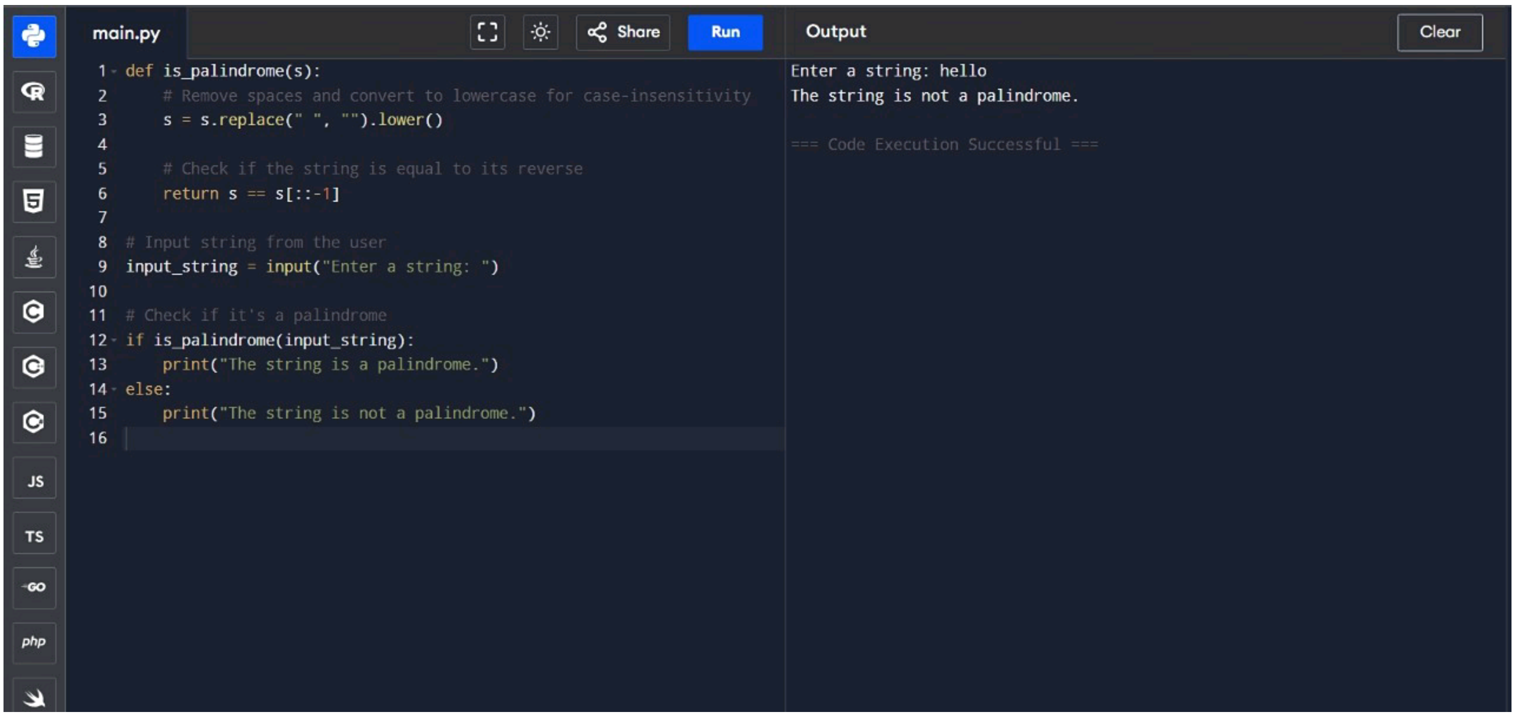
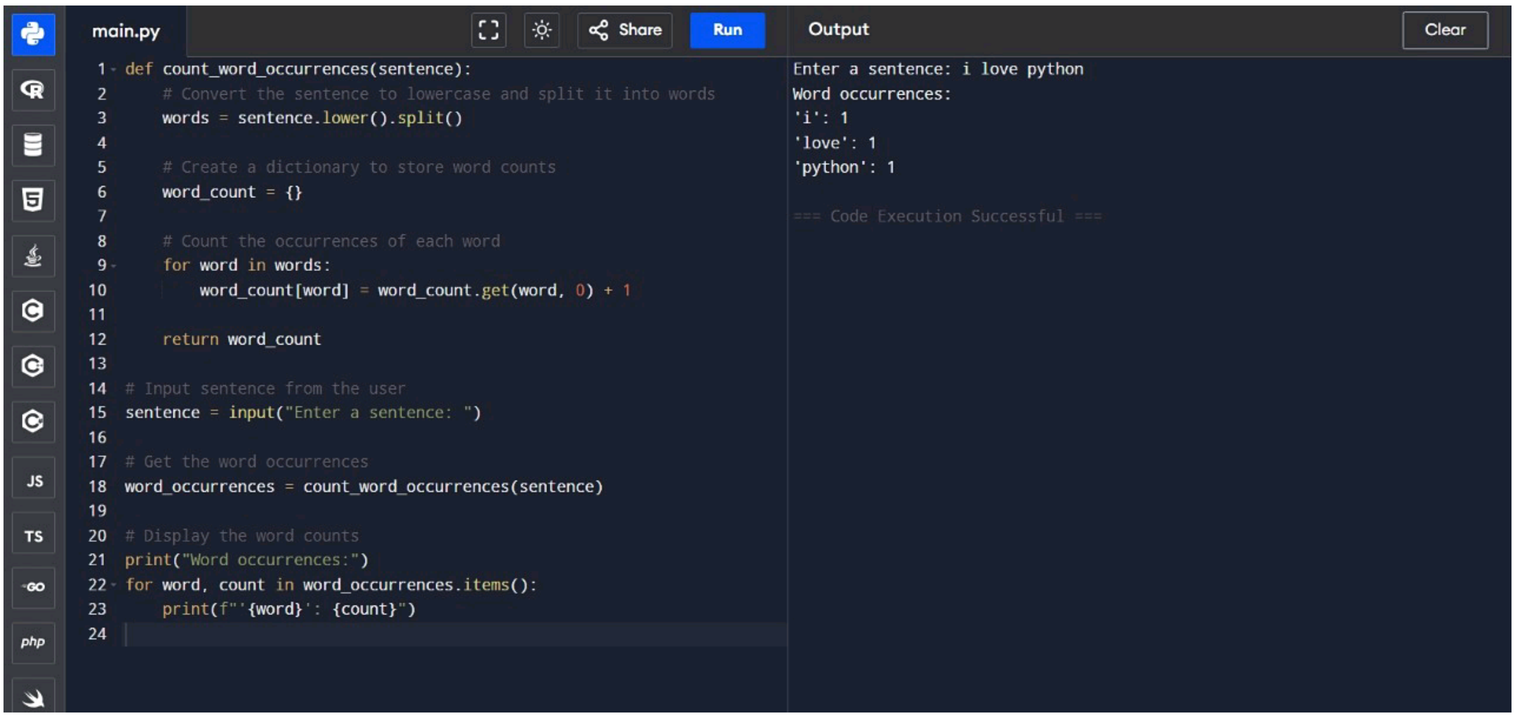


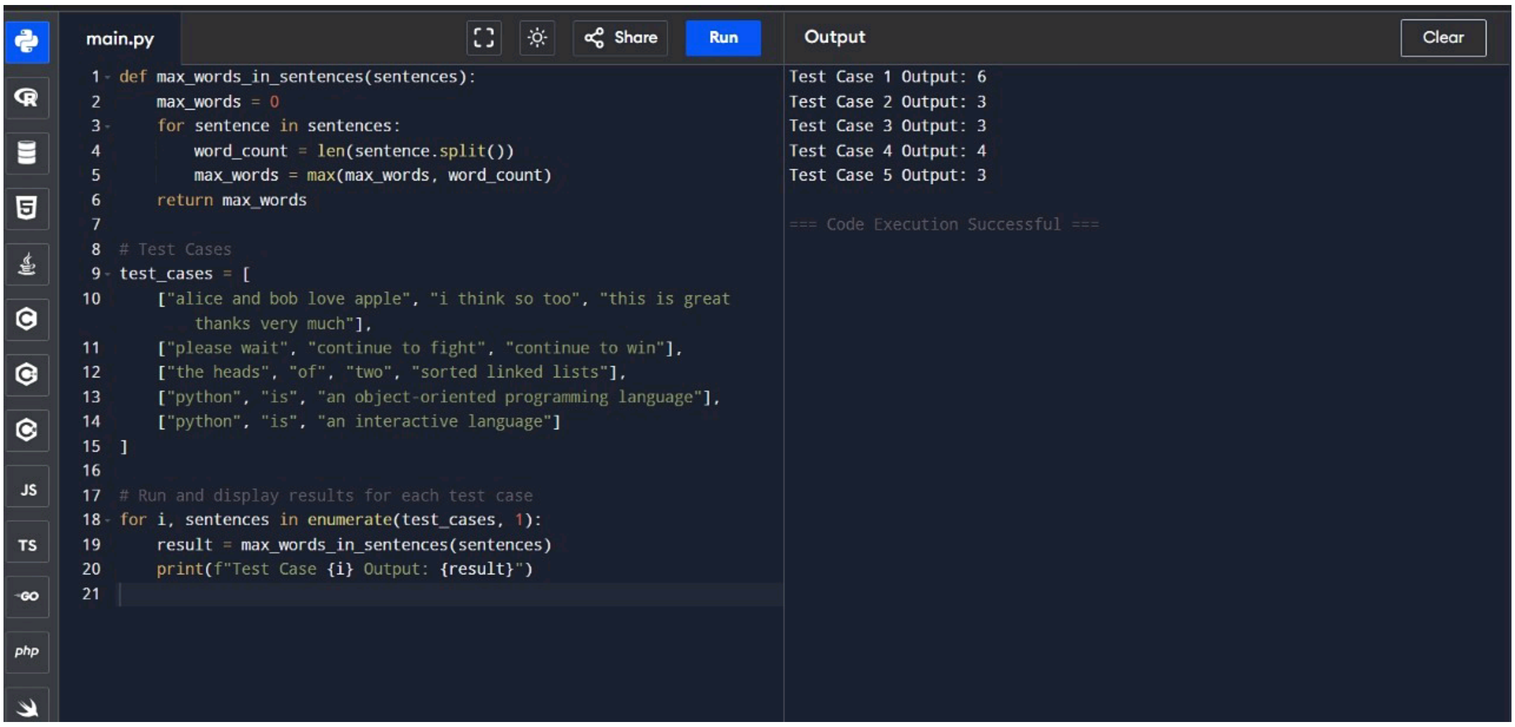
```python
def is_palindrome(s):
    # Remove spaces and convert to lowercase for case-insensitivity
    s = s.replace(" ", "").lower()

    # Check if the string is equal to its reverse
    return s == s[::-1]

# Input string from the user
input_string = input("Enter a string: ")

# Check if it's a palindrome
if is_palindrome(input_string):
    print("The string is a palindrome.")
else:
    print("The string is not a palindrome.")
```

```
Enter a string: hello
The string is not a palindrome.

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



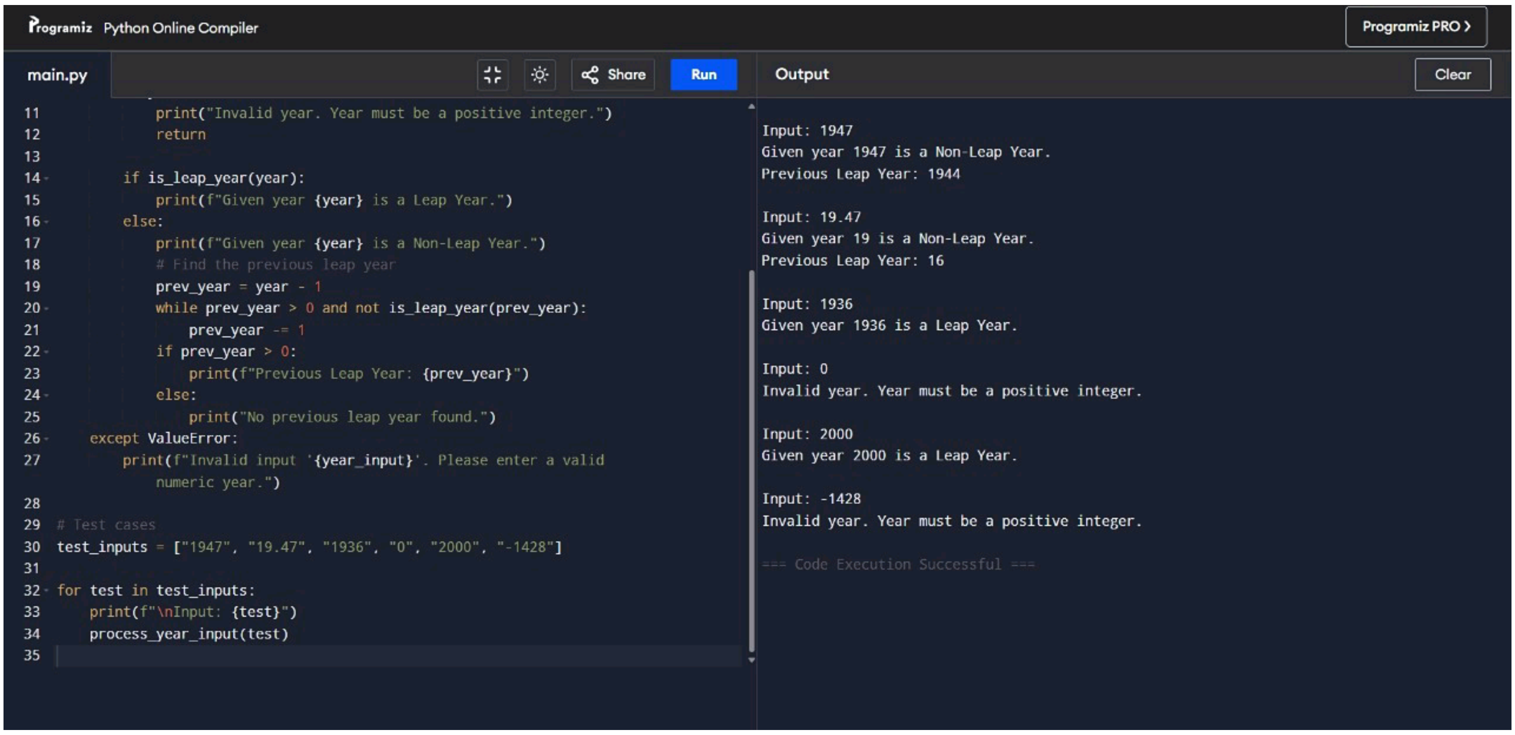
```python
def count_word_occurrences(sentence):
    # Convert the sentence to lowercase and split it into words
    words = sentence.lower().split()

    # Create a dictionary to store word counts
    word_count = {}

    # Count the occurrences of each word
    for word in words:
        word_count[word] = word_count.get(word, 0) + 1

    return word_count

# Input sentence from the user
sentence = input("Enter a sentence: ")

# Get the word occurrences
word_occurrences = count_word_occurrences(sentence)

# Display the word counts
print("Word occurrences:")
for word, count in word_occurrences.items():
    print(f"'{word}': {count}")

```

```
Enter a sentence: i love python
Word occurrences:
'i': 1
'love': 1
'python': 1

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



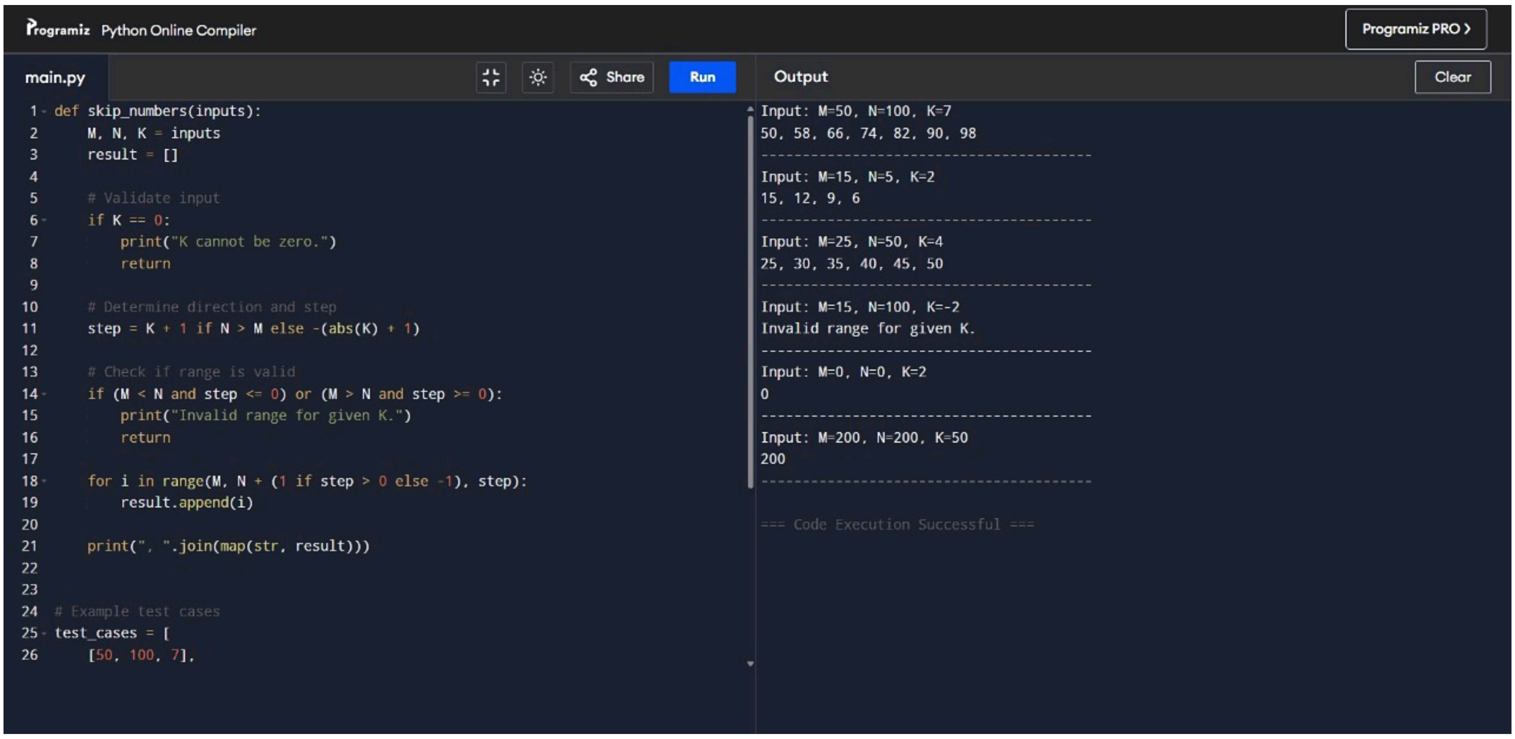
```python
def max_words_in_sentences(sentences):
    max_words = 0
    for sentence in sentences:
        word_count = len(sentence.split())
        max_words = max(max_words, word_count)
    return max_words

# Test Cases
test_cases = [
    ["alice and bob love apple", "i think so too", "this is great
        thanks very much"],
    ["please wait", "continue to fight", "continue to win"],
    ["the heads", "of", "two", "sorted linked lists"],
    ["python", "is", "an object-oriented programming language"],
    ["python", "is", "an interactive language"]
]

# Run and display results for each test case
for i, sentences in enumerate(test_cases, 1):
    result = max_words_in_sentences(sentences)
    print(f"Test Case {i} Output: {result}")
```

```
Test Case 1 Output: 6
Test Case 2 Output: 3
Test Case 3 Output: 3
Test Case 4 Output: 4
Test Case 5 Output: 3

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



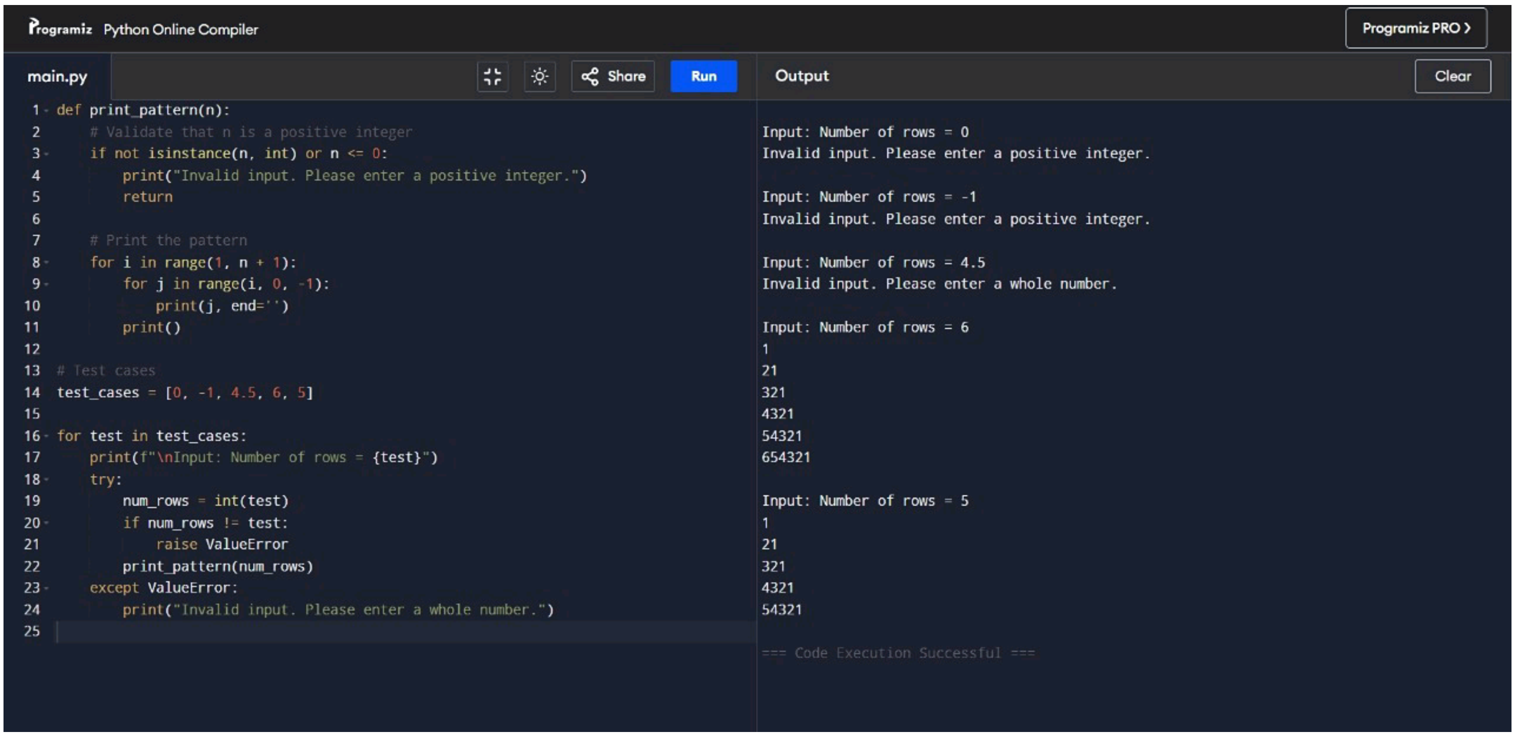
```python
            print("Invalid year. Year must be a positive integer.")
            return

        if is_leap_year(year):
            print(f"Given year {year} is a Leap Year.")
        else:
            print(f"Given year {year} is a Non-Leap Year.")
            # Find the previous leap year
            prev_year = year - 1
            while prev_year > 0 and not is_leap_year(prev_year):
                prev_year -= 1
            if prev_year > 0:
                print(f"Previous Leap Year: {prev_year}")
            else:
                print("No previous leap year found.")
    except ValueError:
        print(f"Invalid input '{year_input}'. Please enter a valid numeric year.")

# Test cases
test_inputs = ["1947", "19.47", "1936", "0", "2000", "-1428"]

for test in test_inputs:
    print(f"\nInput: {test}")
    process_year_input(test)
```

```
Input: 1947
Given year 1947 is a Non-Leap Year.
Previous Leap Year: 1944

Input: 19.47
Given year 19 is a Non-Leap Year.
Previous Leap Year: 16

Input: 1936
Given year 1936 is a Leap Year.

Input: 0
Invalid year. Year must be a positive integer.

Input: 2000
Given year 2000 is a Leap Year.

Input: -1428
Invalid year. Year must be a positive integer.

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



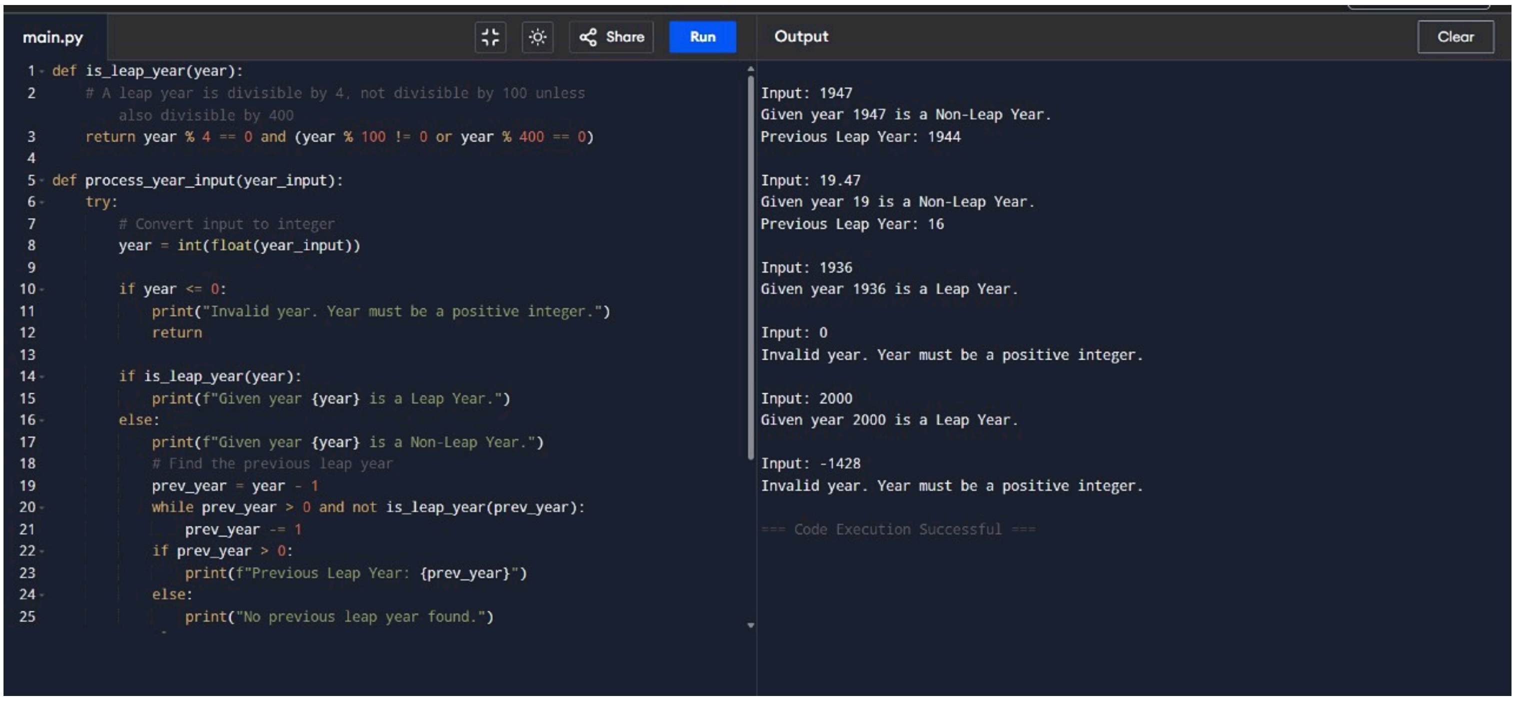
```python
def skip_numbers(inputs):
    M, N, K = inputs
    result = []

    # Validate input
    if K == 0:
        print("K cannot be zero.")
        return

    # Determine direction and step
    step = K + 1 if N > M else -(abs(K) + 1)

    # Check if range is valid
    if (M < N and step <= 0) or (M > N and step >= 0):
        print("Invalid range for given K.")
        return

    for i in range(M, N + (1 if step > 0 else -1), step):
        result.append(i)

    print(", ".join(map(str, result)))


# Example test cases
test_cases = [
    [50, 100, 7],
```

```
Input: M=50, N=100, K=7
50, 58, 66, 74, 82, 90, 98
----------------------------------------
Input: M=15, N=5, K=2
15, 12, 9, 6
----------------------------------------
Input: M=25, N=50, K=4
25, 30, 35, 40, 45, 50
----------------------------------------
Input: M=15, N=100, K=-2
Invalid range for given K.
----------------------------------------
Input: M=0, N=0, K=2
0
----------------------------------------
Input: M=200, N=200, K=50
200
----------------------------------------

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



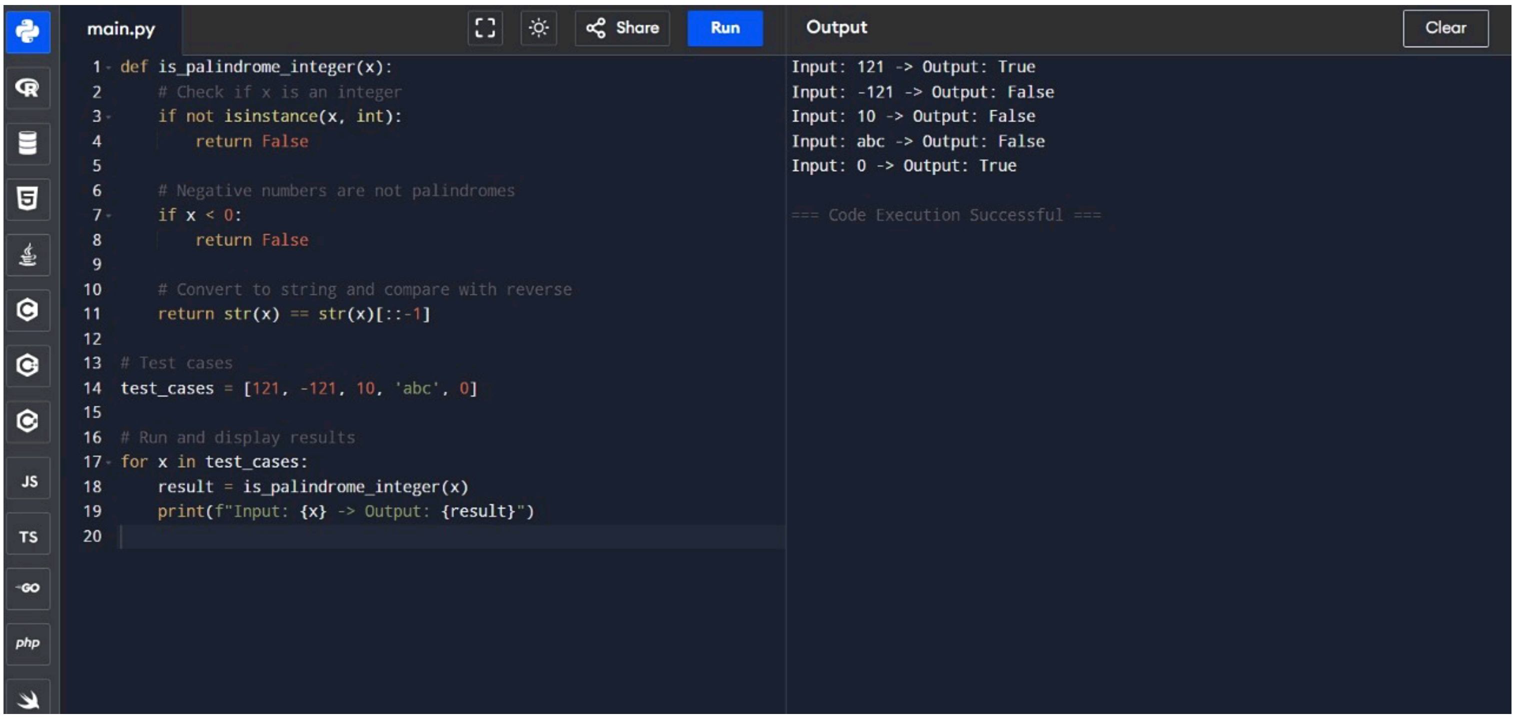
```python
def print_pattern(n):
    # Validate that n is a positive integer
    if not isinstance(n, int) or n <= 0:
        print("Invalid input. Please enter a positive integer.")
        return

    # Print the pattern
    for i in range(1, n + 1):
        for j in range(i, 0, -1):
            print(j, end='')
        print()

# Test cases
test_cases = [0, -1, 4.5, 6, 5]

for test in test_cases:
    print(f"\nInput: Number of rows = {test}")
    try:
        num_rows = int(test)
        if num_rows != test:
            raise ValueError
        print_pattern(num_rows)
    except ValueError:
        print("Invalid input. Please enter a whole number.")

```

```
Input: Number of rows = 0
Invalid input. Please enter a positive integer.

Input: Number of rows = -1
Invalid input. Please enter a positive integer.

Input: Number of rows = 4.5
Invalid input. Please enter a whole number.

Input: Number of rows = 6
1
21
321
4321
54321
654321

Input: Number of rows = 5
1
21
321
4321
54321

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



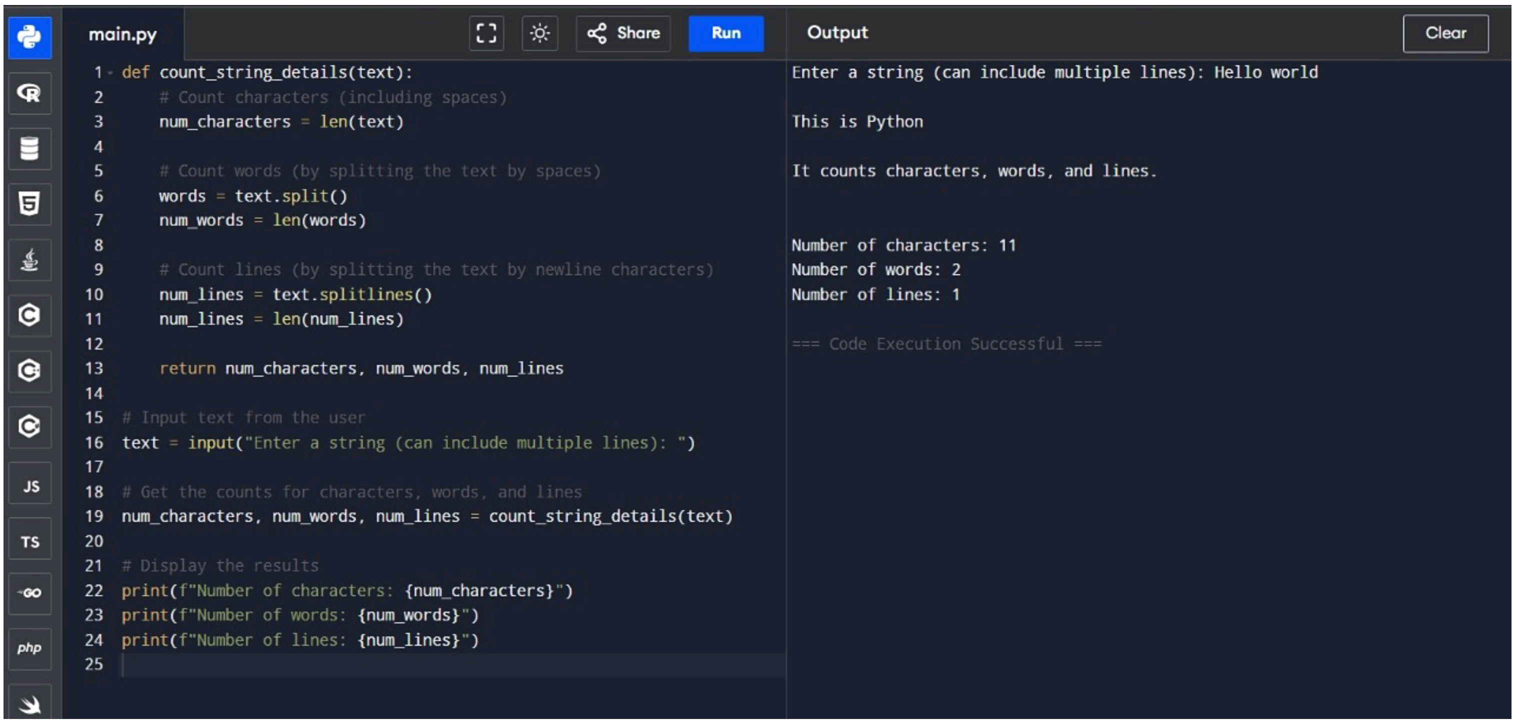
```python
def is_leap_year(year):
    # A leap year is divisible by 4, not divisible by 100 unless also divisible by 400
    return year % 4 == 0 and (year % 100 != 0 or year % 400 == 0)

def process_year_input(year_input):
    try:
        # Convert input to integer
        year = int(float(year_input))

        if year <= 0:
            print("Invalid year. Year must be a positive integer.")
            return

        if is_leap_year(year):
            print(f"Given year {year} is a Leap Year.")
        else:
            print(f"Given year {year} is a Non-Leap Year.")
            # Find the previous leap year
            prev_year = year - 1
            while prev_year > 0 and not is_leap_year(prev_year):
                prev_year -= 1
            if prev_year > 0:
                print(f"Previous Leap Year: {prev_year}")
            else:
                print("No previous leap year found.")
```

```
Input: 1947
Given year 1947 is a Non-Leap Year.
Previous Leap Year: 1944

Input: 19.47
Given year 19 is a Non-Leap Year.
Previous Leap Year: 16

Input: 1936
Given year 1936 is a Leap Year.

Input: 0
Invalid year. Year must be a positive integer.

Input: 2000
Given year 2000 is a Leap Year.

Input: -1428
Invalid year. Year must be a positive integer.

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



```python
def is_palindrome_integer(x):
    # Check if x is an integer
    if not isinstance(x, int):
        return False

    # Negative numbers are not palindromes
    if x < 0:
        return False

    # Convert to string and compare with reverse
    return str(x) == str(x)[::-1]

# Test cases
test_cases = [121, -121, 10, 'abc', 0]

# Run and display results
for x in test_cases:
    result = is_palindrome_integer(x)
    print(f"Input: {x} -> Output: {result}")
```

```
Input: 121 -> Output: True
Input: -121 -> Output: False
Input: 10 -> Output: False
Input: abc -> Output: False
Input: 0 -> Output: True

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



```python
def count_string_details(text):
    # Count characters (including spaces)
    num_characters = len(text)

    # Count words (by splitting the text by spaces)
    words = text.split()
    num_words = len(words)

    # Count lines (by splitting the text by newline characters)
    num_lines = text.splitlines()
    num_lines = len(num_lines)

    return num_characters, num_words, num_lines

# Input text from the user
text = input("Enter a string (can include multiple lines): ")

# Get the counts for characters, words, and lines
num_characters, num_words, num_lines = count_string_details(text)

# Display the results
print(f"Number of characters: {num_characters}")
print(f"Number of words: {num_words}")
print(f"Number of lines: {num_lines}")
```

```
Enter a string (can include multiple lines): Hello world

This is Python

It counts characters, words, and lines.


Number of characters: 11
Number of words: 2
Number of lines: 1

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