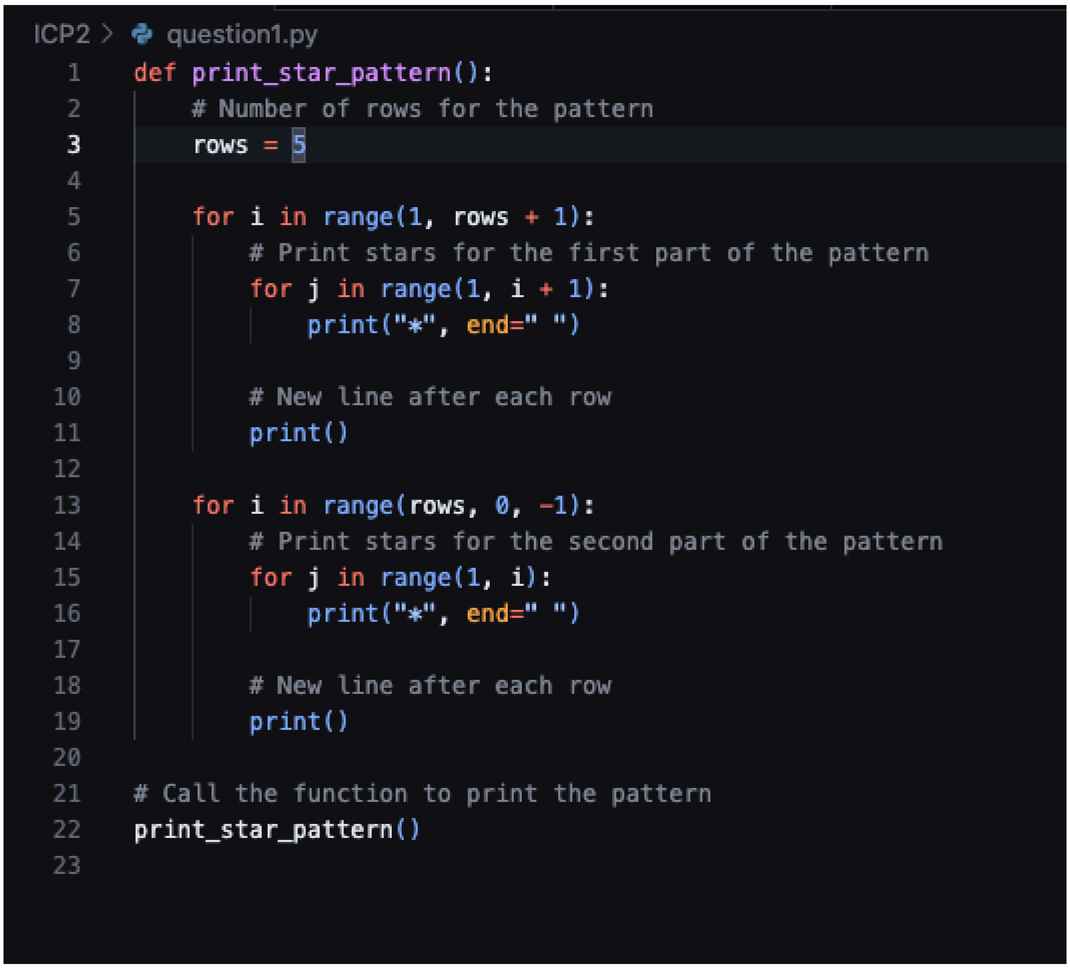
Machine Learning ICP2
Rishma Reddy Nalla
700752916

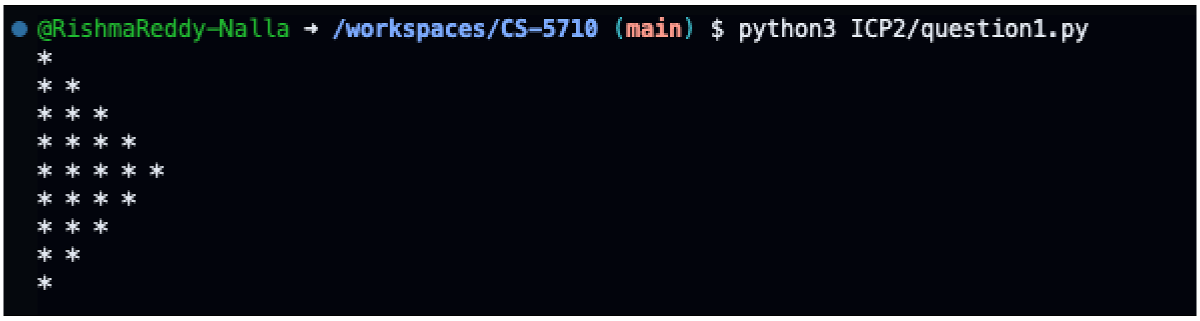
Video Link: https://youtu.be/fPxKuBez4PM
GitHub Link: https://github.com/RishmaReddy-Nalla/CS-5710/tree/main/ICP2

Question1:



ICP2 > question1.py

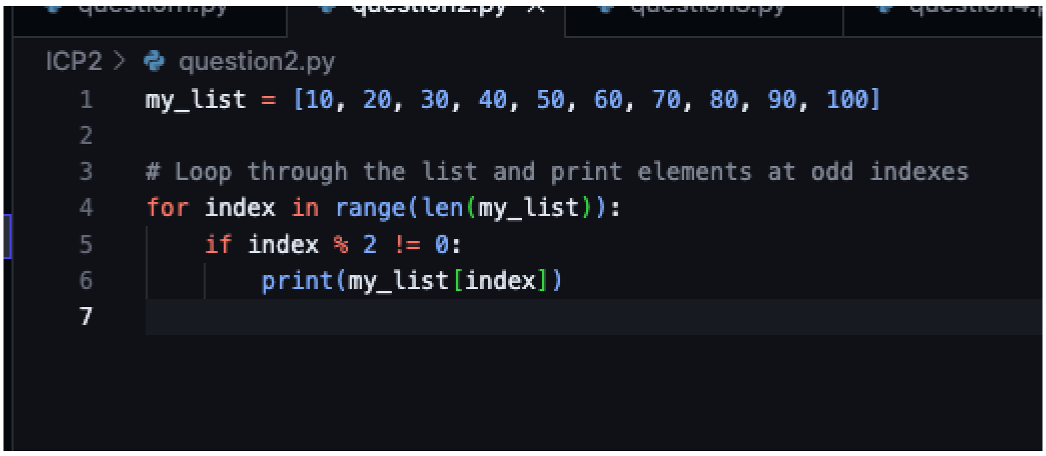
```python
def print_star_pattern():
    # Number of rows for the pattern
    rows = 5

    for i in range(1, rows + 1):
        # Print stars for the first part of the pattern
        for j in range(1, i + 1):
            print("*", end=" ")

        # New line after each row
        print()

    for i in range(rows, 0, -1):
        # Print stars for the second part of the pattern
        for j in range(1, i):
            print("*", end=" ")

        # New line after each row
        print()

# Call the function to print the pattern
print_star_pattern()
```



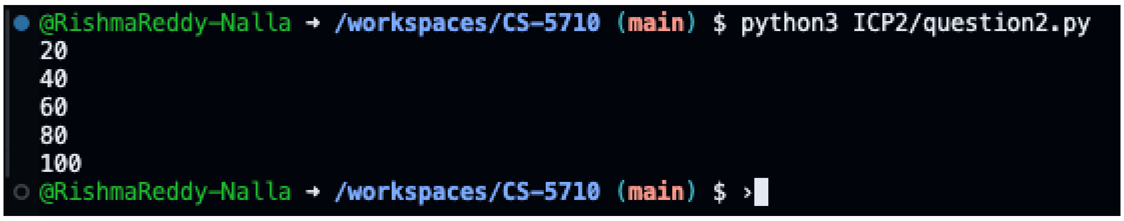
```
@RishmaReddy-Nalla → /workspaces/CS-5710 (main) $ python3 ICP2/question1.py
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

Function: print_star_pattern
The function print_star_pattern prints a star pattern in the shape of a diamond using nested loops.

Question 2:



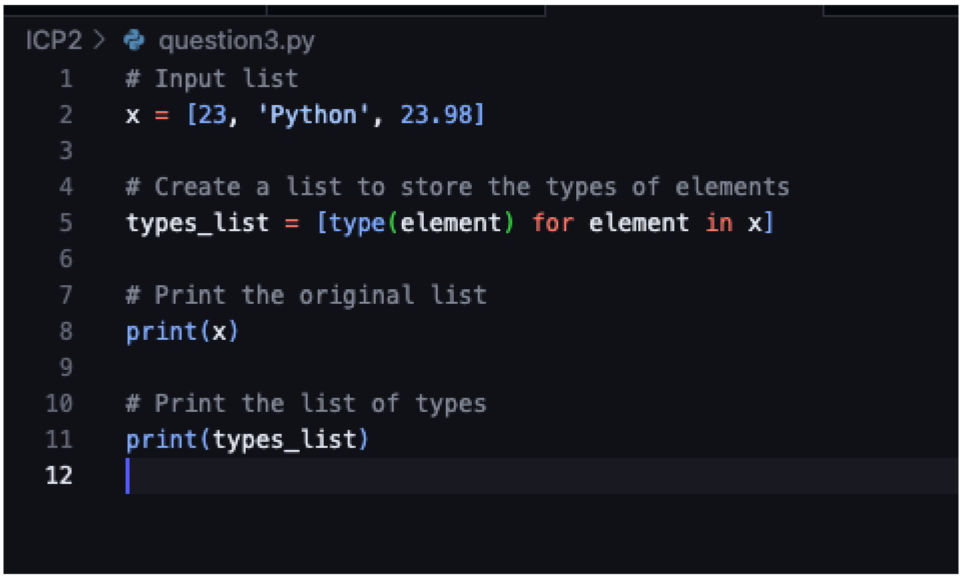
```
ICP2 > question2.py
my_list = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

# Loop through the list and print elements at odd indexes
for index in range(len(my_list)):
    if index % 2 != 0:
        print(my_list[index])
```



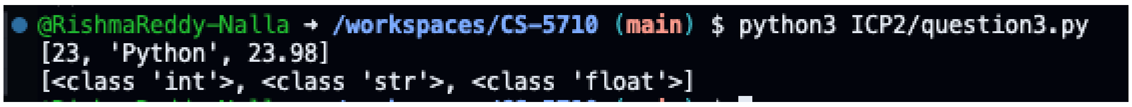
```
@RishmaReddy-Nalla → /workspaces/CS-5710 (main) $ python3 ICP2/question2.py
20
40
60
80
100
@RishmaReddy-Nalla → /workspaces/CS-5710 (main) $
```

The code loops through the `my_list`, checks if each index is odd, and prints the elements that are located at odd indexes.

Question 3:



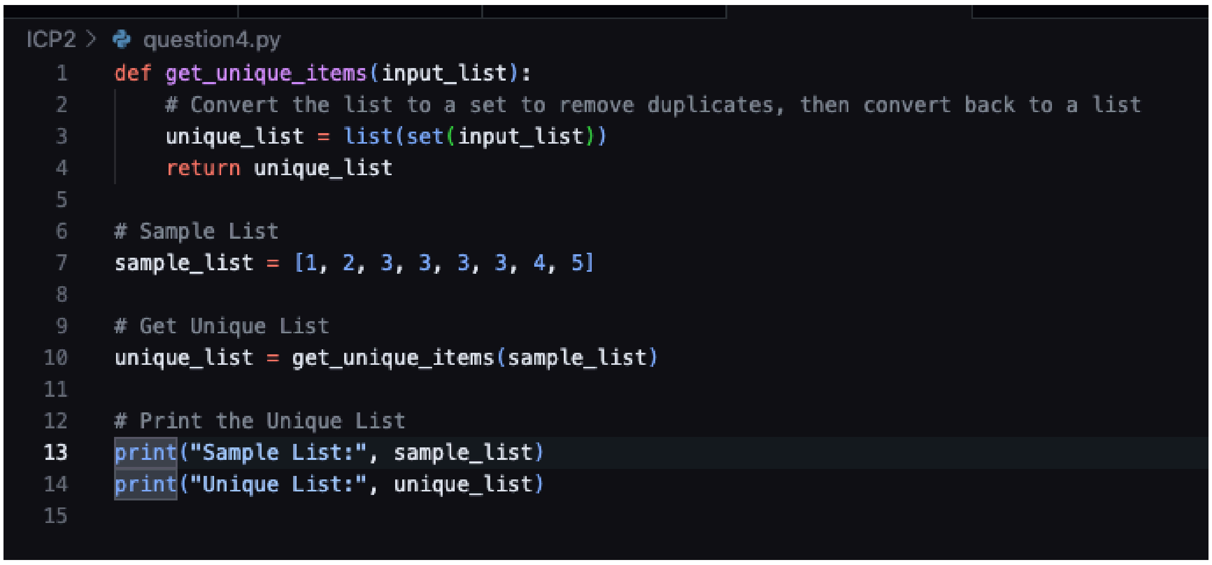
```
ICP2 > question3.py
# Input list
x = [23, 'Python', 23.98]

# Create a list to store the types of elements
types_list = [type(element) for element in x]

# Print the original list
print(x)

# Print the list of types
print(types_list)
```



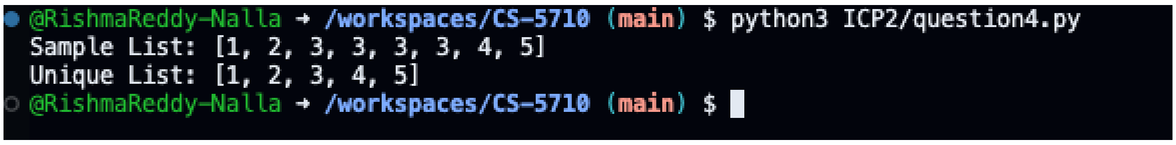
```
@RishmaReddy-Nalla → /workspaces/CS-5710 (main) $ python3 ICP2/question3.py
[23, 'Python', 23.98]
[<class 'int'>, <class 'str'>, <class 'float'>]
```

The code defines a list x with mixed data types.
It creates `types_list` containing the types of each element in x.
Finally, it prints both the original list and the list of types.

Question 4:


ICP2 > question4.py

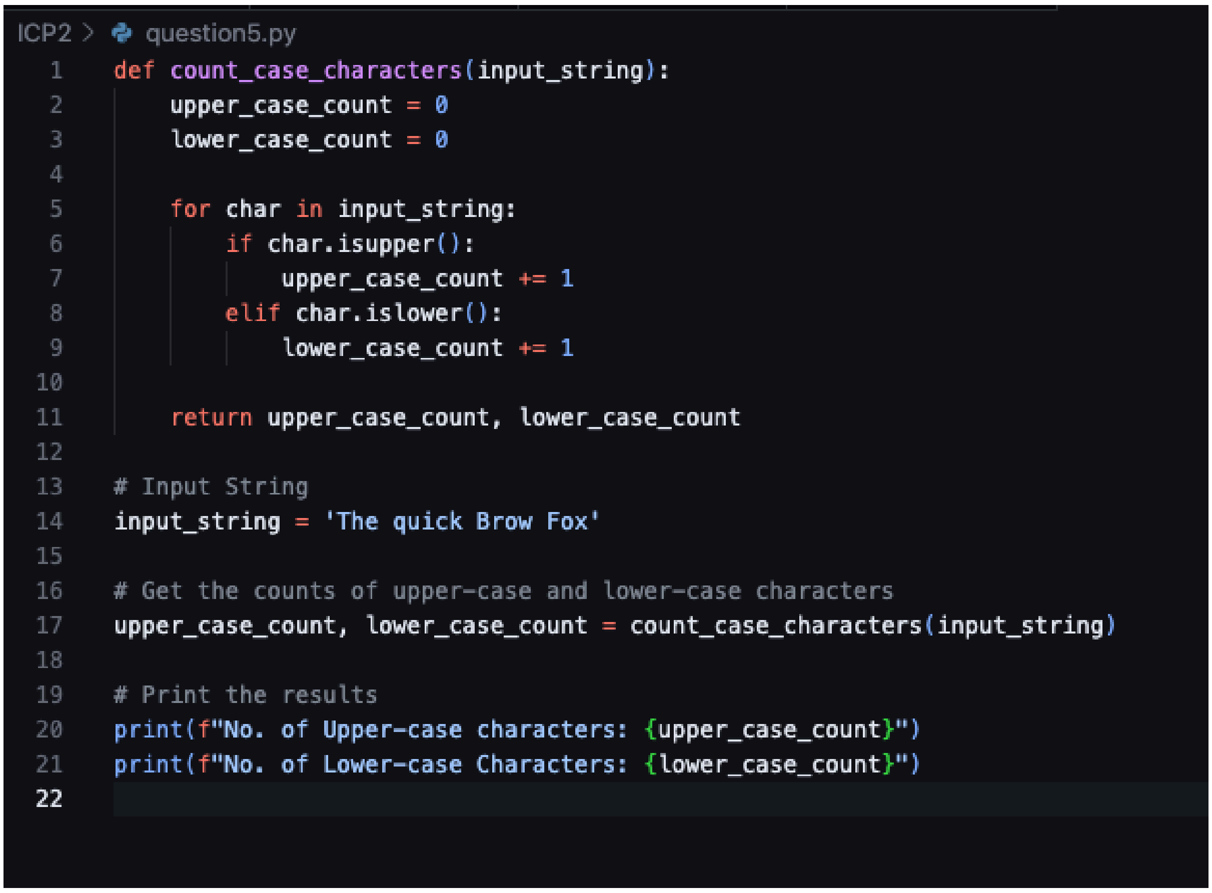
```python
def get_unique_items(input_list):
    # Convert the list to a set to remove duplicates, then convert back to a list
    unique_list = list(set(input_list))
    return unique_list

# Sample List
sample_list = [1, 2, 3, 3, 3, 3, 4, 5]

# Get Unique List
unique_list = get_unique_items(sample_list)

# Print the Unique List
print("Sample List:", sample_list)
print("Unique List:", unique_list)
```



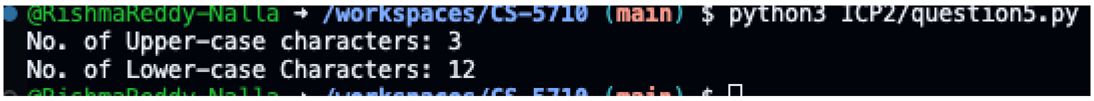
```
@RishmaReddy-Nalla → /workspaces/CS-5710 (main) $ python3 ICP2/question4.py
Sample List: [1, 2, 3, 3, 3, 3, 4, 5]
Unique List: [1, 2, 3, 4, 5]
@RishmaReddy-Nalla → /workspaces/CS-5710 (main) $
```

`get_unique_items(input_list)`: This function takes a list as input, removes duplicate elements by converting the list to a set and back to a list, and returns the list of unique items.

Question5:


ICP2 > question5.py

```python
def count_case_characters(input_string):
    upper_case_count = 0
    lower_case_count = 0

    for char in input_string:
        if char.isupper():
            upper_case_count += 1
        elif char.islower():
            lower_case_count += 1

    return upper_case_count, lower_case_count

# Input String
input_string = 'The quick Brow Fox'

# Get the counts of upper-case and lower-case characters
upper_case_count, lower_case_count = count_case_characters(input_string)

# Print the results
print(f"No. of Upper-case characters: {upper_case_count}")
print(f"No. of Lower-case Characters: {lower_case_count}")

```



```
@RishmaReddy-Nalla → /workspaces/CS-5710 (main) $ python3 ICP2/question5.py
No. of Upper-case characters: 3
No. of Lower-case Characters: 12
```

The `count_case_characters` function takes an input string and counts the number of upper-case and lower-case characters in it. It iterates through each character in the input string, and for each character, it checks whether it's upper-case using the `isupper()` method or lower-case using the `islower()` method. Based on the result, it increments the corresponding count variable (`upper_case_count` or `lower_case_count`). Finally, it returns the counts of upper-case and lower-case characters as a tuple.