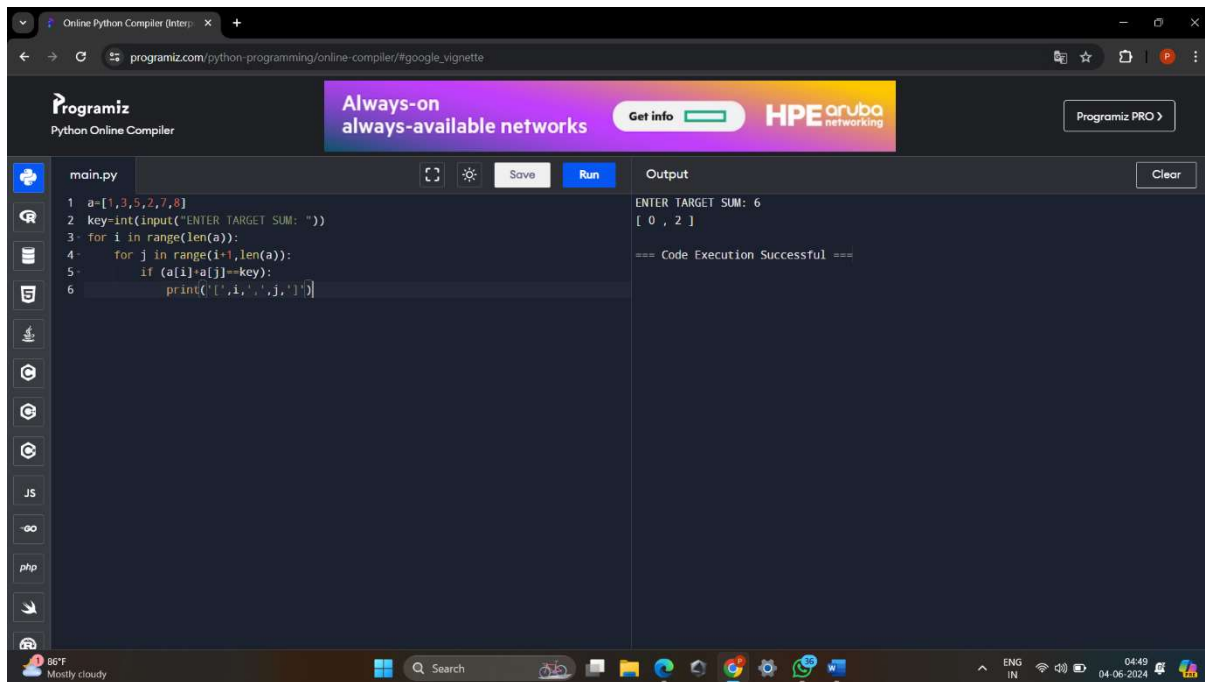


1. Two Sum Given an array of integers `nums` and an integer `target`, return indices of the two numbers such that they add up to `target`. You may assume that each input would have exactly one solution, and you may not use the same element twice. You can return the answer in any order. Example 1:

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]`

A screenshot of the Programiz Online Python Compiler interface. The browser address bar shows 'programiz.com/python-programming/online-compiler/#google_vignette'. The page has a dark theme. At the top, there are banners for 'Programiz Python Online Compiler', 'Always-on always-available networks', and 'HPE aruba networking'. Below the banners, there's a 'main.py' file editor with the following Python code:

```
1 a=[1,3,5,2,7,8]
2 key=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]==key:
6             print(i,j)
```

The 'Run' button is highlighted in blue. To the right of the code editor is the 'Output' panel, which displays the execution results:

```
ENTER TARGET SUM: 6
[ 0 , 2 ]

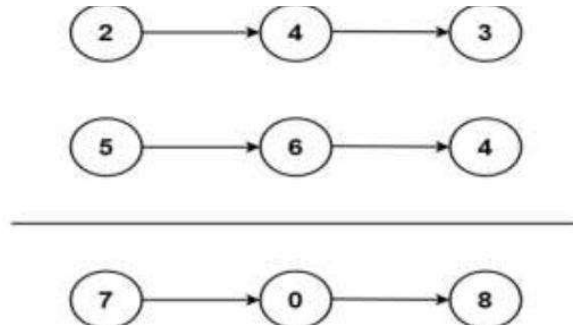
=== Code Execution Successful ===
```

At the bottom of the screen, there's a Windows taskbar showing the date and time as '04-06-2024' and '04:49', along with system icons for network, volume, and battery.

Time : $O(n)$

Space: $O(n)$

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: $342 + 465 = 807$.

The screenshot shows a web-based Python IDE with the following code in `main.py`:

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

The output window displays:

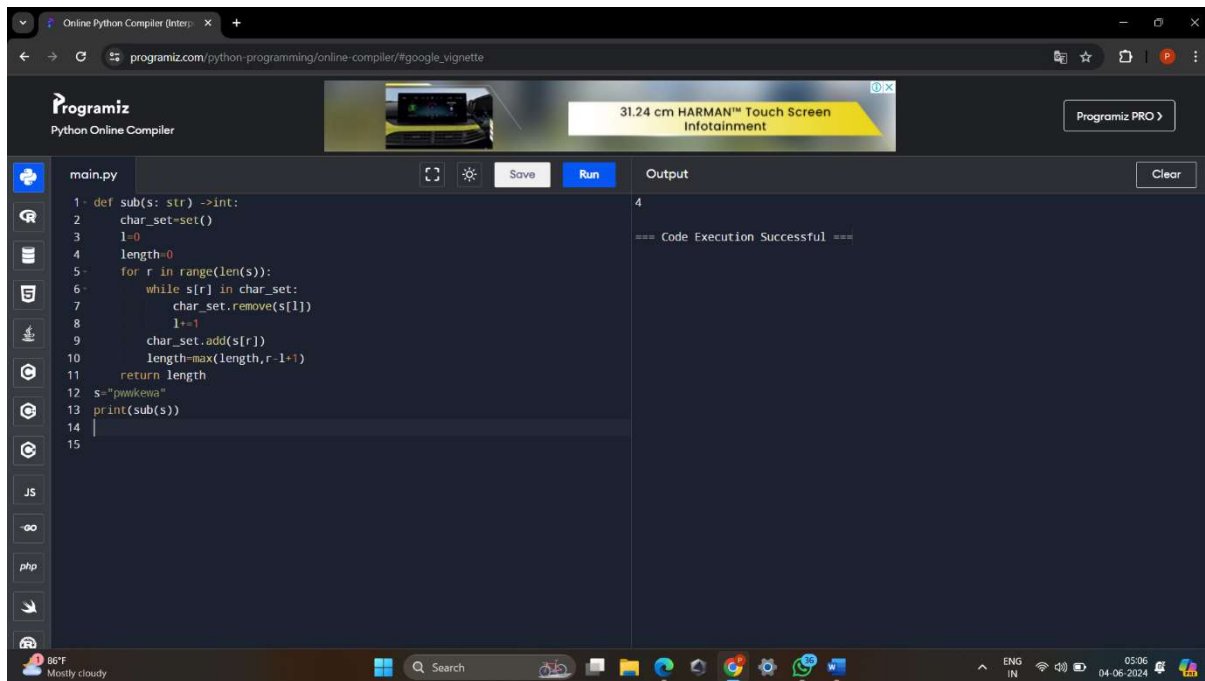
```
[7, 8, 9]
=== Code Execution Successful ===
```

Time : $O((m,n))$

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

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

Example 2: Input: $s = \text{"bbbbb"}$ Output: 1 Explanation: The answer is "b", with the length of 1



The screenshot shows a web browser window with the URL `programiz.com/python-programming/online-compiler/#google_vignette`. The page features the Programiz logo and a banner for a 31.24 cm HARMAN™ Touch Screen Infotainment. Below the banner is a code editor with a file named `main.py`. The code implements a sliding window algorithm to find the longest substring without repeating characters. The output window shows the result `4` and the message `=== Code Execution Successful ===`. The code is as follows:

```
1 def sub(s: str) ->int:
2     char_set=set()
3     l=0
4     length=0
5     for r in range(len(s)):
6         while s[r] in char_set:
7             char_set.remove(s[l])
8             l+=1
9         char_set.add(s[r])
10        length=max(length,r-l+1)
11    return length
12 s="pnmkoon"
13 print(sub(s))
14
15
```

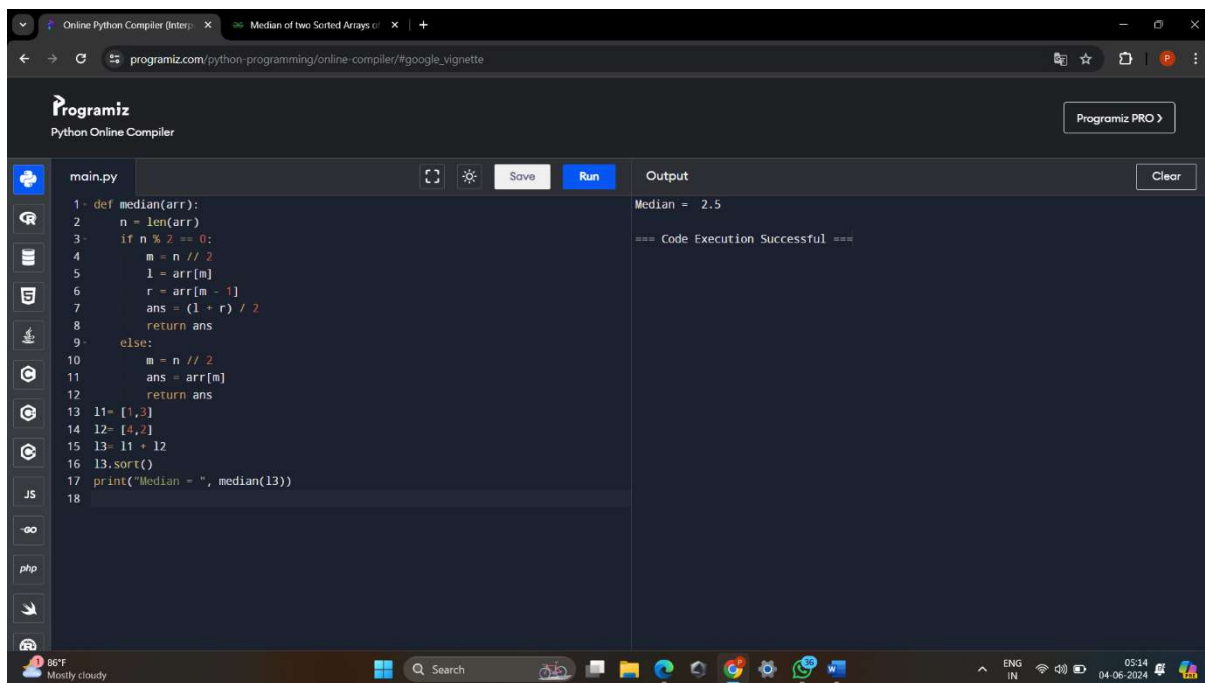
Time: $O(n)$

Space: $O(n)$

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 $(2 + 3) / 2 = 2.5$



The screenshot shows a web browser window with the Programiz Python Online Compiler. The code in the editor is as follows:

```
1- def median(arr):
2-     n = len(arr)
3-     if n % 2 == 0:
4-         m = n // 2
5-         l = arr[m]
6-         r = arr[m - 1]
7-         ans = (l + r) / 2
8-         return ans
9-     else:
10-        m = n // 2
11-        ans = arr[m]
12-        return ans
13- l1= [1,3]
14- l2= [4,2]
15- l3= l1 + l2
16- l3.sort()
17- print("Median = ", median(l3))
18-
```

The output of the code is:

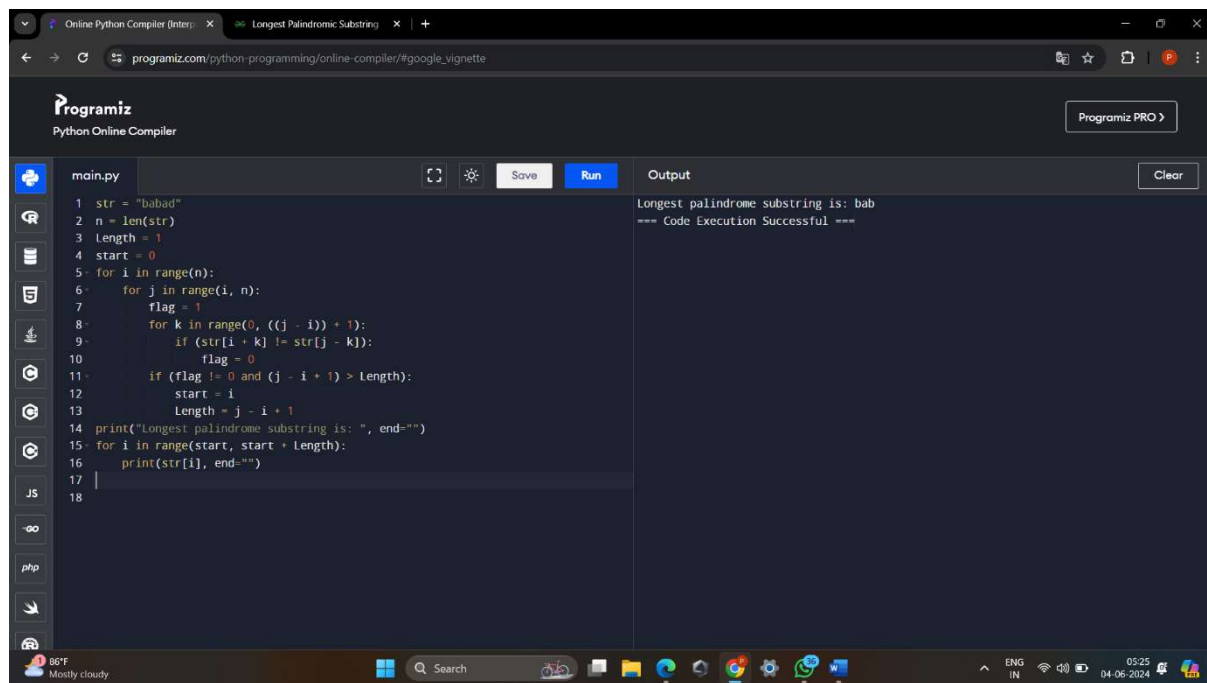
```
Median = 2.5
=== Code Execution Successful ===
```

The browser's address bar shows the URL: programiz.com/python-programming/online-compiler/#google_vignette. The Programiz logo and "Python Online Compiler" text are visible at the top. The bottom of the screen shows a Windows taskbar with the date 04-06-2024 and time 05:14.

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* = "cbabd" Output: "bb"



The screenshot shows a web browser window with the URL `programiz.com/python-programming/online-compiler/#google_vignette`. The page is titled "Programiz Python Online Compiler". The code editor contains a Python script for finding the longest palindromic substring. The output window shows the result: "Longest palindrome substring is: bab" and "=== Code Execution Successful ===".

```
1 str = "babad"
2 n = len(str)
3 Length = 1
4 start = 0
5 for i in range(n):
6     for j in range(i, n):
7         flag = 1
8         for k in range(0, ((j - i)) + 1):
9             if (str[i + k] != str[j - k]):
10                flag = 0
11            if (flag != 0 and (j - i + 1) > Length):
12                start = i
13                Length = j - i + 1
14 print("Longest palindrome substring is: ", end="")
15 for i in range(start, start + Length):
16     print(str[i], end="")
17
18
```

Output: Longest palindrome substring is: bab
=== Code Execution Successful ===

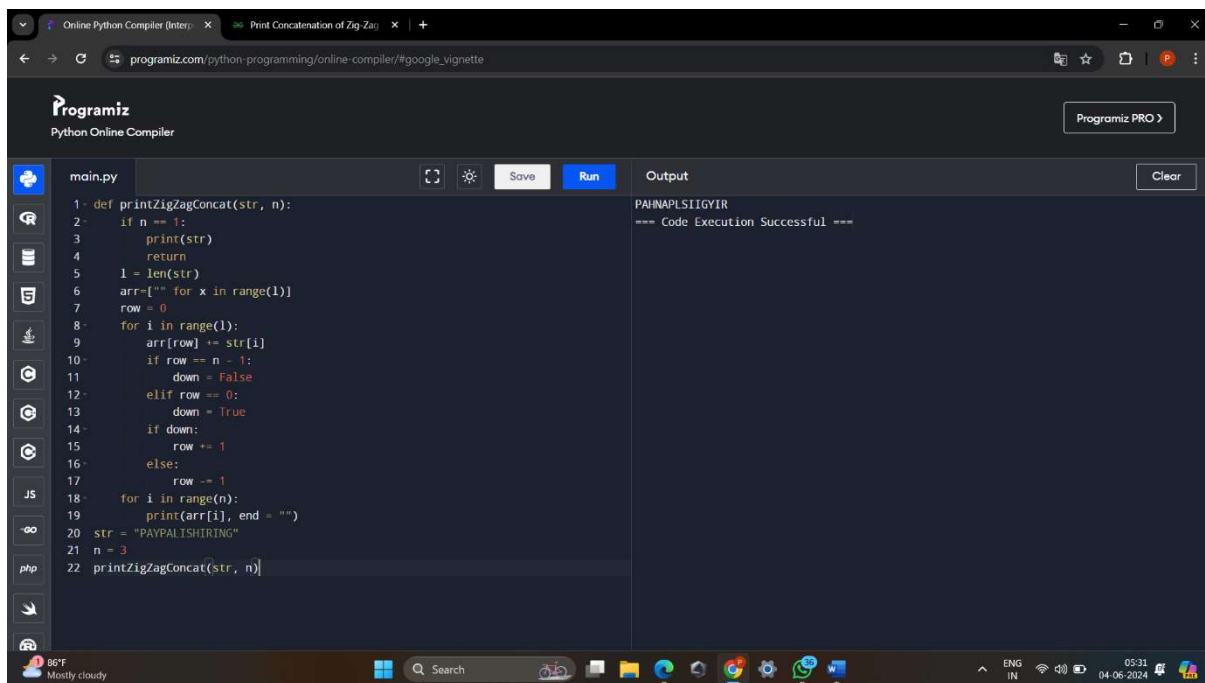
Time: $O(n^2)$

Space: $O(1)$

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"



The screenshot shows a web browser window with the URL `programiz.com/python-programming/online-compiler/#google_vignette`. The page title is "Programiz Python Online Compiler". The code editor shows a file named `main.py` with the following Python code:

```
1 def printZigZagConcat(str, n):
2     if n == 1:
3         print(str)
4         return
5     l = len(str)
6     arr = [''] * l
7     row = 0
8     for i in range(l):
9         arr[row] += str[i]
10        if row == n - 1:
11            down = False
12        elif row == 0:
13            down = True
14        if down:
15            row += 1
16        else:
17            row -= 1
18        for i in range(n):
19            print(arr[i], end = '')
20    str = "PAYPALISHIRING"
21    n = 3
22    printZigZagConcat(str, n)
```

The output window shows the result of the code execution:

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

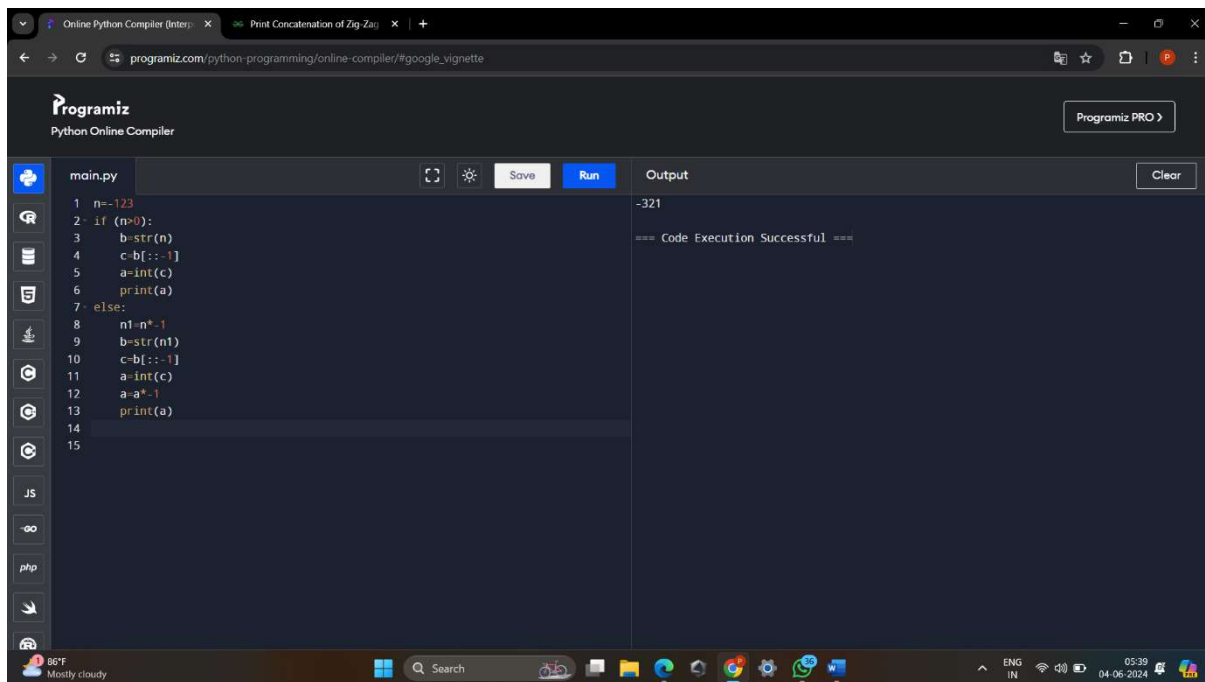
Time: $O(n)$

Space: $O(n)$

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



The screenshot shows a web browser window with the URL `programiz.com/python-programming/online-compiler/#google_vignette`. The page is titled "Programiz Python Online Compiler". The code editor contains a Python script in `main.py` that implements the reverse integer function. The script uses a conditional approach: if the input `n` is non-zero, it converts it to a string, reverses it, and converts it back to an integer. If `n` is zero, it simply prints `n`. The output window shows the result `-321` and a message "=== Code Execution Successful ===".

```
1 n=-123
2 if (n!=0):
3     b=str(n)
4     c=b[::-1]
5     a=int(c)
6     print(a)
7 else:
8     n1=n*-1
9     b=str(n1)
10    c=b[::-1]
11    a=int(c)
12    a=a*-1
13    print(a)
14
15
```

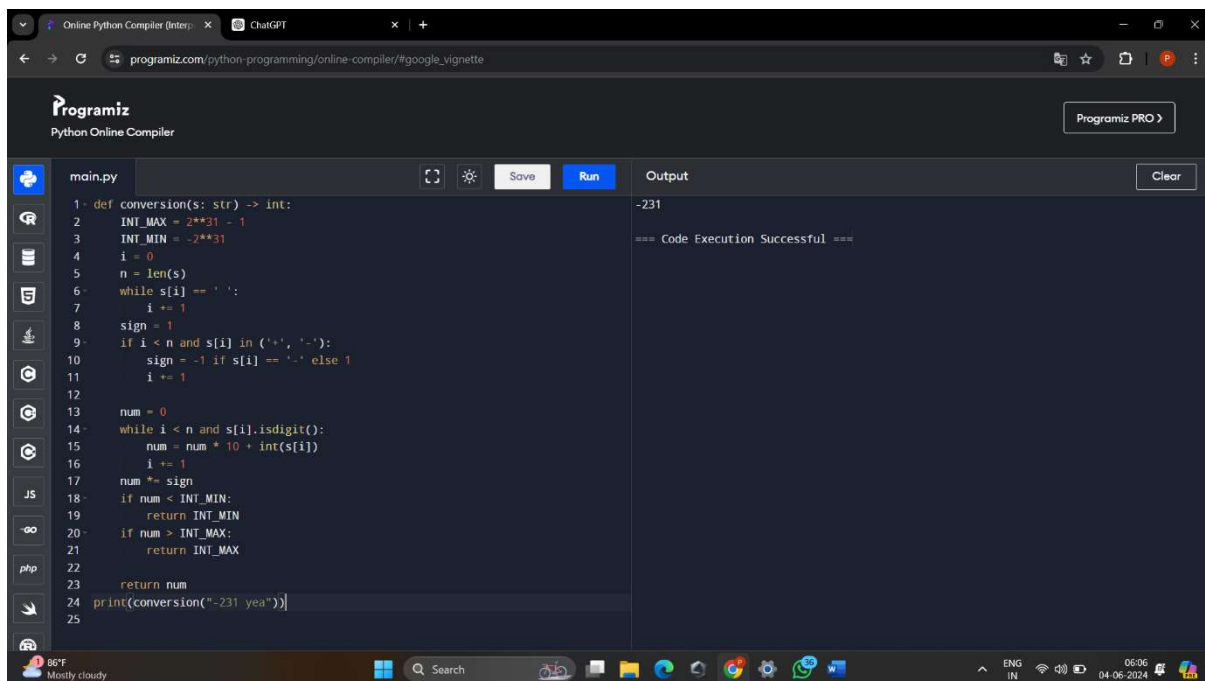
Output: -321
=== Code Execution Successful ===

Time: $O(n)$

Space: $O(n)$

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 a web browser window with the URL `programiz.com/python-programming/online-compiler/#google_vignette`. The page is titled "Programiz Python Online Compiler". The code editor contains the following Python code:

```
1 def conversion(s: str) -> int:
2     INT_MAX = 2**31 - 1
3     INT_MIN = -2**31
4     i = 0
5     n = len(s)
6     while s[i] == ' ':
7         i += 1
8     sign = 1
9     if i < n and s[i] in ('+', '-'):
10        sign = -1 if s[i] == '-' else 1
11        i += 1
12
13    num = 0
14    while i < n and s[i].isdigit():
15        num = num * 10 + int(s[i])
16        i += 1
17    num *= sign
18    if num < INT_MIN:
19        return INT_MIN
20    if num > INT_MAX:
21        return INT_MAX
22    return num
23
24 print(conversion("-231 yea"))
25
```

The output window shows the result: `-231` and the message `=== Code Execution Successful ===`. The bottom status bar shows the temperature as 86°F, the time as 06:06, and the date as 04-06-2024.

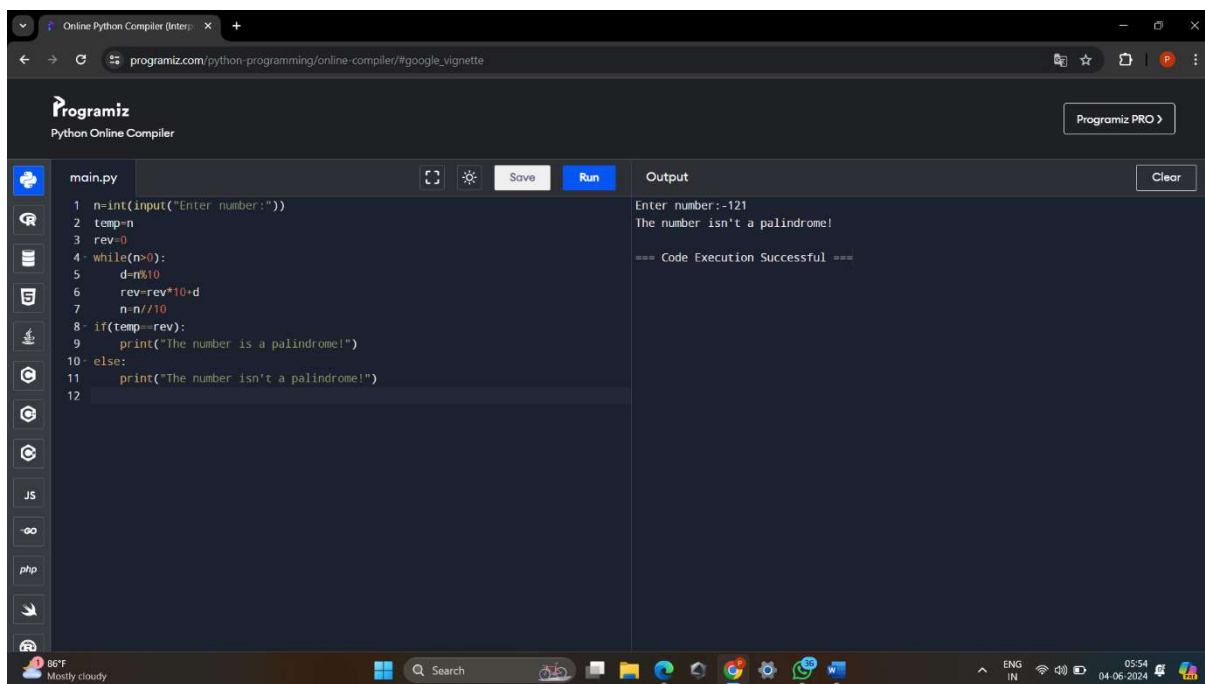
Time : $O(n)$

Space : $O(1)$

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.



The screenshot shows a web browser window with the URL `programiz.com/python-programming/online-compiler/#google_vignette`. The page title is "Programiz Python Online Compiler". The main area is divided into two panels: a code editor on the left and an output panel on the right. The code editor contains a Python script named `main.py` that checks if a number is a palindrome. The script prompts the user to "Enter number:" and receives the input `-121`. The output panel shows the execution results: "Enter number:-121", "The number isn't a palindrome!", and "=== Code Execution Successful ===". The code in the editor is as follows:

```
1 n=int(input("Enter number:"))
2 temp=n
3 rev=0
4 while(n>0):
5     d=n%10
6     rev=rev*10+d
7     n=n//10
8 if(temp==rev):
9     print("The number is a palindrome!")
10 else:
11     print("The number isn't a palindrome!")
12
```

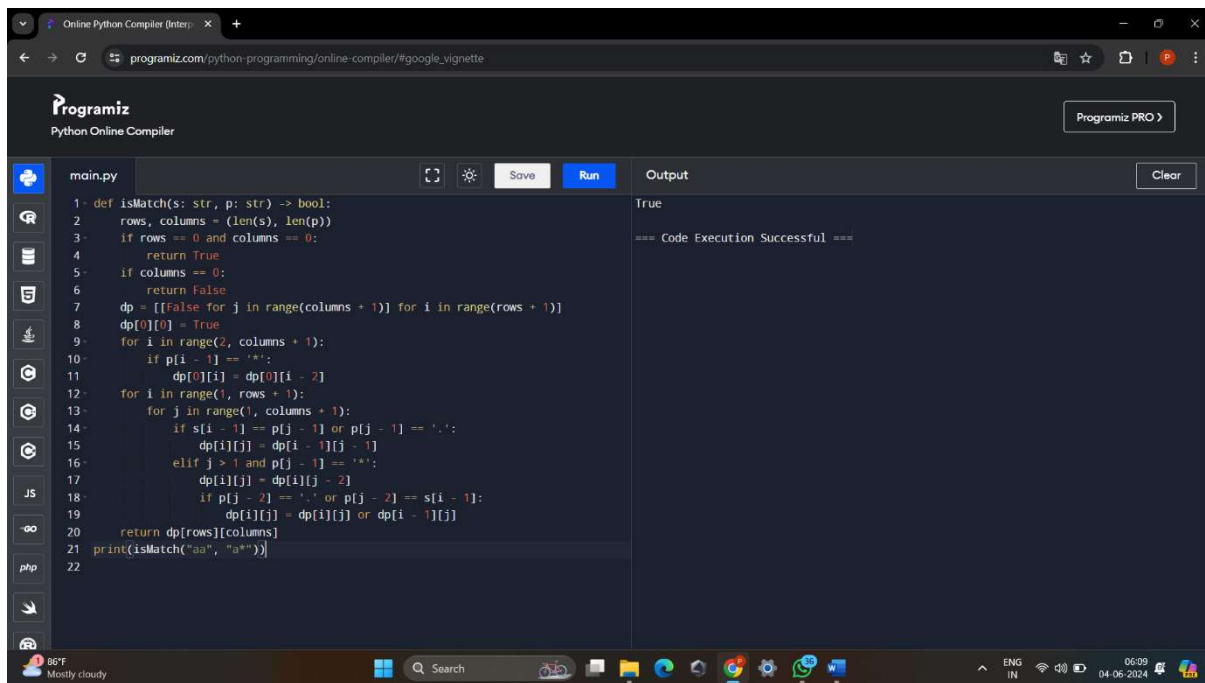
TIME: $O(n)$

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".



The screenshot shows a web browser window with the URL `programiz.com/python-programming/online-compiler/#google_vignette`. The page title is "Programiz Python Online Compiler". The main area displays a Python code editor with the following code:

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

The output panel on the right shows the result: `True` followed by `=== Code Execution Successful ===`. The bottom of the image shows a Windows taskbar with the date and time 04-06-2024 06:39.

Time : $O(m*n)$

Space: $O(m*n)$