


```
main.py
1 # Get input from user
2 number = float(input("Enter a decimal number: "))
3
4 # Calculate square and cube
5 square = number ** 2
6 cube = number ** 3
7
8 # Print results
9 print(f"Given Number: {number}")
10 print(f"Square Number: {square}")
11 print(f"Cube Number: {cube}")
```

Output

```
Enter a decimal number: 0.1
Given Number: 0.1
Square Number: 0.010000000000000002
Cube Number: 0.0010000000000000002

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

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

Sample Input:

Enter the Character to be printed: +

```
main.py
1 # Get input from user
2 char = input("Enter the Character to be printed: ")
3 rows = int(input("Number of rows: "))
4
5 # Print the pattern
6 for i in range(1, rows + 1):
7     print(char*i)
```

Output

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

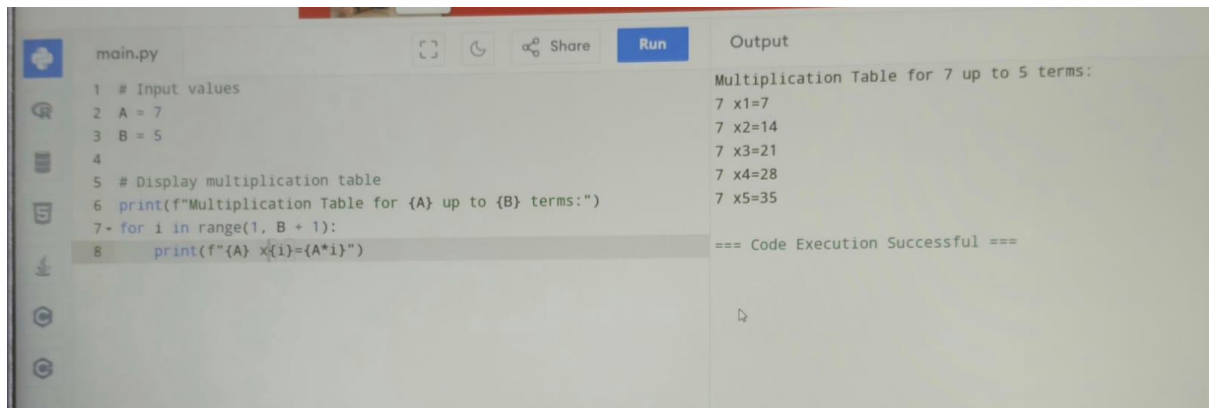
=== Code Execution Successful ===
```

4. Python Program to Display the Multiplication Table

Sample Input:

A=7

B=5



The screenshot shows a Python IDE with a file named 'main.py'. The code defines variables A=7 and B=5, then uses a for loop to print a multiplication table for A up to B terms. The output window shows the resulting table and a success message.

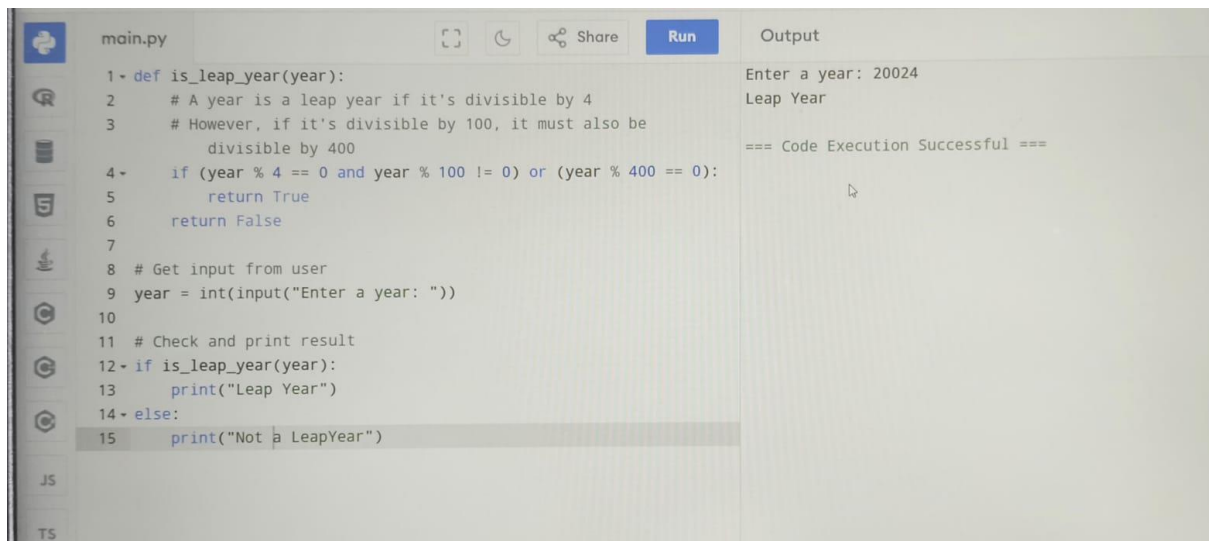
```
main.py
1 # Input values
2 A = 7
3 B = 5
4
5 # Display multiplication table
6 print(f"Multiplication Table for {A} up to {B} terms:")
7 for i in range(1, B + 1):
8     print(f"{A} x {i} = {A*i}")
```

Output

```
Multiplication Table for 7 up to 5 terms:
7 x1=7
7 x2=14
7 x3=21
7 x4=28
7 x5=35
=== Code Execution Successful ===
```

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

Sample Input: 2000



The screenshot shows a Python IDE with a file named 'main.py'. The code defines a function 'is_leap_year' that checks if a year is a leap year based on divisibility rules. It then takes user input and prints the result. The output window shows the input '20024' and the result 'Leap Year'.

```
main.py
1 def is_leap_year(year):
2     # A year is a leap year if it's divisible by 4
3     # However, if it's divisible by 100, it must also be
4     # divisible by 400
5     if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
6         return True
7     return False
8
9 # Get input from user
10 year = int(input("Enter a year: "))
11
12 # Check and print result
13 if is_leap_year(year):
14     print("Leap Year")
15 else:
16     print("Not a LeapYear")
```

Output

```
Enter a year: 20024
Leap Year
=== Code Execution Successful ===
```

6 Write a program to find out the duplicate array

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

The screenshot shows a Python IDE with a file named `main.py`. The code defines a function `find_duplicates(arr)` that converts the array to a set to remove duplicates and then back to a list. A sample input array `[1, 2, 3, 4, 1]` is provided, and the program prints the unique array. The output shows the unique array as `1, 2, 3, 4` and a success message.

```
1 def find_duplicates(arr):
2     # Convert array to set to remove duplicates, then back to
    list
3     unique_arr = list(dict.fromkeys(arr))
4     return unique_arr
5
6 # Sample input
7 array = [1, 2, 3, 4, 1]
8
9 # Find duplicates and get unique array
10 result = find_duplicates(array)
11
12 # Print output in specified format
13 print(f"duplicate array={', '.join(map(str,result))}")
```

Output: duplicate array=1, 2, 3, 4:
=== Code Execution Successful ===

7 .Check whether the number is positive or negative

Sample Input:23

The screenshot shows a Python IDE with a file named `main.py`. The code checks if a number is greater than 0 (positive), less than 0 (negative), or equal to 0 (zero). The sample input is 23, and the output is "positive".

```
1 number = 23
2 if number > 0:
3     print("positive")
4 elif number < 0:
5     print("negative")
6 else:
7     print("zero")
```

Output: positive
=== Code Execution Successful ===

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

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

The screenshot shows a Python IDE with a file named `main.py`. The code defines a function `average_of_stats` that takes a list of numbers and returns the average of its mean, median, and mode. The mode is handled with a try-except block to catch `StatisticsError`. The input list is `[12, 45, 83, 521/4]`, and the output is printed as `Output:48`.

```
1 from statistics import mean, median, mode
2 def average_of_stats(numbers):
3     mean_val = mean(numbers)
4     median_val = median(numbers)
5     try:
6         mode_val = mode(numbers)
7     except:
8         mode_val = mean
9     result = (mean_val + median_val + mode_val) / 3
10
11     return round(result)
12 input_list = [12, 45, 83, 521/4]
13 result = average_of_stats(input_list)
14 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]

The screenshot shows a Python IDE with a file named `main.py`. The code defines a function `sort_non_increasing` that sorts an array in descending order using `sorted(arr, reverse=True)`. The input array is `[1, 8, 3, 4, 0]`, and the output is printed as `Output:[8,4,3,1,0]`.

```
1 def sort_non_increasing(arr):
2     # Sort array in descending order
3     return sorted(arr, reverse=True)
4
5 # Sample input
6 input_array = [1, 8, 3, 4, 0]
7
8 # Get sorted array
9 result = sort_non_increasing(input_array)
10
11 # Print output
12 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

```
1 # Function to find intersection of two tuples
2 def find_intersection(tuple1, tuple2):
3     set1 = set(tuple1)
4     set2 = set(tuple2)
5     intersection = set1.intersection(set2)
6     return tuple(intersection)
7 input1 = input("Enter first tuple (e.g., (2,3,4,5)): ")
8 input2 = input("Enter second tuple (e.g., (3,4,8,6)): ")
9 tuple1 = eval(input1)
10 tuple2 = eval(input2)
11 result = find_intersection(tuple1, tuple2)
12 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 ===

JS

TS