



## ASSIGNMENT NO --3

**Student name:** M. Yugandhar Reddy.

**Reg No** : 192372016.

**Subject name:** Python Programming.

**Sub Code** :Csa0898.



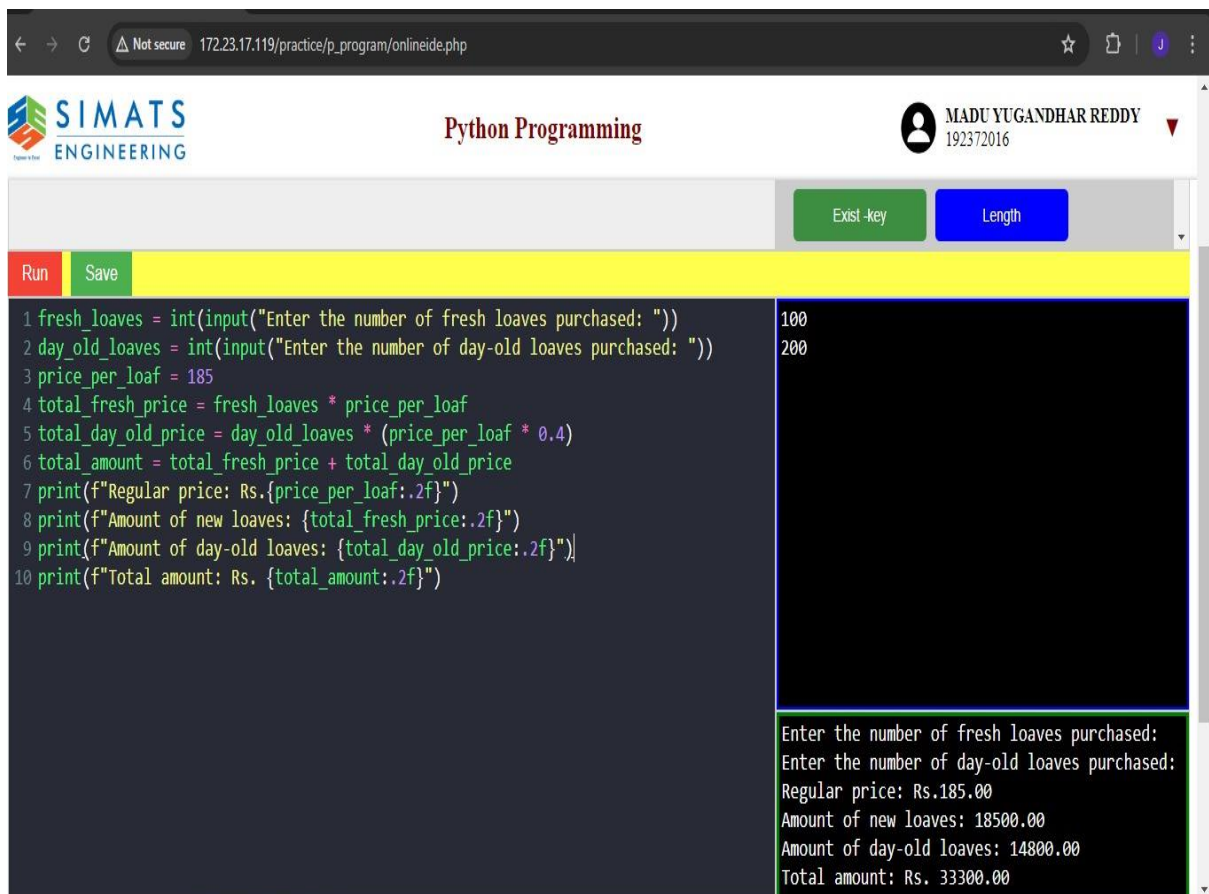
**Under The Guidance:**  
**Dr.Rashmita.**

1. A bakery sells loaves of bread for 185 rupees each. Day old bread is discounted by 60 percent. Write a python program that begins by reading the number of loaves of day old bread being purchased from the user. Then your program should display the regular price for the bread, the discount because it is a day old, and the total price. All of the values should be displayed using two decimal places, and the decimal points in all of the numbers should be aligned when reasonable values are entered by the user.

Sample Input:

Enter the number of fresh loves purchased: 5

Enter the number of day-old loaves purchased: 3



The screenshot shows a web browser window with the URL `172.23.17.119/practice/p_program/onlineide.php`. The page header includes the SIMATS ENGINEERING logo, the text "Python Programming", and a user profile for MADU YUGANDHAR REDDY. Below the header, there are buttons for "Run", "Save", "Exist -key", and "Length". The main area contains a Python program that calculates the total price for fresh and day-old loaves. The program uses `input()` to get the number of loaves and `print()` to display the results with two decimal places. The output shows the regular price, the discount, and the total amount.

```
1 fresh_loaves = int(input("Enter the number of fresh loaves purchased: "))
2 day_old_loaves = int(input("Enter the number of day-old loaves purchased: "))
3 price_per_loaf = 185
4 total_fresh_price = fresh_loaves * price_per_loaf
5 total_day_old_price = day_old_loaves * (price_per_loaf * 0.4)
6 total_amount = total_fresh_price + total_day_old_price
7 print(f"Regular price: Rs.{price_per_loaf:.2f}")
8 print(f"Amount of new loaves: {total_fresh_price:.2f}")
9 print(f"Amount of day-old loaves: {total_day_old_price:.2f}")
10 print(f"Total amount: Rs. {total_amount:.2f}")
```

Enter the number of fresh loaves purchased:  
Enter the number of day-old loaves purchased:  
Regular price: Rs.185.00  
Amount of new loaves: 18500.00  
Amount of day-old loaves: 14800.00  
Total amount: Rs. 33300.00

2. Given two strings “s” and “t”, determine if they are isomorphic. Two strings “s” and “t” are isomorphic if the characters in “s” can be replaced to get “t”. All occurrences of a character must be replaced with another character while preserving the order of characters.

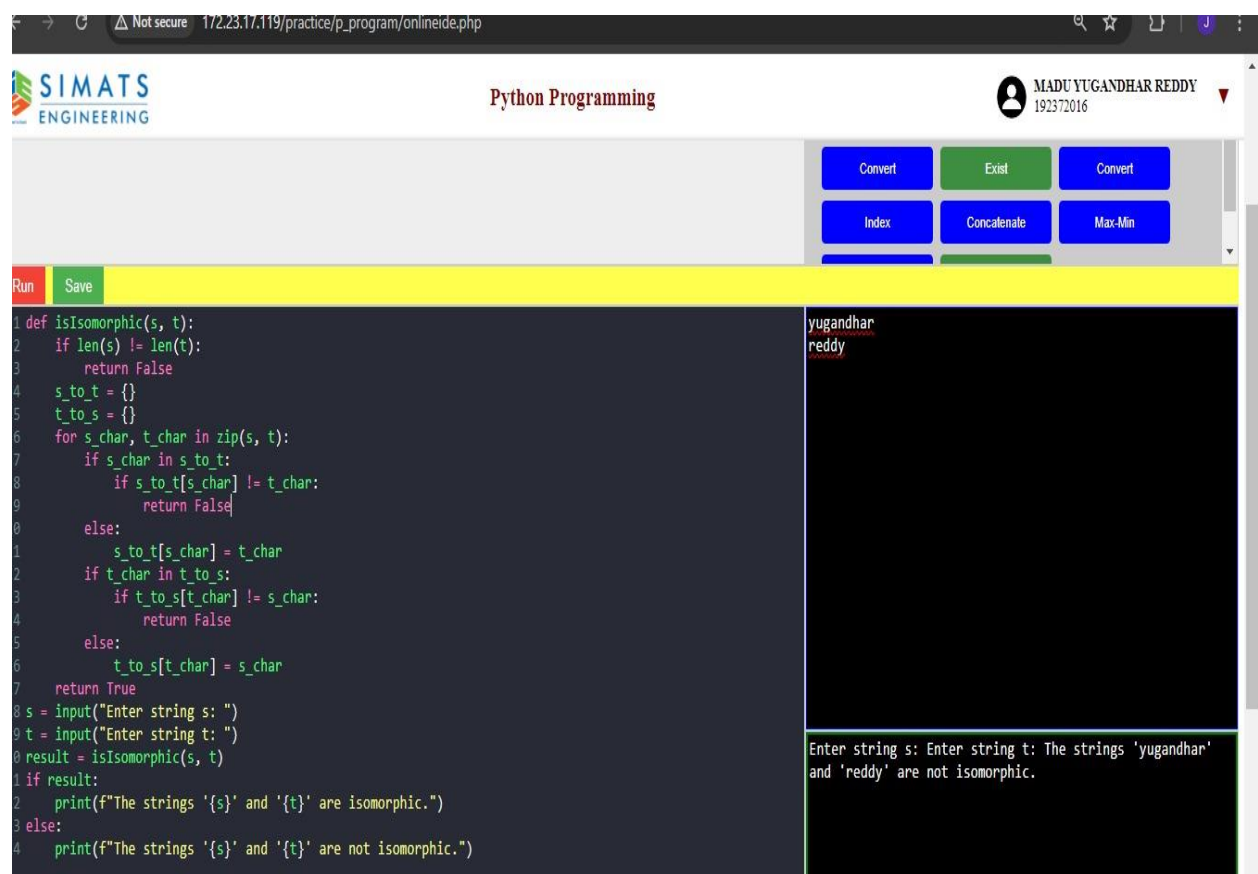
No two characters may map to the same character, but a character may map to itself.

Constraints:

✓ s and t consist of any valid ascii character.

Test Cases:

1.Input: s = "egg", t = "add" Output: true.



The screenshot shows a web browser window with the URL `1/2.23.17.119/practice/p_program/onlineide.php`. The page header includes the SIMATS ENGINEERING logo, the text "Python Programming", and a user profile for "MADU YUGANDHAR REDDY" with ID "192372016". Below the header is a toolbar with buttons for "Convert", "Exist", "Convert", "Index", "Concatenate", and "Max-Min". The main area is a code editor with a dark background. The code defines a function `isIsomorphic(s, t)` that checks if two strings are isomorphic. It uses two dictionaries, `s_to_t` and `t_to_s`, to map characters from one string to the other. The function returns `True` if the strings are isomorphic and `False` otherwise. The code is executed, and the output is displayed in a terminal window on the right. The output shows the input strings "yugandhar" and "reddy" and the result "The strings 'yugandhar' and 'reddy' are not isomorphic."

```
1 def isIsomorphic(s, t):
2     if len(s) != len(t):
3         return False
4     s_to_t = {}
5     t_to_s = {}
6     for s_char, t_char in zip(s, t):
7         if s_char in s_to_t:
8             if s_to_t[s_char] != t_char:
9                 return False
10        else:
11            s_to_t[s_char] = t_char
12        if t_char in t_to_s:
13            if t_to_s[t_char] != s_char:
14                return False
15        else:
16            t_to_s[t_char] = s_char
17    return True
18 s = input("Enter string s: ")
19 t = input("Enter string t: ")
20 result = isIsomorphic(s, t)
21 if result:
22     print(f"The strings '{s}' and '{t}' are isomorphic.")
23 else:
24     print(f"The strings '{s}' and '{t}' are not isomorphic.")
```

Output:

```
Enter string s: Enter string t: The strings 'yugandhar'
and 'reddy' are not isomorphic.
```

3. Given  $n$  non-negative integers  $a_1, a_2, a_3, \dots, a_n$  where each represents a point at coordinate

$(i, a_i)$ . 'n' vertical lines are drawn such that the two endpoints of line  $i$  is at  $(i, a_i)$  and

$(i, 0)$ . Find two lines, which together with x-axis forms a container, such that the container contains the most water. The program should return an integer which corresponds to the maximum area of water that can be contained (maximum area instead of maximum volume

sounds weird but this is the 2D plane we are working with for simplicity).

Note:

You may not slant the container.

Test case:

1. Input: array = [1, 5, 4, 3] Output: 6

The screenshot shows a web-based Python IDE. At the top, there's a header with 'SIMATS ENGINEERING' on the left, 'Python Programming' in the center, and a user profile 'MADU YUGANDHAR REDDY' with ID '192372016' on the right. Below the header is a toolbar with buttons: 'Convert', 'Exit', 'Convert', 'Index', 'Concatenate', and 'Max-Min'. A yellow status bar indicates 'Run', 'Save', and 'Changes Updated, Saved 141'. The main area is split into two panes. The left pane contains Python code for a function 'maxArea' that uses a two-pointer approach to find the maximum area between two lines. The right pane shows the input '1 8 6 2 5 4 8 3 7' and the output 'Enter the heights separated by space: The maximum area between any two lines is: 49'.

```
1 def maxArea(height):
2     n = len(height)
3     max_area = 0
4     left = 0
5     right = n - 1
6     while left < right:
7         current_area = min(height[left], height[right]) * (right - left)
8         max_area = max(max_area, current_area)
9         if height[left] < height[right]:
10             left += 1
11         else:
12             right -= 1
13     return max_area
14 height = list(map(int, input("Enter the heights separated by space: ").split()))
15 result = maxArea(height)
16 print(f"The maximum area between any two lines is: {result}")
```

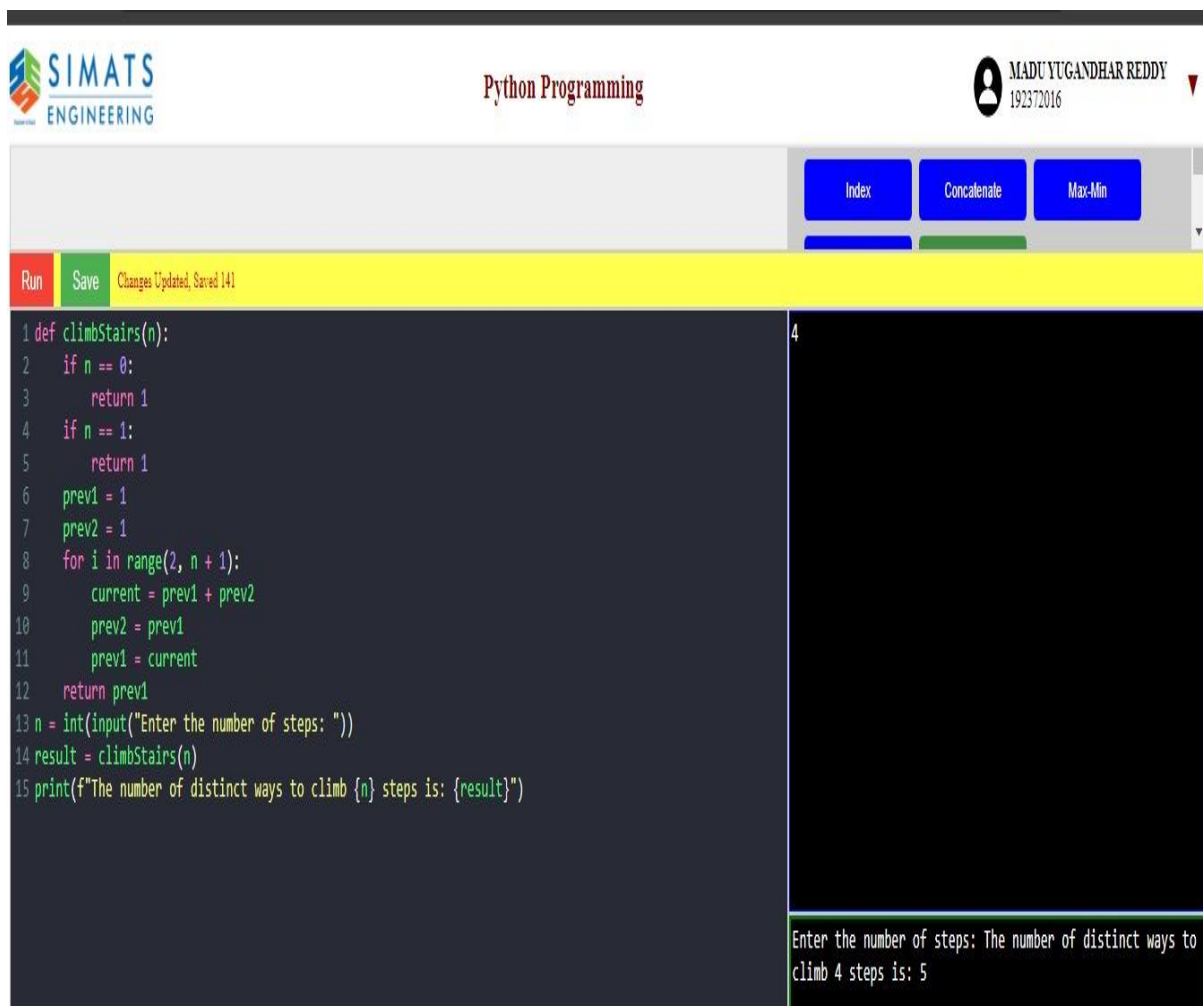
1 8 6 2 5 4 8 3 7

Enter the heights separated by space: The maximum area between any two lines is: 49

4. You are climbing a staircase. It takes  $n$  steps to reach the top. Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

Test Case:

1. Input:  $n = 2$  Output: 2.



The screenshot shows a web-based Python IDE interface. At the top, there is a header with the 'SIMATS ENGINEERING' logo on the left, the text 'Python Programming' in the center, and a user profile 'MADU YUGANDHAR REDDY' with the ID '192372016' on the right. Below the header, there is a toolbar with buttons for 'Index', 'Concatenate', and 'Max-Min'. A yellow status bar indicates 'Run' and 'Save' actions, with a message 'Changes Updated, Saved 141'. The main editor area contains a Python script for calculating the number of distinct ways to climb  $n$  steps. The script defines a function `climbStairs(n)` that uses a sliding window of two previous values to calculate the current value. The input is taken from the user, and the result is printed. The output shows the number 4, which corresponds to the input  $n=2$  (though the script output for  $n=2$  is 2, the screenshot shows 4, likely due to a typo in the input or output display). The console at the bottom shows the prompt 'Enter the number of steps: ' followed by the output 'The number of distinct ways to climb 4 steps is: 5'.

```
1 def climbStairs(n):
2     if n == 0:
3         return 1
4     if n == 1:
5         return 1
6     prev1 = 1
7     prev2 = 1
8     for i in range(2, n + 1):
9         current = prev1 + prev2
10        prev2 = prev1
11        prev1 = current
12    return prev1
13 n = int(input("Enter the number of steps: "))
14 result = climbStairs(n)
15 print(f"The number of distinct ways to climb {n} steps is: {result}")
```

4

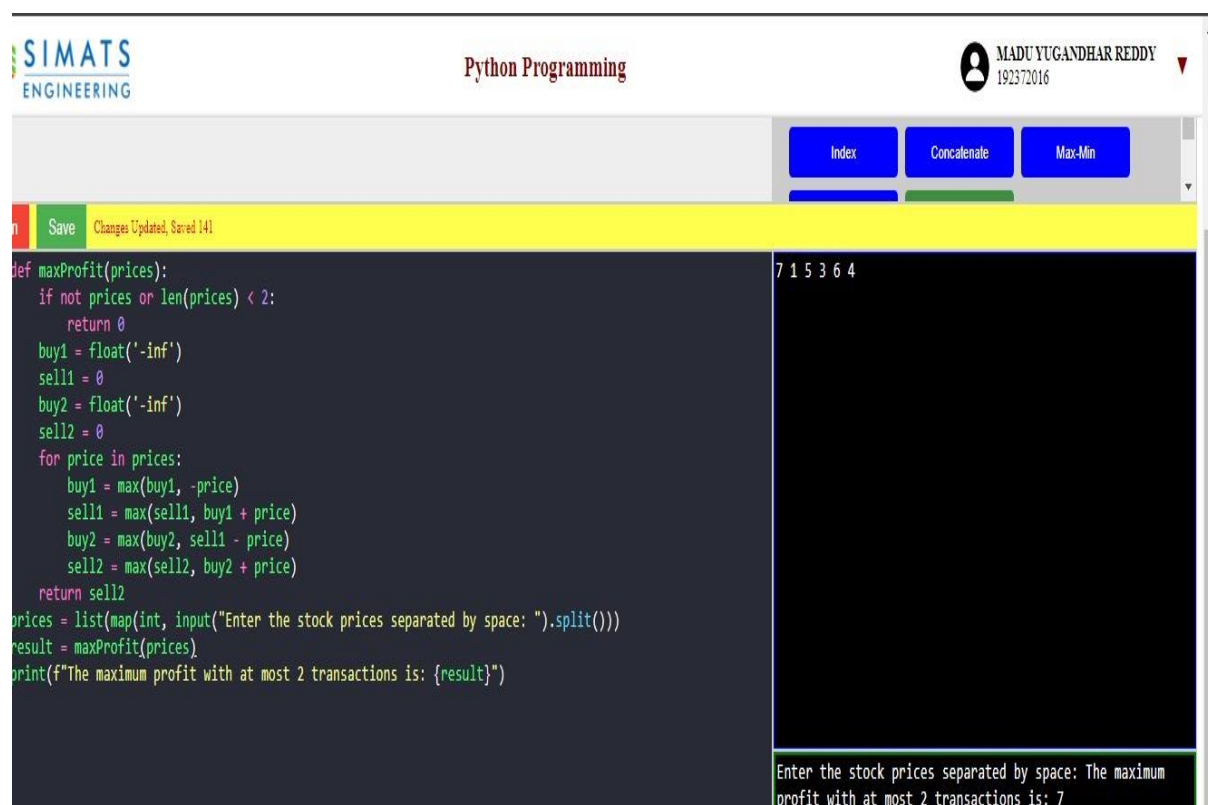
Enter the number of steps: The number of distinct ways to climb 4 steps is: 5

5. In daily share trading, a buyer buys shares in the morning and sells them on the same day.

If the trader is allowed to make at most 2 transactions in a day, whereas the second transaction can only start after the first one is complete (Buy->sell->Buy->sell). Given stock prices throughout the day, find out the maximum profit that a share trader could have made.

Test Case:

1.Input: prices = [7,1,5,3,6,4] Output: 7



The screenshot shows a Python programming environment with the following components:

- Header:** SIMATS ENGINEERING logo, Python Programming title, and user profile MADU YUGANDHAR REDDY 192372016.
- Buttons:** Index, Concatenate, and Max-Min.
- Code Editor:** Contains the following Python code:

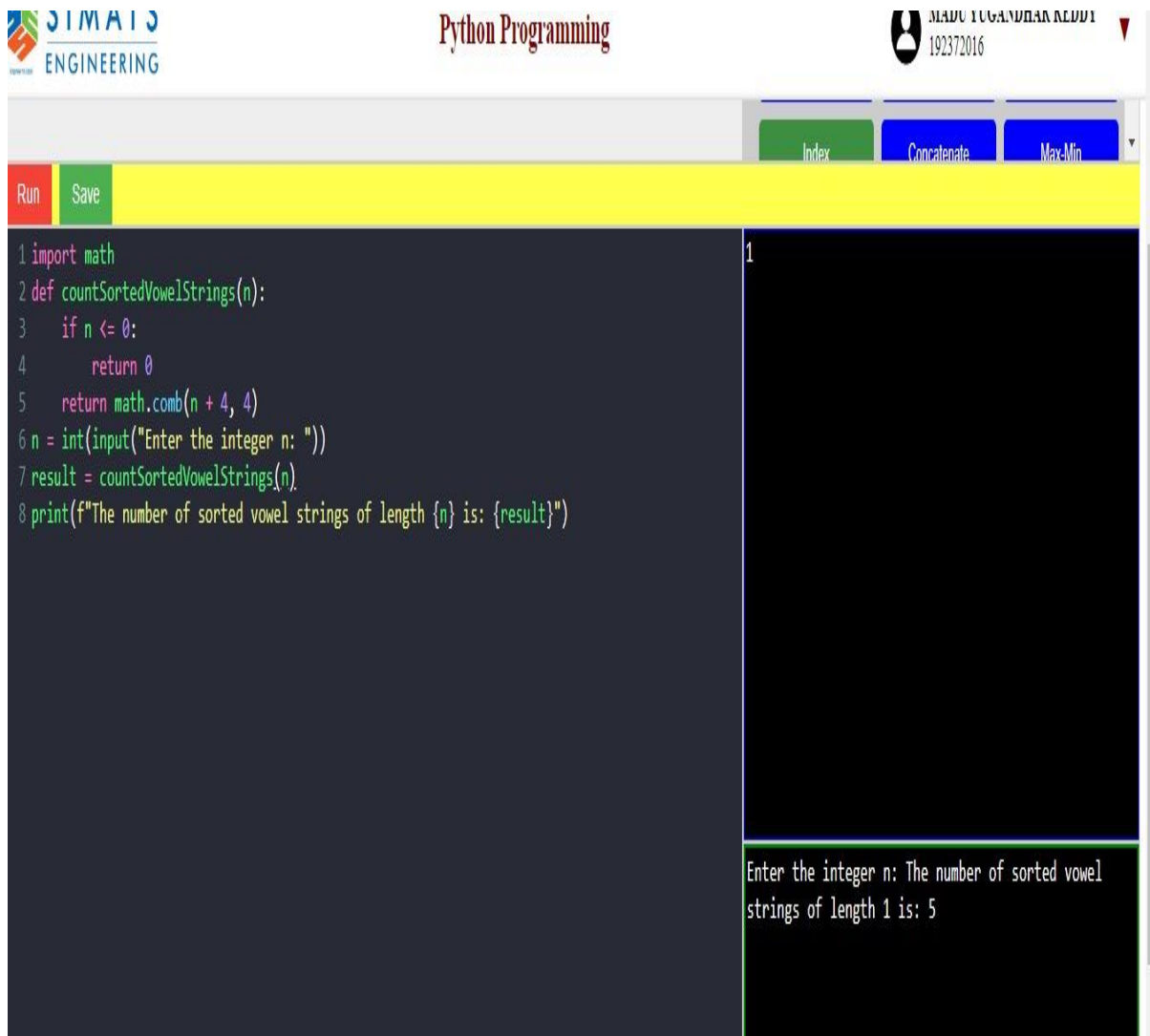
```
def maxProfit(prices):  
    if not prices or len(prices) < 2:  
        return 0  
    buy1 = float('-inf')  
    sell1 = 0  
    buy2 = float('-inf')  
    sell2 = 0  
    for price in prices:  
        buy1 = max(buy1, -price)  
        sell1 = max(sell1, buy1 + price)  
        buy2 = max(buy2, sell1 - price)  
        sell2 = max(sell2, buy2 + price)  
    return sell2  
prices = list(map(int, input("Enter the stock prices separated by space: ").split()))  
result = maxProfit(prices)  
print(f"The maximum profit with at most 2 transactions is: {result}")
```
- Output Console:** Displays the input "7 1 5 3 6 4" and the output "Enter the stock prices separated by space: The maximum profit with at most 2 transactions is: 7".

6. Given an integer  $n$ , return the number of strings of length  $n$  that consist only of vowels (a, e, i, o, u) and are lexicographically sorted. A string  $s$  is lexicographically sorted if for all valid  $i$ ,  $s[i]$  is the same as or comes before  $s[i+1]$  in the alphabet.

Test Cases:

1. Input:  $n = 1$  Output: 5

Explanation: The 5 sorted strings that consist of vowels only are ["a", "e", "i", "o", "u"].



The screenshot shows a Python IDE interface. At the top, there is a header with the SIMATS ENGINEERING logo on the left, the text "Python Programming" in the center, and a user profile icon with the name "MADU YUGANDEAN ALDUI" and ID "192372016" on the right. Below the header, there is a toolbar with buttons for "Run", "Save", "Index", "Concatenate", and "Max-Min". The main editor area contains the following Python code:

```
1 import math
2 def countSortedVowelStrings(n):
3     if n <= 0:
4         return 0
5     return math.comb(n + 4, 4)
6 n = int(input("Enter the integer n: "))
7 result = countSortedVowelStrings(n)
8 print(f"The number of sorted vowel strings of length {n} is: {result}")
```

On the right side of the editor, there is a console output area. It shows the number "1" at the top, followed by the input prompt "Enter the integer n:" and the output "The number of sorted vowel strings of length 1 is: 5".

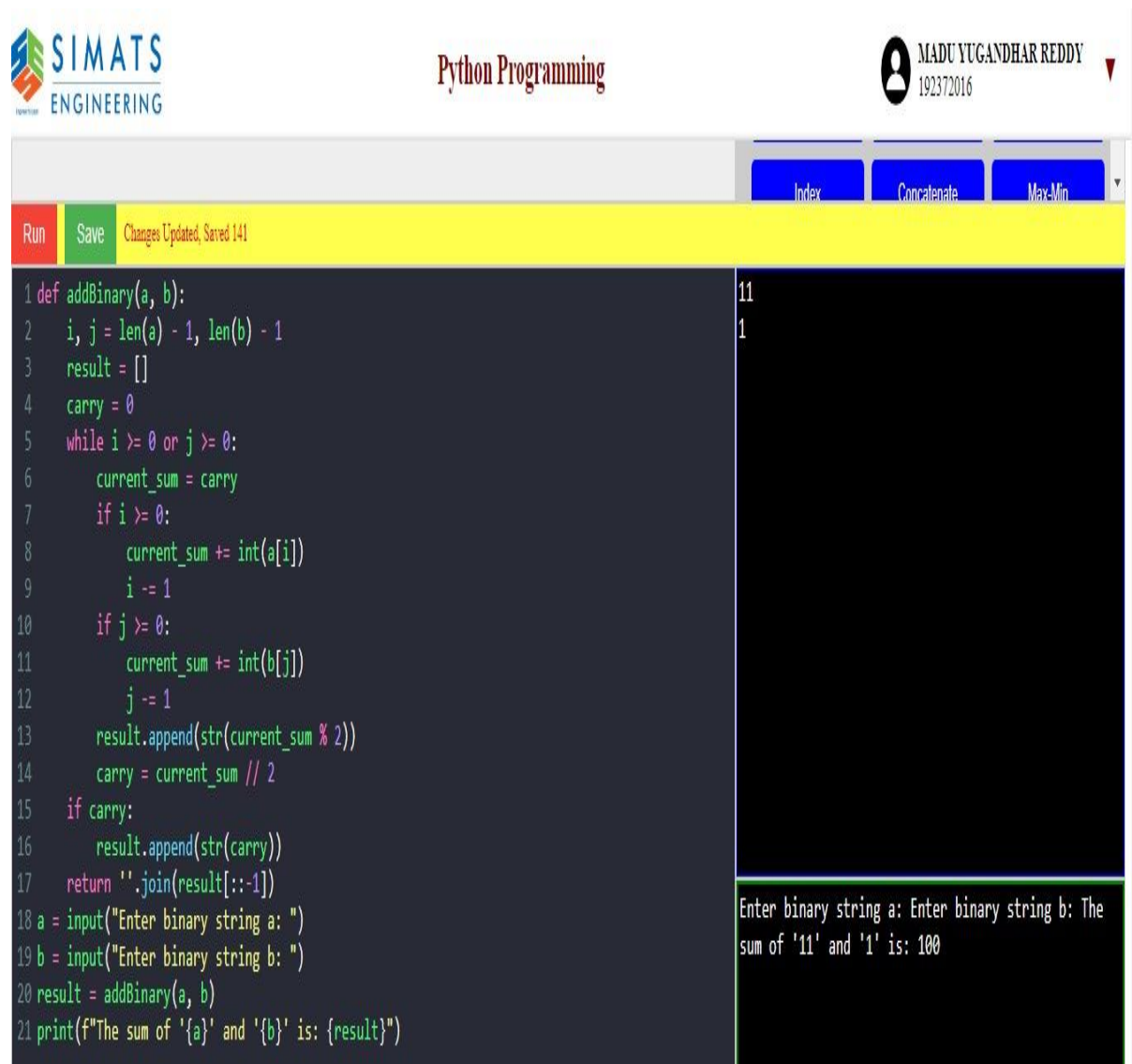


7. Given two binary strings *a* and *b*, return their sum as a binary string.

- *a* and *b* consist only of '0' or '1' characters.
- Each string does not contain leading zeros except for the zero itself.

Test cases:

1. Input: *a* = "11", *b* = "1" Output: "100"



The screenshot shows a Python IDE interface. At the top left is the SIMATS ENGINEERING logo. In the center is the text "Python Programming". At the top right is a user profile for MADU YUGANDHAR REDDY with ID 192372016. Below the header is a toolbar with buttons for "Run", "Save", and "Changes Updated, Saved 141". To the right of the toolbar are tabs for "Index", "Concatenate", and "Max-Min". The main area is a code editor with a dark background. It contains a Python function `addBinary(a, b)` that calculates the sum of two binary strings. The function uses a while loop to iterate over the characters of *a* and *b* from right to left, calculating the sum and carry. The result is built as a list of strings, which is then joined into a single string. Below the code editor is a console window. It shows the input "11" and "1", and the output "The sum of '11' and '1' is: 100".

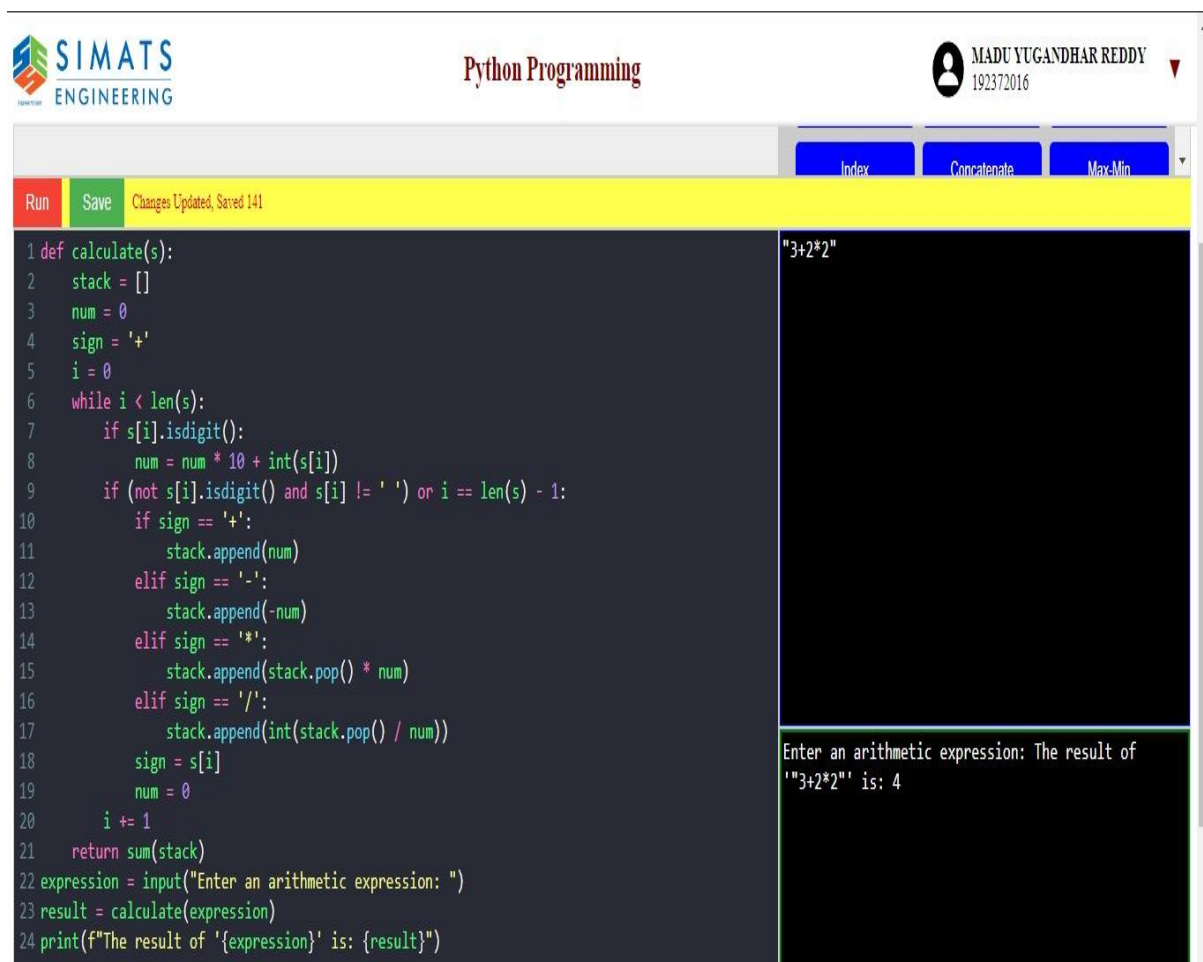
```
1 def addBinary(a, b):
2     i, j = len(a) - 1, len(b) - 1
3     result = []
4     carry = 0
5     while i >= 0 or j >= 0:
6         current_sum = carry
7         if i >= 0:
8             current_sum += int(a[i])
9             i -= 1
10        if j >= 0:
11            current_sum += int(b[j])
12            j -= 1
13        result.append(str(current_sum % 2))
14        carry = current_sum // 2
15    if carry:
16        result.append(str(carry))
17    return ''.join(result[::-1])
18 a = input("Enter binary string a: ")
19 b = input("Enter binary string b: ")
20 result = addBinary(a, b)
21 print(f"The sum of '{a}' and '{b}' is: {result}")
```

Enter binary string a: 11  
Enter binary string b: 1  
The sum of '11' and '1' is: 100



**8. Basic Calculator II** Given a string *s* which represents an expression, evaluate this expression and return its value. The integer division should truncate toward zero. You may assume that the given expression is always valid. All intermediate results will be in the range of  $[-231, 231 - 1]$ .

**1.**Input: *s* = "3+2\*2" Output: 7



The screenshot shows a Python IDE interface. At the top, there is a header with the SIMATS ENGINEERING logo on the left, the text "Python Programming" in the center, and a user profile for "MADU YUGANDHAR REDDY" with the ID "192372016" on the right. Below the header is a toolbar with buttons for "Run", "Save", "Index", "Concatenate", and "Max-Min". The "Run" button is highlighted in red. The main area is divided into two panes. The left pane contains a Python script for a calculator. The right pane shows the output of the script. The script defines a function `calculate(s)` that evaluates an arithmetic expression using a stack. It then takes user input, calls the function, and prints the result. The output pane shows the input expression "3+2\*2" and the resulting output "Enter an arithmetic expression: The result of '3+2\*2' is: 4".

```
1 def calculate(s):
2     stack = []
3     num = 0
4     sign = '+'
5     i = 0
6     while i < len(s):
7         if s[i].isdigit():
8             num = num * 10 + int(s[i])
9         if (not s[i].isdigit() and s[i] != ' ') or i == len(s) - 1:
10            if sign == '+':
11                stack.append(num)
12            elif sign == '-':
13                stack.append(-num)
14            elif sign == '*':
15                stack.append(stack.pop() * num)
16            elif sign == '/':
17                stack.append(int(stack.pop() / num))
18            sign = s[i]
19            num = 0
20            i += 1
21    return sum(stack)
22 expression = input("Enter an arithmetic expression: ")
23 result = calculate(expression)
24 print(f"The result of '{expression}' is: {result}")
```

3+2\*2

Enter an arithmetic expression: The result of '3+2\*2' is: 4


9. Raju, has again started troubling people in your city. The people have turned on to you for getting rid of Raju. Raju presents to you a number consisting of numbers from 0 to 9 characters. He wants you to reverse it from the final answer such that the number becomes


Mirror number. A Mirror is a number which equals its reverse. The hope of people are on you so you have to solve the riddle. You have to tell if some number exists which you would reverse to convert the number into Mirror

Sample input:

Enter the number: 123456

Sample output: Mirror image: 654321

Python Programming

MADU YUGANDHAR REDDY  
192372016

RunSaveChanges Updated, Saved 141

```
1 def is_mirror_number(number):
2     original_str = str(number)
3     reversed_str = original_str[::-1]
4     reversed_number = int(reversed_str)
5     if reversed_number == number:
6         return reversed_number
7     else:
8         return None
9 def main():
10     try:
11         number = int(input("Enter the number: "))
12         mirror_number = is_mirror_number(number)
13         if mirror_number is not None:
14             print(f"Mirror image: {mirror_number}")
15         else:
16             print("No such number exists which would reverse to convert the number into a Mirror")
17     except ValueError:
18         print("Invalid input. Please enter a valid number.")
19 if __name__ == "__main__":
20     main()
```

123456

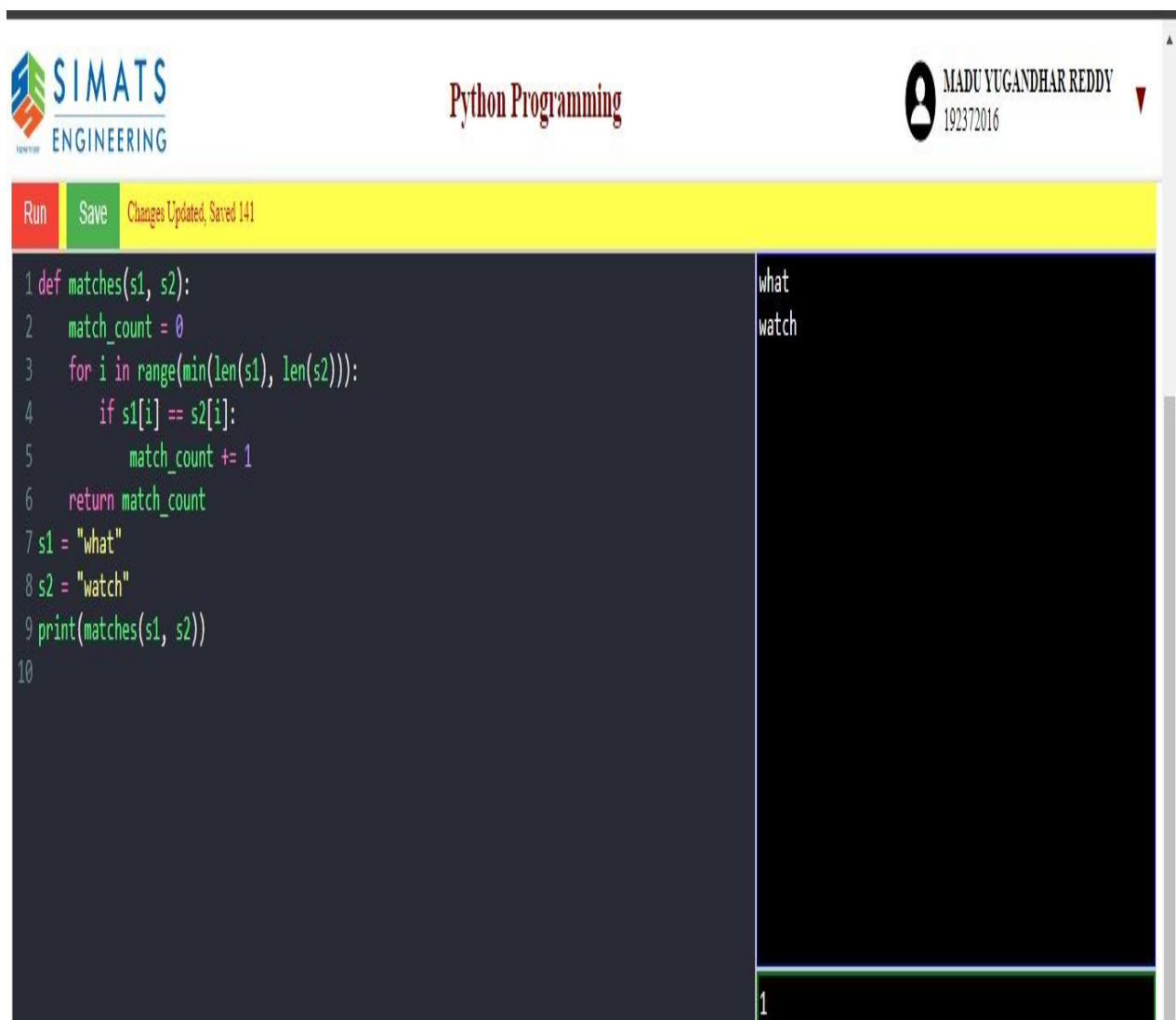
Enter the number: No such number exists which would reverse to convert the number into a Mirror

10. Write a python function called matches that takes two strings as arguments and returns

how many matches there are between the strings. A match is where the two strings have the same character at the same index.

Test Cases:

1. Input: s1= "what" s2= "watch" Output: 1



The screenshot shows a Python IDE interface. At the top, there is a header with the SIMATS ENGINEERING logo on the left, the text "Python Programming" in the center, and a user profile icon with the name "MADU YUGANDHAR REDDY" and ID "192372016" on the right. Below the header is a yellow bar with "Run" and "Save" buttons, and a status message "Changes Updated, Saved 141". The main area is a dark-themed code editor with the following Python code:

```
1 def matches(s1, s2):
2     match_count = 0
3     for i in range(min(len(s1), len(s2))):
4         if s1[i] == s2[i]:
5             match_count += 1
6     return match_count
7 s1 = "what"
8 s2 = "watch"
9 print(matches(s1, s2))
10
```

To the right of the code editor is a console window. It displays the input strings "what" and "watch" on separate lines, and the output "1" at the bottom.