SAVEETHA SCHOOL OF ENGINEERING

DEPARTMENT OF COMPUTERSCIENCE AND ENGINEERING

CSA0889 – Python Programming

Assignment - 3

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

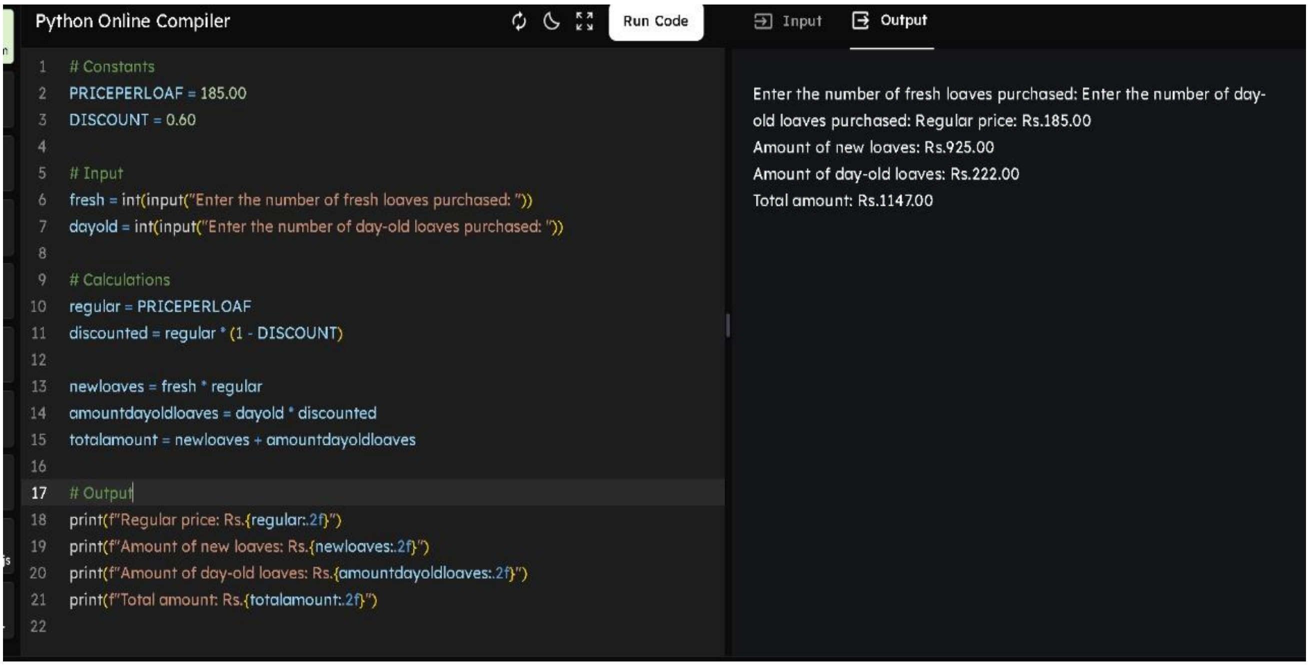
Sample Output:

Regular price: Rs.185.00 Amount of new loaves: 925.00

Amount of day-old loaves: 333.00

Total amount: Rs. 1258.00

Test cases: 1. 4, 6 2. -1,5 3. 0,6 4. 7,8 5. 3,4



```python
# Constants
PRICEPERLOAF = 185.00
DISCOUNT = 0.60

# Input
fresh = int(input("Enter the number of fresh loaves purchased: "))
dayold = int(input("Enter the number of day-old loaves purchased: "))

# Calculations
regular = PRICEPERLOAF
discounted = regular * (1 - DISCOUNT)

newloaves = fresh * regular
amountdayoldloaves = dayold * discounted
totalamount = newloaves + amountdayoldloaves

# Output
print(f"Regular price: Rs.{regular:.2f}")
print(f"Amount of new loaves: Rs.{newloaves:.2f}")
print(f"Amount of day-old loaves: Rs.{amountdayoldloaves:.2f}")
print(f"Total amount: Rs.{totalamount:.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: Rs.925.00
Amount of day-old loaves: Rs.222.00
Total amount: Rs.1147.00
```

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:

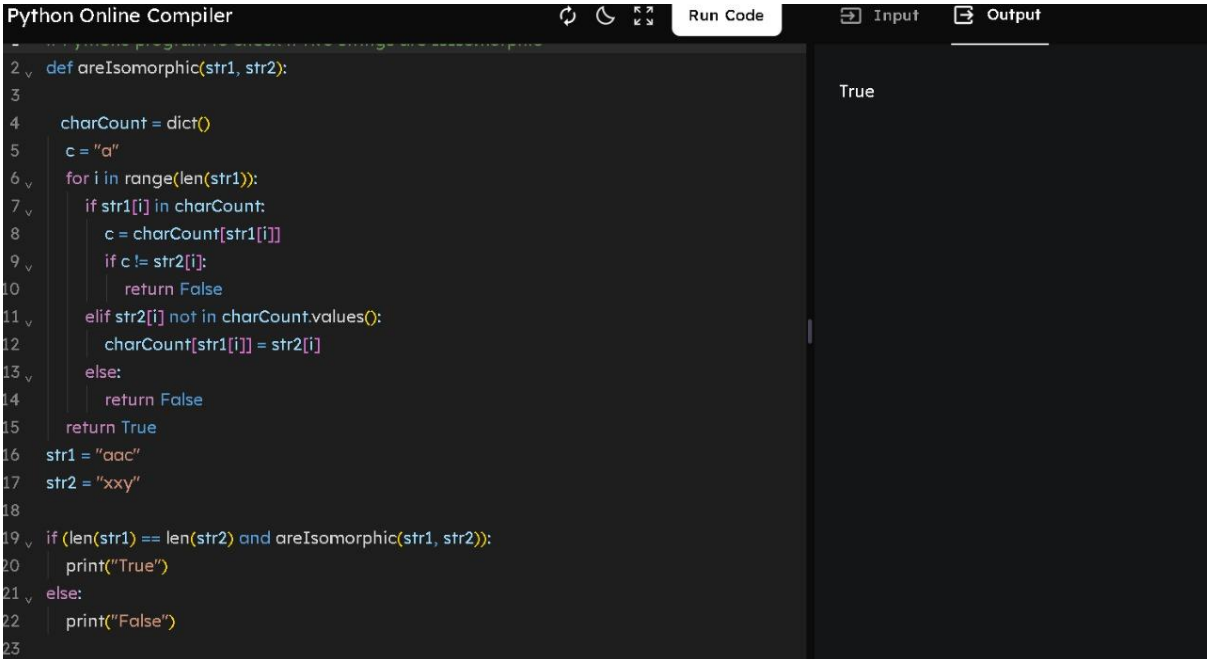
1.Input: s = "egg", t = "add" Output: true

2.Input: s = "foo", t = "bar" Output: false

3.Input: s = "paper", t = "title" Output: true

4.Input: s = "fry", t = "sky" Output: true

5. Input: s = "apples", t = "apple" Output: false



```python
def areIsomorphic(str1, str2):

    charCount = dict()
    c = "a"
    for i in range(len(str1)):
        if str1[i] in charCount:
            c = charCount[str1[i]]
            if c != str2[i]:
                return False
        elif str2[i] not in charCount.values():
            charCount[str1[i]] = str2[i]
        else:
            return False
    return True
str1 = "aac"
str2 = "xxy"

if (len(str1) == len(str2) and areIsomorphic(str1, str2)):
    print("True")
else:
    print("False")
```

```
True
```

3. Given n non-negative integers a1, a2,a3,…an where each represents a point at coordinate (i, ai) . ‘ n ‘ vertical lines are drawn such that the two endpoints of line i is at (i, ai) 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:

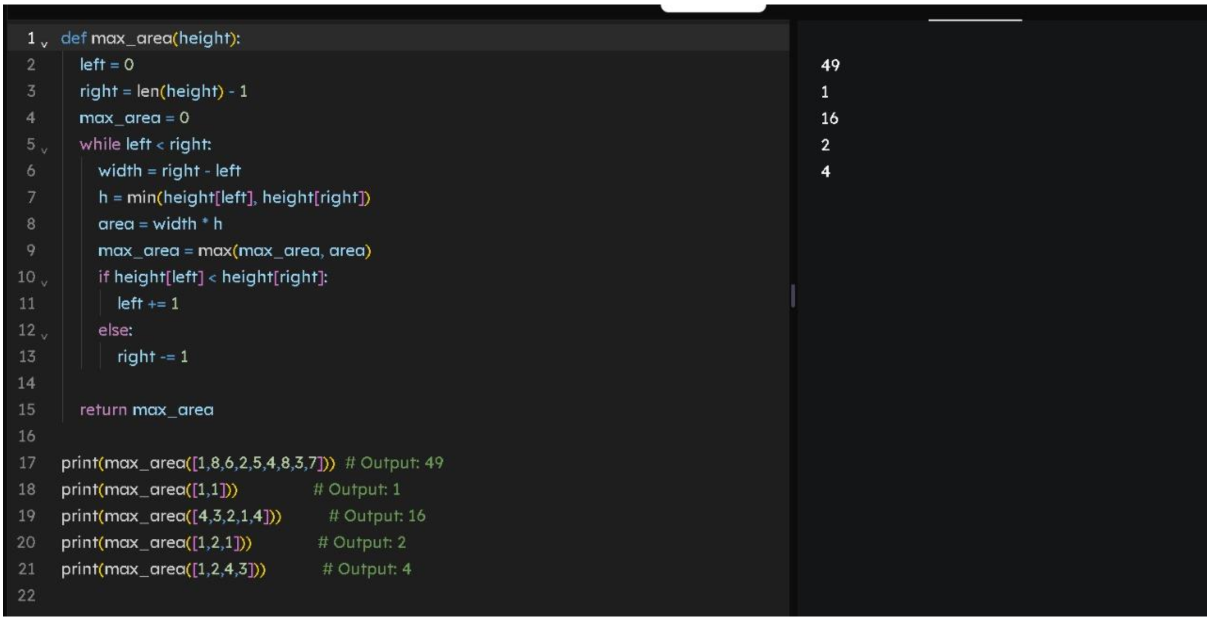
1.Input: array = [1, 5, 4, 3] Output: 6

2.Input: array = [3, 1, 2, 4, 5] Output: 12

3.Input: array = [1,8,6,2,5,4,8,3,7] Output: 49

4.Input: array = [1,1] Output: 1

5.Input: array = [7,3] Output: 3 at coordinate (i, ai)



```python
def max_area(height):
    left = 0
    right = len(height) - 1
    max_area = 0
    while left < right:
        width = right - left
        h = min(height[left], height[right])
        area = width * h
        max_area = max(max_area, area)
        if height[left] < height[right]:
            left += 1
        else:
            right -= 1

    return max_area

print(max_area([1,8,6,2,5,4,8,3,7])) # Output: 49
print(max_area([1,1]))           # Output: 1
print(max_area([4,3,2,1,4]))       # Output: 16
print(max_area([1,2,1]))          # Output: 2
print(max_area([1,2,4,3]))         # Output: 4
```

```
49
1
16
2
4
```

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:

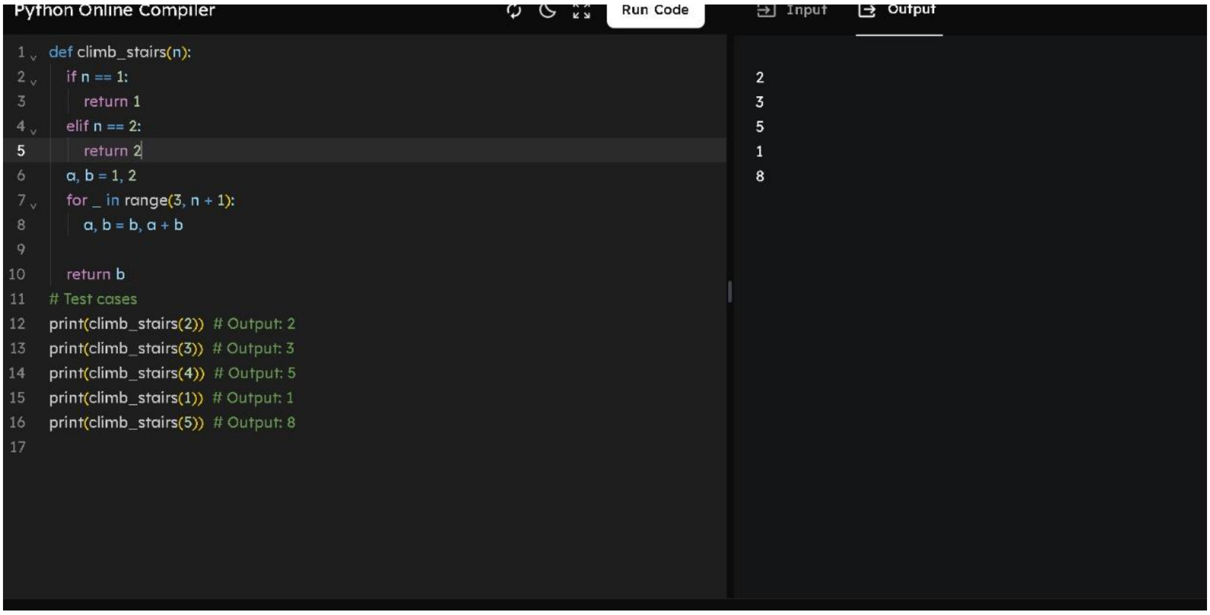
1.Input: n = 2 Output: 2

2.Input: n = 3 Output: 3

3.Input: n = 4 Output: 5

4.Input: n = 1 Output: 1

5.Input: n = 5 Output: 8



```
def climb_stairs(n):
    if n == 1:
        return 1
    elif n == 2:
        return 2
    a, b = 1, 2
    for _ in range(3, n + 1):
        a, b = b, a + b

    return b
# Test cases
print(climb_stairs(2))  # Output: 2
print(climb_stairs(3))  # Output: 3
print(climb_stairs(4))  # Output: 5
print(climb_stairs(1))  # Output: 1
print(climb_stairs(5))  # Output: 8
```

```
2
3
5
1
8
```

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:

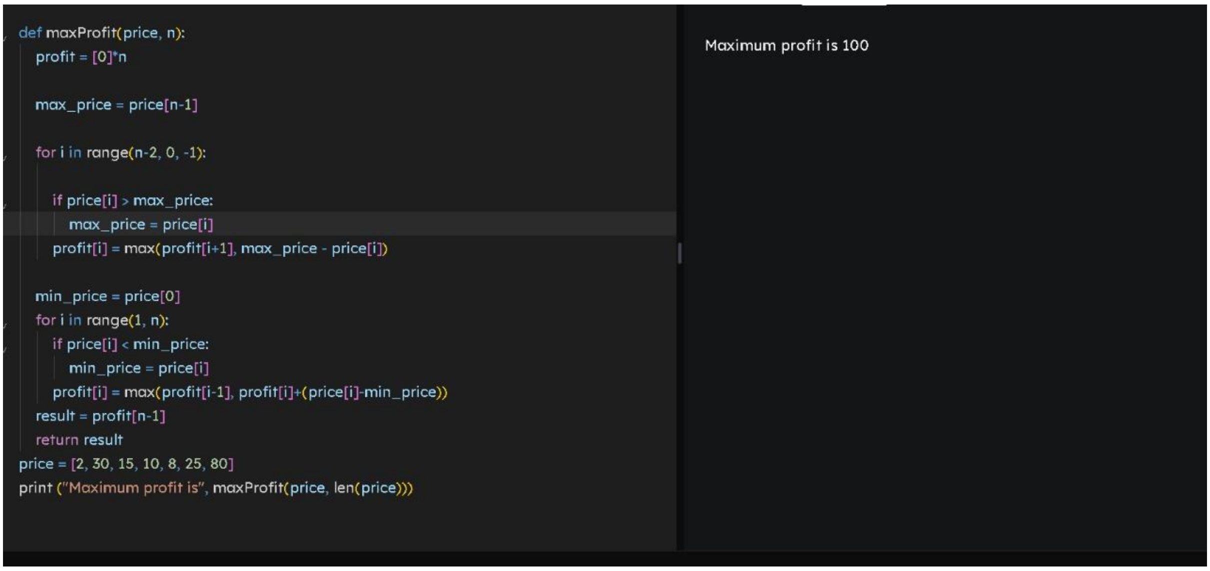
1.Input: prices = [7,1,5,3,6,4] Output: 7

2.Input: prices = [7,6,4,3,1] Output: 0

3.Input: [10, 22, 5, 75, 65, 80] Output:87

4.Input: [2, 30, 15, 10, 8, 25, 80] Output:100

5. Input: [5,25,3,10,7,9] Output:27



```
def maxProfit(price, n):
    profit = [0]*n

    max_price = price[n-1]

    for i in range(n-2, 0, -1):

        if price[i] > max_price:
            max_price = price[i]
        profit[i] = max(profit[i+1], max_price - price[i])

    min_price = price[0]
    for i in range(1, n):
        if price[i] < min_price:
            min_price = price[i]
        profit[i] = max(profit[i-1], profit[i]+(price[i]-min_price))
    result = profit[n-1]
    return result
price = [2, 30, 15, 10, 8, 25, 80]
print ("Maximum profit is", maxProfit(price, len(price)))
```

```
Maximum profit is 100
```

SAVEETHA SCHOOL OF ENGINEERING

DEPARTMENT OF COMPUTERSCIENCE AND ENGINEERING

CSA0889 – Python Programming

Assignment – 4

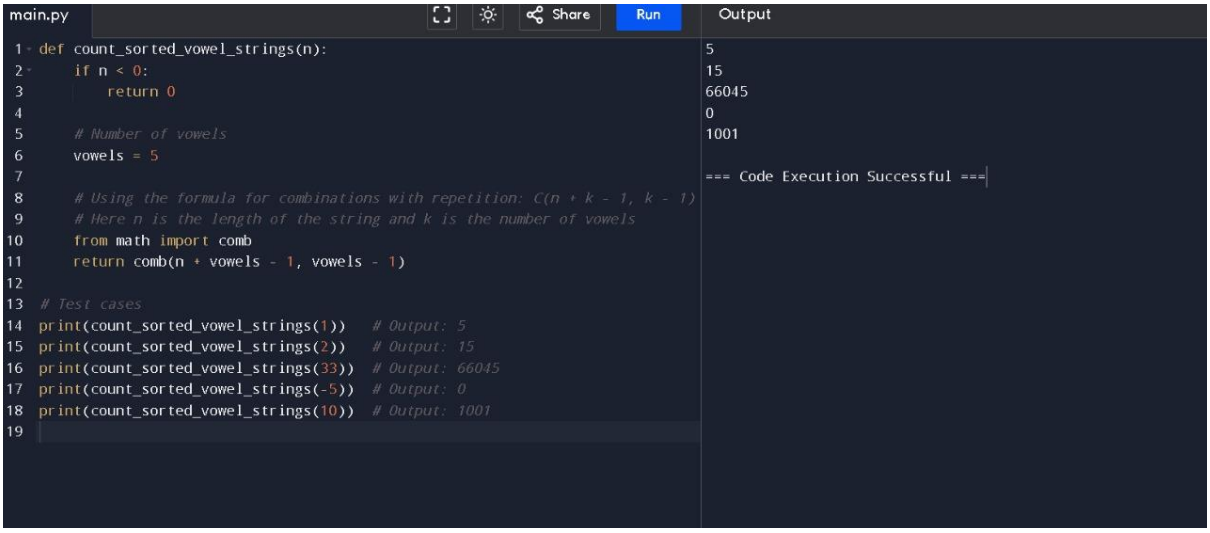
1. Given an integer n, return the number of strings of length n that consist only of vowels (a, e, i, o, u) and are lexicographically sorted. A string s is lexicographically sorted if for all valid i, s[i] is the same as or comes before s[i+1] in the alphabet.

Test Cases:

1.Input: n = 1 Output: 5 Explanation: The 5 sorted strings that consist of vowels only are ["a","e","i","o","u"].

2.Input: n = 2 Output: 15 Explanation: The 15 sorted strings that consist of vowels only are ["aa","ae","ai","ao","au","ee","ei","eo","eu","ii","io","iu","oo","ou","uu"]. Note that "ea" is not a valid string since 'e' comes after 'a' in the alphabet.

3. Input: n = 33 Output: 66045 4.n=-5 5.n=1



```python
def count_sorted_vowel_strings(n):
    if n < 0:
        return 0

    # Number of vowels
    vowels = 5

    # Using the formula for combinations with repetition: C(n + k - 1, k - 1)
    # Here n is the length of the string and k is the number of vowels
    from math import comb
    return comb(n + vowels - 1, vowels - 1)

# Test cases
print(count_sorted_vowel_strings(1))   # Output: 5
print(count_sorted_vowel_strings(2))   # Output: 15
print(count_sorted_vowel_strings(33))  # Output: 66045
print(count_sorted_vowel_strings(-5))  # Output: 0
print(count_sorted_vowel_strings(10))  # Output: 1001
```

```
5
15
66045
0
1001

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

2.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:

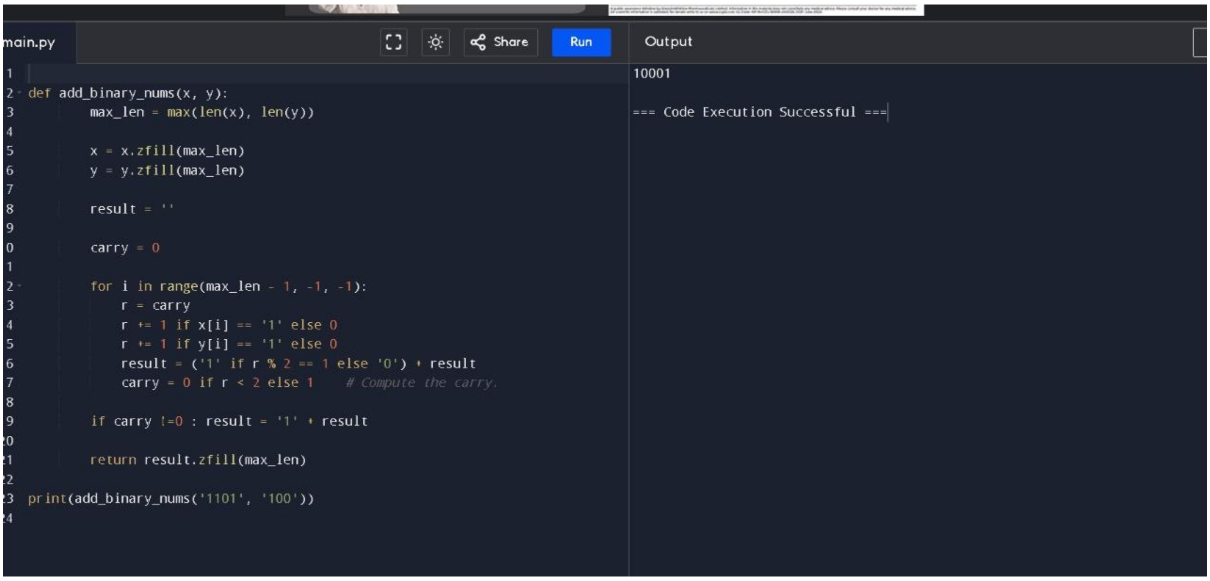
1.Input: a = "11", b = "1" Output: "100"

2.Input: a = "1010", b = "1011" Output: "10101" ‘

3.a= “1111”, b= “1010”

4.a= “101101”, b= “1100”

5.a= “1011” b= “1111”



```python

def add_binary_nums(x, y):
        max_len = max(len(x), len(y))

        x = x.zfill(max_len)
        y = y.zfill(max_len)

        result = ''

        carry = 0

        for i in range(max_len - 1, -1, -1):
            r = carry
            r += 1 if x[i] == '1' else 0
            r += 1 if y[i] == '1' else 0
            result = ('1' if r % 2 == 1 else '0') + result
            carry = 0 if r < 2 else 1     # Compute the carry.

        if carry !=0 : result = '1' + result

        return result.zfill(max_len)

print(add_binary_nums('1101', '100'))
```

```
10001

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

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]. • s consists of integers and operators ('+', '-', '*', '/') separated by some number of spaces. • s represents a valid expression. • All the integers in the expression are non-negative integers in the range [0, 231 - 1]. The answer is guaranteed to fit in a 32-bit integer. Note: You are not allowed to use any built-in function which evaluates strings as mathematical expressions, such as eval().

Test cases:

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

2.Input: s = " 3/2 " Output: 1

3.Input: s = " 3+5 / 2 " Output: 5

4.s= “-1+5” 5.s= “2+3+5”

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

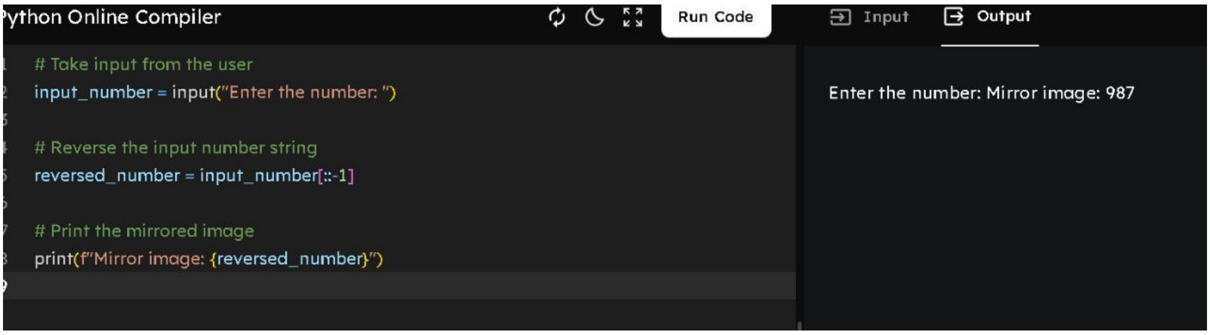
Sample input:

Enter the number: 123456

Sample output: Mirror image: 654321

Test cases:

1. Sell123
2. 5489236
3. Abc-abc
4. %$$$$^&
5. -123456



```
# Take input from the user
input_number = input("Enter the number: ")

# Reverse the input number string
reversed_number = input_number[::-1]

# Print the mirrored image
print(f"Mirror image: {reversed_number}")
```

```
Enter the number: Mirror image: 987
```

5. 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”

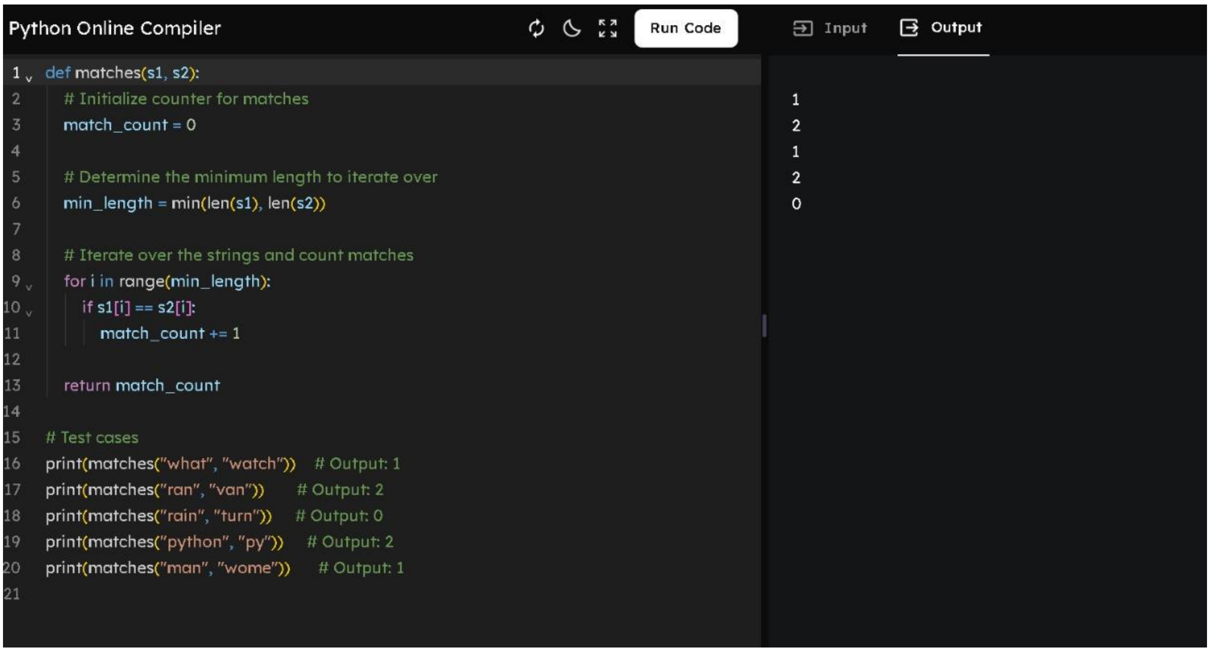
Output:

2.Input: s1= “ ran” s2= “van”

3. Input : s1 = “ rain” s2 = “ turn”

4.Input : s1 = “ python” s2 = “py”

5. Inppput: s1= “man” s2= “women



```python
def matches(s1, s2):
    # Initialize counter for matches
    match_count = 0

    # Determine the minimum length to iterate over
    min_length = min(len(s1), len(s2))

    # Iterate over the strings and count matches
    for i in range(min_length):
        if s1[i] == s2[i]:
            match_count += 1

    return match_count

# Test cases
print(matches("what", "watch"))   # Output: 1
print(matches("ran", "van"))      # Output: 2
print(matches("rain", "turn"))    # Output: 0
print(matches("python", "py"))    # Output: 2
print(matches("man", "wome"))     # Output: 1
```

```
1
2
1
2
0
```