

MACHINE LEARNING #1

Name: Gowtham Paluri

ID: 700741225

Github link: <https://github.com/gowtham8666/Machine-Learning-Assignments/blob/main/Machine%20learning%20Assignment%202.py>

Q1. Use a python code to display the following star pattern using the for loop

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

➔ The first loop for i in $\text{range}(0, \text{rows})$ prints out the top half of the pyramid, and the second loop for i in $\text{range}(\text{rows}, 0, -1)$ prints out the bottom half of the pyramid.

The inner loops for j in $\text{range}(0, i + 1)$ and for j in $\text{range}(0, i - 1)$ control the number of asterisks to be printed on each row. The `end=' '` argument in the `print()` statement is used to specify that a space should be printed after each asterisk

```

rows = input("Enter the height of pyramid")
rows = int (rows)
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")

```

Enter the height of pyramid5

```

*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*

```

Q2. 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]?

→ The loop for i in range(1, len(my_list), 2) iterates over the indices of the elements in the list, starting from 1 (the first odd index), and increments the index by 2 on each iteration.

```

▶ my_list = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
print("The elements at odd places are : ")
for i in range(1, len(my_list), 2):
    print(my_list[i])

```

The elements at odd places are :

20
40
60
80
100

Q3. 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'>]

➔The loop for i in range(len(s)) iterates over the indices of the elements in the list s, and the x.append(type(s[i])) statement is used to append the data type of the current element in s to the list x.

```

s = [23, 'Python', 23.98]
x = []
for i in range(len(s)):
    x.append(type(s[i]))
print(s)
print(x)

```

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

Q4. 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]

→The code defines a function `unique_list` that takes in a list `l` as input. The function creates an empty list `x`.

Then, it iterates over the elements of the input list `l`. For each element `a` in `l`, it checks if it is already in the list `x`. If `a` is not in `x`, it appends `a` to `x`.

Finally, the function returns the unique list `x` that contains all unique elements from the input list `l`.

The code then calls the function `unique_list` with the list `[1, 2, 3, 3, 3, 3, 4, 5]` as an argument and prints the result, which is `[1, 2, 3, 4, 5]`.

```
def unique_list(l):  
    x = []  
    for a in l:  
        if a not in x:  
            x.append(a)  
    return x  
  
print(unique_list([1,2,3,3,3,3,4,5]))  
  
[1, 2, 3, 4, 5]
```

Q5. 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

→The code defines a function `up_low` that takes in a string `string` as input. The function initializes two variables `uppers` and `lowers` to 0, which will be used to count the number of uppercase and lowercase characters in the input string.

Then, it iterates over the characters in the input string. For each character `char` in `string`, it checks if `char` is a lowercase letter using the method `islower()`. If `char` is a lowercase letter, it increments the `lowers` count by 1. If `char` is not a lowercase letter, it checks if `char` is an uppercase letter using the method `isupper()`. If `char` is an uppercase letter, it increments the `uppers` count by 1.

Finally, the function returns a tuple of the count of uppercase characters (`uppers`) and the count of lowercase characters (`lowers`).

```
def up_low(string):  
    uppers = 0  
    lowers = 0  
    for char in string:  
        if char.islower():  
            lowers += 1  
        elif char.isupper():  
            uppers += 1  
    return (uppers, lowers)  
  
print(up_low('Welcome to Machine Learning'))
```

(3, 21)