MACHINE LEARNING –ICP#2

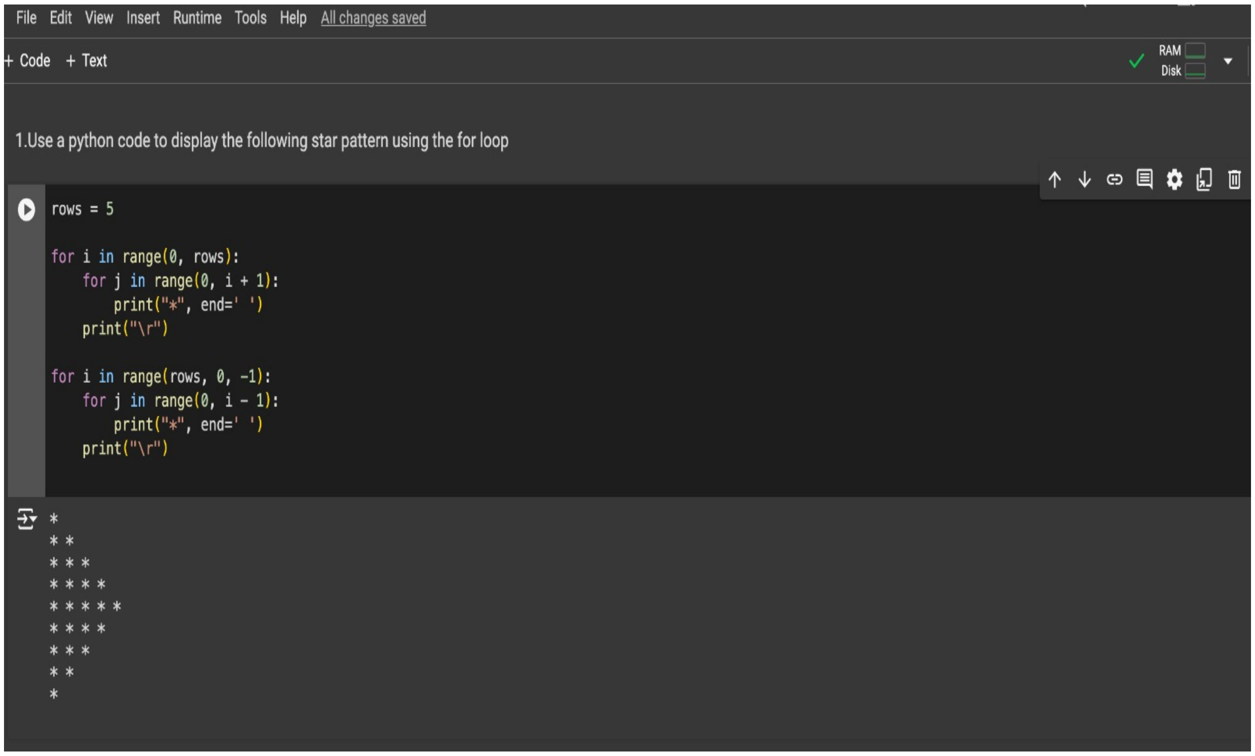
Student Name : Likhitha Parvathi Tadikonda

Student Id: 700752941

Github Link: https://github.com/LikhithaTadikonda/Machine-Learning-ICP-s/tree/master/ICP-2

Question 1:

The number of rows in the pattern is represented by the variable rows, which is first initialized to 5. After that, the star pattern is printed using two nested for loops. The inner loop prints i+1 stars in each row while the outside loop iterates from 0 to rows-1 in the first half of the pattern. The inner loop prints i-1 stars in each row while the outer loop iterates from rows down to 1 in steps of -1 in the second part of the pattern.



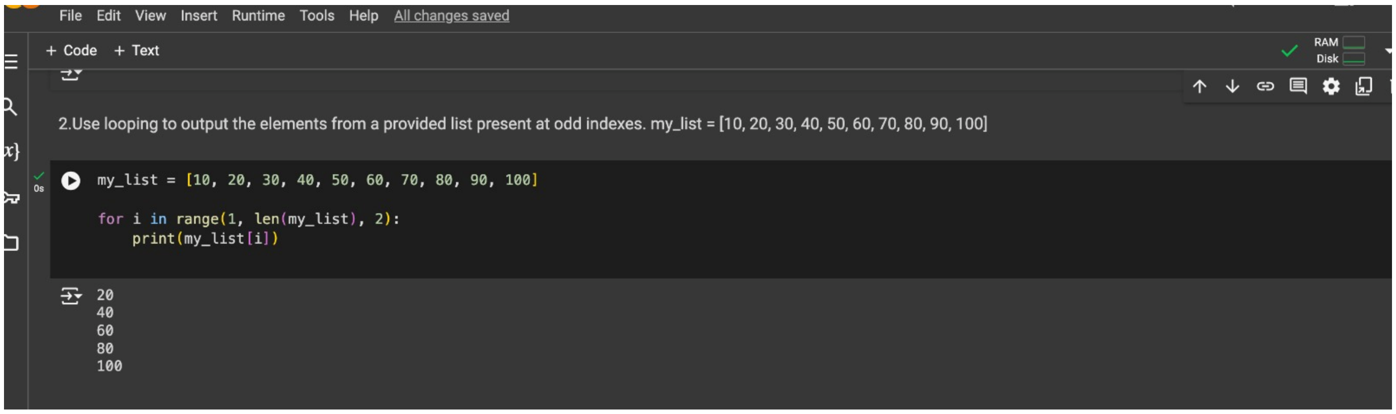
1.Use a python code to display the following star pattern using the for loop

```
rows = 5

for i in range(0, rows):
    for j in range(0, i + 1):
        print("*", end=' ')
    print("\r")

for i in range(rows, 0, -1):
    for j in range(0, i - 1):
        print("*", end=' ')
    print("\r")
```

```
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

Question 2:

The code initializes a list named my_list containing integers. It uses a for loop with the range() function to iterate over the indices of my_list. The loop starts from index 1 because it wants to print elements at odd indices. The loop goes up to len(my_list) (the length of the list) with a step of 2 ensuring it only visits odd indices. Inside the loop, it prints the element at the current odd index using my_list[i].



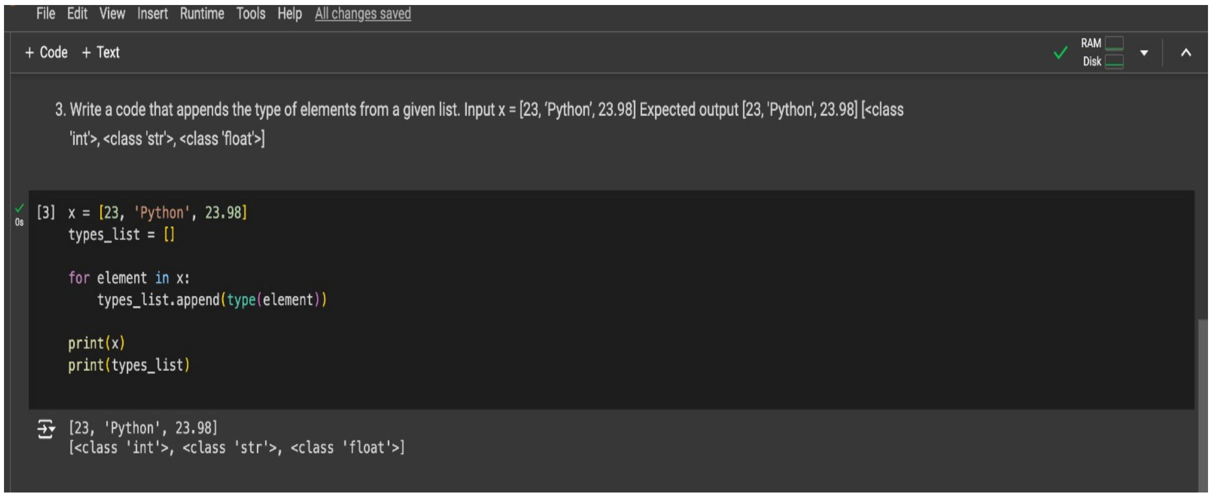
2.Use looping to output the elements from a provided list present at odd indexes. my_list = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

```
my_list = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

for i in range(1, len(my_list), 2):
    print(my_list[i])
```

```
20
40
60
80
100
```

Question 3:

We iterate over each element in the list x. For each element, we use the type() function to determine its type, and then append that type to the types_list. Finally, we print both the original list x and the list containing types types_list.



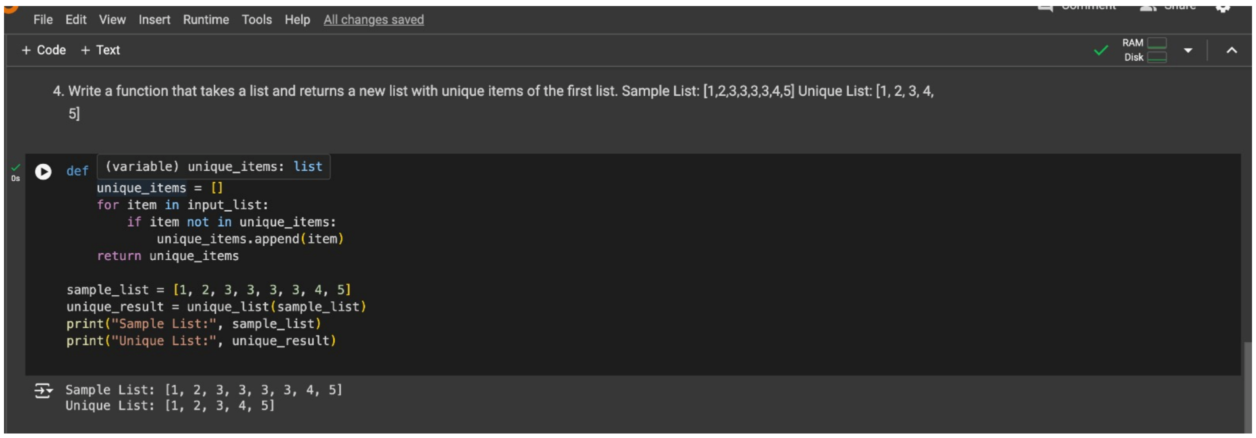
3. Write a code that appends the type of elements from a given list. Input x = [23, 'Python', 23.98] Expected output [23, 'Python', 23.98] [<class 'int'>, <class 'str'>, <class 'float'>]

```
[3] x = [23, 'Python', 23.98]
    types_list = []

    for element in x:
        types_list.append(type(element))

    print(x)
    print(types_list)
```

```
[23, 'Python', 23.98]
[<class 'int'>, <class 'str'>, <class 'float'>]
```

Question 4:

The function unique_list takes a list input_list as input. It initializes an empty list unique_items to store unique elements. It iterates through each element in the input list. For each element, it checks if it's already in the unique_items list. If not, it appends it. Finally, it returns the list of unique items.



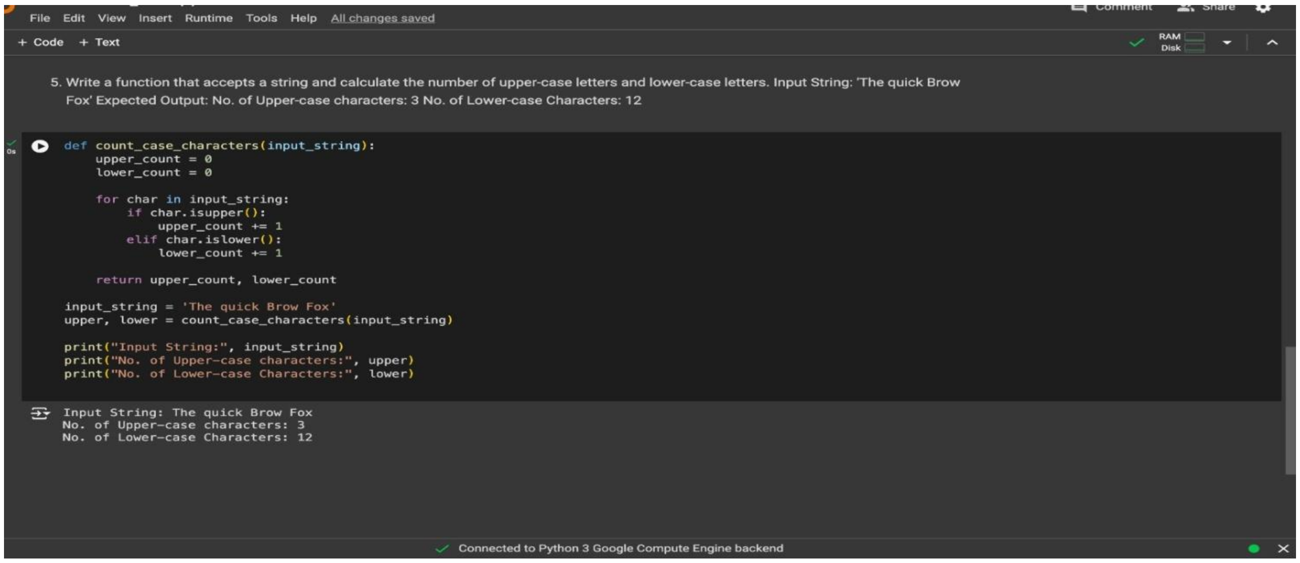
4. Write a function that takes a list and returns a new list with unique items of the first list. Sample List: [1,2,3,3,3,3,4,5] Unique List: [1, 2, 3, 4, 5]

```
def (variable) unique_items: list
    unique_items = []
    for item in input_list:
        if item not in unique_items:
            unique_items.append(item)
    return unique_items

sample_list = [1, 2, 3, 3, 3, 3, 4, 5]
unique_result = unique_list(sample_list)
print("Sample List:", sample_list)
print("Unique List:", unique_result)
```

```
Sample List: [1, 2, 3, 3, 3, 3, 4, 5]
Unique List: [1, 2, 3, 4, 5]
```

Question 5:

The function count_case_characters accepts a string input_string as input. It initializes variables upper_count and lower_count to store the counts of uppercase and lowercase letters, respectively.It iterates through each character in the input string.For each character, it checks if it's uppercase using the isupper() method and increments the upper_count if it's true. Similarly, it checks if the character is lowercase using the islower() method and increments the lower_count if it's true. Finally, it returns the counts of uppercase and lowercase characters



5. Write a function that accepts a string and calculate the number of upper-case letters and lower-case letters. Input String: 'The quick Brow Fox' Expected Output: No. of Upper-case characters: 3 No. of Lower-case Characters: 12

```
def count_case_characters(input_string):
    upper_count = 0
    lower_count = 0

    for char in input_string:
        if char.isupper():
            upper_count += 1
        elif char.islower():
            lower_count += 1

    return upper_count, lower_count

input_string = 'The quick Brow Fox'
upper, lower = count_case_characters(input_string)

print("Input String:", input_string)
print("No. of Upper-case characters:", upper)
print("No. of Lower-case Characters:", lower)
```

```
Input String: The quick Brow Fox
No. of Upper-case characters: 3
No. of Lower-case Characters: 12
```

Connected to Python 3 Google Compute Engine backend