```
1 usage
def fibonacci(n):
    fib_sequence = [0, 1]
    while len(fib_sequence) < n:
        fib_sequence.append(fib_sequence[-1] + fib_sequence[-2])
    return fib_sequence

print(fibonacci(11))

```

Run main ×

```
C:\Users\hp\PycharmProjects\pythonProject2\venv\Scripts\python.exe C:\U
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

```
1 usage
def find_missing_numbers(numbers):
    if not numbers:
        return []
    min_num = min(numbers)
    max_num = max(numbers)
    full_range = set(range(min_num, max_num + 1))
    actual_numbers = set(numbers)
    missing_numbers = full_range - actual_numbers
    return sorted(missing_numbers)
1 usage
def validate_input(numbers):
    if not isinstance(numbers, list) or not all(isinstance(num, int) for num in numbers):
        raise ValueError("Input must be a list of integers.")
    return numbers

try:
    input_numbers = [2, 1, 5, 4, 6, 9, 7, 8, 10]
    validated_numbers = validate_input(input_numbers)
    missing_numbers = find_missing_numbers(validated_numbers)
    print("Missing numbers:", missing_numbers)
except ValueError as e:
    print(e)

```

find_missing_numbers()

Run main ×

```
C:\Users\hp\PycharmProjects\pythonProject2\venv\Scripts\python.exe C:\Users\hp\PycharmProjects\pythonP
Missing numbers: [3]
```