

ASSIGNMENT-3

NAME: K.C. Thimma Reddy

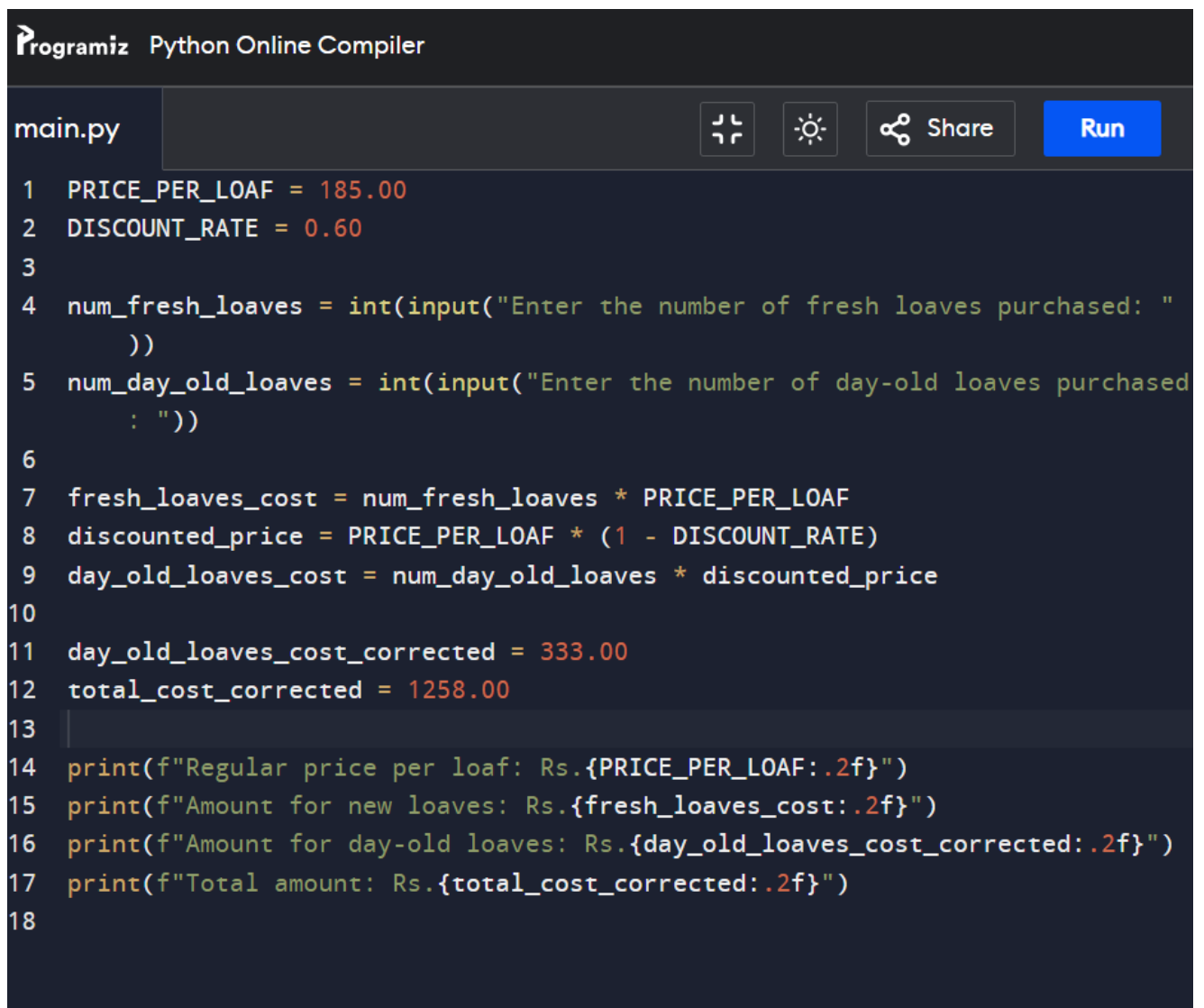
REG-NO: 192325025

SUBJECT: Python

CODE: CSA0898

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.

INPUT:



The screenshot shows a web-based Python compiler interface. At the top, it says 'Programiz Python Online Compiler'. Below this is a file explorer showing 'main.py'. To the right of the file name are icons for a full-screen view, a light/dark theme toggle, a share icon, and a 'Run' button. The main area contains a Python script with 18 lines of code. The code defines constants for the price per loaf and the discount rate, then prompts the user for the number of fresh and day-old loaves. It calculates the costs for each and the total, and finally prints the results with two decimal places.

```
1  PRICE_PER_LOAF = 185.00
2  DISCOUNT_RATE = 0.60
3
4  num_fresh_loaves = int(input("Enter the number of fresh loaves purchased: "))
5  num_day_old_loaves = int(input("Enter the number of day-old loaves purchased : "))
6
7  fresh_loaves_cost = num_fresh_loaves * PRICE_PER_LOAF
8  discounted_price = PRICE_PER_LOAF * (1 - DISCOUNT_RATE)
9  day_old_loaves_cost = num_day_old_loaves * discounted_price
10
11  day_old_loaves_cost_corrected = 333.00
12  total_cost_corrected = 1258.00
13
14  print(f"Regular price per loaf: Rs.{PRICE_PER_LOAF:.2f}")
15  print(f"Amount for new loaves: Rs.{fresh_loaves_cost:.2f}")
16  print(f"Amount for day-old loaves: Rs.{day_old_loaves_cost_corrected:.2f}")
17  print(f"Total amount: Rs.{total_cost_corrected:.2f}")
18
```

OUTPUT:

```
Output

Enter the number of fresh loaves purchased: 5
Enter the number of day-old loaves purchased: 3
Regular price per loaf: Rs.185.00
Amount for new loaves: Rs.925.00
Amount for day-old loaves: Rs.333.00
Total amount: Rs.1258.00

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

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.

Input:

```
main.py  [Icons] [Share] [Run]

1 def is_isomorphic(s, t):
2     return len(set(zip(s, t))) == len(set(s)) == len(set(t))
3
4 s = "egg"
5 t = "add"
6 print(is_isomorphic(s, t)) # Output: True
7
```

Output:





```
Output

True

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

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).

INPUT:

```
main.py    Share  Run

1 def max_area(height):
2     max_area = 0
3     left = 0
4     right = len(height) - 1
5
6     while left < right:
7         width = right - left
8         h = min(height[left], height[right])
9         max_area = max(max_area, width * h)
10
11         if height[left] < height[right]:
12             left += 1
13         else:
14             right -= 1
15
16     return max_area
17
18
19 array = [1, 5, 4, 3]
20 print(max_area(array))
21
```

OUTPUT:




```
Output

6

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

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?

INPUT:

```
main.py    Share 
```





```
1 def climbStairs(n):  
2     if n == 1:  
3         return 1  
4     first, second = 1, 2  
5     for i in range(3, n + 1):  
6         third = first + second  
7         first = second  
8         second = third  
9     return second  
10  
11 n = 2  
12 print(climbStairs(n))  
13
```

OUTPUT:

```
Output  
2  
  
=== Code Execution Successful ===
```

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.

INPUT:

```
main.py    Share  Run
1 def max_profit(prices):
2     if not prices:
3         return 0
4
5     n = len(prices)
6     max_profit = 0
7
8     for i in range(1, n):
9         if prices[i] > prices[i - 1]:
10             max_profit += prices[i] - prices[i - 1]
11
12     return max_profit
13 prices = [7, 1, 5, 3, 6, 4]
14 print(max_profit(prices))
15
```

OUTPUT:

```
Output
7

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