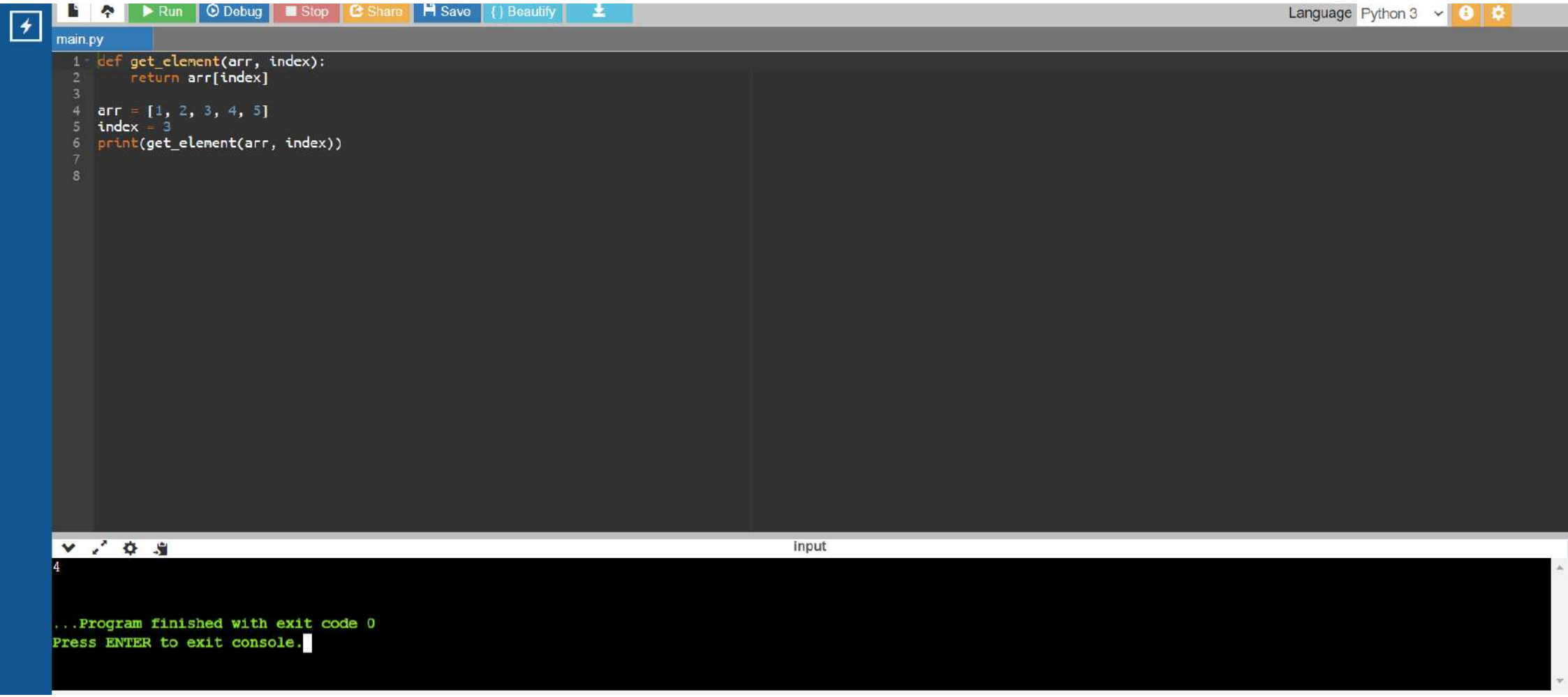
Run Debug Stop Share Save { } Beautify Language Python 3

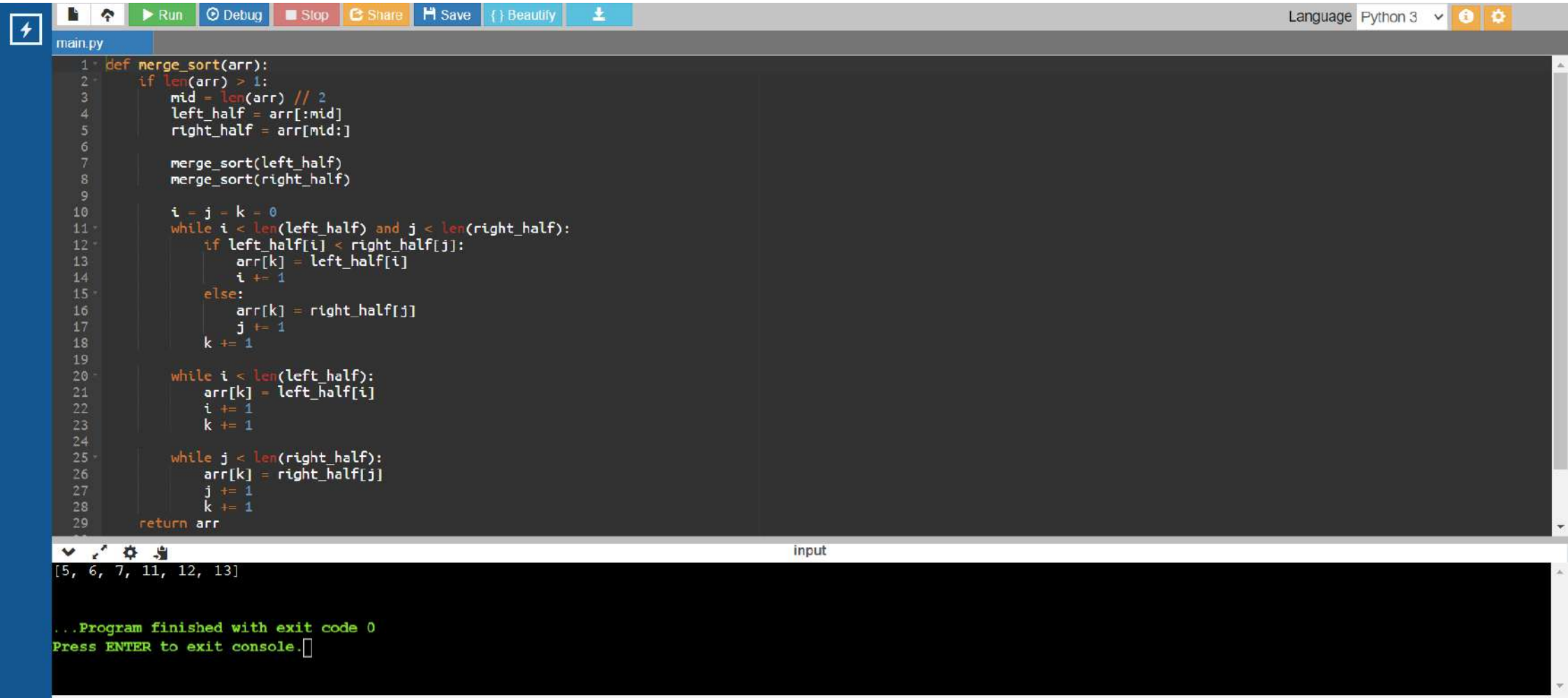
main.py

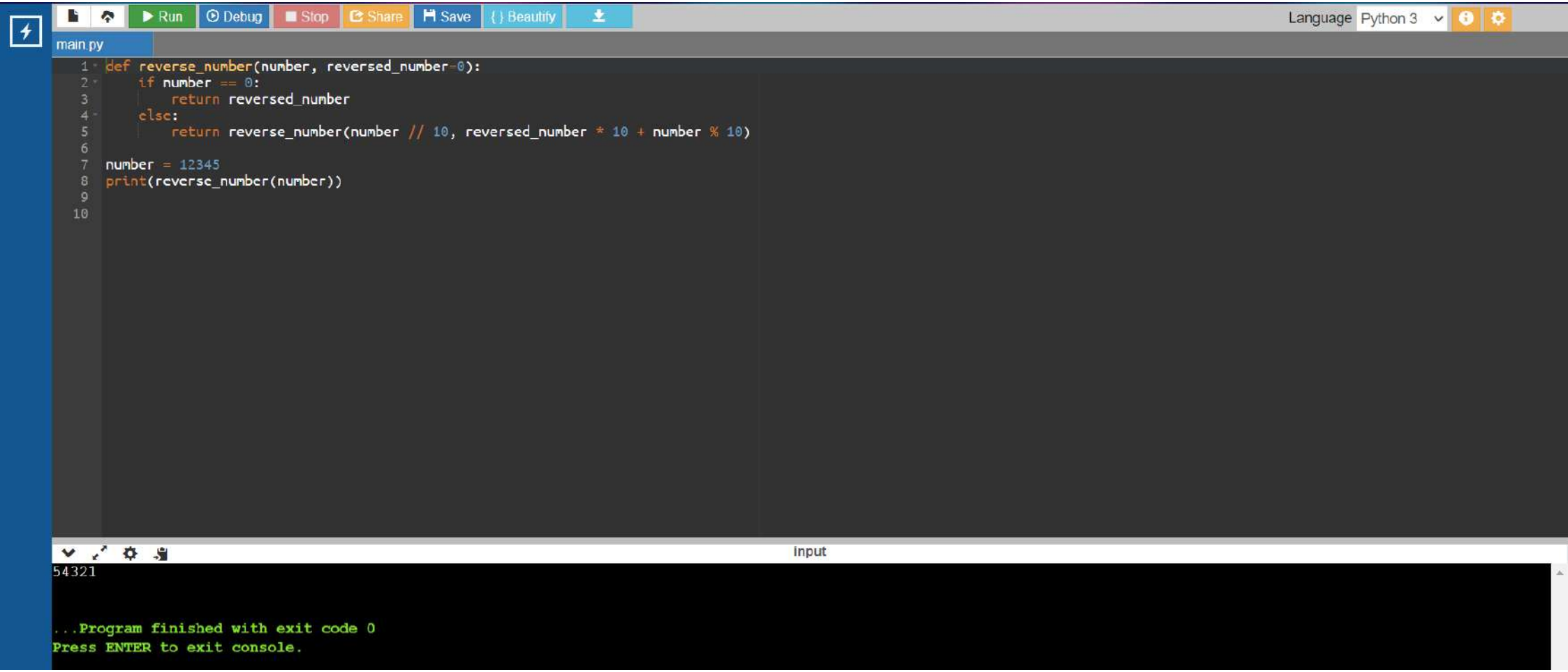
```python
def is_perfect(n):
    sum_divisors = sum(i for i in range(1, n) if n % i == 0)
    return sum_divisors == n

limit = 10000
for num in range(1, limit):
    if is_perfect(num):
        print(num)

```

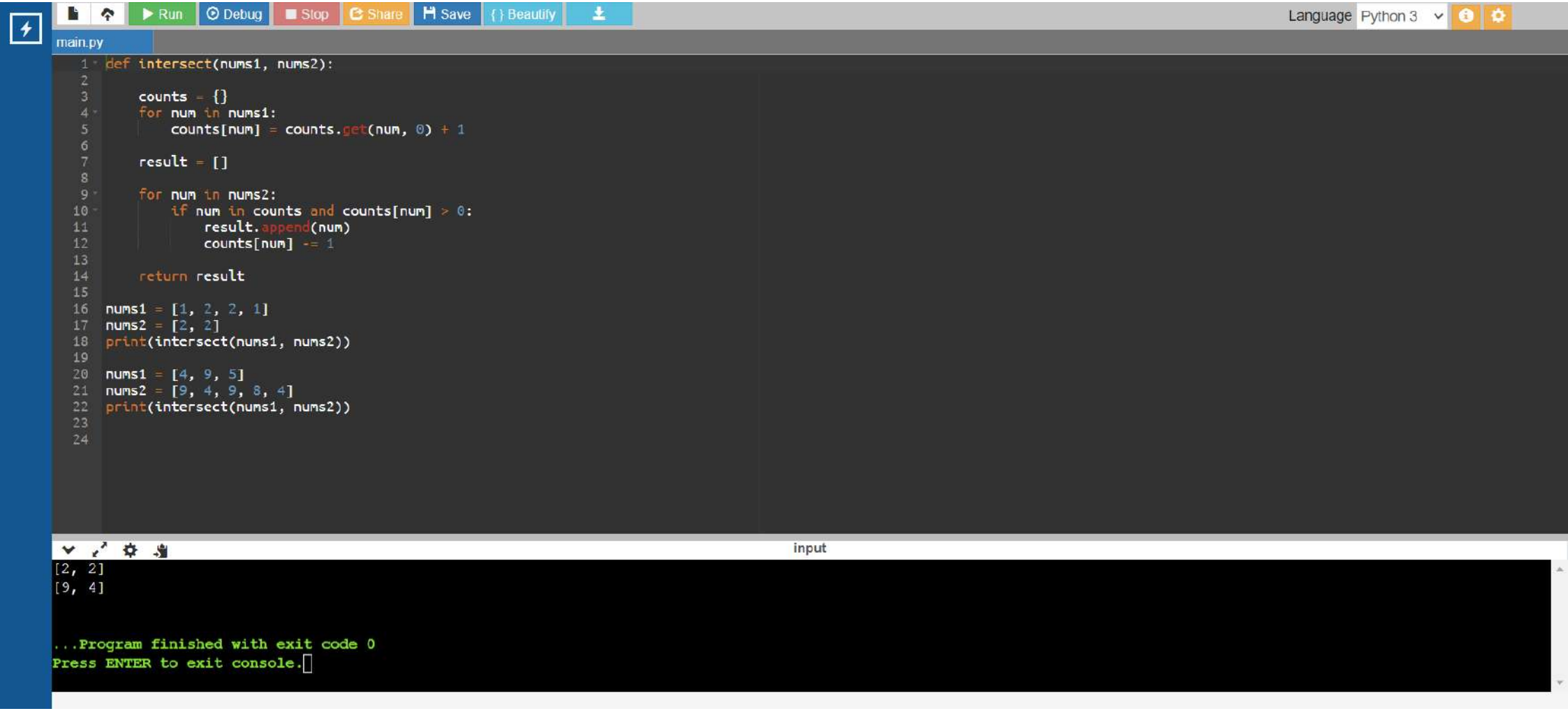
input

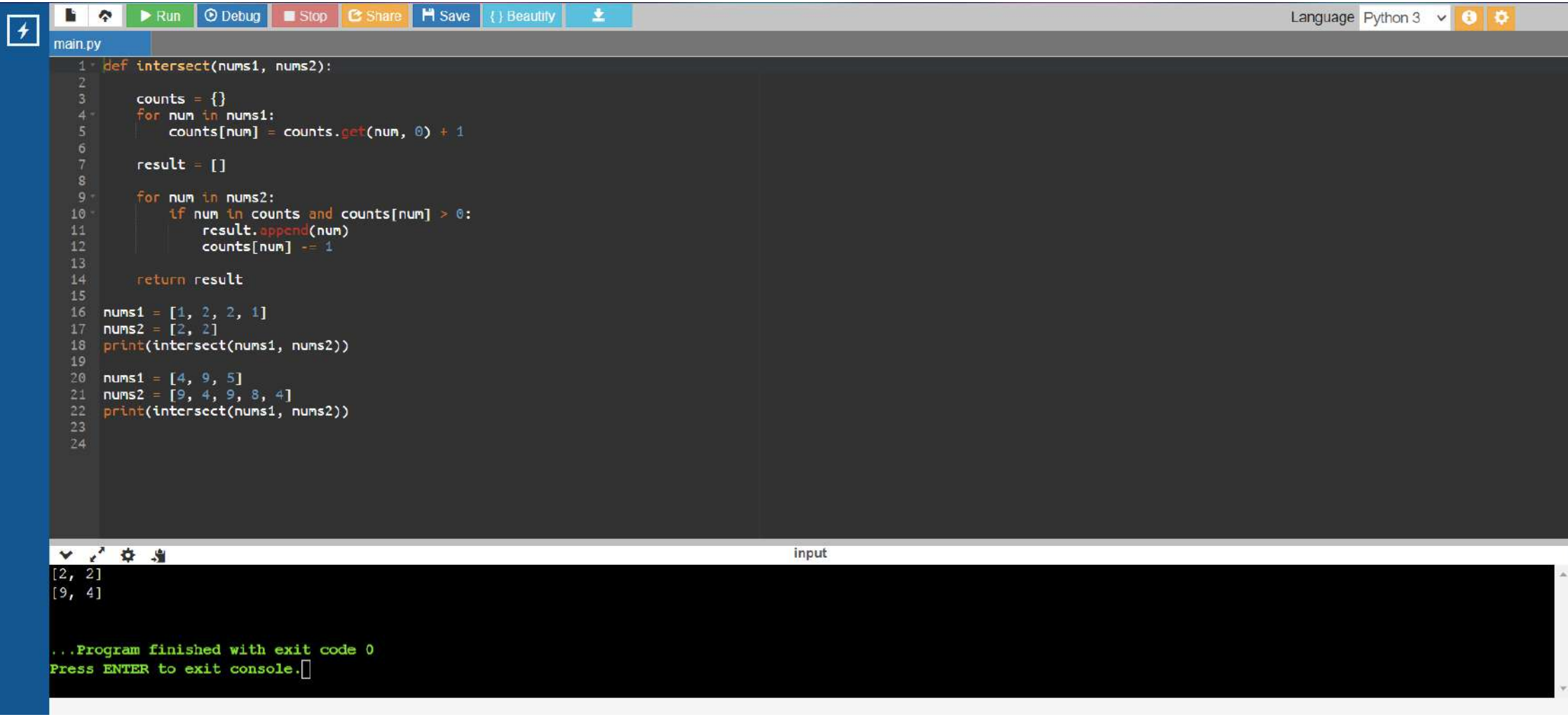
```
28
496
8128


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

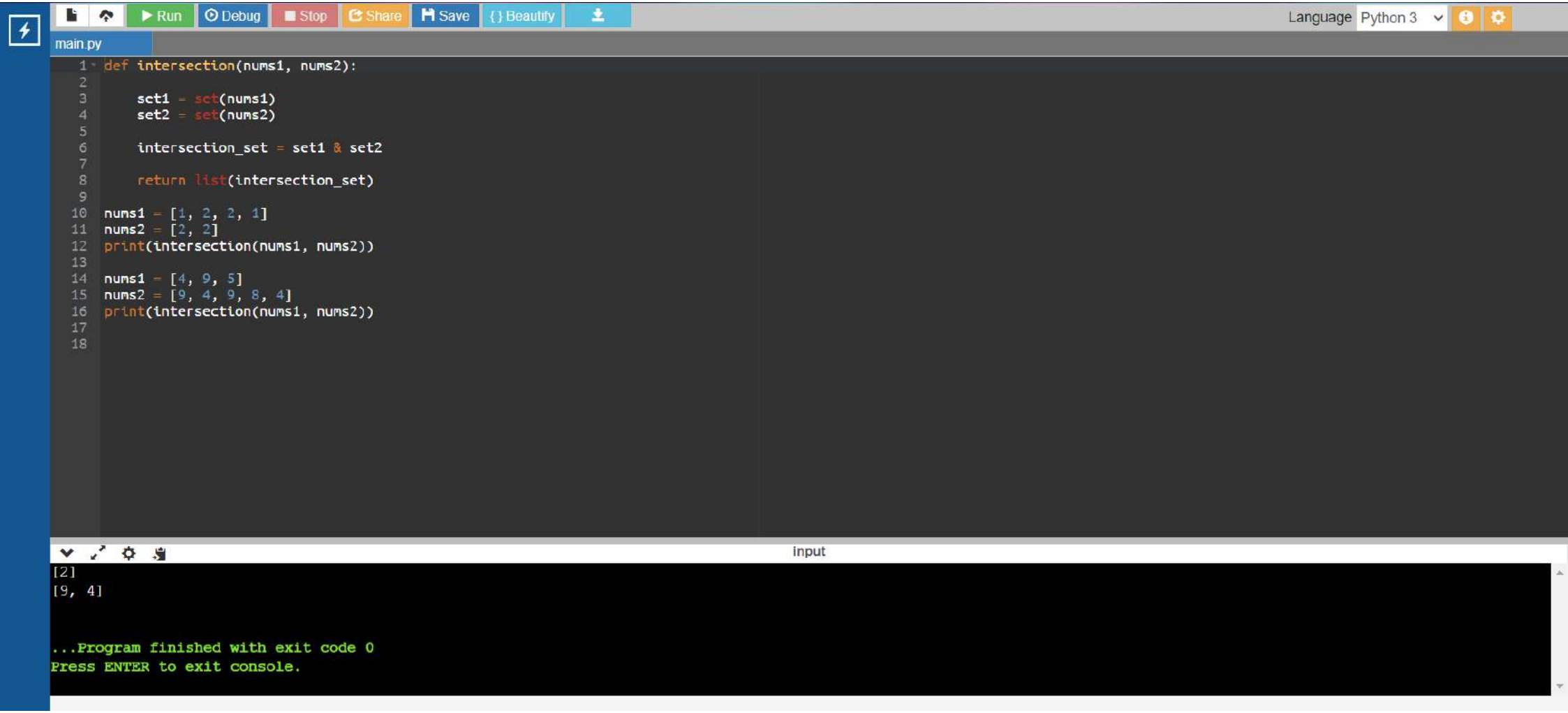

Run
Debug
Stop
Share
Save
{ } Beautify
Language
Python 3
main.py
1 def get_element(arr, index):
2 return arr[index]
3
4 arr = [1, 2, 3, 4, 5]
5 index = 3
6 print(get_element(arr, index))
7
8
input
4
...Program finished with exit code 0
Press ENTER to exit console.

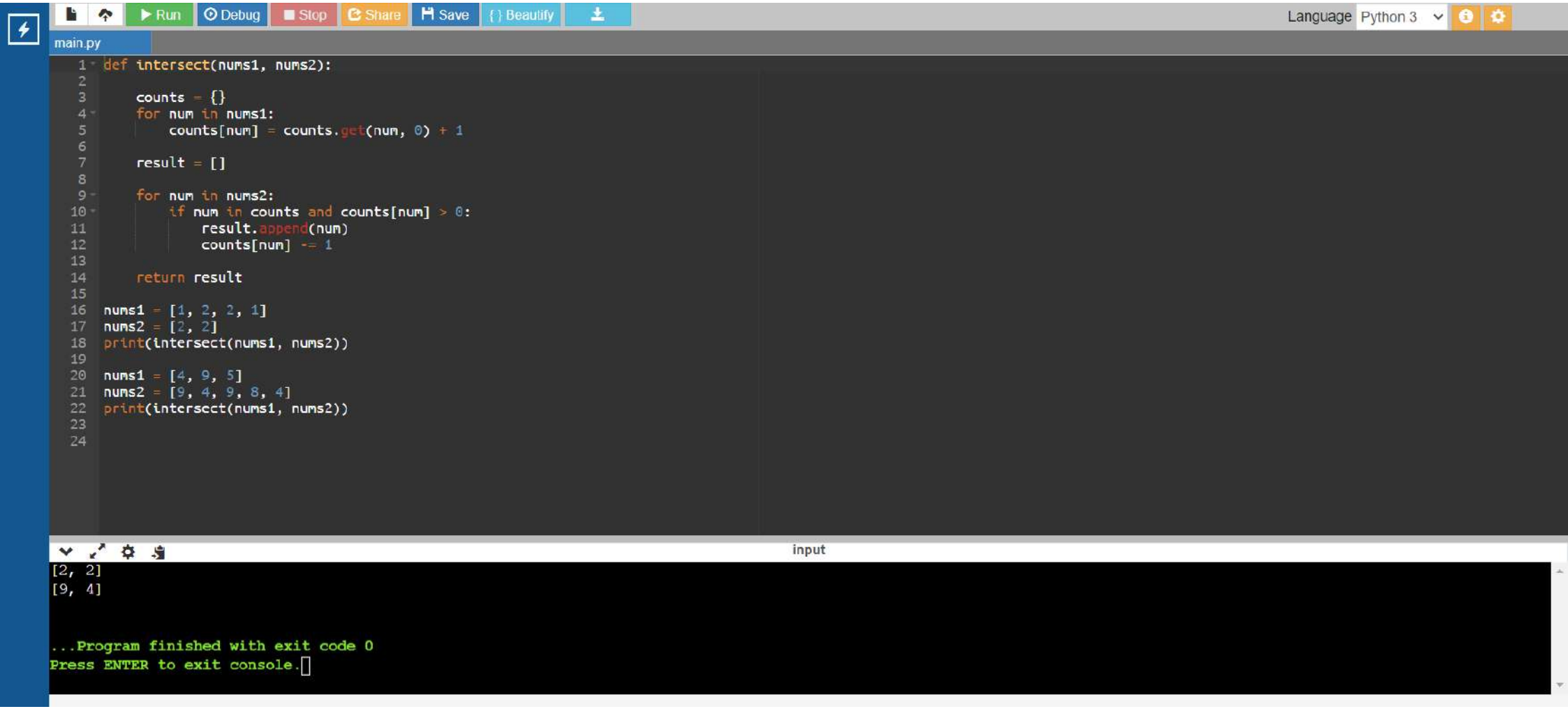

Run
Debug
Stop
Share
Save
{ } Beautify
Language
Python 3
main.py
def merge_sort(arr):
if len(arr) > 1:
mid = len(arr) // 2
left_half = arr[:mid]
right_half = arr[mid:]
merge_sort(left_half)
merge_sort(right_half)
i = j = k = 0
while i < len(left_half) and j < len(right_half):
if left_half[i] < right_half[j]:
arr[k] = left_half[i]
i += 1
else:
arr[k] = right_half[j]
j += 1
k += 1
while i < len(left_half):
arr[k] = left_half[i]
i += 1
k += 1
while j < len(right_half):
arr[k] = right_half[j]
j += 1
k += 1
return arr
input
[5, 6, 7, 11, 12, 13]
...Program finished with exit code 0
Press ENTER to exit console.

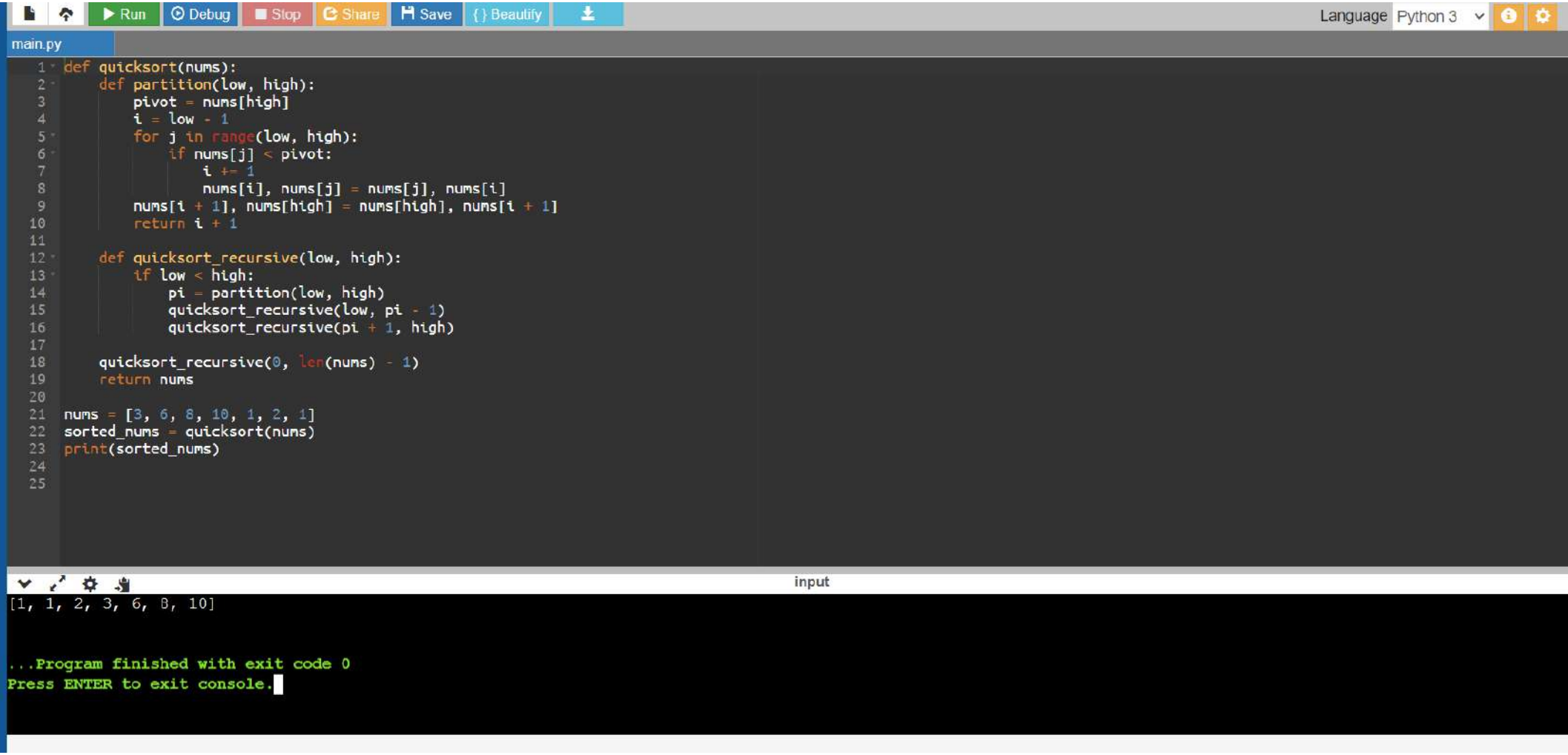

Run
Debug
Stop
Share
Save
{ } Beautify
Language
Python 3
main.py
def reverse_number(number, reversed_number=0):
if number == 0:
return reversed_number
else:
return reverse_number(number // 10, reversed_number * 10 + number % 10)
number = 12345
print(reverse_number(number))
Input
54321
...Program finished with exit code 0
Press ENTER to exit console.


Run
Debug
Stop
Share
Save
{ } Beautify
Language
Python 3
main.py
def intersect(nums1, nums2):
counts = {}
for num in nums1:
counts[num] = counts.get(num, 0) + 1
result = []
for num in nums2:
if num in counts and counts[num] > 0:
result.append(num)
counts[num] -= 1
return result
nums1 = [1, 2, 2, 1]
nums2 = [2, 2]
print(intersect(nums1, nums2))
nums1 = [4, 9, 5]
nums2 = [9, 4, 9, 8, 4]
print(intersect(nums1, nums2))
input
[2, 2]
[9, 4]
...Program finished with exit code 0
Press ENTER to exit console.


Run
Debug
Stop
Share
Save
{ } Beautify
Language
Python 3
main.py
def intersect(nums1, nums2):
counts = {}
for num in nums1:
counts[num] = counts.get(num, 0) + 1
result = []
for num in nums2:
if num in counts and counts[num] > 0:
result.append(num)
counts[num] -= 1
return result
nums1 = [1, 2, 2, 1]
nums2 = [2, 2]
print(intersect(nums1, nums2))
nums1 = [4, 9, 5]
nums2 = [9, 4, 9, 8, 4]
print(intersect(nums1, nums2))
input
[2, 2]
[9, 4]
...Program finished with exit code 0
Press ENTER to exit console.


Run
Debug
Stop
Share
Save
{} Beautify
Language
Python 3
main.py
def intersection(nums1, nums2):
sct1 = sct(nums1)
set2 = set(nums2)
intersection_set = set1 & set2
return list(intersection_set)
nums1 = [1, 2, 2, 1]
nums2 = [2, 2]
print(intersection(nums1, nums2))
nums1 = [4, 9, 5]
nums2 = [9, 4, 9, 8, 4]
print(intersection(nums1, nums2))
input
[2]
[9, 4]
...Program finished with exit code 0
Press ENTER to exit console.


Run
Debug
Stop
Share
Save
{ } Beautify
Language
Python 3
main.py
def intersect(nums1, nums2):
counts = {}
for num in nums1:
counts[num] = counts.get(num, 0) + 1
result = []
for num in nums2:
if num in counts and counts[num] > 0:
result.append(num)
counts[num] -= 1
return result
nums1 = [1, 2, 2, 1]
nums2 = [2, 2]
print(intersect(nums1, nums2))
nums1 = [4, 9, 5]
nums2 = [9, 4, 9, 8, 4]
print(intersect(nums1, nums2))
input
[2, 2]
[9, 4]
...Program finished with exit code 0
Press ENTER to exit console.


Run
Debug
Stop
Share
Save
{} Beautify
Language
Python 3
main.py
def quicksort(nums):
def partition(low, high):
pivot = nums[high]
i = low - 1
for j in range(low, high):
if nums[j] < pivot:
i += 1
nums[i], nums[j] = nums[j], nums[i]
nums[i + 1], nums[high] = nums[high], nums[i + 1]
return i + 1
def quicksort_recursive(low, high):
if low < high:
pi = partition(low, high)
quicksort_recursive(low, pi - 1)
quicksort_recursive(pi + 1, high)
quicksort_recursive(0, len(nums) - 1)
return nums
nums = [3, 6, 8, 10, 1, 2, 1]
sorted_nums = quicksort(nums)
print(sorted_nums)
input
[1, 1, 2, 3, 6, 8, 10]
...Program finished with exit code 0
Press ENTER to exit console.

Run | Debug | Stop | Share | Save | { } Beautify | Language Python 3

main.py

```python
def sort_array_by_parity(nums):

    odds = [num for num in nums if num % 2 != 0]
    evens = [num for num in nums if num % 2 == 0]

    result = [0] * len(nums)

    odd_index = 1
    even_index = 0

    for num in evens:
        result[even_index] = num
        even_index += 2

    for num in odds:
        result[odd_index] = num
        odd_index += 2

    return result

nums = [4, 2, 5, 7]
sorted_nums = sort_array_by_parity(nums)
print(sorted_nums)
```

input

```
[4, 5, 2, 7]


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