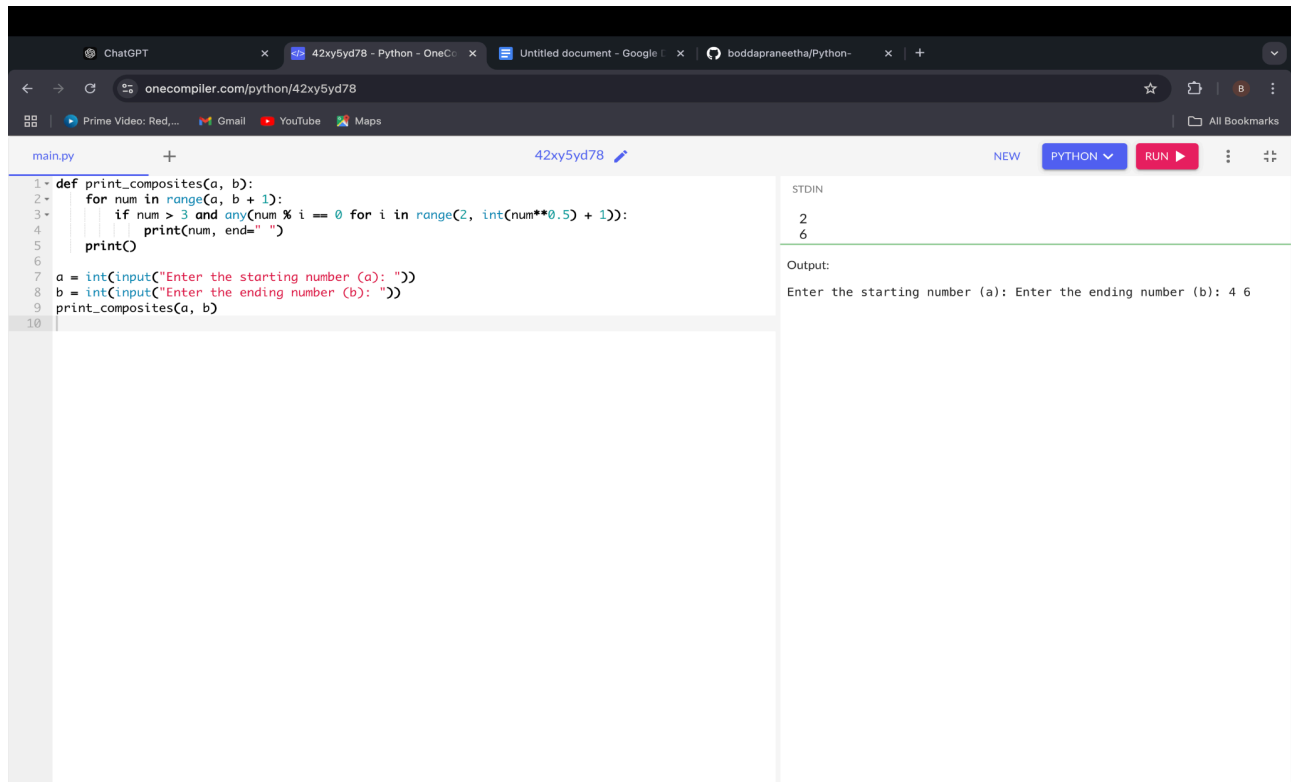


41. Write a program to print all the composite numbers between a and b?



The screenshot shows a web browser with the URL `onecompiler.com/python/42xy5yd78`. The code editor contains the following Python code:

```
1 def print_composites(a, b):
2     for num in range(a, b + 1):
3         if num > 3 and any(num % i == 0 for i in range(2, int(num**0.5) + 1)):
4             print(num, end=" ")
5     print()
6
7 a = int(input("Enter the starting number (a): "))
8 b = int(input("Enter the ending number (b): "))
9 print_composites(a, b)
10
```

The right-hand side of the interface shows the execution results:

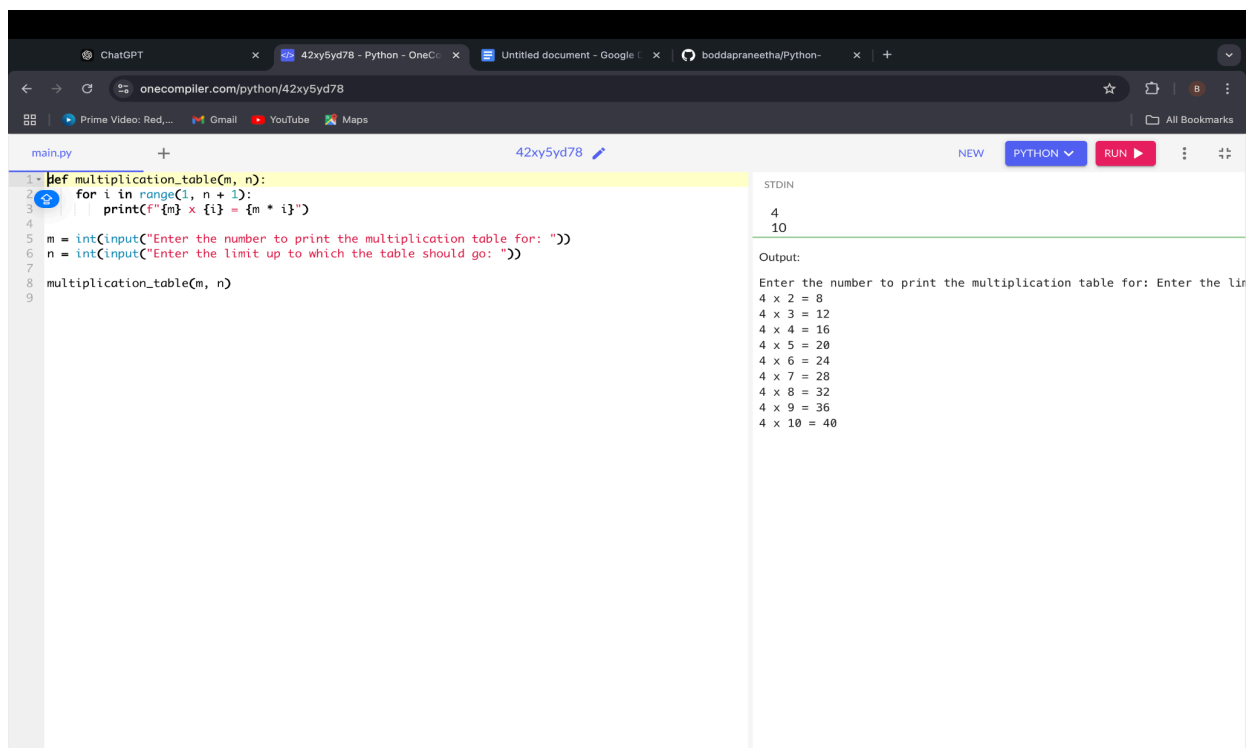
STDIN

```
2
6
```

Output:

```
Enter the starting number (a): Enter the ending number (b): 4 6
```

42. Write a program to print the multiplication table of number m up to n.



The screenshot shows the same online Python compiler with a new program. The code editor contains the following Python code:

```
1 def multiplication_table(m, n):
2     for i in range(1, n + 1):
3         print(f"{m} x {i} = {m * i}")
4
5 m = int(input("Enter the number to print the multiplication table for: "))
6 n = int(input("Enter the limit up to which the table should go: "))
7
8 multiplication_table(m, n)
9
```

The right-hand side of the interface shows the execution results:

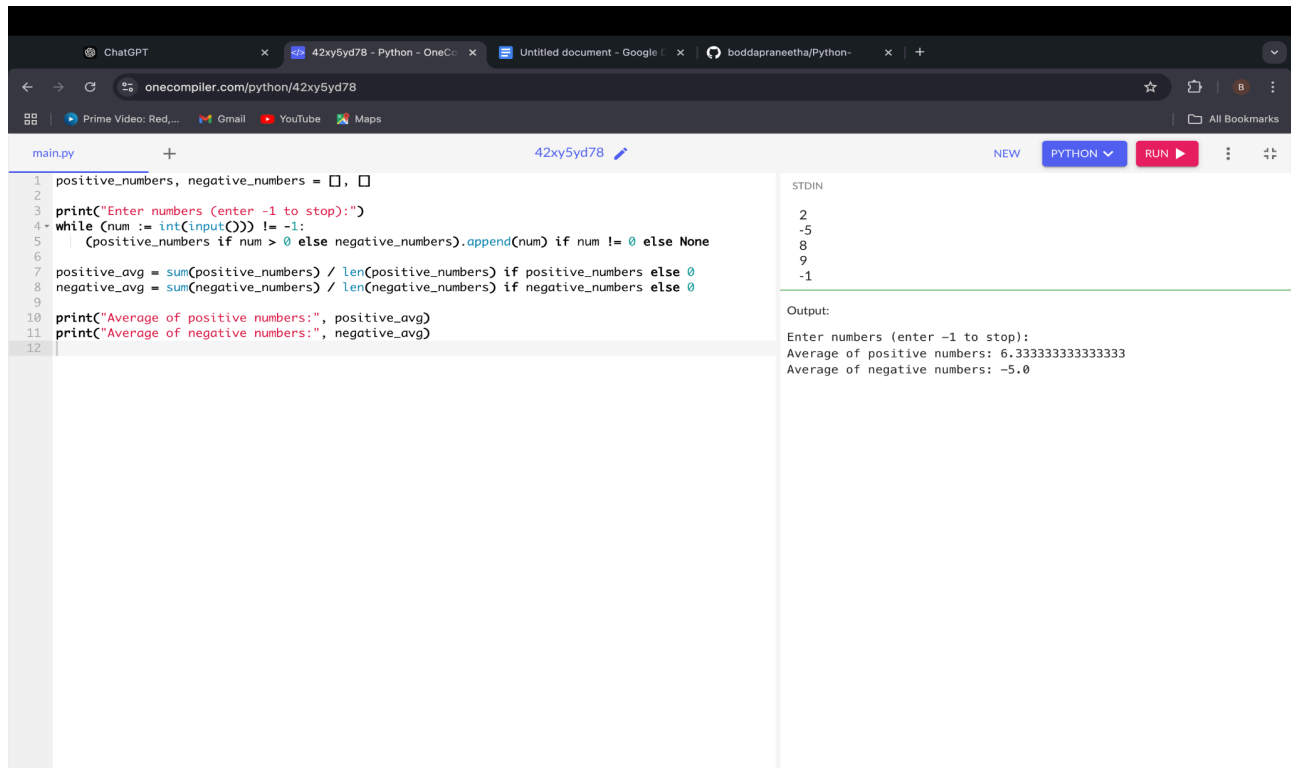
STDIN

```
4
10
```

Output:

```
Enter the number to print the multiplication table for: Enter the limit up to which the table should go:
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40
```

43. Write a program to read the numbers until -1 is encountered. Find the average of positive numbers and negative numbers entered by user



The screenshot shows a web browser with the URL `onecompiler.com/python/42xy5yd78`. The code editor contains a Python script named `main.py` that reads numbers from the user until -1 is entered. It calculates the average of positive and negative numbers. The output shows the averages for the input sequence 2, -5, 8, 9, -1.

```
1 positive_numbers, negative_numbers = [], []
2
3 print("Enter numbers (enter -1 to stop):")
4 while (num := int(input())) != -1:
5     (positive_numbers if num > 0 else negative_numbers).append(num) if num != 0 else None
6
7 positive_avg = sum(positive_numbers) / len(positive_numbers) if positive_numbers else 0
8 negative_avg = sum(negative_numbers) / len(negative_numbers) if negative_numbers else 0
9
10 print("Average of positive numbers:", positive_avg)
11 print("Average of negative numbers:", negative_avg)
12
```

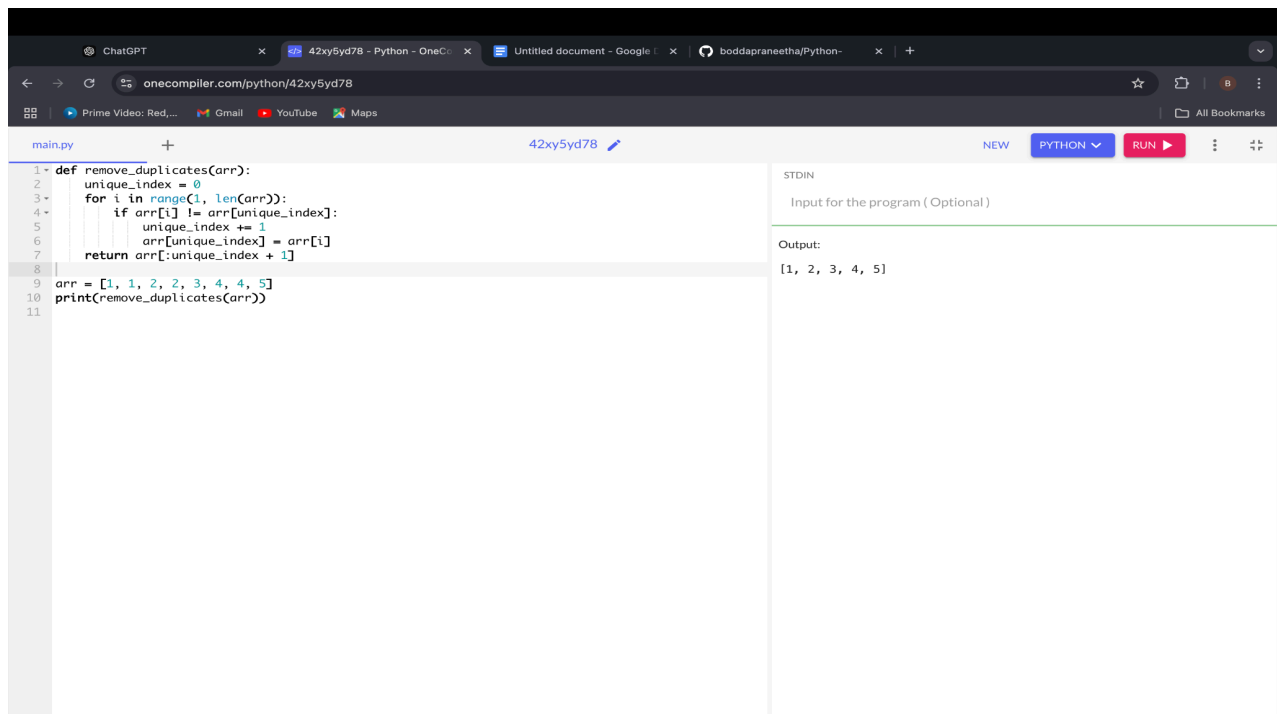
STDIN

```
2
-5
8
9
-1
```

Output:

```
Enter numbers (enter -1 to stop):
Average of positive numbers: 6.333333333333333
Average of negative numbers: -5.0
```

44. Program to remove duplicates from the sorted array



The screenshot shows a web browser with the URL `onecompiler.com/python/42xy5yd78`. The code editor contains a Python script named `main.py` that defines a function `remove_duplicates` to remove duplicates from a sorted array. The output shows the result for the input array `[1, 1, 2, 2, 3, 4, 4, 5]`.

```
1 def remove_duplicates(arr):
2     unique_index = 0
3     for i in range(1, len(arr)):
4         if arr[i] != arr[unique_index]:
5             unique_index += 1
6             arr[unique_index] = arr[i]
7     return arr[:unique_index + 1]
8
9 arr = [1, 1, 2, 2, 3, 4, 4, 5]
10 print(remove_duplicates(arr))
11
```

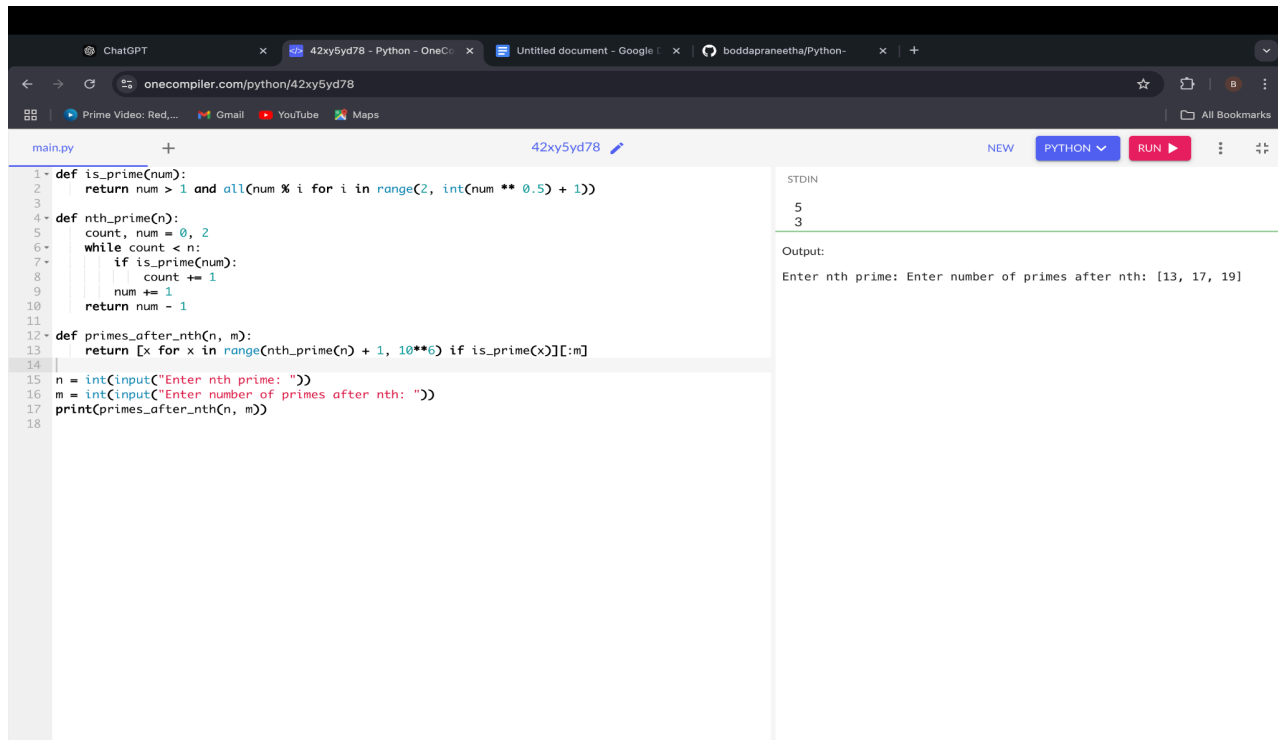
STDIN

```
Input for the program ( Optional )
```

Output:

```
[1, 2, 3, 4, 5]
```

45. Write a program to print n prime numbers after nth Prime number



The screenshot shows a web browser with the URL `oncompiler.com/python/42xy5yd78`. The code editor contains the following Python code:

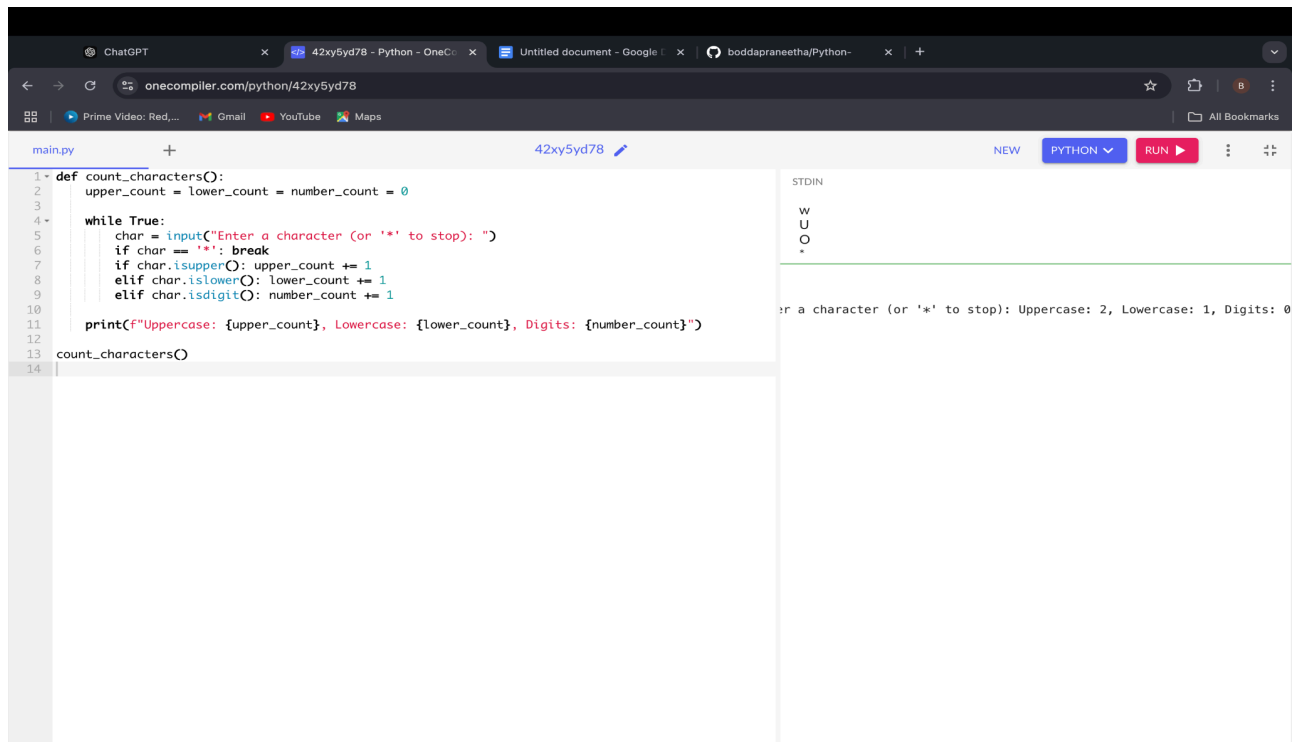
```
1- def is_prime(num):
2-     return num > 1 and all(num % i for i in range(2, int(num ** 0.5) + 1))
3-
4- def nth_prime(n):
5-     count, num = 0, 2
6-     while count < n:
7-         if is_prime(num):
8-             count += 1
9-         num += 1
10-    return num - 1
11-
12- def primes_after_nth(n, m):
13-     return [x for x in range(nth_prime(n) + 1, 10**6) if is_prime(x)][m:]
14-
15- n = int(input("Enter nth prime: "))
16- m = int(input("Enter number of primes after nth: "))
17- print(primes_after_nth(n, m))
18-
```

The output section shows the following text:

```
STDIN
5
3

Output:
Enter nth prime: Enter number of primes after nth: [13, 17, 19]
```

46. write a program to read a character until a * is encountered. Also count the number of uppercase, lowercase, and numbers entered by the users.



The screenshot shows a web browser with the URL `oncompiler.com/python/42xy5yd78`. The code editor contains the following Python code:

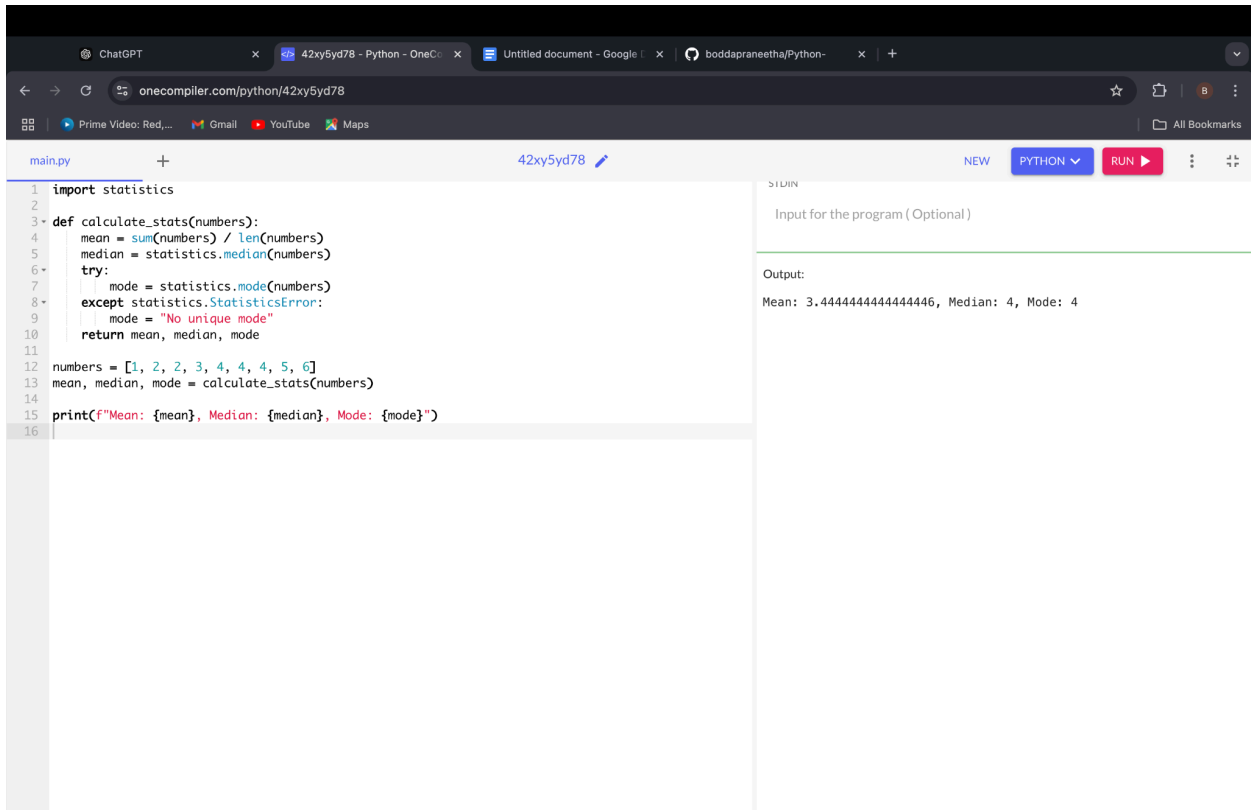
```
1- def count_characters():
2-     upper_count = lower_count = number_count = 0
3-
4-     while True:
5-         char = input("Enter a character (or '*' to stop): ")
6-         if char == '*': break
7-         if char.isupper(): upper_count += 1
8-         elif char.islower(): lower_count += 1
9-         elif char.isdigit(): number_count += 1
10-
11-     print(f"Uppercase: {upper_count}, Lowercase: {lower_count}, Digits: {number_count}")
12-
13- count_characters()
14-
```

The output section shows the following text:

```
STDIN
W
U
O
*

Enter a character (or '*' to stop): Uppercase: 2, Lowercase: 1, Digits: 0
```

47. write a program to Find the Mean, Median, Mode of the array of numbers?

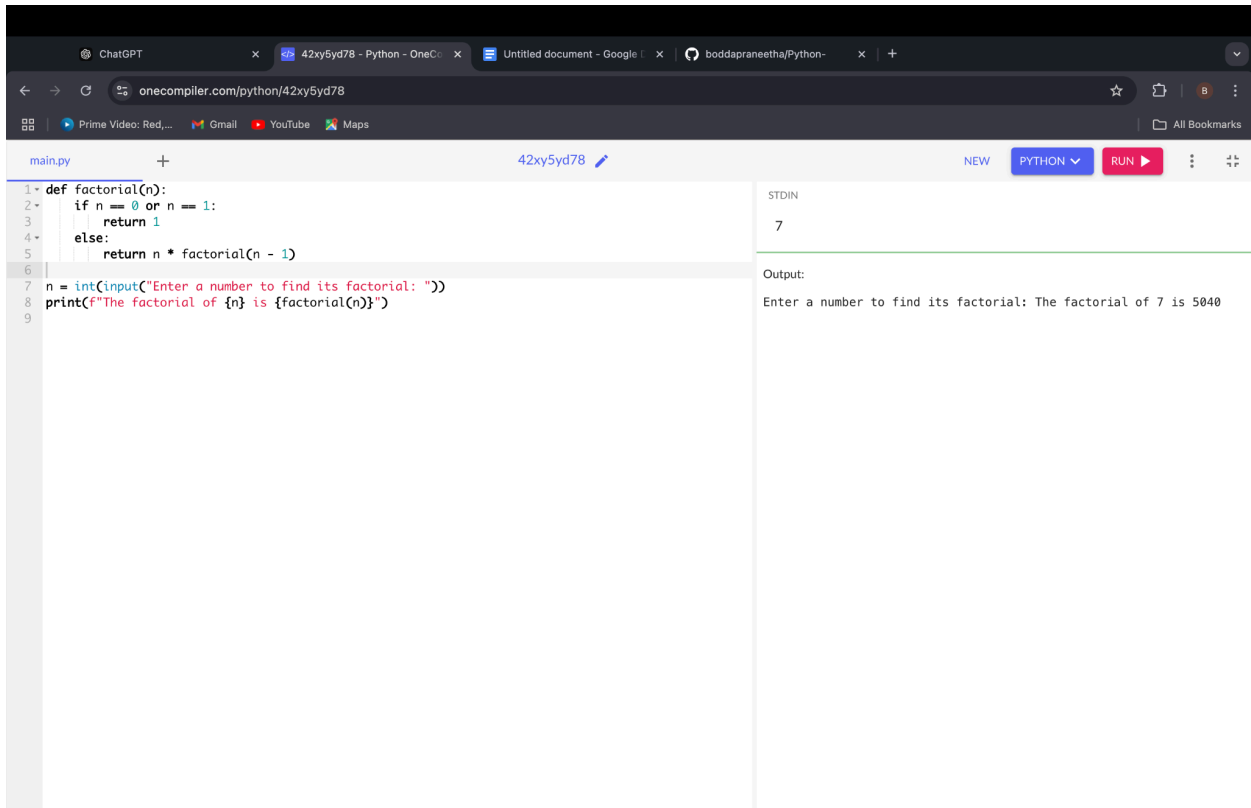


The screenshot shows a web browser with the URL `onecompiler.com/python/42xy5yd78`. The IDE interface includes a file explorer with `main.py`, a code editor, and a console. The code in `main.py` imports the `statistics` module and defines a function `calculate_stats` that takes a list of numbers and returns the mean, median, and mode. The mode is calculated using a try-except block to handle `StatisticsError`. The main program defines a list `numbers = [1, 2, 2, 3, 4, 4, 4, 5, 6]` and prints the results.

```
1 import statistics
2
3 def calculate_stats(numbers):
4     mean = sum(numbers) / len(numbers)
5     median = statistics.median(numbers)
6     try:
7         mode = statistics.mode(numbers)
8     except statistics.StatisticsError:
9         mode = "No unique mode"
10    return mean, median, mode
11
12 numbers = [1, 2, 2, 3, 4, 4, 4, 5, 6]
13 mean, median, mode = calculate_stats(numbers)
14
15 print(f"Mean: {mean}, Median: {median}, Mode: {mode}")
16
```

The console output shows: `Mean: 3.4444444444444446, Median: 4, Mode: 4`.

48. write a program to Find the factorial of n?

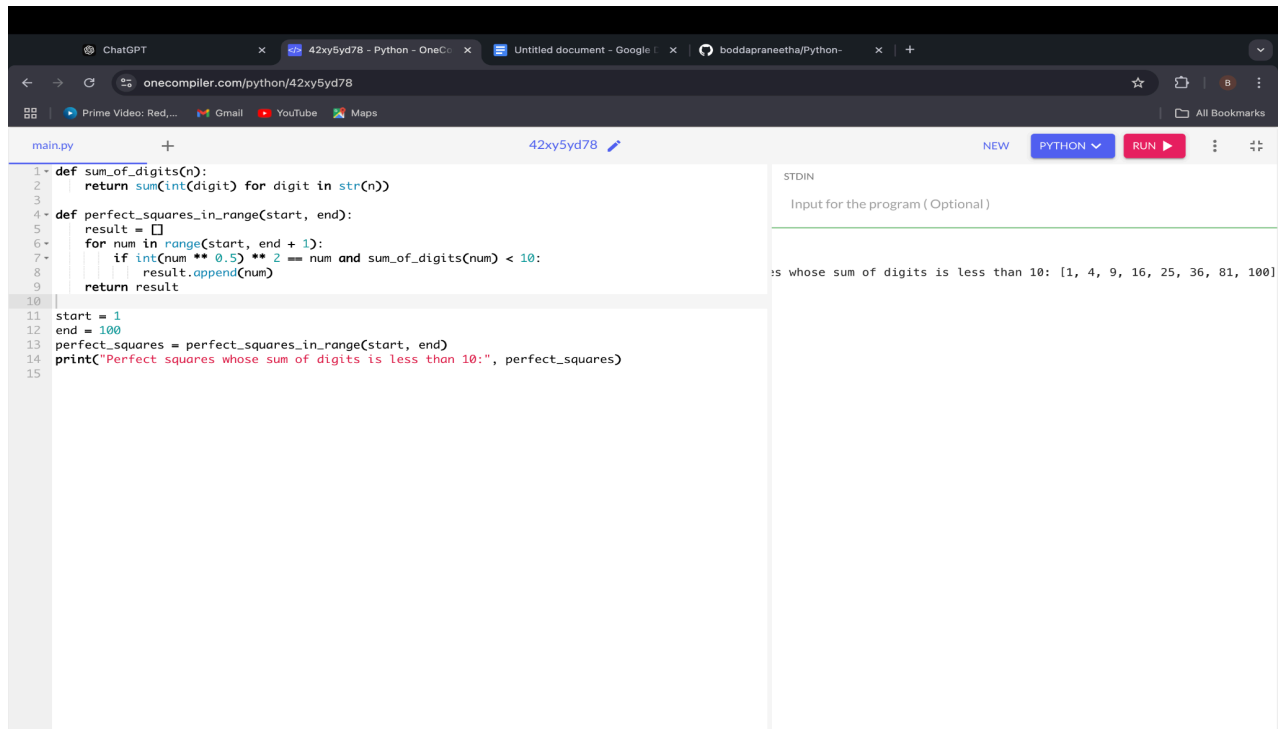


The screenshot shows the same web-based Python IDE with a new program to calculate the factorial of a number `n`. The code defines a recursive function `factorial` that returns 1 for `n == 0` or `n == 1`, and `n * factorial(n - 1)` otherwise. The main program prompts the user to enter a number and prints the factorial result.

```
1 def factorial(n):
2     if n == 0 or n == 1:
3         return 1
4     else:
5         return n * factorial(n - 1)
6
7 n = int(input("Enter a number to find its factorial: "))
8 print(f"The factorial of {n} is {factorial(n)}")
9
```

The console shows the input `7` and the output `Enter a number to find its factorial: The factorial of 7 is 5040`.

49. Write a Python Program to create a list of all numbers in a range which are perfect squares and the sum of the digits of the number is less than 10



The screenshot shows a web browser with the OneCompiler Python IDE. The code in the editor is as follows:

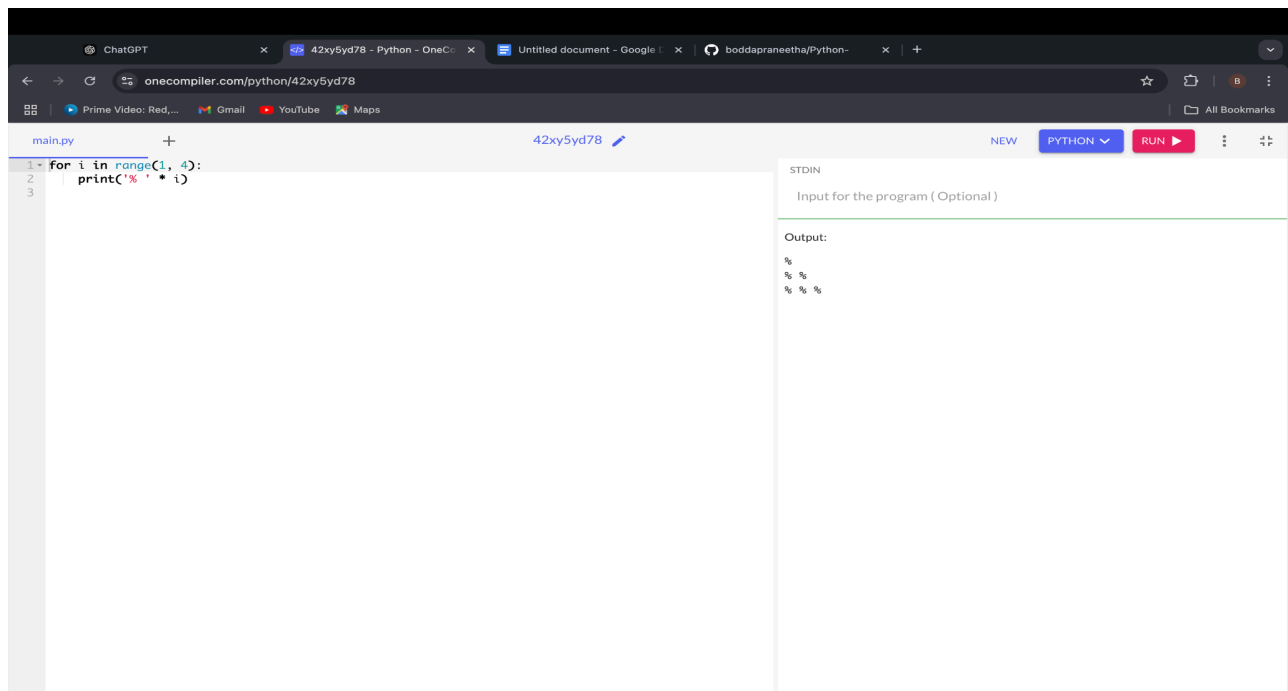
```
1 def sum_of_digits(n):
2     return sum(int(digit) for digit in str(n))
3
4 def perfect_squares_in_range(start, end):
5     result = []
6     for num in range(start, end + 1):
7         if int(num ** 0.5) ** 2 == num and sum_of_digits(num) < 10:
8             result.append(num)
9     return result
10
11 start = 1
12 end = 100
13 perfect_squares = perfect_squares_in_range(start, end)
14 print("Perfect squares whose sum of digits is less than 10:", perfect_squares)
15
```

The output on the right side of the IDE shows the result of the program:

```
Perfect squares whose sum of digits is less than 10: [1, 4, 9, 16, 25, 36, 81, 100]
```

50. Write a program to print the following pattern

```
%
% %
% % %
```



The screenshot shows a web browser with the OneCompiler Python IDE. The code in the editor is as follows:

```
1 for i in range(1, 4):
2     print('% * * i)
```

The output on the right side of the IDE shows the result of the program:

```
%
% %
% % %
```