P.DILLI BABU

192472284

CSA0815

PYTHON PROGRAMMING FOR BLOCK CHAIN PROJECTS

SLOT-B

1. Read the number until -1 is encounter. find the avg of positive numbers and negative numbers entered by user

Sample Input:

Enter -1 to exit, enter the numbers ,-1

main.py

```python
pos_sum = neg_sum = pos_count = neg_count = 0

while True:
    num = float(input("Enter -1 to exit, enter a number: "))
    if num == -1:
        break
    if num > 0:
        pos_sum += num
        pos_count += 1
    elif num < 0:
        neg_sum += num
        neg_count += 1

avg_pos = pos_sum // pos_count if pos_count > 0 else 0
avg_neg = neg_sum // neg_count if neg_count > 0 else 0

print(f"avg negative number is {avg_neg}")
print(f"avg positive number is {avg_pos}")
```

Output

```
Enter -1 to exit, enter a number: -1
avg negative number is 0
avg positive number is 0

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

2. Write a python program to find the square, cube of the given decimal number.

Sample Input:

Given Number: 0.6

main.py

```python
# Get input from user
number = float(input("Enter a decimal number: "))

# Calculate square and cube
square = number ** 2
cube = number ** 3

# Print results
print(f"Given Number: {number}")
print(f"Square Number: {square}")
print(f"Cube Number: {cube}")
```

Output

```
Enter a decimal number: 0.1
Given Number: 0.1
Square Number: 0.010000000000000002
Cube Number: 0.001000000000000002

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

3.Write a python program to print the following pattern.

Sample Input:

Enter the Character to be printed:+

main.py

```python
# Get input from user
char = input("Enter the Character to be printed: ")
rows = int(input("Number of rows: "))

# Print the pattern
for i in range(1, rows + 1):
    print(char*i)
```

Output

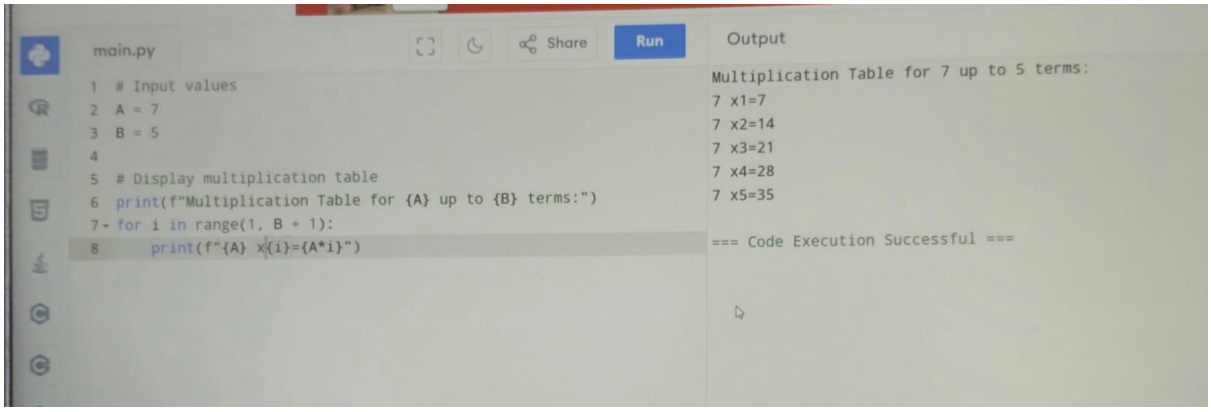
```
Enter the Character to be printed: +
Number of rows: 5
+
++
+++
++++
+++++

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

4.Python Program to Display the Multiplication Table

Sample Input:

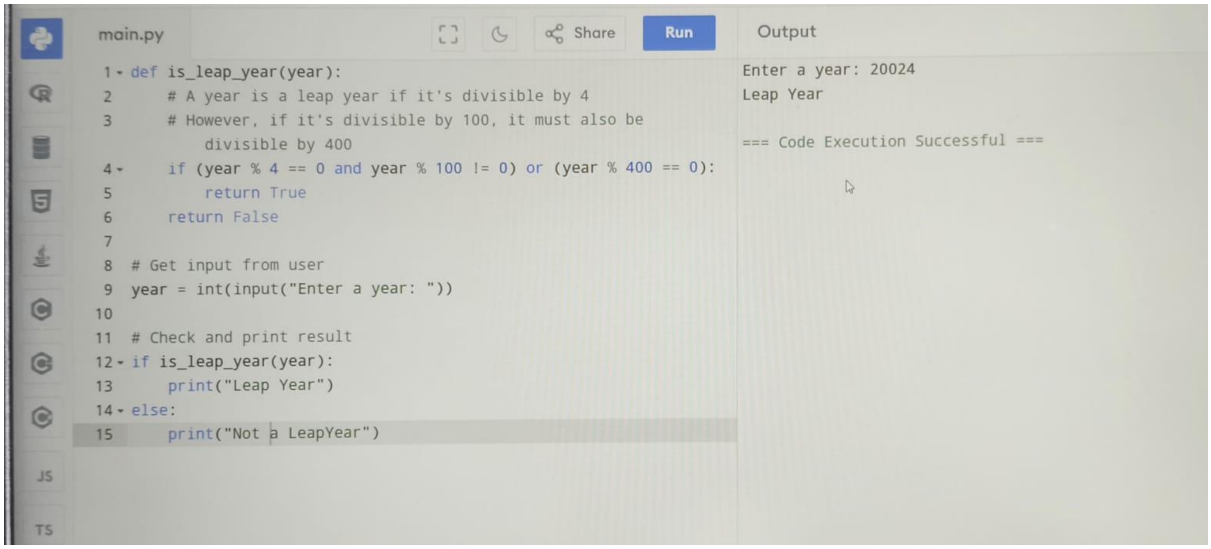
A=7

B=5



5. Write a program to find whether it is leap year or not?

Sample Input: 2000



6 Write a program to find out the duplicate array

Sample Input: array={1,2,3,4,1}

```python
def find_duplicates(arr):
    # Convert array to set to remove duplicates, then back to list
    unique_arr = list(dict.fromkeys(arr))
    return unique_arr

# Sample input
array = [1, 2, 3, 4, 1]

# Find duplicates and get unique array
result = find_duplicates(array)

# Print output in specified format
print(f"duplicate array=:{', '.join(map(str,result))}:")
```

```
duplicate array=:1, 2, 3, 4:

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

7 .Check whether the number is positive or negative

Sample Input:23

```python
number = 23
if number > 0:
    print("positive")
elif number < 0:
    print("negative")
else:
    print("zero")
```

```
positive

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

8 .Write a python program to find the average of mean median mode

Sample Input: [12,45,83,52]/4

```python
from statistics import mean, median, mode
def average_of_stats(numbers):
    mean_val = mean(numbers)
    median_val = median(numbers)
    try:
        mode_val = mode(numbers)
    except:
        mode_val = mean
    result = (mean_val + median_val + mode_val) / 3

    return round(result)
input_list = [12, 45, 83, 521/4]
result = average_of_stats(input_list)
print(f"Output:{result}")
```

```
Output:48

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

9. Write a python program to store the arrays in non-increasing order

Sample Input:[1,8,3,4,0]

```python
def sort_non_increasing(arr):
    # Sort array in descending order
    return sorted(arr, reverse=True)

# Sample input
input_array = [1, 8, 3, 4, 0]

# Get sorted array
result = sort_non_increasing(input_array)

# Print output
print(result)  # Output:[8,4,3,1,0]
```

```
[8, 4, 3, 1, 0]

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

10. Write a Python Program to Intersecting an elements

Sample Input:

(2,3,4,5)

(3,4,8,6)

main.py

Share Run

```python
# Function to find intersection of two tuples
def find_intersection(tuple1, tuple2):
    set1 = set(tuple1)
    set2 = set(tuple2)
    intersection = set1.intersection(set2)
    return tuple(intersection)
input1 = input("Enter first tuple (e.g., (2,3,4,5)): ")
input2 = input("Enter second tuple (e.g., (3,4,8,6)): ")
tuple1 = eval(input1)
tuple2 = eval(input2)
result = find_intersection(tuple1, tuple2)
print("Output:",result)
```

Output

```
Enter first tuple (e.g., (2,3,4,5)): (2,3,4,5)
Enter second tuple (e.g., (3,4,8,6)): (3,4,8,6)
Output: (3, 4)

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