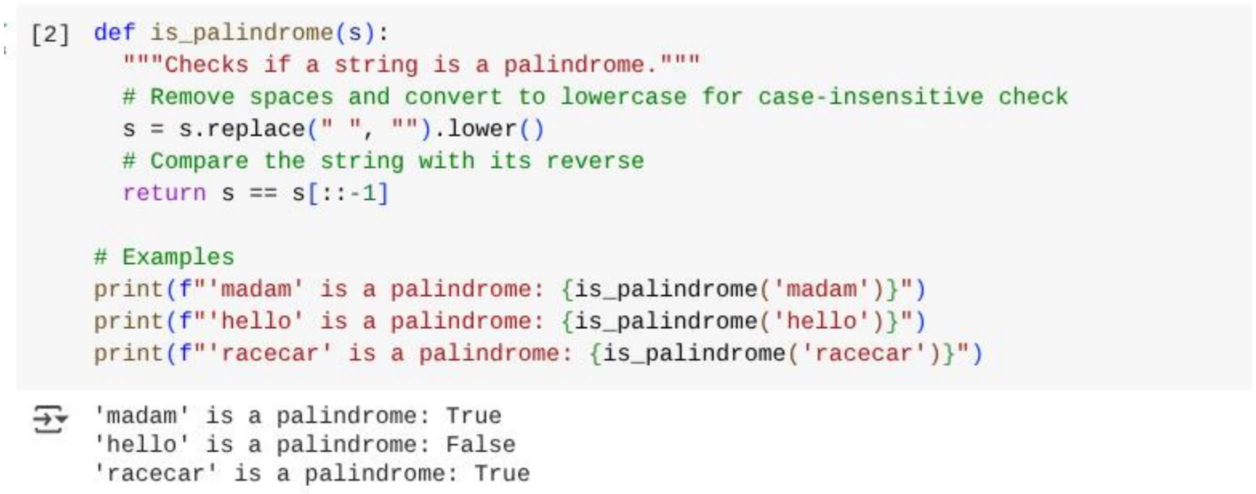
Lab Test-1

Task-1:

write a simple python function that checks whether a string is a palindrome using few short with 2-3 examples

Code and Output:

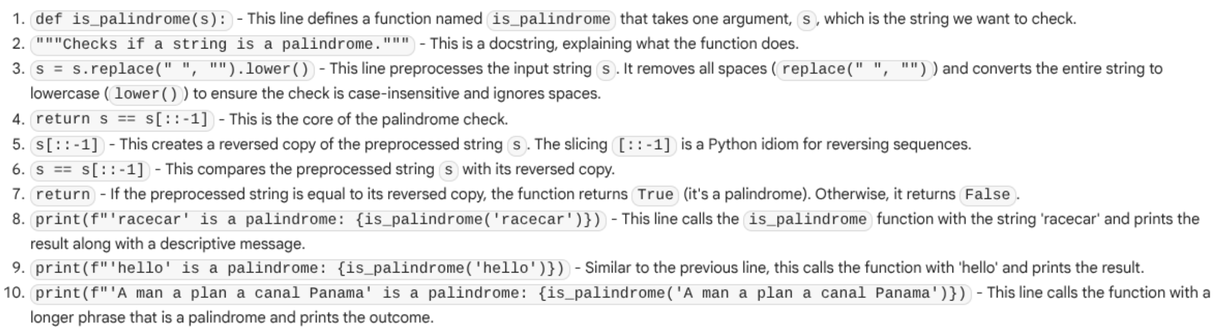


```
[2] def is_palindrome(s):
      """Checks if a string is a palindrome."""
      # Remove spaces and convert to lowercase for case-insensitive check
      s = s.replace(" ", "").lower()
      # Compare the string with its reverse
      return s == s[::-1]

    # Examples
    print(f"'madam' is a palindrome: {is_palindrome('madam')}")
    print(f"'hello' is a palindrome: {is_palindrome('hello')}")
    print(f"'racecar' is a palindrome: {is_palindrome('racecar')}")
```

```
'madam' is a palindrome: True
'hello' is a palindrome: False
'racecar' is a palindrome: True
```

Explanation:

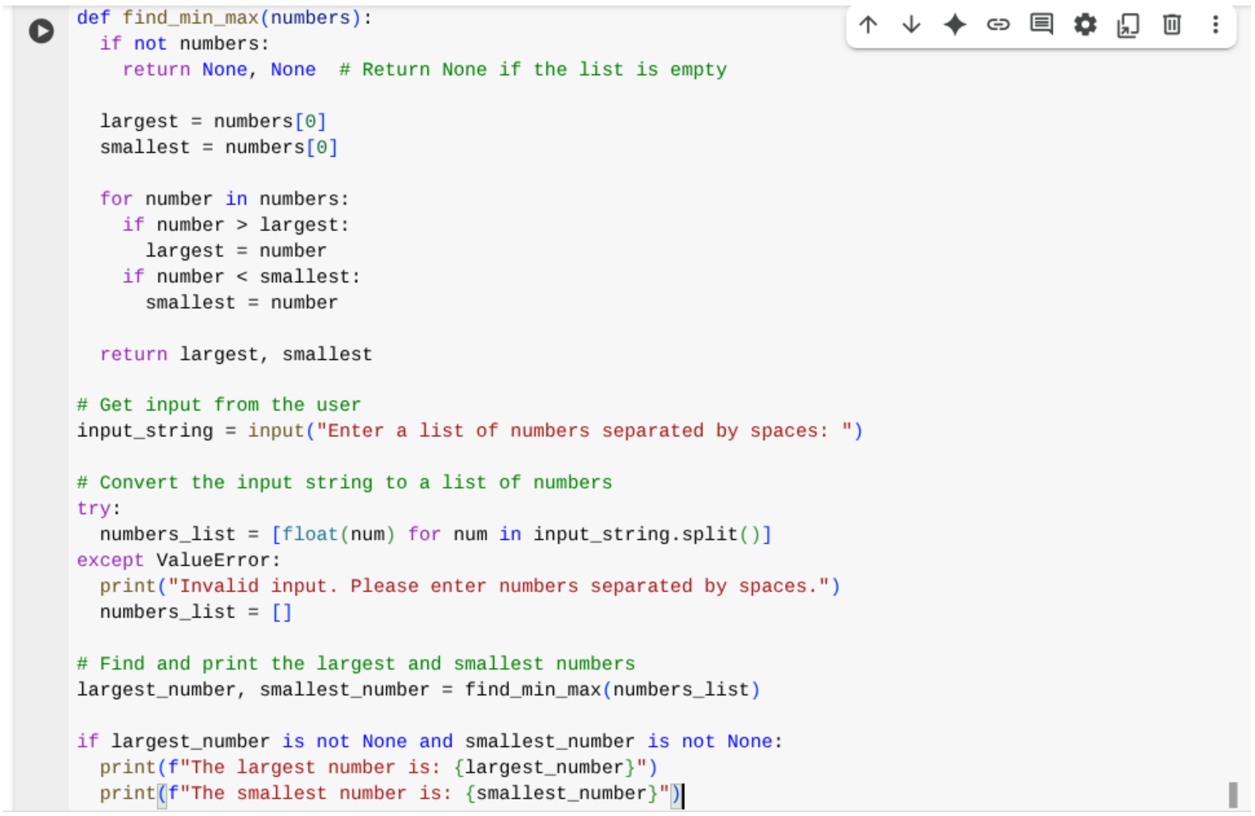


1. `def is_palindrome(s):` - This line defines a function named `is_palindrome` that takes one argument, `s`, which is the string we want to check.
2. `"""Checks if a string is a palindrome."""` - This is a docstring, explaining what the function does.
3. `s = s.replace(" ", "").lower()` - This line preprocesses the input string `s`. It removes all spaces (`replace(" ", "")`) and converts the entire string to lowercase (`lower()`) to ensure the check is case-insensitive and ignores spaces.
4. `return s == s[::-1]` - This is the core of the palindrome check.
5. `s[::-1]` - This creates a reversed copy of the preprocessed string `s`. The slicing `[::-1]` is a Python idiom for reversing sequences.
6. `s == s[::-1]` - This compares the preprocessed string `s` with its reversed copy.
7. `return` - If the preprocessed string is equal to its reversed copy, the function returns `True` (it's a palindrome). Otherwise, it returns `False`.
8. `print(f"'racecar' is a palindrome: {is_palindrome('racecar')}")` - This line calls the `is_palindrome` function with the string 'racecar' and prints the result along with a descriptive message.
9. `print(f"'hello' is a palindrome: {is_palindrome('hello')}")` - Similar to the previous line, this calls the function with 'hello' and prints the result.
10. `print(f"'A man a plan a canal Panama' is a palindrome: {is_palindrome('A man a plan a canal Panama')}")` - This line calls the function with a longer phrase that is a palindrome and prints the outcome.

Task-2:

write a simple python program that finds the largest and smallest numbers in a user provided list

Code:



```
def find_min_max(numbers):
  if not numbers:
    return None, None  # Return None if the list is empty

  largest = numbers[0]
  smallest = numbers[0]

  for number in numbers:
    if number > largest:
      largest = number
    if number < smallest:
      smallest = number

  return largest, smallest

# Get input from the user
input_string = input("Enter a list of numbers separated by spaces: ")

# Convert the input string to a list of numbers
try:
  numbers_list = [float(num) for num in input_string.split()]
except ValueError:
  print("Invalid input. Please enter numbers separated by spaces.")
  numbers_list = []

# Find and print the largest and smallest numbers
largest_number, smallest_number = find_min_max(numbers_list)

if largest_number is not None and smallest_number is not None:
  print(f"The largest number is: {largest_number}")
  print(f"The smallest number is: {smallest_number}")
```

Output:

```
Enter a list of numbers separated by spaces: 4 5 8 3 9 2 0 7
The largest number is: 9.0
The smallest number is: 0.0
```

Explanation:

1. `def find_min_max(numbers):` defines a function to find min/max.
2. It checks if the input list is empty.
3. If empty, it returns `None`.
4. Otherwise, it initializes `largest` and `smallest` with the first number.
5. It loops through the rest of the numbers.
6. Inside the loop, it updates `largest` if a bigger number is found.
7. It updates `smallest` if a smaller number is found.
8. It returns the final `largest` and `smallest`.
9. The main part of the program gets numbers from the user.
10. It calls the function and prints the results.