

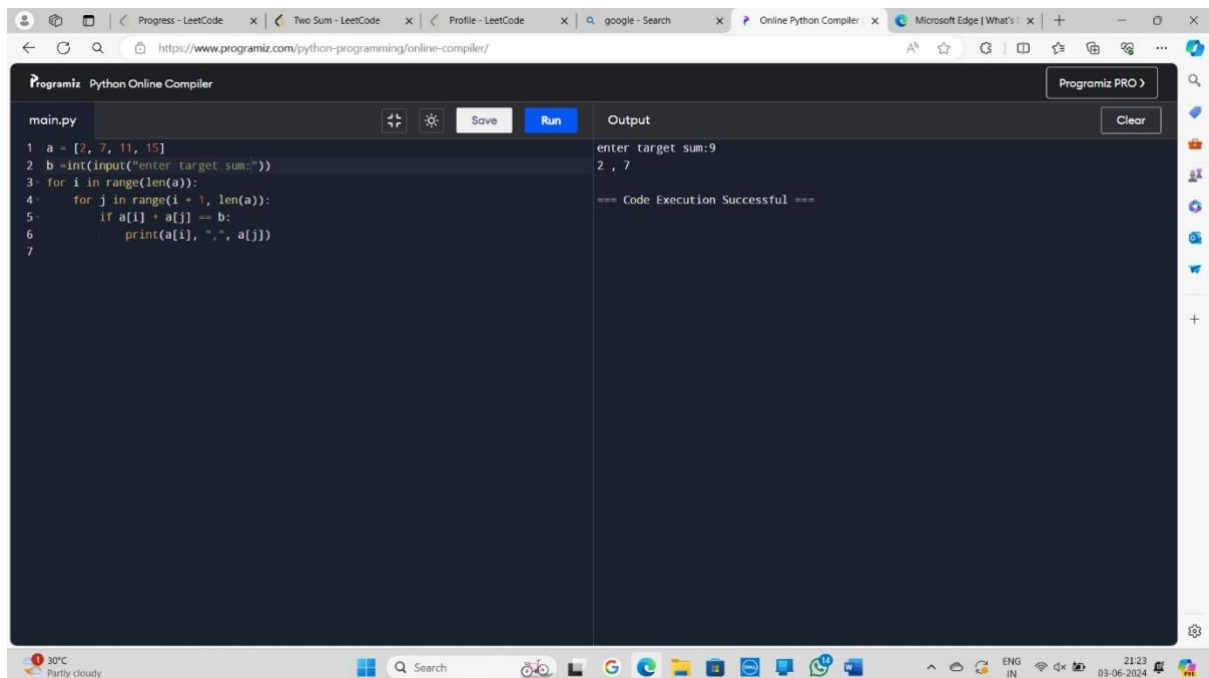
ASSIGNMENT-1

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. Example 1:

Input: `nums = [2,7,11,15]`, `target = 9`

Output: `[0,1]`

Explanation: Because `nums[0] + nums[1] == 9`, we return `[0, 1]`. Example 2: Input: `nums = [3,2,4]`, `target = 6` Output: `[1,2]` Example 3: Input: `nums = [3,3]`, `target = 6` Output: `[0,1]`



The screenshot shows a web browser window with the URL `https://www.programiz.com/python-programming/online-compiler/`. The browser has several tabs open, including 'Progress - LeetCode', 'Two Sum - LeetCode', 'Profile - LeetCode', 'google - Search', 'Online Python Compiler', and 'Microsoft Edge | What's...'. The main content area displays the 'Programiz Python Online Compiler' interface. On the left, a code editor shows a Python script named `main.py` with the following code:

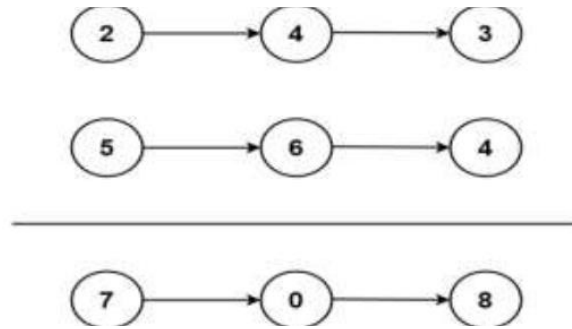
```
1 a = [2, 7, 11, 15]
2 b = int(input("enter target sum:"))
3 for i in range(len(a)):
4     for j in range(i + 1, len(a)):
5         if a[i] + a[j] == b:
6             print(a[i], ",", a[j])
7
```

On the right, the 'Output' panel shows the execution results:

```
enter target sum:9
2 , 7
=== Code Execution Successful ===
```

The bottom of the image shows a Windows taskbar with a search bar, several application icons, and a system tray displaying the date and time as '21:23 03-06-2024'.

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.



Example 1: Input: l1 = [2,4,3], l2 = [5,6,4] Output: [7,0,8] Explanation: 342 + 465 = 807.

```

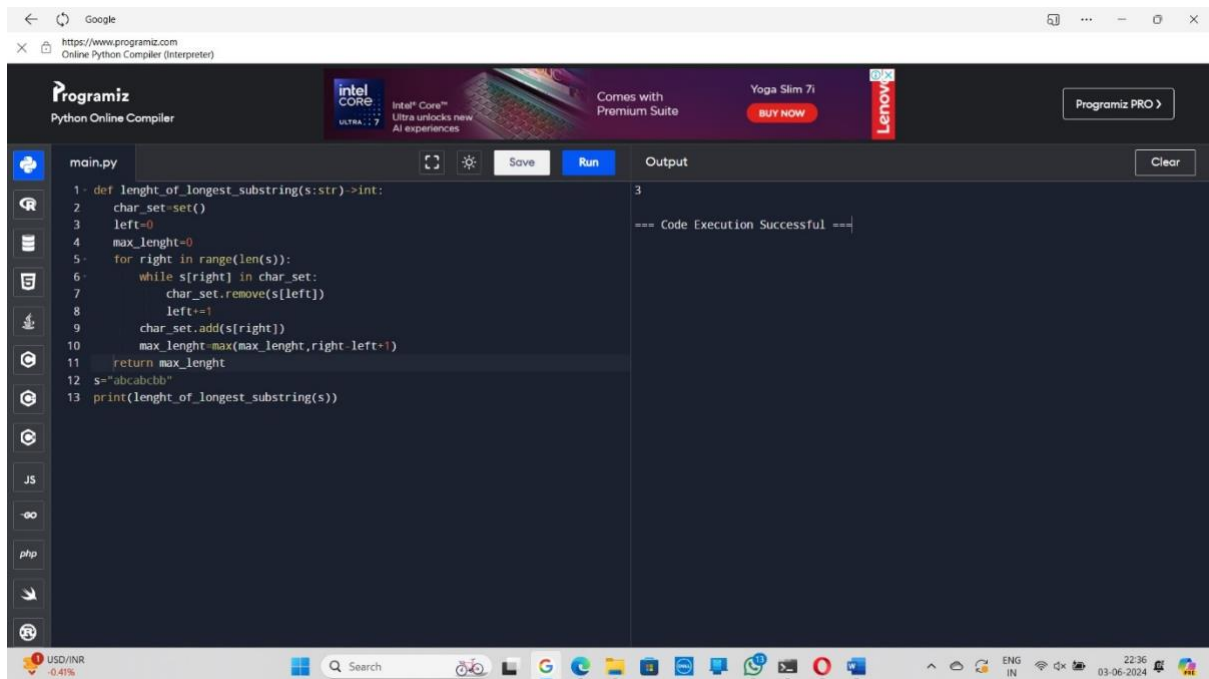
1 l1=[2,4,3]
2 l2=[5,6,4]
3 l3=[]
4 for i in range(len(l1)):
5     s=l1[i]+l2[i]
6     if s>=10:
7         l3.append(int(str(s)[-1]))
8     else:
9         l3.append(s)
10 for j in range(len(l3)):
11     print(l3[j],end=" ")
  
```

Output: 7,0,7,
 === Code Execution Successful ===

3. Longest Substring without Repeating Characters Given a string s, find the length of the longest substring without repeating characters.

Example 1: Input: s = "abcabcbb" Output: 3 Explanation: The answer is "abc", with the length of 3.

Example 2: Input: s = "bbbbbb" Output: 1 Explanation: The answer is "b", with the length of 1



The screenshot shows the Programiz Python Online Compiler interface. The code in the editor is as follows:

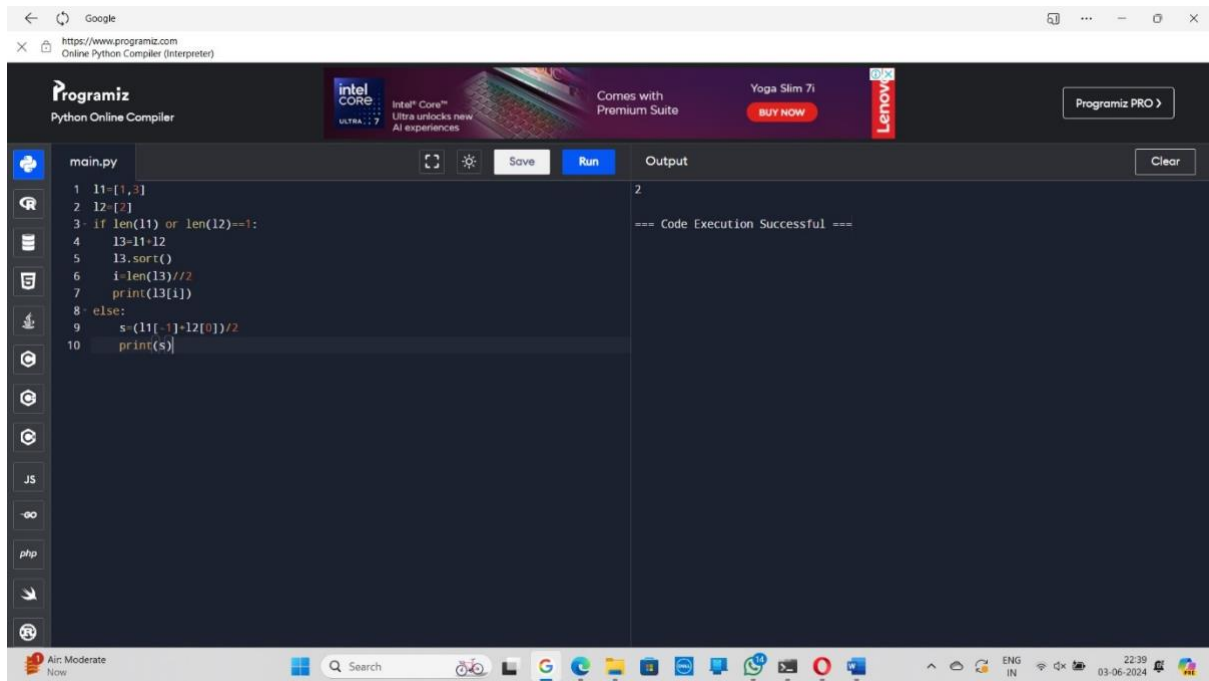
```
1 def length_of_longest_substring(s:str)->int:
2     char_set=set()
3     left=0
4     max_length=0
5     for right in range(len(s)):
6         while s[right] in char_set:
7             char_set.remove(s[left])
8             left+=1
9         char_set.add(s[right])
10        max_length=max(max_length,right-left+1)
11    return max_length
12 s="abcbcb"
13 print(length_of_longest_substring(s))
```

The output on the right shows "3" and "=== Code Execution Successful ===".

4. 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. The overall run time complexity should be $O(\log(m+n))$.

Example 1: Input: nums1 = [1,3], nums2 = [2] Output: 2.00000 Explanation: merged array = [1,2,3] and median is 2.

Example 2: Input: nums1 = [1,2], nums2 = [3,4] Output: 2.50000 Explanation: merged array = [1,2,3,4] and median is $(2 + 3) / 2 = 2.5$



5. Longest Palindromic Substring Given a string s, return the longest palindromic substring in s.

Example 1: Input: s = "babad" Output: "bab" Explanation: "aba" is also a valid answer.

Example 2: Input: s = "cbbd" Output: "bb"

```
1 def longest_palindrome(s:str)->str:
2     if len(s)==0:
3         return ""
4     start,end=0,0
5     for i in range(len(s)):
6         len1=expand_around_center(s,i,i)
7         len2=expand_around_center(s,i,i+1)
8         max_len=max(len1,len2)
9         if max_len>end:
10             start=i-(max_len-1)//2
11             end=i+max_len//2
12     return s[start:end+1]
13 def expand_around_center(s,left,right):
14     while left>=0 and right<len(s) and s[left]==s[right]:
15         left-=1
16         right+=1
17     return right-left-1
18 s="bab"
19 print(longest_palindrome(s))
```

b

=== Code Execution Successful ===

6. 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);

Example 1: Input: s = "PAYPALISHIRING", numRows = 3 Output: "PAHNAPLSIIGYIR"

Example 2: Input: s = "PAYPALISHIRING", numRows = 4 Output: "PINALSIGYAHRPI"

```
1. def convert(s: str, numRows: int) -> str:
2.     if numRows == 1 or numRows >= len(s):
3.         return s
4.     rows = [''] * numRows
5.     cur_row = 0
6.     going_down = False
7.     for char in s:
8.         rows[cur_row] += char
9.         if cur_row == 0 or cur_row == numRows - 1:
10.            going_down = not going_down
11.            cur_row += 1 if going_down else -1
12.     return ''.join(rows)
13. s = "PAYPALISHIRING"
14. numRows = 3
15. print(convert(s, numRows))
```

PAHN APLSIIG YIR

=== Code Execution Successful ===

7. 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 $[-2^{31}, 2^{31} - 1]$, then return 0. Assume the environment does not allow you to store 64-bit integers (signed or unsigned).

Example 1: Input: $x = 123$ Output: 321

Example 2: Input: $x = -123$ Output: -321

```
1. def convert(s: str, numRows: int) -> str:
2.     if numRows == 1 or numRows >= len(s):
3.         return s
4.     rows = [''] * numRows
5.     cur_row = 0
6.     going_down = False
7.     for char in s:
8.         rows[cur_row] += char
9.         if cur_row == 0 or cur_row == numRows - 1:
10.            going_down = not going_down
11.            cur_row += 1 if going_down else -1
12.     return ''.join(rows)
13. s = "PAYPALISHIRING"
14. numRows = 3
15. print(convert(s, numRows))
```

PAHN APLSIIG YIR

=== Code Execution Successful ===

8. 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 $[-2^{31}, 2^{31} - 1]$, then clamp the integer so that it remains in the range. Specifically, integers less than -2^{31} should be clamped to -2^{31} , and integers greater than $2^{31} - 1$ should be clamped to $2^{31} - 1$.
6. Return the integer as the final result

The screenshot shows the Programiz Online Python Compiler interface. The code in the editor is as follows:

```
1 def myAtoi(s: str) -> int:
2     s = s.strip()
3
4     sign = 1
5     if s and (s[0] == '-' or s[0] == '+'):
6         if s[0] == '-':
7             sign = -1
8         s = s[1:]
9
10    num = 0
11    for char in s:
12        if not char.isdigit():
13            break
14        num = num * 10 + int(char)
15
16    num *= sign
17
18    INT_MAX = 2**31 - 1
19    INT_MIN = -2**31
20    if num > INT_MAX:
21        return INT_MAX
22    elif num < INT_MIN:
23        return INT_MIN
24    else:
25        return num
26
27 s = "-42"
28 print(myAtoi(s))
29
30 s = "4193 with words"
31 print(myAtoi(s))
32
33 s = "words and 487"
34 print(myAtoi(s))
35 s = "-4153 with words"
36 print(myAtoi(s))
37
38 s = "words and 487"
39 print(myAtoi(s))
```

The output on the right shows the results of the function calls:

```
-42
4193
0
4193
0
```

Below the output, it says "Code Execution Successful".

9. Palindrome Number Given an integer x, return true if x is a palindrome, and false otherwise.

Example 1: Input: x = 121 Output: true Explanation: 121 reads as 121 from left to right and from right to left.

Example 2: Input: x = -121 Output: false Explanation: From left to right, it reads -121. From right to left, it becomes 121-. Therefore it is not a palindrome.


```
1 def isPalindrome(x: int) -> bool:
2     return str(x) == str(x)[::-1]
3
4 x = 121
5 print(isPalindrome(x))
6
7 x = -121
8 print(isPalindrome(x))
9
10 x = 10
11 print(isPalindrome(x))
```

True
False
False
=== Code Execution Successful ===

10. 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).

Example 1: Input: *s* = "aa", *p* = "a" Output: false Explanation: "a" does not match the entire string "aa".

Google

https://www.programiz.com
Online Python Compiler (Interpreter)

Programiz

Python Online Compiler

Programiz PRO

main.py

Save

Run

Output

Clear

```
1 def isMatch(s: str, p: str) -> bool:
2     dp = [[False] * (len(p) + 1) for _ in range(len(s) + 1)]
3     dp[0][0] = True
4
5     for j in range(1, len(p) + 1):
6         if p[j - 1] == '*':
7             dp[0][j] = dp[0][j - 2]
8
9     for i in range(1, len(s) + 1):
10        for j in range(1, len(p) + 1):
11            if p[j - 1] == '.' or p[j - 1] == s[i - 1]:
12                dp[i][j] = dp[i - 1][j - 1]
13            elif p[j - 1] == '*':
14                dp[i][j] = dp[i][j - 2] or (dp[i - 1][j] and (s[i - 1] == p[j - 2] or p[j - 2] == '.'))
15
16    return dp[-1][-1]
17
18 s = "aa"
19 p = "a*"
20 print(isMatch(s, p))
21
22 s = "mississippi"
23 p = "mis*is*p*."
24 print(isMatch(s, p))
```

```
True
False

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

USD/INR
-0.41%

Search

ENG
IN

23:04
03-06-2024