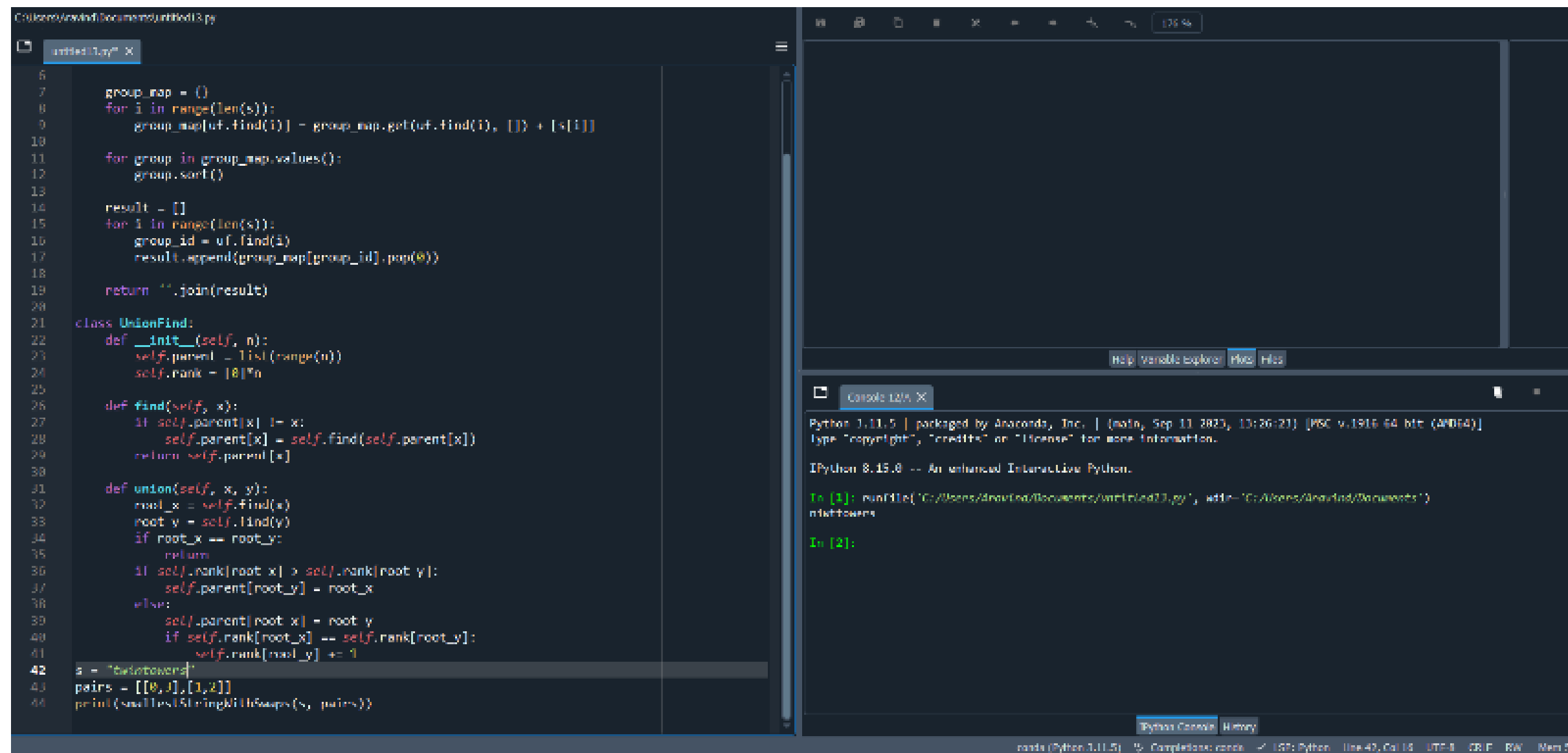
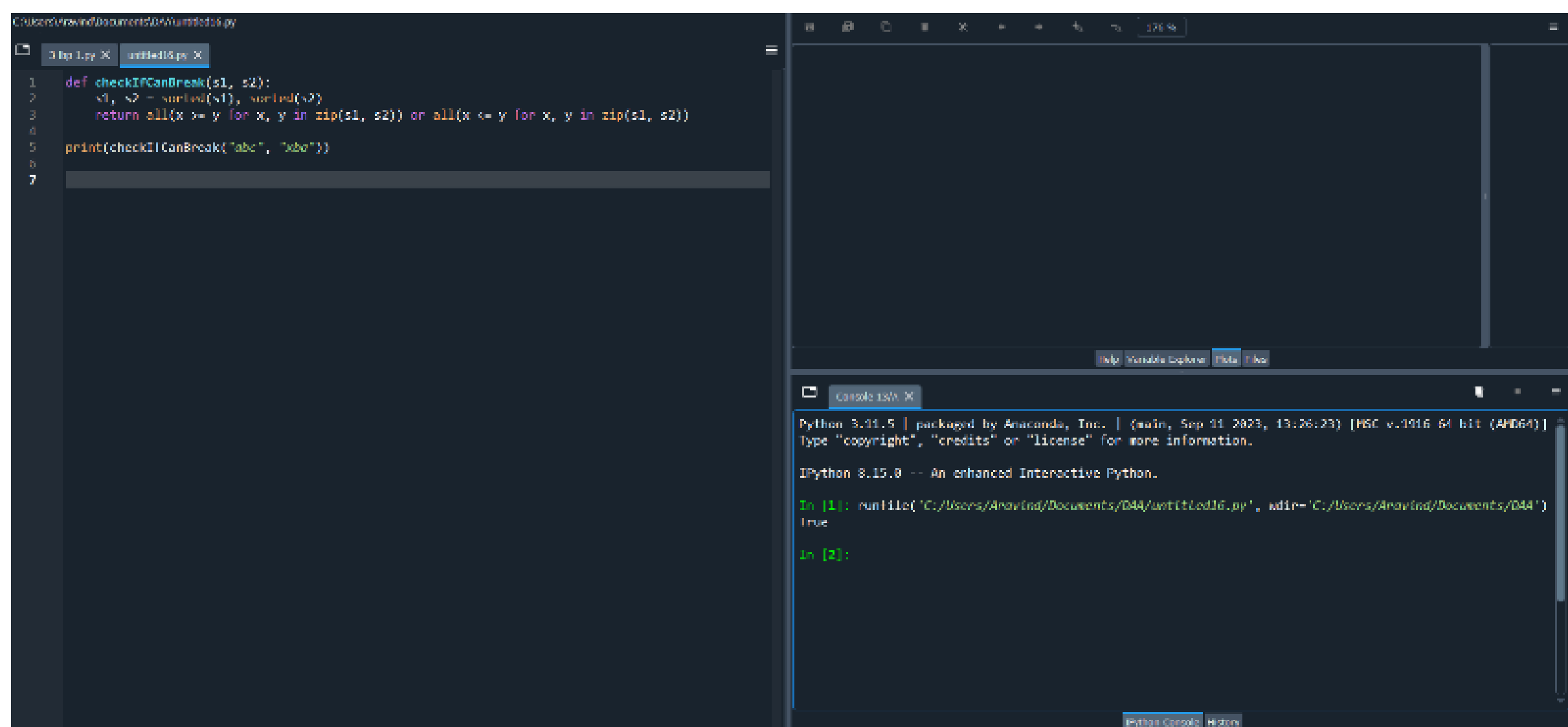


1. You are given a string  $s$ , and an array of pairs of indices in the string  $\text{pairs}$  where  $\text{pairs}[i] = [a, b]$  indicates 2 indices (0-indexed) of the string. You can swap the characters at any pair of indices in the given pairs any number of times. Return the lexicographically smallest string that  $s$  can be changed to after using the swaps.



```
6
7     group_map = {}
8     for i in range(len(s)):
9         group_map.setdefault(uf.find(i), []).append(s[i])
10
11     for group in group_map.values():
12         group.sort()
13
14     result = []
15     for i in range(len(s)):
16         group_id = uf.find(i)
17         result.append(group_map[group_id].pop(0))
18
19     return ''.join(result)
20
21 class UnionFind:
22     def __init__(self, n):
23         self.parent = list(range(n))
24         self.rank = [0]*n
25
26     def find(self, x):
27         if self.parent[x] != x:
28             self.parent[x] = self.find(self.parent[x])
29         return self.parent[x]
30
31     def union(self, x, y):
32         root_x = self.find(x)
33         root_y = self.find(y)
34         if root_x == root_y:
35             return
36         if self.rank[root_x] > self.rank[root_y]:
37             self.parent[root_y] = root_x
38         else:
39             self.parent[root_x] = root_y
40             if self.rank[root_x] == self.rank[root_y]:
41                 self.rank[root_y] += 1
42
43 s = "baotowers"
44 pairs = [[0,1],[1,2]]
45 print(smallestStringWithSwaps(s, pairs))
```

Given two strings:  $s_1$  and  $s_2$  with the same size, check if some permutation of string  $s_1$  can break some permutation of string  $s_2$  or vice-versa. In other words  $s_2$  can break  $s_1$  or vice-versa. A string  $x$  can break string  $y$  (both of size  $n$ ) if  $x[i] \geq y[i]$  (in alphabetical order) for all  $i$  between 0 and  $n-1$ .



```
1 def checkIfCanBreak(s1, s2):
2     s1, s2 = sorted(s1), sorted(s2)
3     return all(x >= y for x, y in zip(s1, s2)) or all(x <= y for x, y in zip(s1, s2))
4
5 print(checkIfCanBreak("abc", "bca"))
6
7
```

Given two strings: s1 and s2 with the same size, check if some permutation of string s1 can break some permutation of string s2 or vice-versa. In other words s2 can break s1 or vice-versa. A string x can break string y (both of size n) if  $x[i] \geq y[i]$  (in alphabetical order) for all i between 0 and n-1.

The screenshot shows a Jupyter Notebook interface. The left pane displays a Python script in a file named 'untitled16.py'. The script defines a function `checkIfCanBreak(s1, s2)` that sorts both strings and compares them character by character. It returns `True` if `s1[i] >= s2[i]` for all `i`, and `False` otherwise. The function is called with `checkIfCanBreak("abc", "xbo")`, and the result is printed. The right pane shows the Jupyter Notebook's console, which displays the output of the function call: `True`.

```

1 def checkIfCanBreak(s1, s2):
2     s1, s2 = sorted(s1), sorted(s2)
3     return all(x >= y for x, y in zip(s1, s2)) or all(x <= y for x, y in zip(s1, s2))
4
5 print(checkIfCanBreak("abc", "xbo"))
6
7

```

```

Python 3.11.5 | packaged by Anaconda, Inc. | (main, Sep 11 2023, 13:26:23) [AMD64]
Type "copyright", "credits" or "license()" for more information.

IPython 8.15.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/Aravind/Documents/D44/untitled16.py', wdir='C:/Users/Aravind/Documents/D44')
True

In [2]:

```

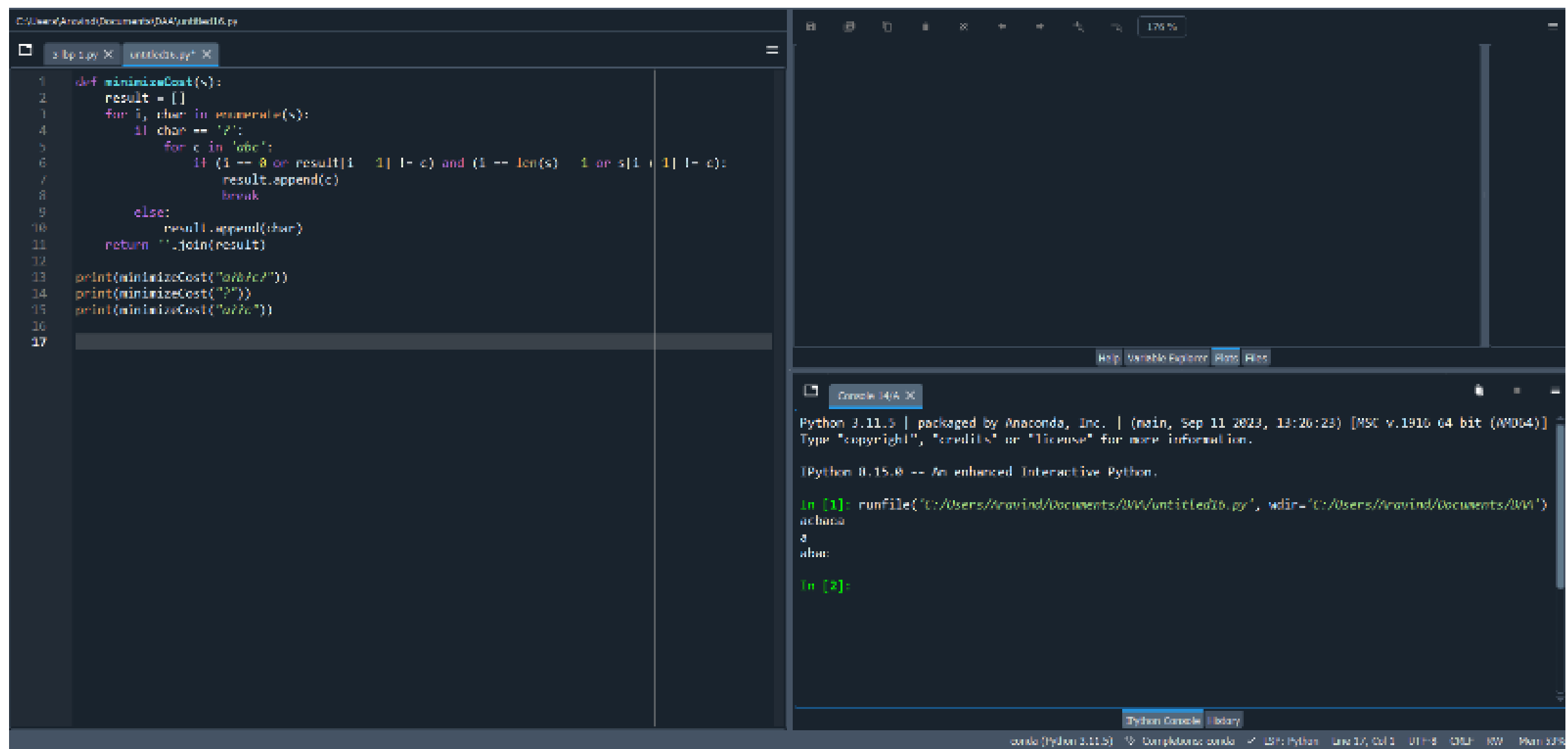
You are given a string s.  $s[i]$  is either a lowercase English letter or '?'. For a string t having length m containing only lowercase English letters, we define the function  $\text{cost}(i)$  for an index i as the number of characters equal to  $t[i]$  that appeared before it, i.e. in the range  $[0, i-1]$ . The value of t is the sum of  $\text{cost}(i)$  for all indices i. For example, for the string  $t = \text{"aab"}$ :

$$\text{cost}(0) = 0$$

$$\text{cost}(1) = 1$$

$$\text{cost}(2) = 0$$

Hence, the value of "aab" is  $0 + 1 + 0 = 1$ . Your task is to replace all occurrences of '?' in s with any lowercase English letter so that the value of s is minimized.



```
1 def minimizeCost(s):
2     result = []
3     for i, char in enumerate(s):
4         if char == 'Z':
5             for c in 'abc':
6                 if (i == 0 or result[i-1] != c) and (i == len(s)-1 or s[i+1] != c):
7                     result.append(c)
8                     break
9             else:
10                result.append(char)
11     return ''.join(result)
12
13 print(minimizeCost("aabbcc"))
14 print(minimizeCost("b"))
15 print(minimizeCost("a"))
16
17
```

```
Python 3.11.5 | packaged by Anaconda, Inc. | (main, Sep 11 2023, 13:26:23) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license()" for more information.

Python 3.11.5 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/Aravind/Documents/DA/untitled16.py', wdir='C:/Users/Aravind/Documents/DA')
aachaa
a
aahaa

In [2]:
```

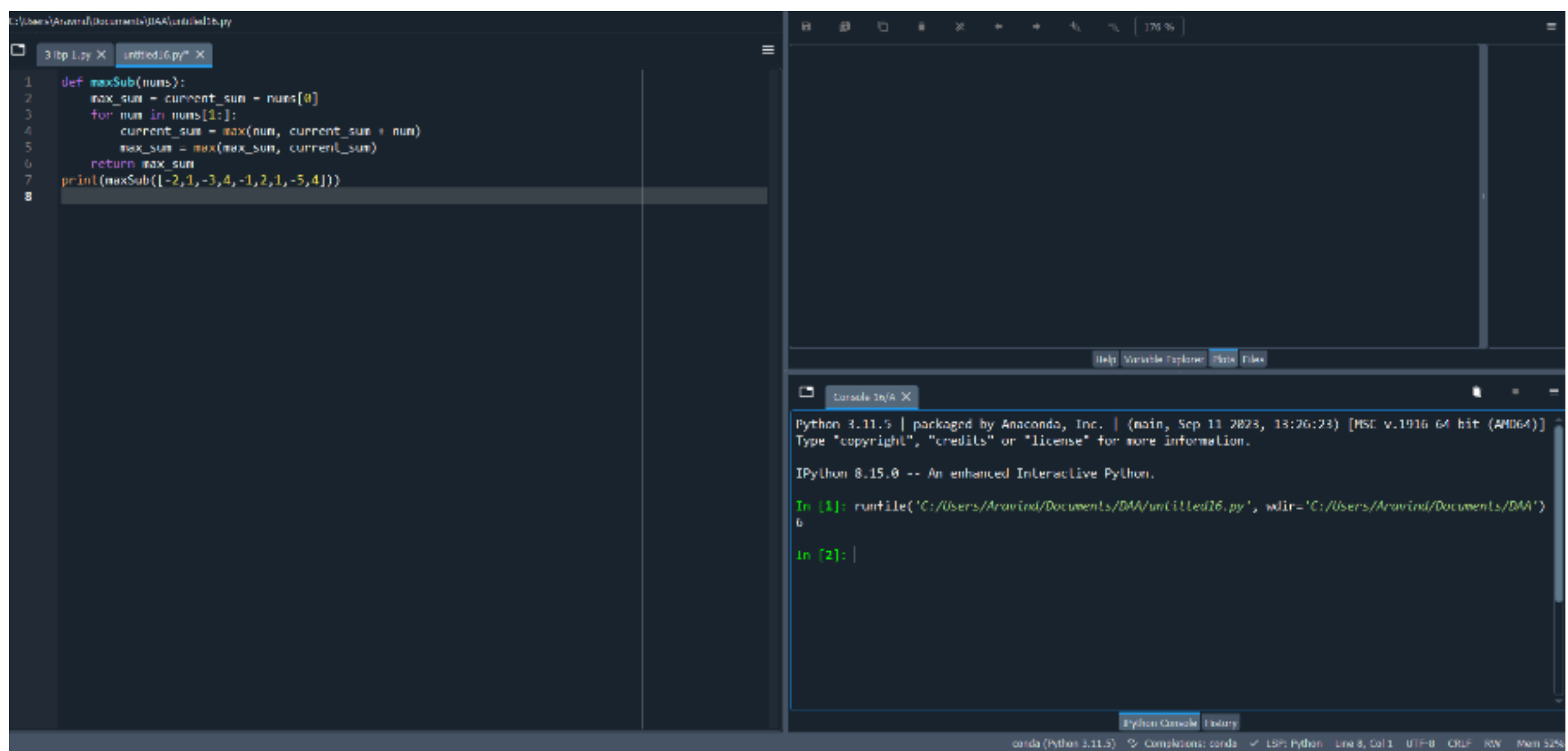
Given an integer array `nums`, find the subarray with the largest sum, and return its sum.

Example 1:

Input: `nums = [-2,1,-3,4,-1,2,1,-5,4]`

Output: 6

Explanation: The subarray `[4,-1,2,1]` has the largest sum 6.



```
1 def maxSub(nums):
2     max_sum = current_sum = nums[0]
3     for num in nums[1:]:
4         current_sum = max(num, current_sum + num)
5         max_sum = max(max_sum, current_sum)
6     return max_sum
7 print(maxSub([-2,1,-3,4,-1,2,1,-5,4]))
8
```

```
Python 3.11.5 | packaged by Anaconda, Inc. | (main, Sep 11 2023, 13:26:24) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license()" for more information.

IPython 8.15.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/Aravind/Documents/DA/untitled16.py', wdir='C:/Users/Aravind/Documents/DA')
6

In [2]:
```

You are given an integer array `nums` with no duplicates. A maximum binary tree can be built recursively from `nums` using the following algorithm: Create a root node whose value is the maximum value in `nums`. Recursively build the left subtree on the subarray prefix to the left of the maximum value. Recursively build the right subtree on the subarray suffix to the right of the maximum value. Return the maximum binary tree built from `nums`.

```
C:\Users\Aravind\Documents\DMA\untitled6.py
untitled6.py X
1 class TreeNode:
2     def __init__(self, x):
3         self.val = x
4         self.left = None
5         self.right = None
6
7 def constructMaximumBinaryTree(nums):
8     if not nums:
9         return None
10
11     max_val = max(nums)
12     max_idx = nums.index(max_val)
13
14     root = TreeNode(max_val)
15     root.left = constructMaximumBinaryTree(nums[:max_idx])
16     root.right = constructMaximumBinaryTree(nums[max_idx+1:])
17
18     return root
19
20 nums = [3, 2, 1, 6, 0, 5]
21 root = constructMaximumBinaryTree(nums)
22
23 # Print the tree
24 def print_tree(node, level=0):
25     if node:
26         print_tree(node.right, level+1)
27         print(' ' * level + str(node.val))
28         print_tree(node.left, level+1)
29
30 print_tree(root)
```

```
Python 3.11.5 | packaged by Anaconda, Inc. | (main, Sep 11 2023, 13:26:23) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license()" for more information.

IPython 8.15.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/Aravind/Documents/DMA/untitled6.py', wdir='C:/Users/Aravind/Documents/DMA')
5
0
6
1
2
3
In [2]: |
```

Given a circular integer array `nums` of length `n`, return the maximum possible sum of a non-empty subarray of `nums`. A circular array means the end of the array connects to the beginning of the array. Formally, the next element of `nums[i]` is `nums[(i + 1) % n]` and the previous element of `nums[i]` is `nums[(i - 1 + n) % n]`. A subarray may only include each element of the fixed buffer `nums` at most once. Formally, for a subarray `nums[i], nums[i + 1], ..., nums[j]`, there does not exist  $i \leq k_1, k_2 \leq j$  with  $k_1 \% n == k_2 \% n$ .

```
C:\Users\Aravind\Documents\DMA\untitled6.py
untitled6.py X
1 def maxCircularSum(nums):
2     n = len(nums)
3     max_sum = float('-inf')
4     min_sum = float('inf')
5     total_sum = 0
6     curr_max_sum = 0
7     curr_min_sum = 0
8
9     for num in nums:
10         total_sum += num
11         curr_max_sum = max(num, curr_max_sum + num)
12         max_sum = max(max_sum, curr_max_sum)
13         curr_min_sum = min(num, curr_min_sum + num)
14         min_sum = min(min_sum, curr_min_sum)
15
16     if total_sum == min_sum:
17         return max_sum
18     else:
19         return max(max_sum, total_sum - min_sum)
20
21 a = [11, 10, -20, 5, -4, -5, 8, -13, 10]
22 print("Maximum circular sum is", maxCircularSum(a))
```

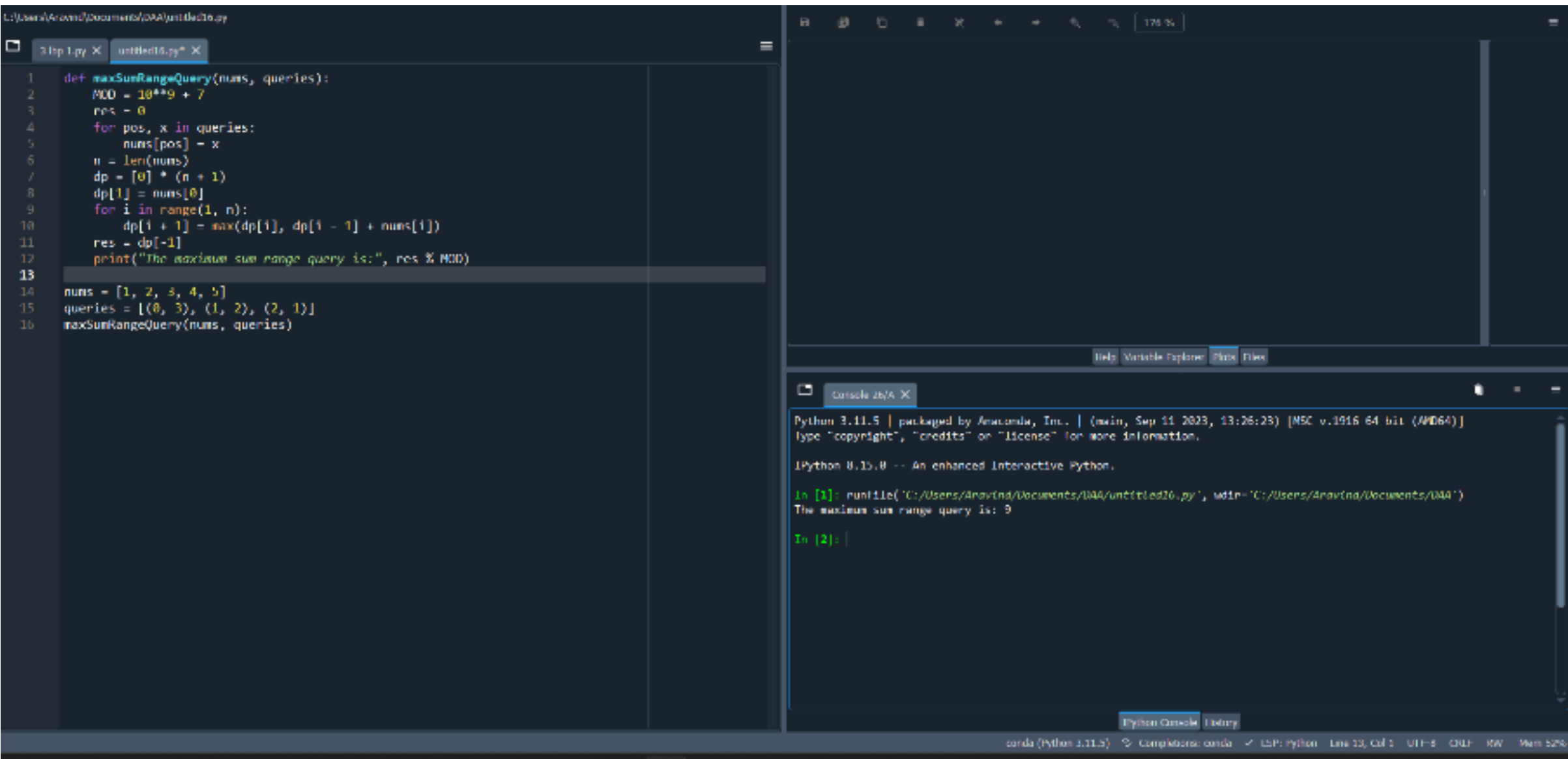
```
Python 3.11.5 | packaged by Anaconda, Inc. | (main, Sep 11 2023, 13:26:23) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license()" for more information.

IPython 8.15.0 -- An enhanced Interactive Python.

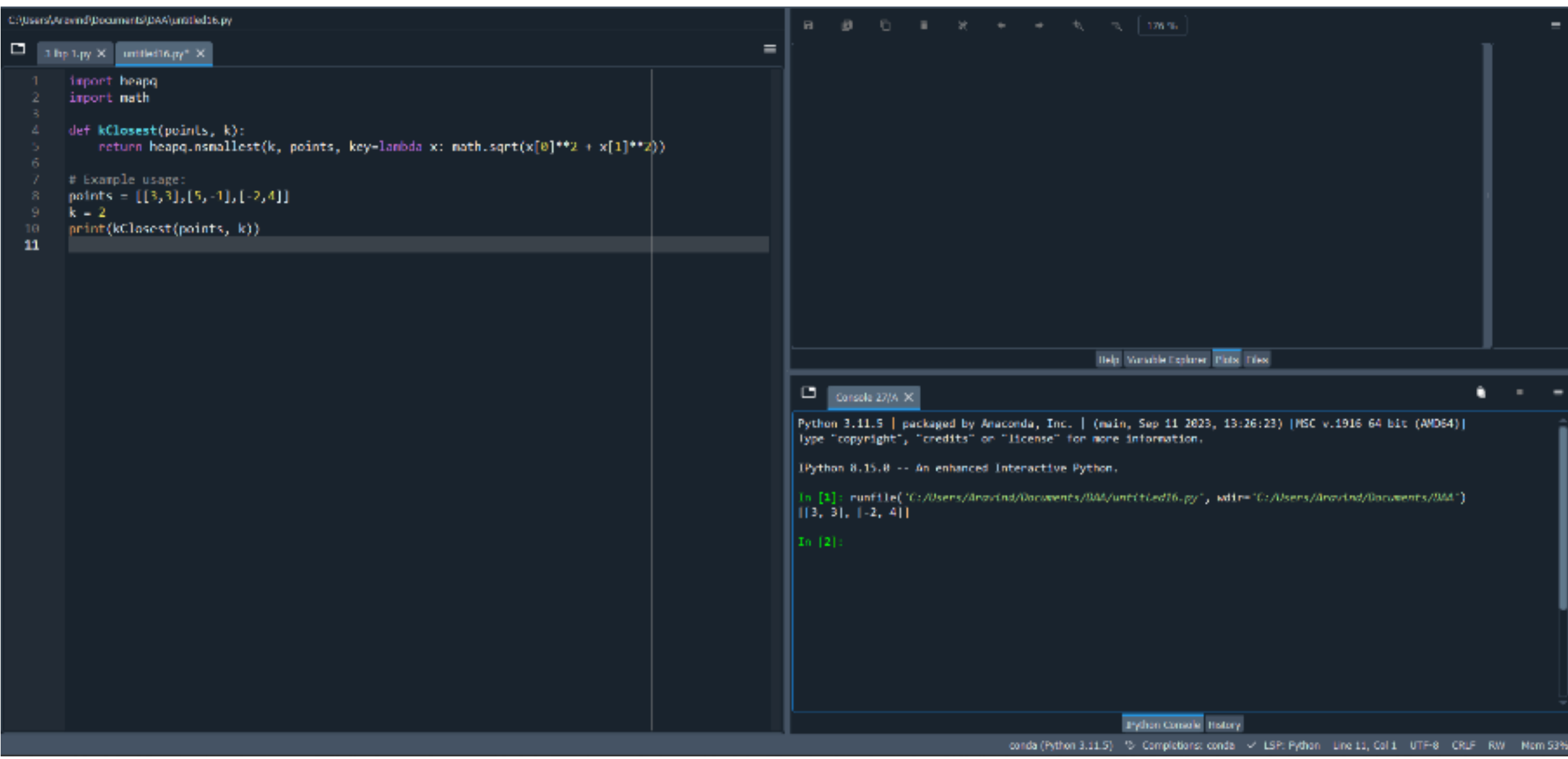
In [3]: runfile('C:/Users/Aravind/Documents/DMA/untitled6.py', wdir='C:/Users/Aravind/Documents/DMA')
Maximum circular sum is 31
In [2]: |
```

You are given an array `nums` consisting of integers. You are also given a 2D array `queries`, where `queries[i] = [posi, xi]`. For query `i`, we first set `nums[posi]` equal to `xi`, then we calculate the answer to query `i` which is the maximum sum of a subsequence of `nums` where no two adjacent elements are selected. Return the sum of the answers to all queries. Since the final answer may be very large,

return it modulo  $10^9 + 7$ . A subsequence is an array that can be derived from another array by deleting some or no elements without changing the order of the remaining elements.



Given an array of points where `points[i] = [xi, yi]` represents a point on the X-Y plane and an integer `k`, return the `k` closest points to the origin `(0,0)`. The distance between two points on the X-Y plane is the Euclidean distance (i.e.,  $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ ). You may return the answer in any order. The answer is guaranteed to be unique (except for the order that it is in).



Given two sorted arrays `nums1` and `nums2` of size `m` and `n` respectively, return the median of the two sorted arrays. The overall run time complexity should be  $O(\log(m+n))$ .

