test.py > ...

```python
def sum_of_evens(numbers):
    even_sum = 0
    for number in numbers:
        if number % 2 == 0:
            even_sum += number
    return even_sum
```

```python
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, n):
        if n % i == 0:
            return False
    return True

num = int(input("Enter a number: "))
if is_prime(num):
    print(f"{num} is a prime number")
else:
    print(f"{num} is not a prime number")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
ya/test.py
Enter a number: 3
3 is a prime number
```

test.py > ...

```python
def find_min_max(numbers):
    min_num = numbers[0]
    max_num = numbers[0]
    for num in numbers:
        if num < min_num:
            min_num = num
        if num > max_num:
            max_num = num
    return (min_num, max_num)

numbers = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
print("The list of numbers is: ", numbers)
min_num, max_num = find_min_max(numbers)
print("The smallest number is: ", min_num)
print("The largest number is: ", max_num)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
The list of numbers is:  [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
The smallest number is:  10
The largest number is:  100
PS D:\samarthya>
```

```python
def count_vowels_consonants(input_string):
    vowels = "aeiouAEIOU"
    vowels_count = 0
    consonants_count = 0

    for char in input_string:
        if char in vowels:
            vowels_count += 1
        elif char.isalpha():
            sentants_count += 1

    return {"vowels": vowels_count, "consonants": consonants_count}
```

```python
def reverse_string(s):
    result = ""
    for char in s:
        result = char + result
    return result

input_string = "hello"
print(reverse_string(input_string))
```

```python
from collections import Counter

def most_common_element(string):
    # Create a dictionary to store the count of each character in the string
    char_counts = Counter(string)

    # Find the character with the highest count
    most_common = char_counts.most_common(1)

    # Return the character with the highest count
    return most_common[0][0]

# Test the program with a sample string
string = "hello world"
print(most_common_element(string))
```

```python
def is_palindrome(string):
    string = string.lower() # Convert the string to lowercase
    string = ''.join(e for e in string if e.isalnum()) # Remove any non-alphanumeric charact
    return string == string[::-1] # Check if the string is equal to its reverse

# Test the function
print(is_palindrome("A man, a plan, a canal, Panama!")) # Output: True
print(is_palindrome("Hello, World!")) # Output: False
```

```python
def fibonacci(n):
    if n <= 0:
        return []
    elif n == 1:
        return [0]
    elif n == 2:
        return [0, 1]
    else:
        fib_list = [0, 1]
        for i in range(2, n):
            next_num = fib_list[i - 1] + fib_list[i - 2]
            fib_list.append(next_num)
        return fib_list

n = int(input("Enter the number of Fibonacci numbers to generate: "))
print("The first", n, "Fibonacci numbers are:", fibonacci(n))
```

test.py > ...

```python
def common_elements(list1, list2):
    return [element for element in list1 if element in list2]
list1 = [1, 2, 3, 4, 5]
list2 = [4, 5, 6, 7, 8]

common = common_elements(list1, list2)
print(common)
```

Welcome | test.py | A.txt

test.py > ...

```python
def factorial_recursion(n):
    if n == 1:
        return 1
    else:
        return n * factorial_recursion(n-1)

num = int(input("Enter a number: "))
print("The factorial of", num, "is", factorial_recursion(num))
def factorial_iteration(n):
    result = 1
    for i in range(1, n+1):
        result = result * i
    return result

num = int(input("Enter a number: "))
print("The factorial of", num, "is", factorial_iteration(num))
```

```python
def remove_spaces(input_string):
    result = ""
    for char in input_string:
        if char != " ":
            result += char
    return result
```

test.py > ...

```python
def is_leap_year(year):
    if year % 400 == 0:
        return True
    elif year % 100 == 0:
        return False
    elif year % 4 == 0:
        return True
    else:
        return False

year = int(input("Enter a year: "))
if is_leap_year(year):
    print(year, "is a leap year.")
else:
    print(year, "is not a leap year.")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
ya/test.py
Enter a year: 2003
2003 is not a leap year.
PS D:\samarthya>
```

test.py > ...

```python
    numbers.sort()
    if len(numbers) % 2 == 0:
        m1 = numbers[len(numbers)//2-1]
        m2 = numbers[len(numbers)//2]
        m = (m1 + m2) / 2
    else:
        m = numbers[len(numbers)//2]
    return m

if __name__ == '__main__':
    numbers = [int(x) for x in input("Enter numbers separated by spac
    mean_result = mean(numbers)
    median_result = median(numbers)
    print("Mean: ", mean_result)
    print("Median: ", median_result)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Enter numbers separated by space: 2 3 4 5
Mean:  3.5
Median:  3.5
PS D:\samarthya>
```

test.py > ...

```python
def letter_frequency(s):
    frequency = {}
    for letter in s:
        if letter.isalpha():
            if letter in frequency:
                frequency[letter] += 1
            else:
                frequency[letter] = 1
    return frequency
```

```python
def get_primes(n):
    primes = []
    for num in range(2, n + 1):
        is_prime = True
        for i in range(2, int(num ** 0.5) + 1):
            if num % i == 0:
                is_prime = False
                break
        if is_prime:
            primes.append(num)
    return primes
```

test.py > ...

```python
def decimal_to_other_bases(decimal_num):
    binary_num = bin(decimal_num)
    octal_num = oct(decimal_num)
    hexadecimal_num = hex(decimal_num)
    return binary_num, octal_num, hexadecimal_num

decimal_num = int(input("Enter a decimal number: "))
binary_num, octal_num, hexadecimal_num = decimal_to_other_bases(decimal_num)
print(f"Binary: {binary_num}, Octal: {octal_num}, Hexadecimal: {hexadecimal_num}")
```

Welcome | test.py | A.txt

test.py > ...

```python
def length_of_longest_substring(s):
    char_set = set()
    max_length, start = 0, 0
    for i, char in enumerate(s):
        while char in char_set:
            char_set.remove(s[start])
            start += 1
        char_set.add(char)
        max_length = max(max_length, i - start + 1)
    return max_length
```

test.py > ...

```python
def union(list1, list2):
    return list(set(list1) | set(list2))

def intersection(list1, list2):
    return list(set(list1) & set(list2))

list1 = [1, 2, 3, 4, 5]
list2 = [4, 5, 6, 7, 8]

print("Union of the two lists:", union(list1, 
print("Intersection of the two lists:", inters

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
ya/test.py
Union of the two lists: [1, 2, 3, 4, 5, 6, 7, 8]
Intersection of the two lists: [4, 5]
PS D:\samarthya>
```

```python
def sum_and_product(lst):
    new_lst = []
    for tup in lst:
        new_tup = (sum(tup), reduce(lambda x, y: x * y, tup))
        new_lst.append(new_tup)
    return new_lst
```

test.py > ...

```python
def count_keys(dict_list):
    key_counts = {}
    for d in dict_list:
        for key in d:
            if key in key_counts:
                key_counts[key] += 1
            else:
                key_counts[key] = 1
    return key_counts
```