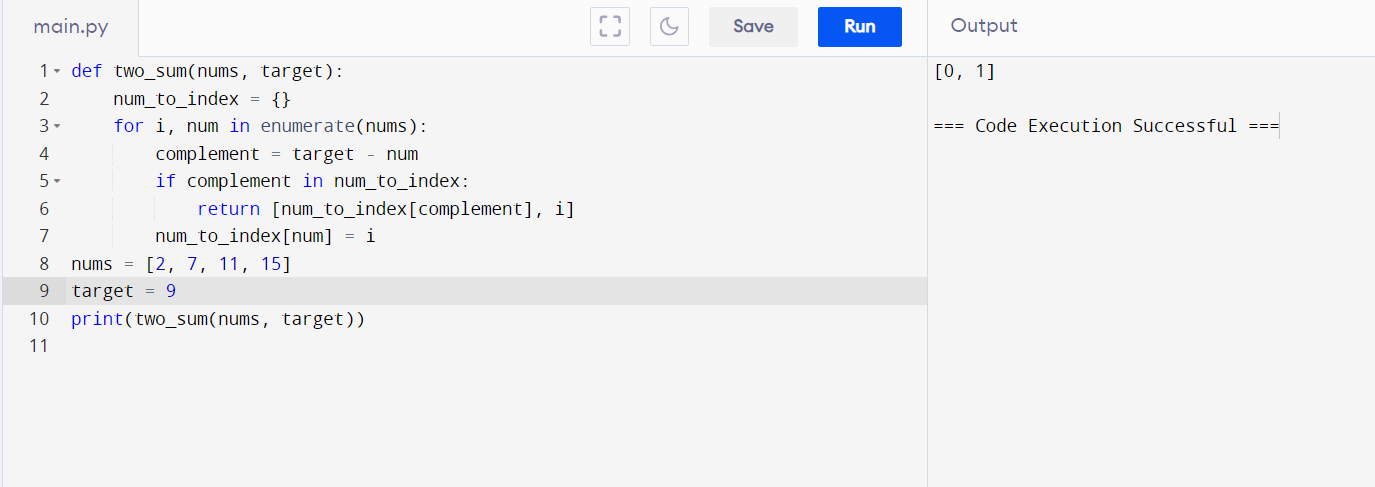
# **ASSIGNMENT-5**

# **1.Two Sum**

Given an array of integers nums and an integer target, return *indices of the two numbers such that they add up to target*.

You may assume that each input would have *exactly* one solution, and you may not use the *same* element twice.

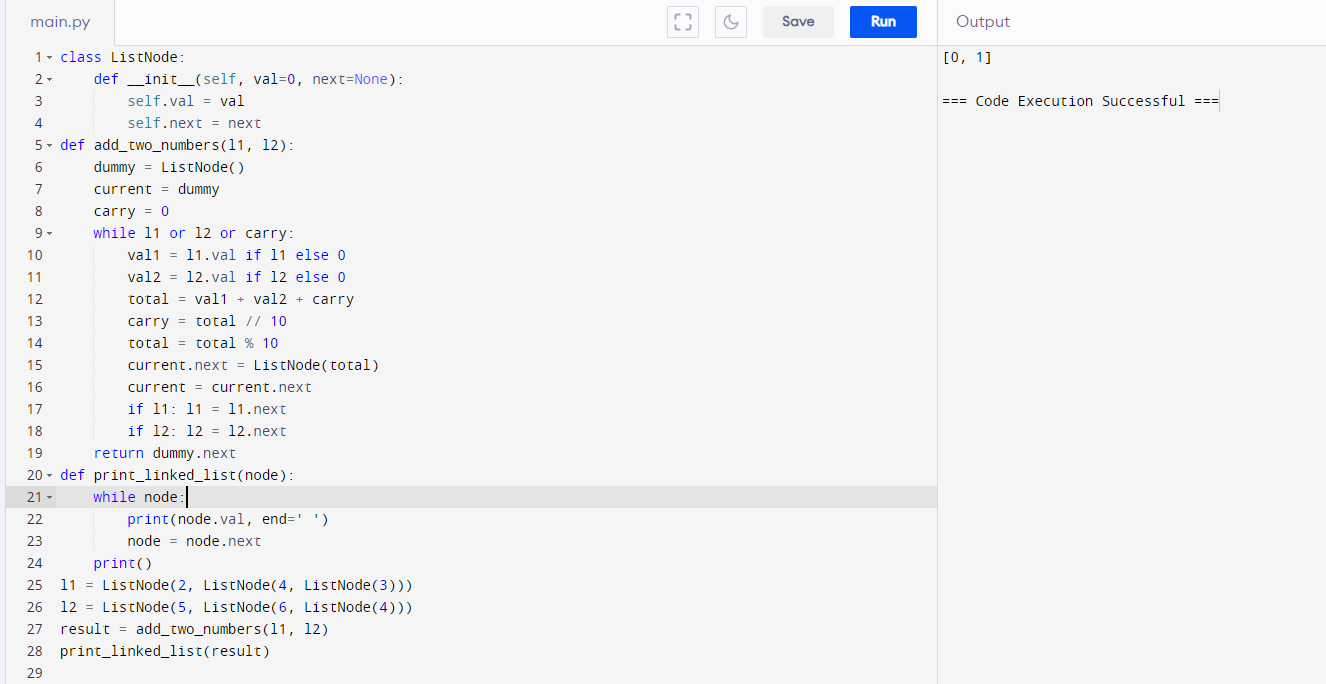
You can return the answer in any order.



# **2.Add Two Numbers**

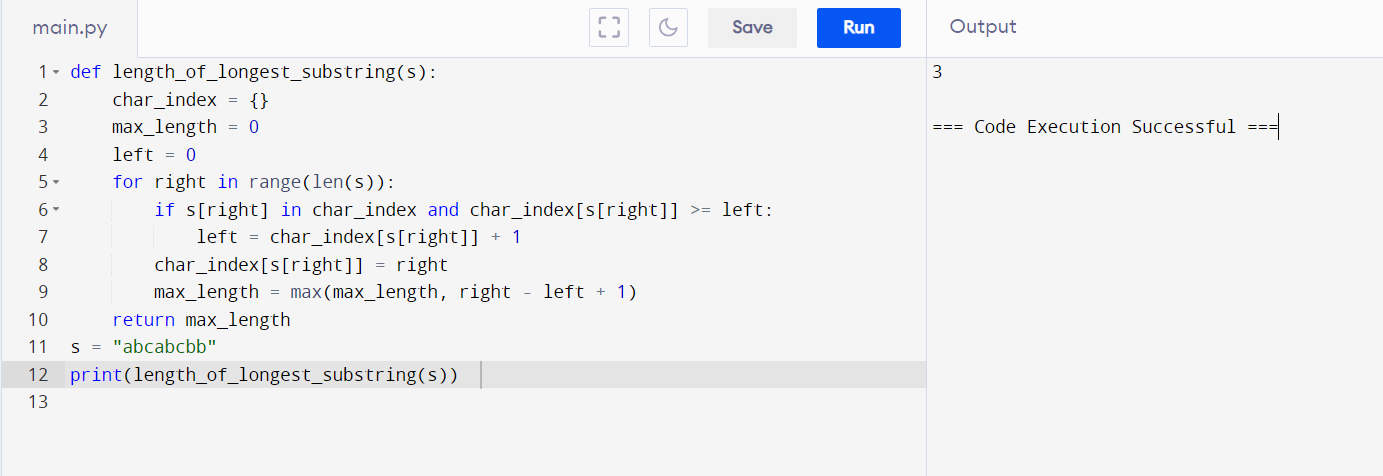
You are given two non-empty linked lists representing two non-negative integers. The digits are stored in reverse order, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.



# **Longest Substring without Repeating Characters**

Given a string s, find the length of the longest substring without repeating characters.

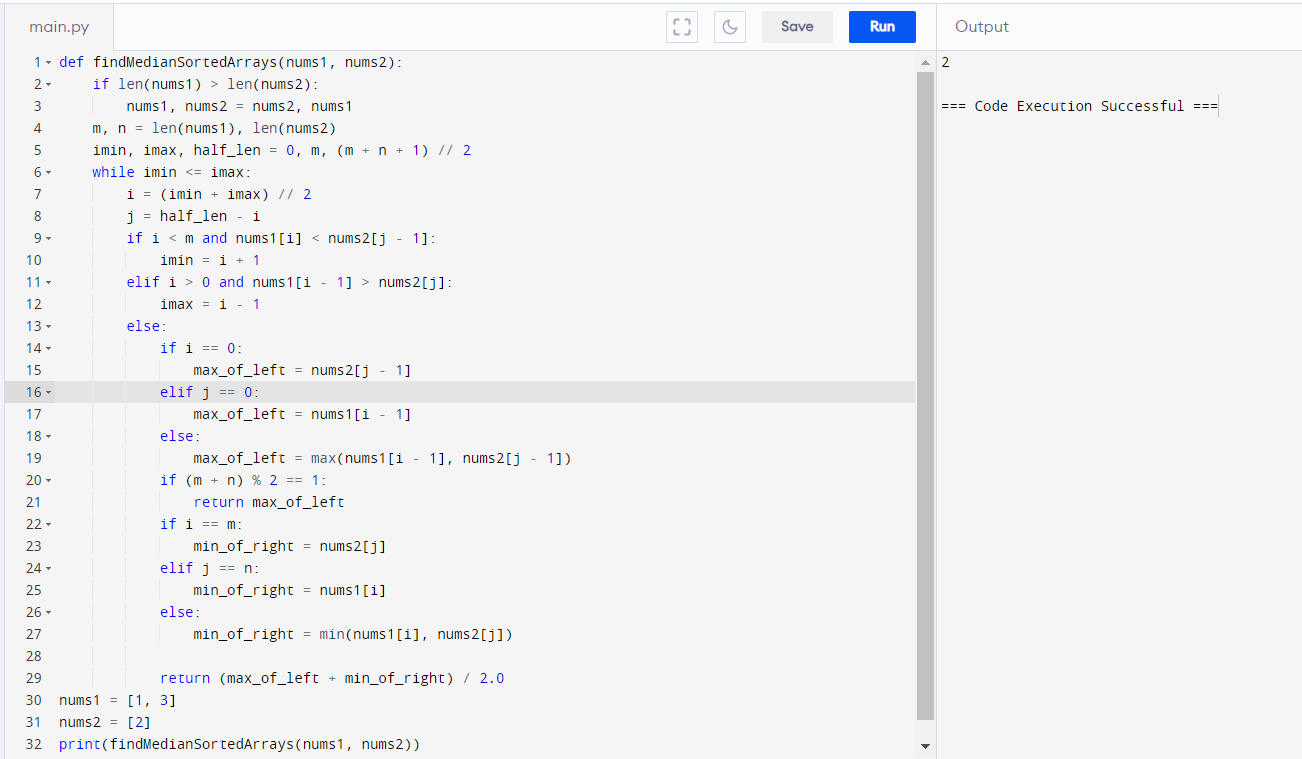


# **Median of Two Sorted Arrays**

Given two sorted arrays nums1 and nums2 of size m and n respectively, return the median of the two sorted arrays.

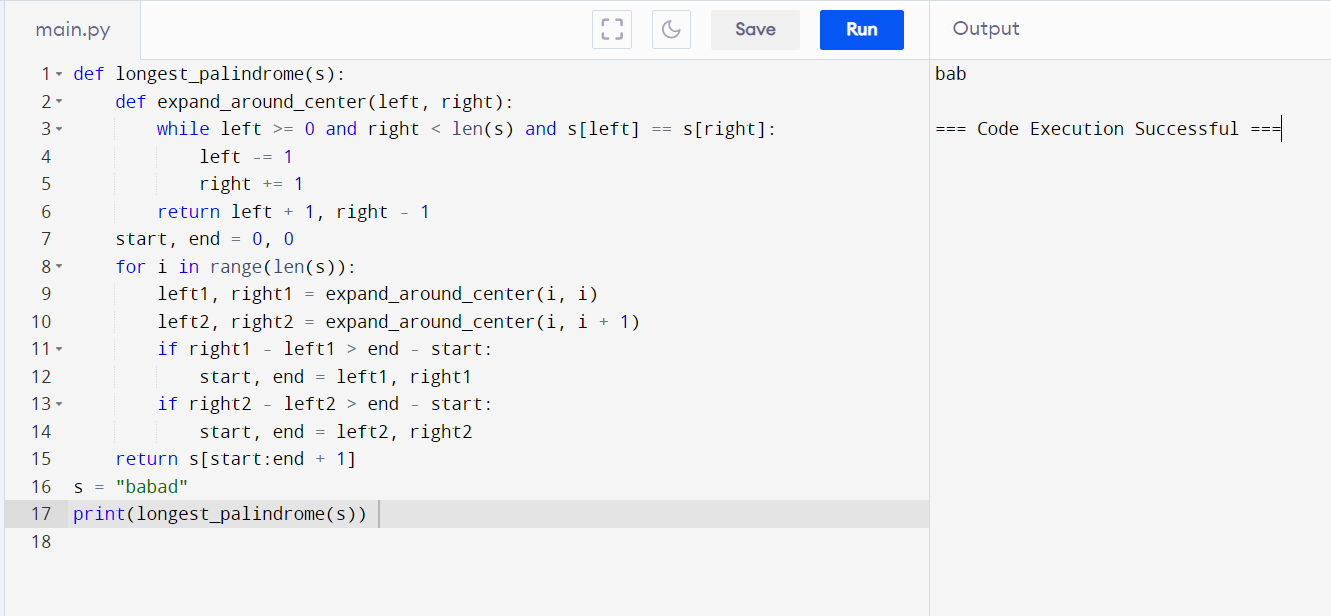
|  |
| --- |
| O(log (m+n)) |

The overall run time complexity should be .



# **Longest Palindromic Substring**

Given a string s, return *the longest* *palindromic* *substring* in s.



# **Zigzag Conversion**

The string "PAYPALISHIRING" is written in a zigzag pattern on a given number of rows like this: (you may want to display this pattern in a fixed font for better legibility) P A H N

A P L S I I G

Y I R

And then read line by line: "PAHNAPLSIIGYIR"

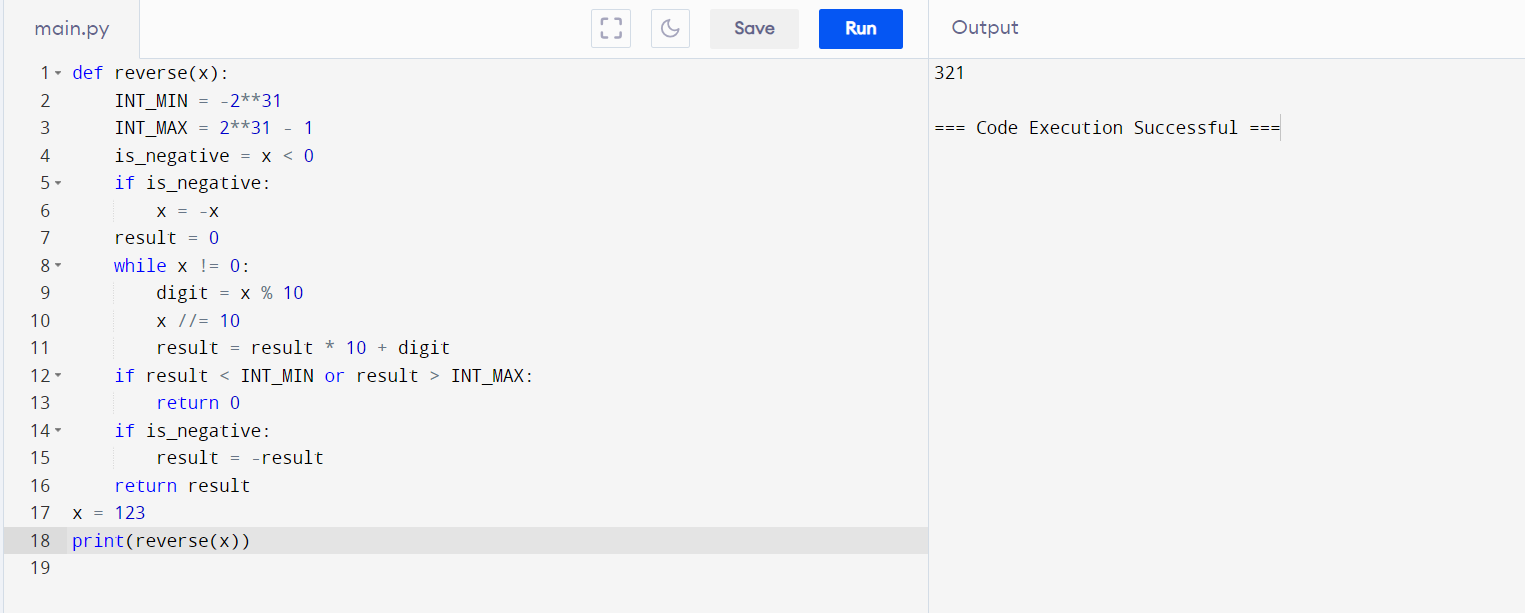
Write the code that will take a string and make this conversion given a number of rows:

string convert(string s, int numRows);



# **Reverse Integer**

Given a signed 32-bit integer x, return x *with its digits reversed*. If reversing x causes the value to go outside the signed 32-bit integer range [-231, 231 - 1], then return 0. Assume the environment does not allow you to store 64-bit integers (signed or unsigned).



# **String to Integer (atoi)**

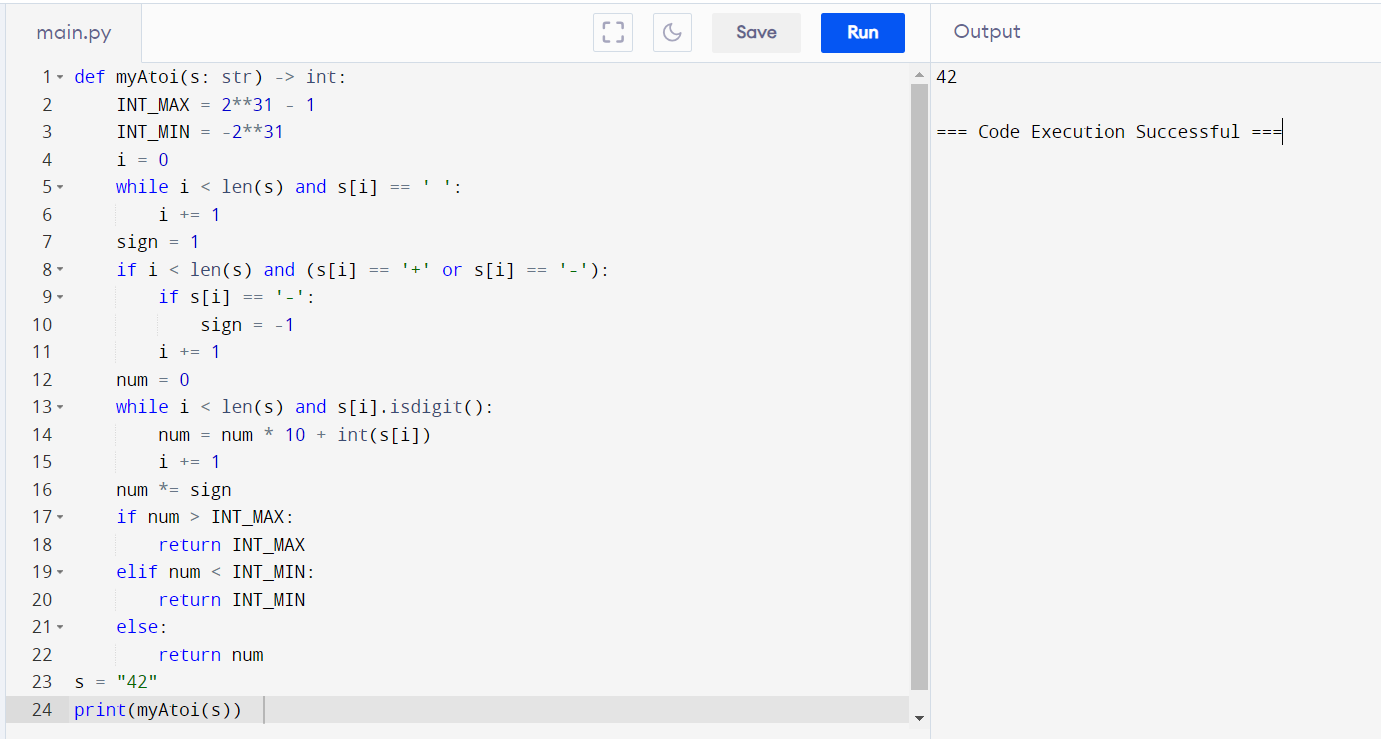
Implement the myAtoi(string s) function, which converts a string to a 32-bit signed integer (similar to C/C++'s atoi function).

The algorithm for myAtoi(string s) is as follows:

1. Read in and ignore any leading whitespace.
2. Check if the next character (if not already at the end of the string) is '-' or '+'. Read this character in if it is either. This determines if the final result is negative or positive respectively. Assume the result is positive if neither is present.
3. Read in next the characters until the next non-digit character or the end of the input is reached. The rest of the string is ignored.
4. Convert these digits into an integer (i.e. "123" -> 123, "0032" -> 32). If no digits were read, then the integer is 0. Change the sign as necessary (from step 2).
5. If the integer is out of the 32-bit signed integer range [-231, 231 - 1], then clamp the integer so that it remains in the range. Specifically, integers less than -231 should be clamped to -231, and integers greater than 231 - 1 should be clamped to 231 - 1.
6. Return the integer as the final result. Note:

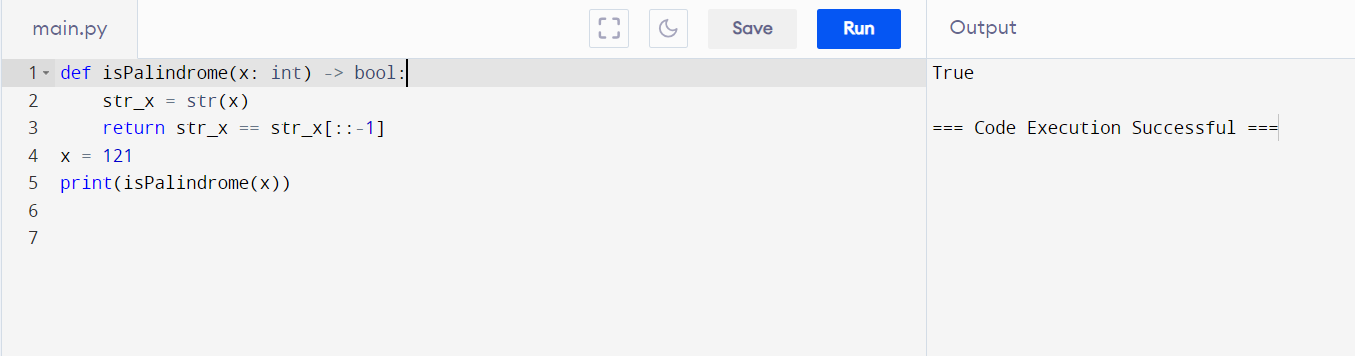
|  |  |  |  |
| --- | --- | --- | --- |
| ● | Only the space character ' | ' is considered a whitespace character. |  |
| ● | Do not ignore any characters other than the leading whitespace or the rest of the | | |

string after the digits.



# **Palindrome Number**

Given an integer x, return true *if* x *is a palindrome, and* false *otherwise*.

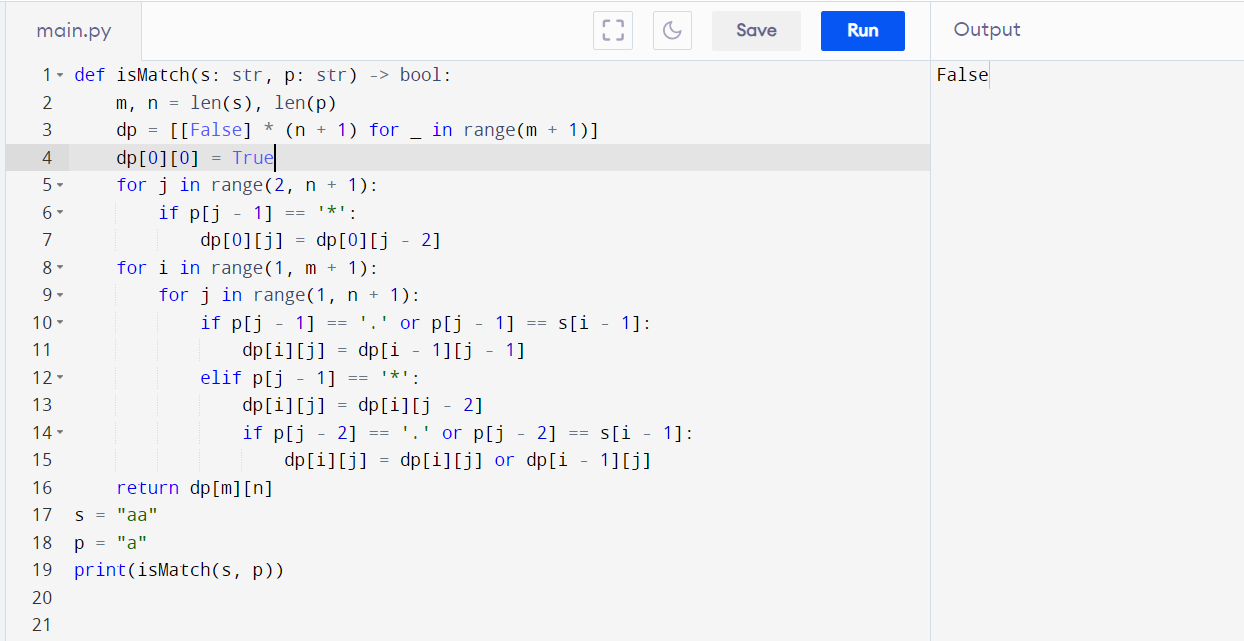


# **Regular Expression Matching**

Given an input string s and a pattern p, implement regular expression matching with support for '.' and '\*' where:

* '.' Matches any single character.
* '\*' Matches zero or more of the preceding element.

The matching should cover the entire input string (not partial).



# **Container With Most Water**

You are given an integer array height of length n. There are n vertical lines drawn such that the two endpoints of the ith line are (i, 0) and (i, height[i]). Find two lines that together with the x-axis form a container, such that the container contains the most water.

Return *the maximum amount of water a container can store*. Notice that you may not slant the container.



# **Integer to Roman**

Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M. Symbol Value

I 1

V 5

X 10

L 50

1. 100
2. 500

M 1000



# **Roman to Integer**

Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M. Symbol Value

I 1

V 5

X 10

L 50

1. 100
2. 500

M 1000

# Longest Common Prefix

|  |
| --- |
| "" |

Write a function to find the longest common prefix string amongst an array of strings. If there is no common prefix, return an empty string .



# **3Sum**

Given an integer array nums, return all the triplets [nums[i], nums[j], nums[k]] such that i != j, i != k, and j != k, and nums[i] + nums[j] + nums[k] == 0.

Notice that the solution set must not contain duplicate triplets.

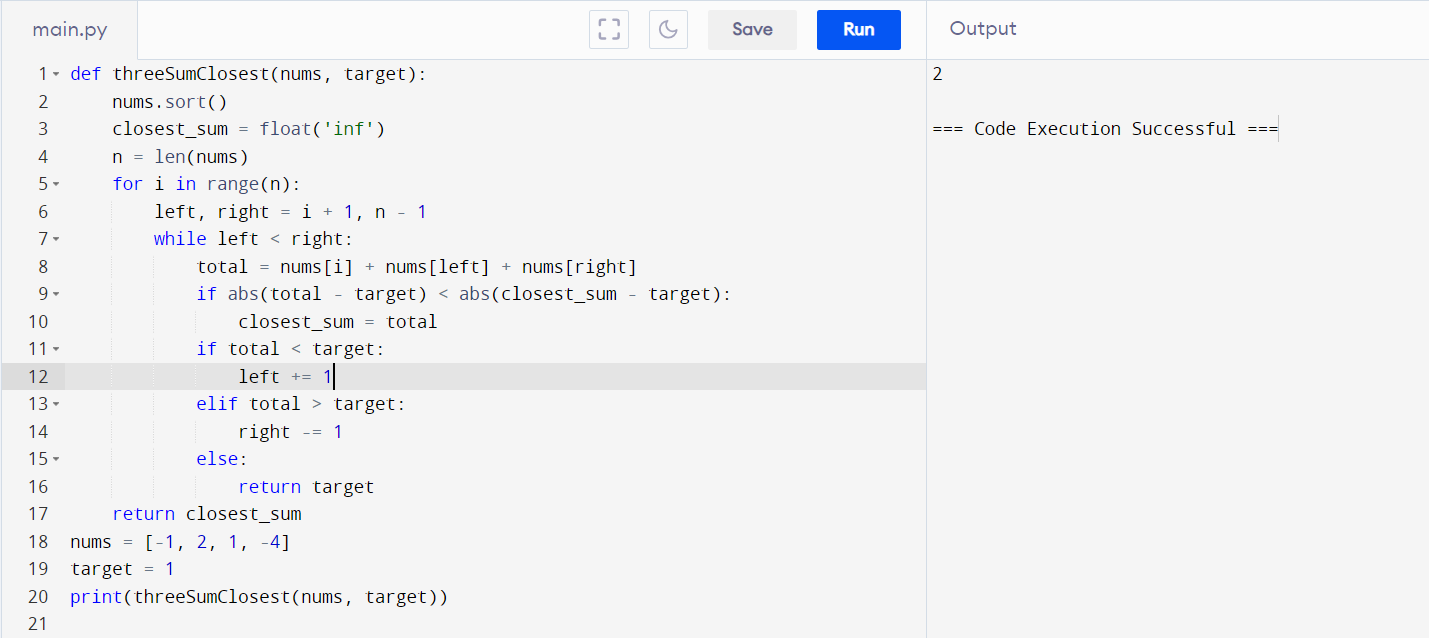


# 3Sum Closest

Given an integer array nums of length n and an integer target, find three integers in nums such that the sum is closest to target.

Return *the sum of the three integers*.

You may assume that each input would have exactly one solution.

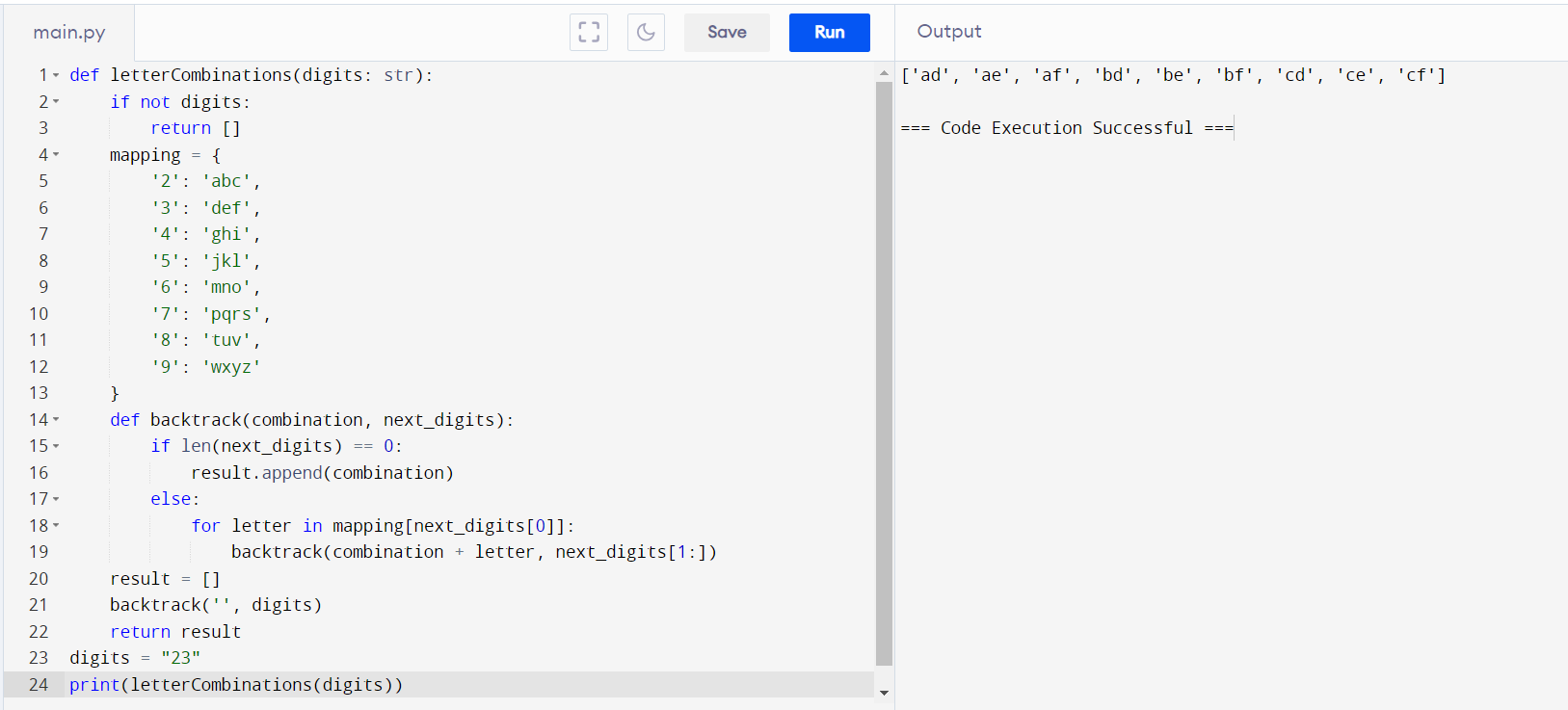


# **Letter Combinations of a Phone Number**

Given a string containing digits from 2-9 inclusive, return all possible letter combinations that the number could represent. Return the answer in any order.

A mapping of digits to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.

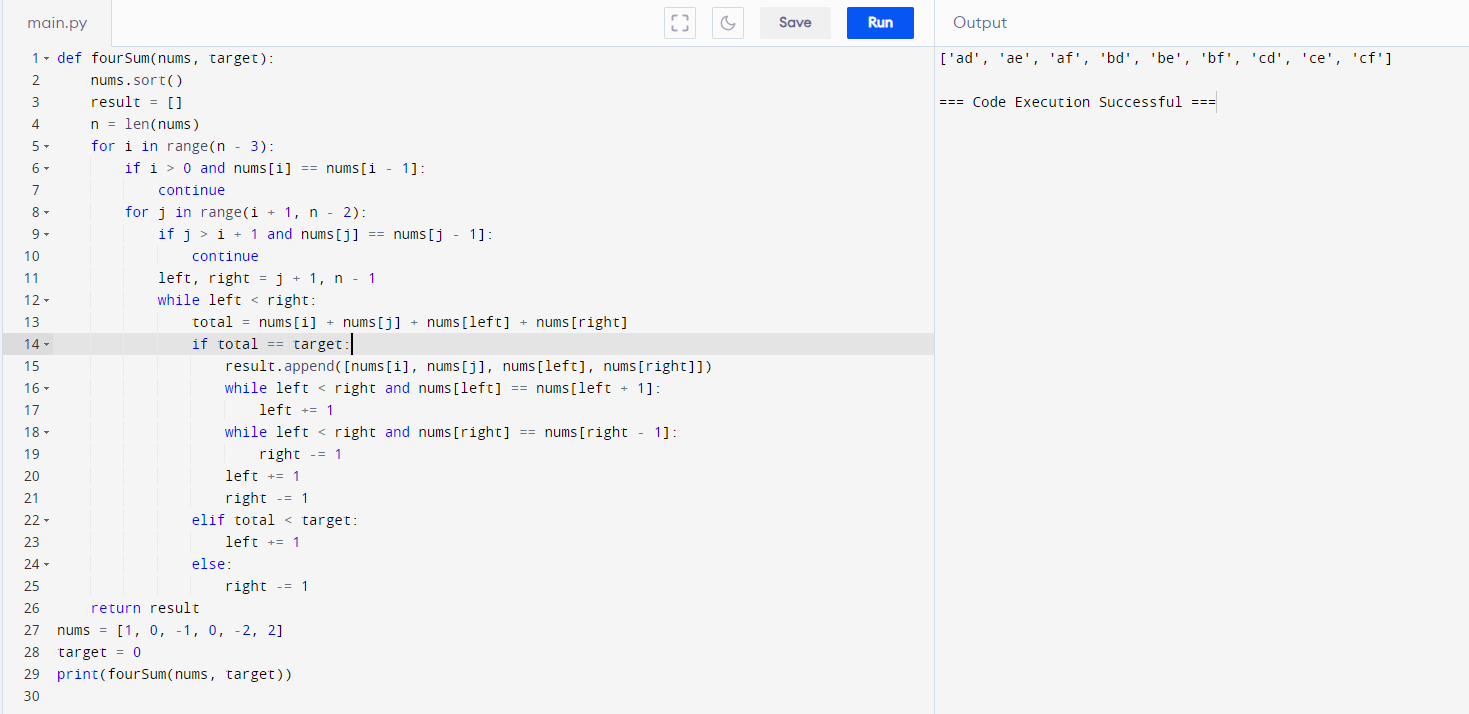




# **4Sum**

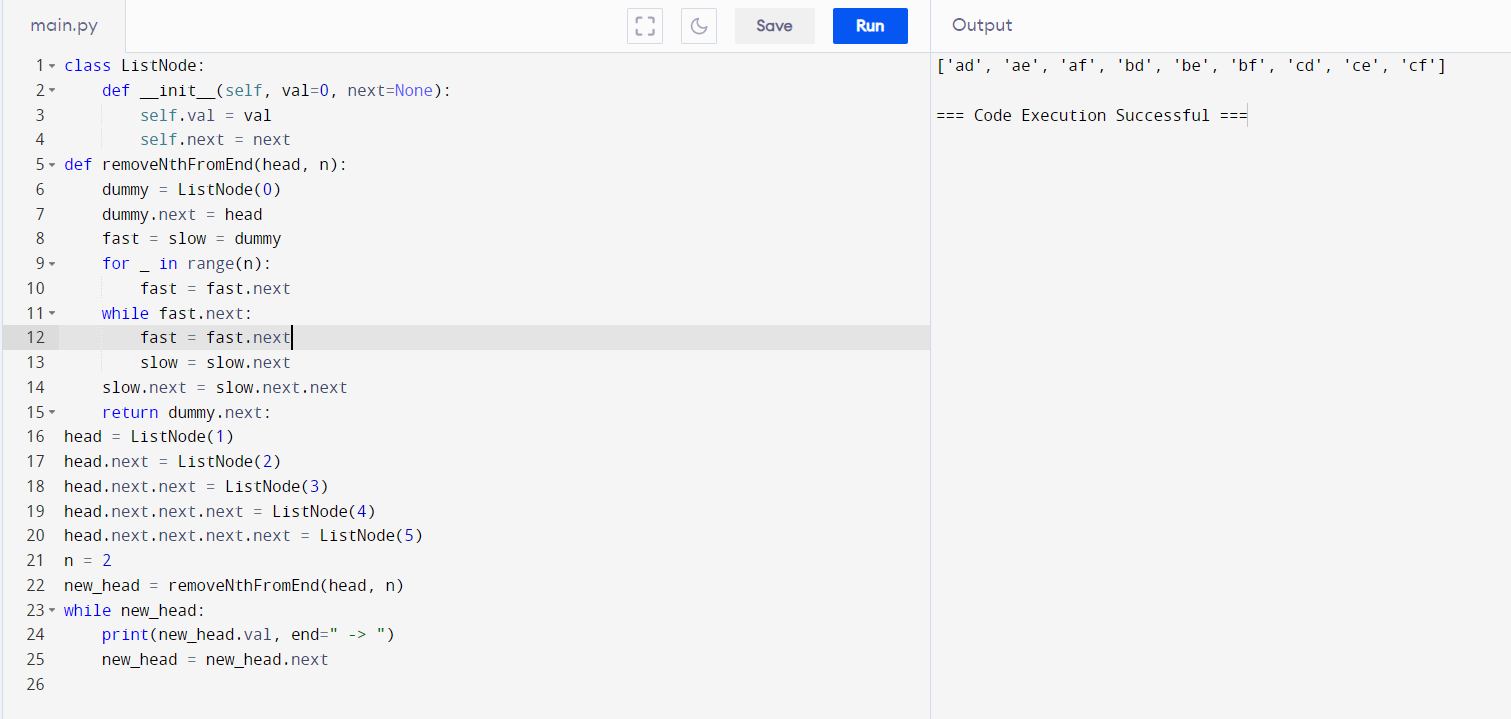
Given an array nums of n integers, return *an array of all the unique quadruplets* [nums[a], nums[b], nums[c], nums[d]] such that: ● 0 <= a, b, c, d < n

* a, b, c, and d are distinct.
* nums[a] + nums[b] + nums[c] + nums[d] == target You may return the answer in any order.



# **Remove Nth Node From End of List**

Given the head of a linked list, remove the nth node from the end of the list and return its head.



.

20.

**Valid Parentheses**

Given a string

s

containing just the characters

'('

,

')'

,

'{'

,

'}'

,

'['

and

']'

, determine if the

input string is valid.

An input string is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.
3. Every close bracket has a corresponding open bracket of the same type.

