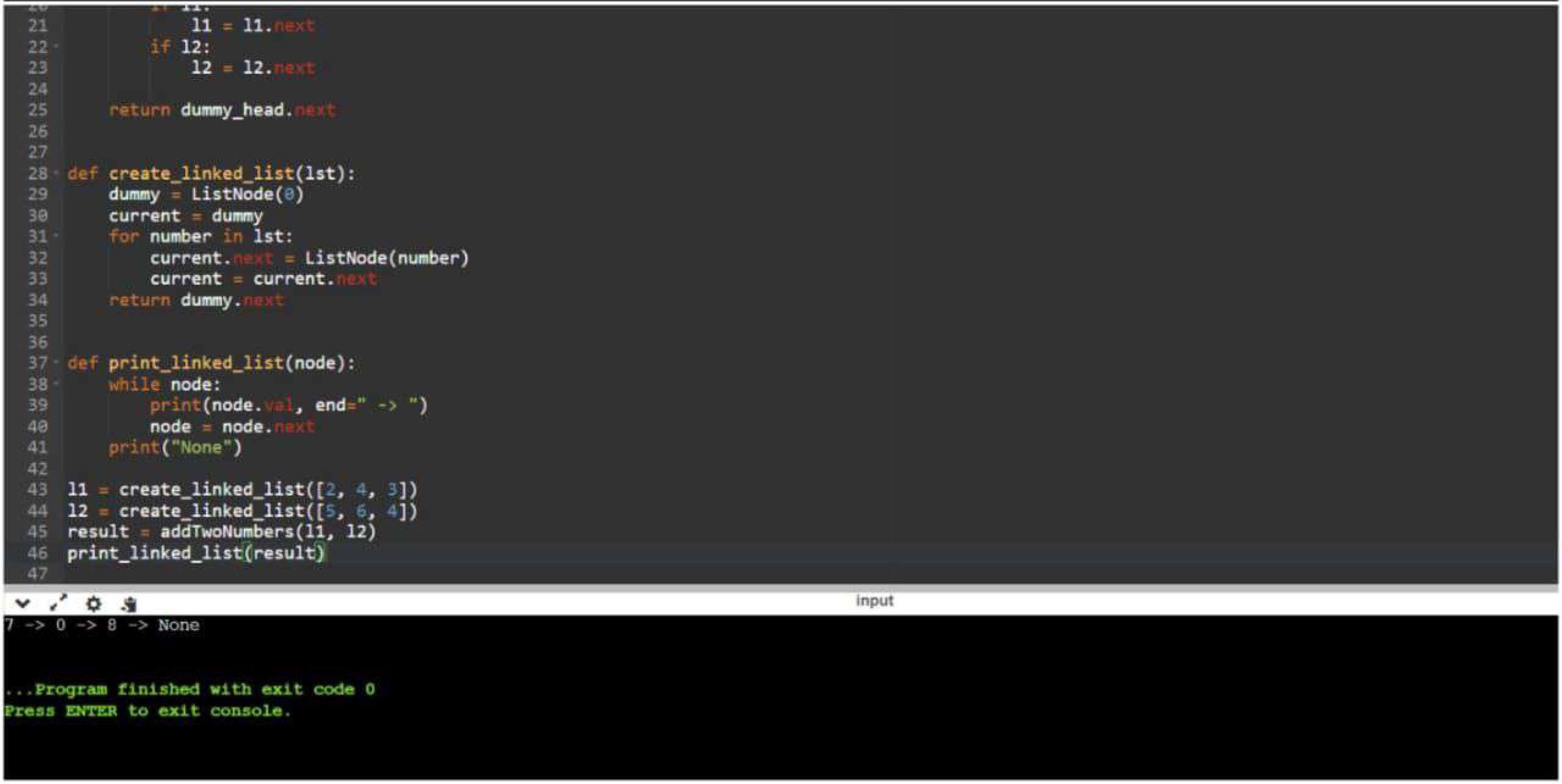
```
def two_sum(nums, target):
    num_dict = {}
    for i, num in enumerate(nums):
        complement = target - num
        if complement in num_dict:
            return [num_dict[complement], i]
        num_dict[num] = i

# Get user input
nums = list(map(int, input("Enter the list of numbers separated by space: ").split()))
target = int(input("Enter the target sum: "))

result = two_sum(nums, target)
print(result)

```

```
nter the list of numbers separated by space: 3 2 4
nter the target sum: 6
1, 2]


..Program finished with exit code 0
ress ENTER to exit console.
```



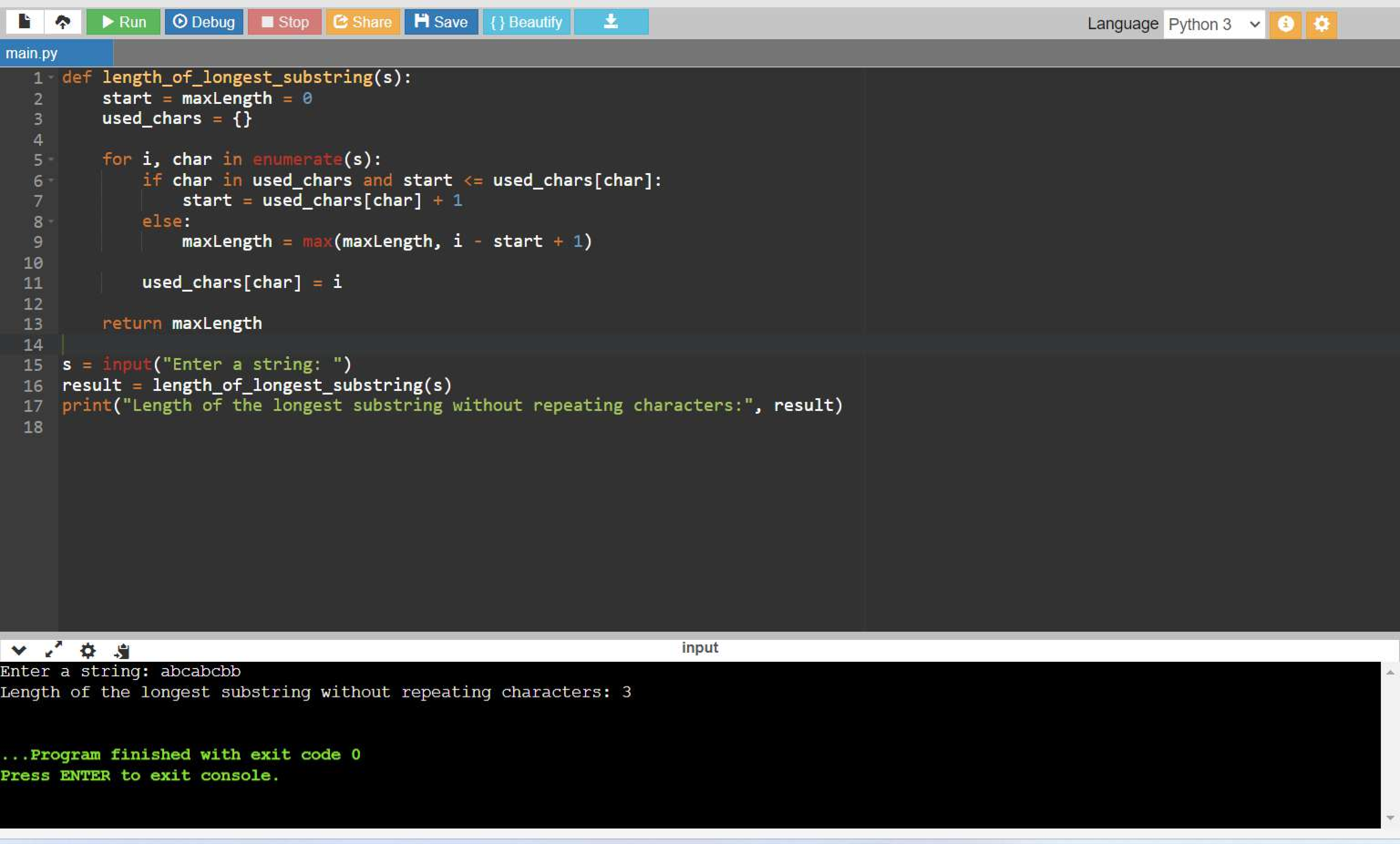
```python
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

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

    while l1 or l2 or carry:
        val1 = l1.val if l1 else 0
        val2 = l2.val if l2 else 0

        total = val1 + val2 + carry
        carry = total // 10
        current.next = ListNode(total % 10)
        current = current.next

        if l1:
            l1 = l1.next
        if l2:
            l2 = l2.next

    return dummy_head.next


def create_linked_list(lst):
```

```
 -> 0 -> 8 -> None


..Program finished with exit code 0
ress ENTER to exit console.
```



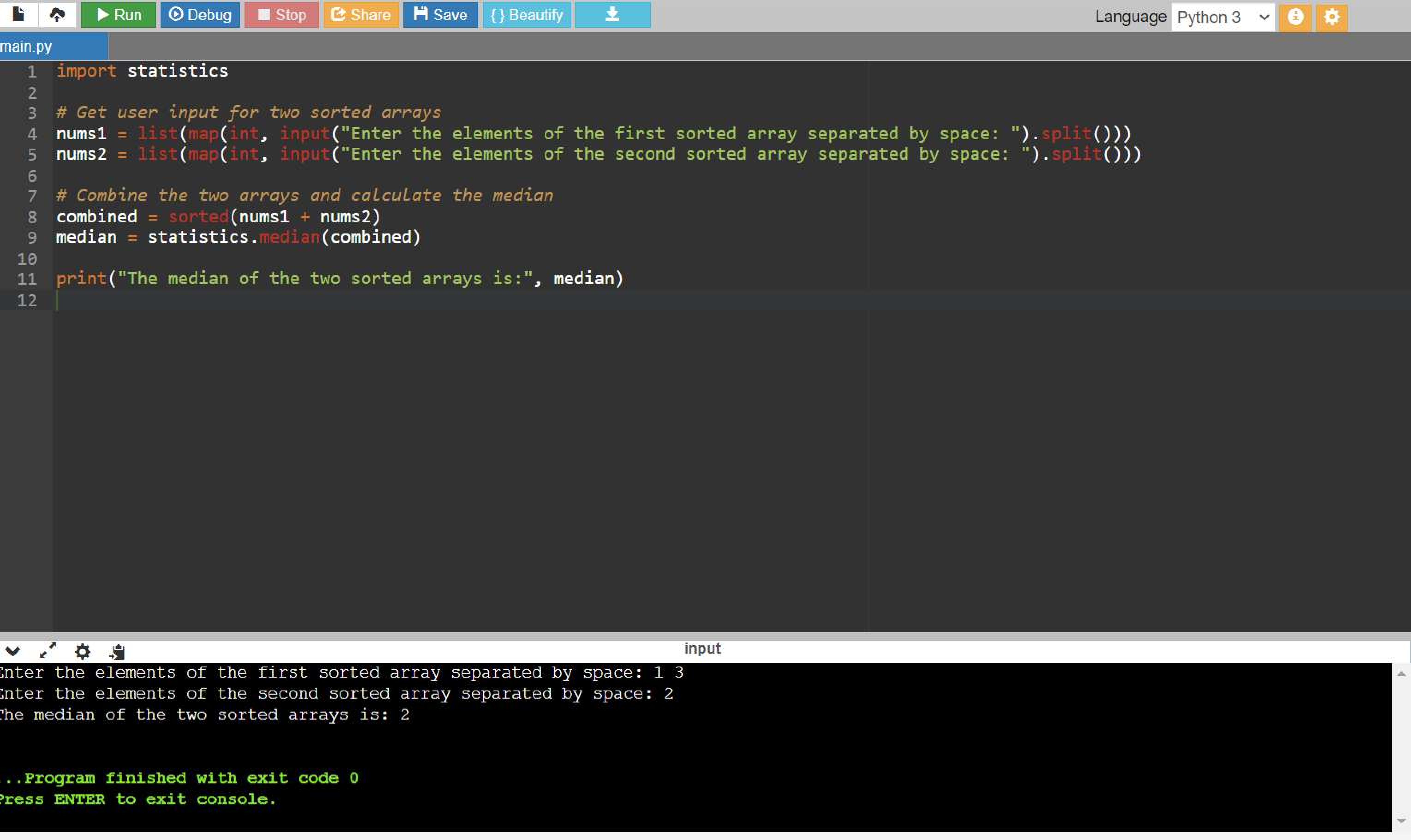
```python
            l1 = l1.next
        if l2:
            l2 = l2.next

    return dummy_head.next


def create_linked_list(lst):
    dummy = ListNode(0)
    current = dummy
    for number in lst:
        current.next = ListNode(number)
        current = current.next
    return dummy.next


def print_linked_list(node):
    while node:
        print(node.val, end=" -> ")
        node = node.next
    print("None")

l1 = create_linked_list([2, 4, 3])
l2 = create_linked_list([5, 6, 4])
result = addTwoNumbers(l1, l2)
print_linked_list(result)
```

```
7 -> 0 -> 8 -> None


...Program finished with exit code 0
Press ENTER to exit console.
```



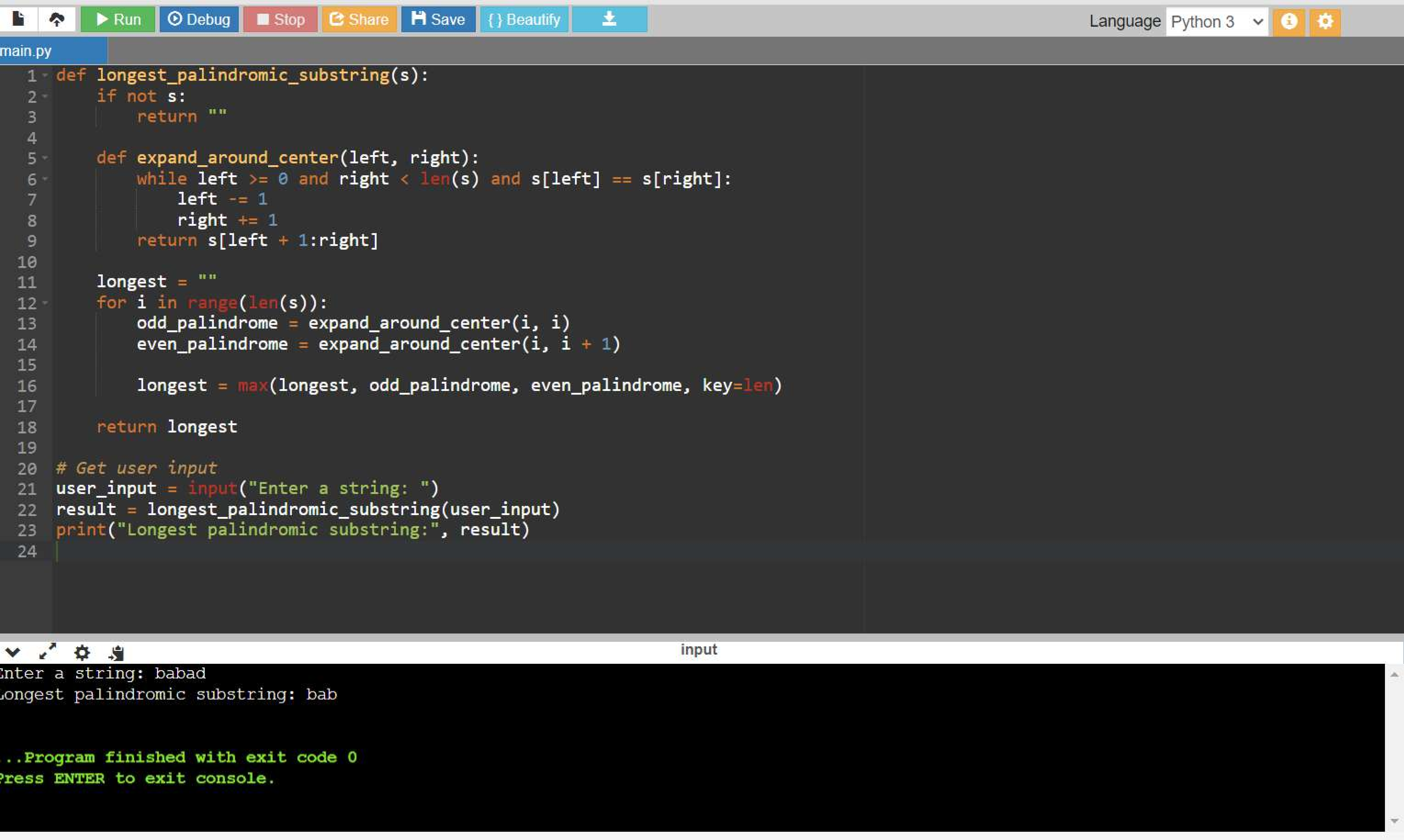
```python
def length_of_longest_substring(s):
    start = maxLength = 0
    used_chars = {}

    for i, char in enumerate(s):
        if char in used_chars and start <= used_chars[char]:
            start = used_chars[char] + 1
        else:
            maxLength = max(maxLength, i - start + 1)

        used_chars[char] = i

    return maxLength

s = input("Enter a string: ")
result = length_of_longest_substring(s)
print("Length of the longest substring without repeating characters:", result)
```

```
Enter a string: abcabcbb
Length of the longest substring without repeating characters: 3


...Program finished with exit code 0
Press ENTER to exit console.
```



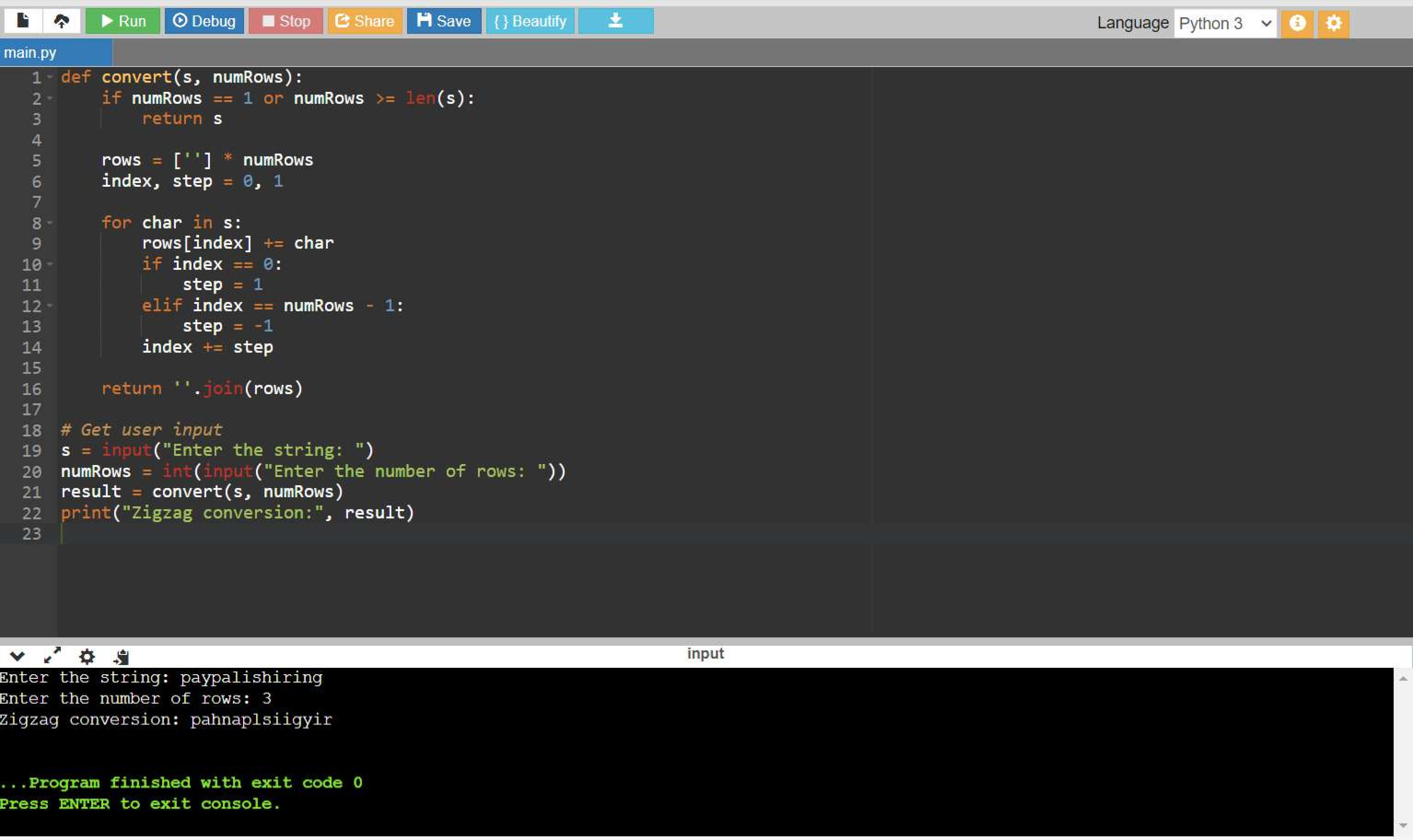
```python
import statistics

# Get user input for two sorted arrays
nums1 = list(map(int, input("Enter the elements of the first sorted array separated by space: ").split()))
nums2 = list(map(int, input("Enter the elements of the second sorted array separated by space: ").split()))

# Combine the two arrays and calculate the median
combined = sorted(nums1 + nums2)
median = statistics.median(combined)

print("The median of the two sorted arrays is:", median)
```

```
nter the elements of the first sorted array separated by space: 1 3
nter the elements of the second sorted array separated by space: 2
he median of the two sorted arrays is: 2


..Program finished with exit code 0
ress ENTER to exit console.
```



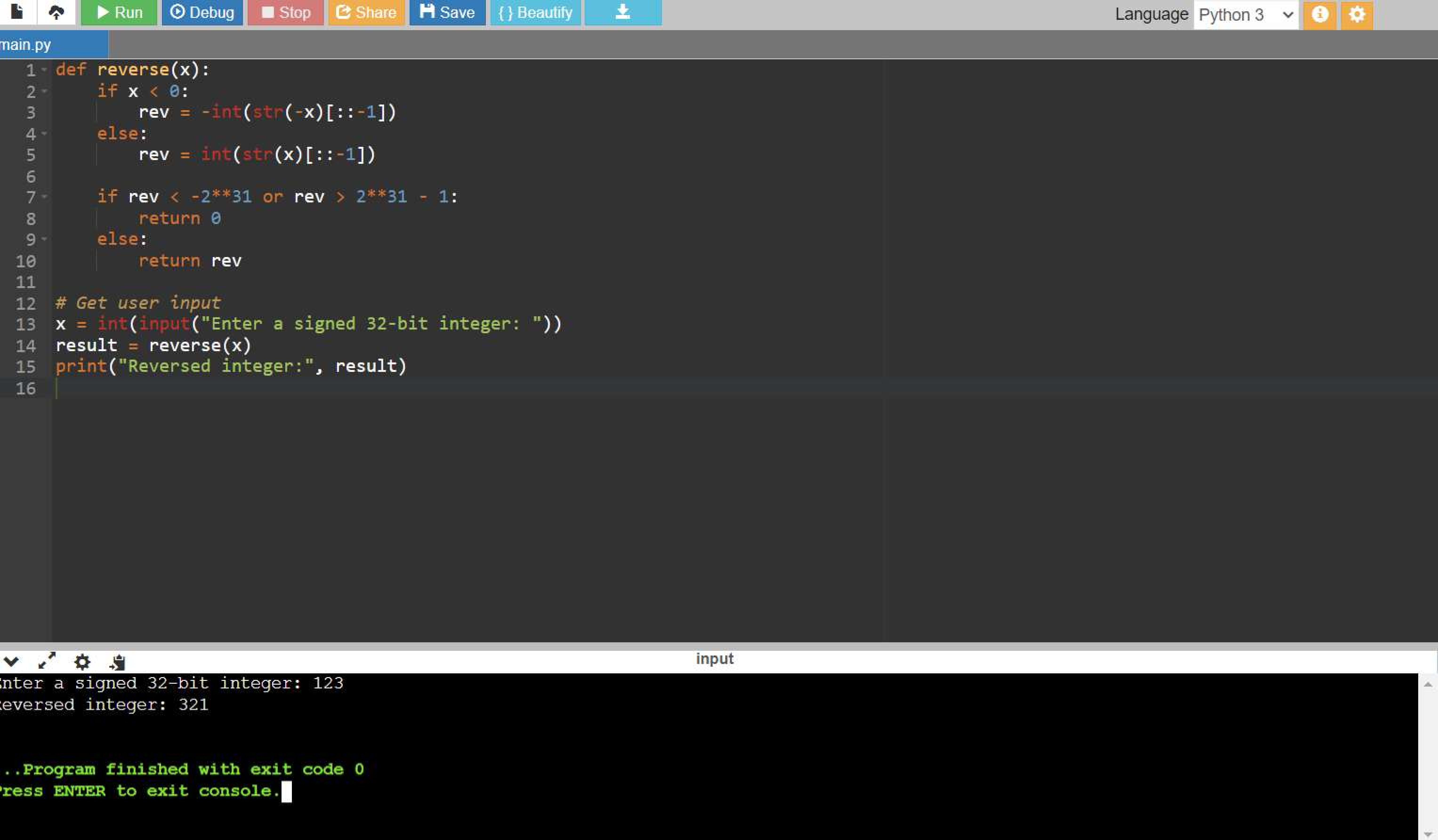
```python
def longest_palindromic_substring(s):
    if not s:
        return ""

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

    longest = ""
    for i in range(len(s)):
        odd_palindrome = expand_around_center(i, i)
        even_palindrome = expand_around_center(i, i + 1)

        longest = max(longest, odd_palindrome, even_palindrome, key=len)

    return longest

# Get user input
user_input = input("Enter a string: ")
result = longest_palindromic_substring(user_input)
print("Longest palindromic substring:", result)
```

```
Enter a string: babad
Longest palindromic substring: bab


...Program finished with exit code 0
Press ENTER to exit console.
```



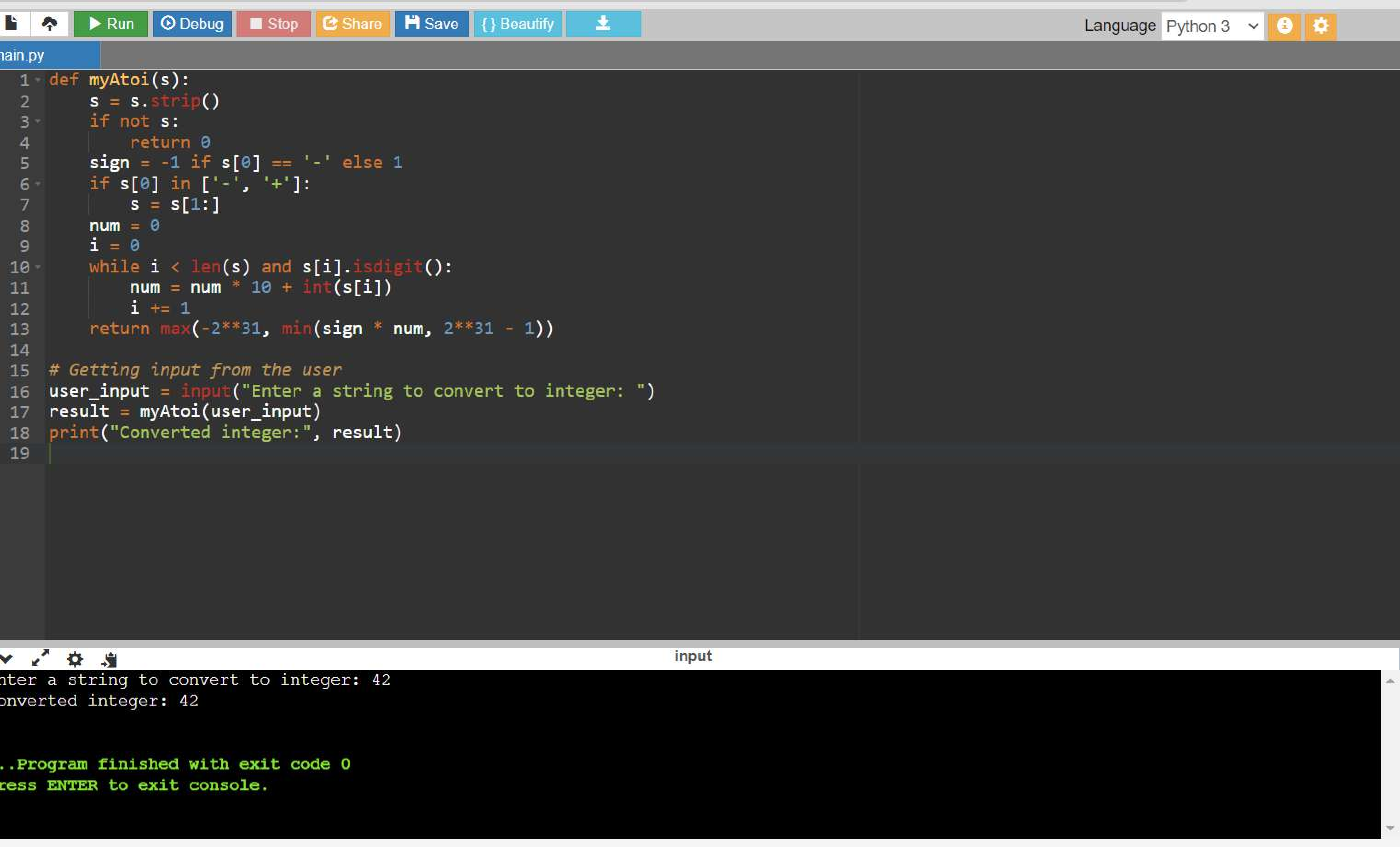
```python
def convert(s, numRows):
    if numRows == 1 or numRows >= len(s):
        return s

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

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

    return ''.join(rows)

# Get user input
s = input("Enter the string: ")
numRows = int(input("Enter the number of rows: "))
result = convert(s, numRows)
print("Zigzag conversion:", result)
```

```
Enter the string: paypalishiring
Enter the number of rows: 3
Zigzag conversion: pahnaplsiigyir


...Program finished with exit code 0
Press ENTER to exit console.
```



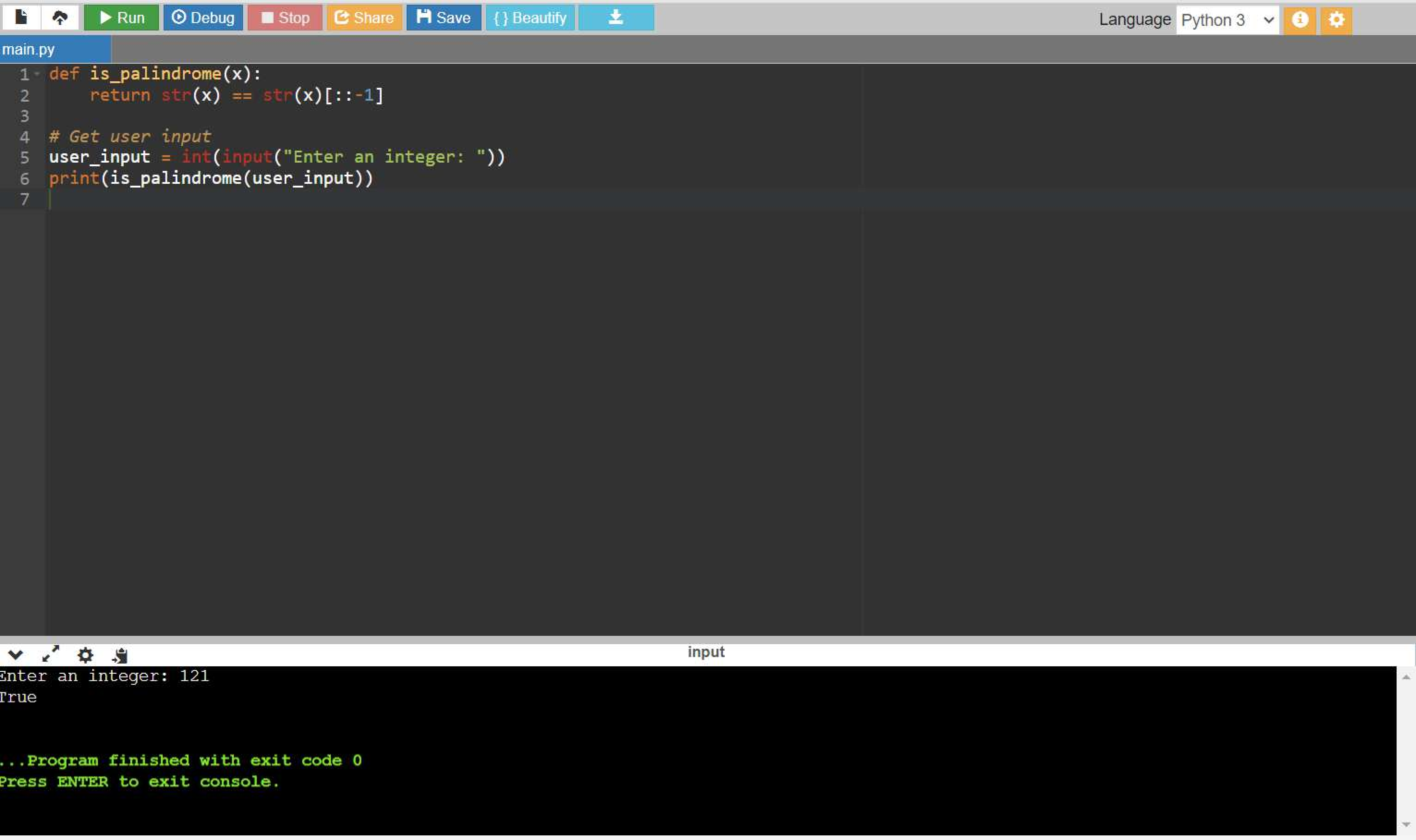
```python
def reverse(x):
    if x < 0:
        rev = -int(str(-x)[::-1])
    else:
        rev = int(str(x)[::-1])

    if rev < -2**31 or rev > 2**31 - 1:
        return 0
    else:
        return rev

# Get user input
x = int(input("Enter a signed 32-bit integer: "))
result = reverse(x)
print("Reversed integer:", result)
```

```
nter a signed 32-bit integer: 123
eversed integer: 321


..Program finished with exit code 0
ress ENTER to exit console.
```



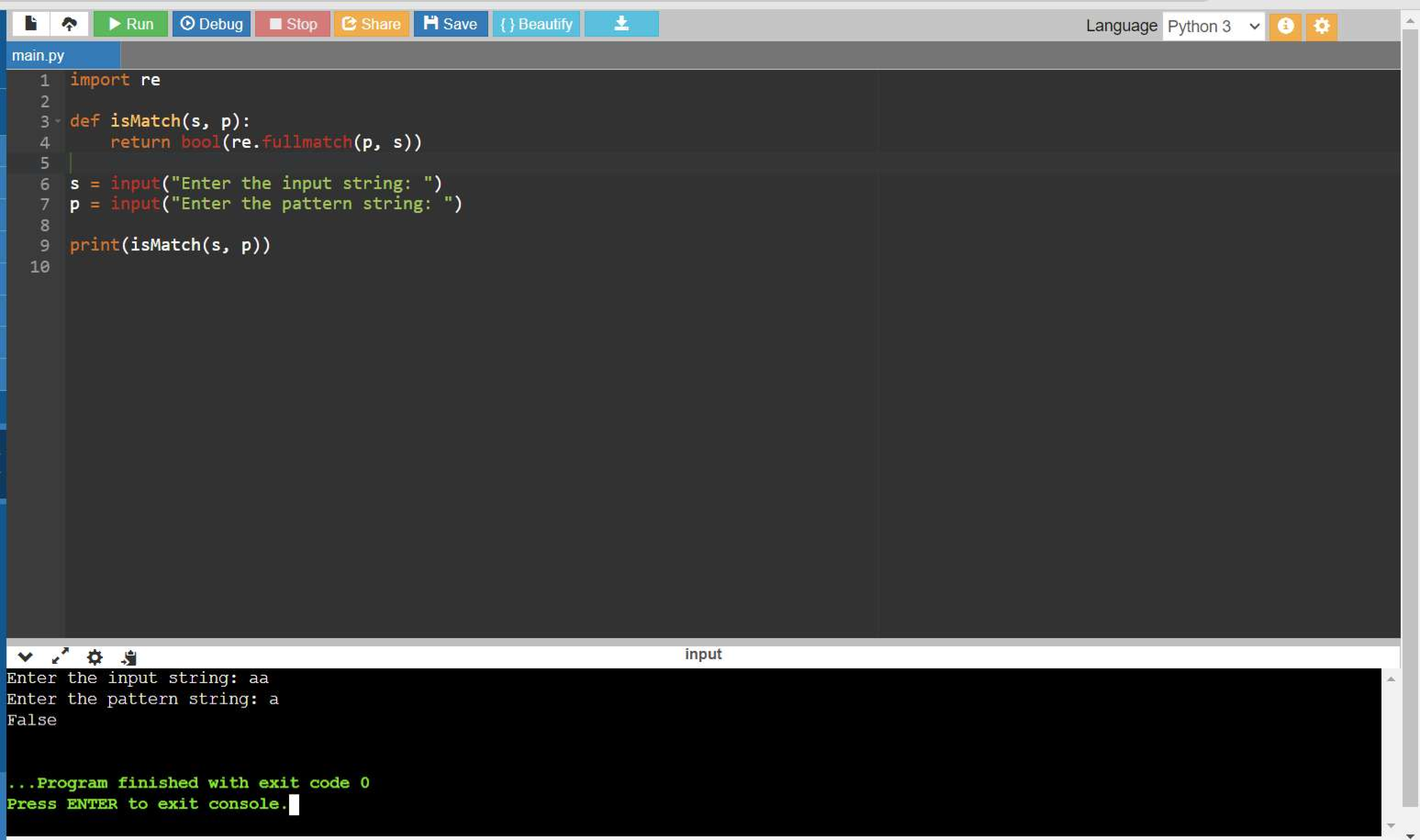
```python
def myAtoi(s):
    s = s.strip()
    if not s:
        return 0
    sign = -1 if s[0] == '-' else 1
    if s[0] in ['-', '+']:
        s = s[1:]
    num = 0
    i = 0
    while i < len(s) and s[i].isdigit():
        num = num * 10 + int(s[i])
        i += 1
    return max(-2**31, min(sign * num, 2**31 - 1))

# Getting input from the user
user_input = input("Enter a string to convert to integer: ")
result = myAtoi(user_input)
print("Converted integer:", result)
```

```
nter a string to convert to integer: 42
onverted integer: 42


..Program finished with exit code 0
ress ENTER to exit console.
```



```python
def is_palindrome(x):
    return str(x) == str(x)[::-1]

# Get user input
user_input = int(input("Enter an integer: "))
print(is_palindrome(user_input))
```

```
Enter an integer: 121
True


...Program finished with exit code 0
Press ENTER to exit console.
```



```python
import re

def isMatch(s, p):
    return bool(re.fullmatch(p, s))

s = input("Enter the input string: ")
p = input("Enter the pattern string: ")

print(isMatch(s, p))
```

```
Enter the input string: aa
Enter the pattern string: a
False


...Program finished with exit code 0
Press ENTER to exit console.
```