

Lab Test-1

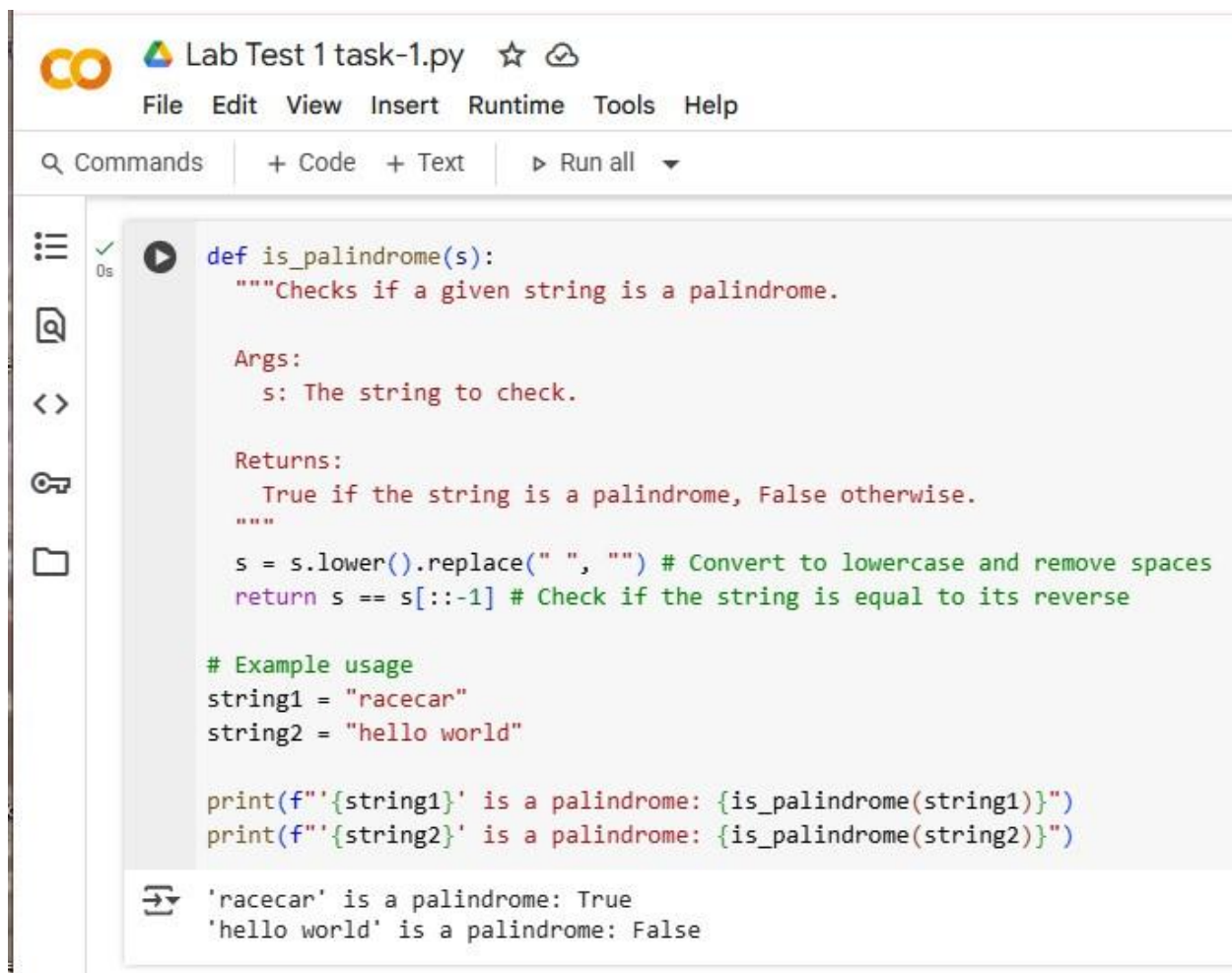
Question 1:

Task-1

Prompt:

#write a python program to check whether a given string is palindrome or not using functions.

Code with Output:



The screenshot shows a code editor window titled "Lab Test 1 task-1.py". The editor contains a Python function `is_palindrome(s)` that checks if a string is a palindrome. The function uses `s.lower().replace(" ", "")` to convert the string to lowercase and remove spaces, then compares it with its reverse `s[::-1]`. Example usage is provided with `string1 = "racecar"` and `string2 = "hello world"`. The output at the bottom shows that "racecar" is a palindrome (True) and "hello world" is not (False).

```
def is_palindrome(s):  
    """Checks if a given string is a palindrome.  
  
    Args:  
        s: The string to check.  
  
    Returns:  
        True if the string is a palindrome, False otherwise.  
    """  
    s = s.lower().replace(" ", "") # Convert to lowercase and remove spaces  
    return s == s[::-1] # Check if the string is equal to its reverse  
  
# Example usage  
string1 = "racecar"  
string2 = "hello world"  
  
print(f'{string1} is a palindrome: {is_palindrome(string1)}')  
print(f'{string2} is a palindrome: {is_palindrome(string2)}')
```

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

Explanation:

- `def is_palindrome(s):` : This line defines the function named `is_palindrome` that accepts one argument, `s`, which represents the string you want to check.
- `"""Checks if a given string is a palindrome...."""` : This is a docstring, which explains what the function does, its arguments, and what it returns. It's good practice to include these for code clarity.
- `s = s.lower().replace(" ", "")` : This line processes the input string `s`.
 - `.lower()` converts the entire string to lowercase. This makes the palindrome check case-insensitive (e.g., "Racecar" will be treated the same as "racecar").
 - `.replace(" ", "")` removes all spaces from the string. This allows phrases like "nurses run" to be considered palindromes.
- `return s == s[::-1]` : This is the core of the palindrome check.
 - `s[::-1]` creates a reversed version of the processed string `s`. The slicing `[::-1]` means start from the end of the string and go backwards to the beginning, taking every character.
 - `s == s[::-1]` compares the original processed string with its reversed version.
 - The `return` statement sends the result of this comparison (either `True` or `False`) back as the output of the function.
- `# Example usage` : This is a comment indicating the code below shows how to use the function.
- `string1 = "racecar"` and `string2 = "hello world"` : These lines create two example strings to test the function.
- `print(f'{string1} is a palindrome: {is_palindrome(string1)}')` and `print(f'{string2} is a palindrome: {is_palindrome(string2)}')` : These lines call the `is_palindrome` function with the example strings and print the results using an f-string for formatted output.

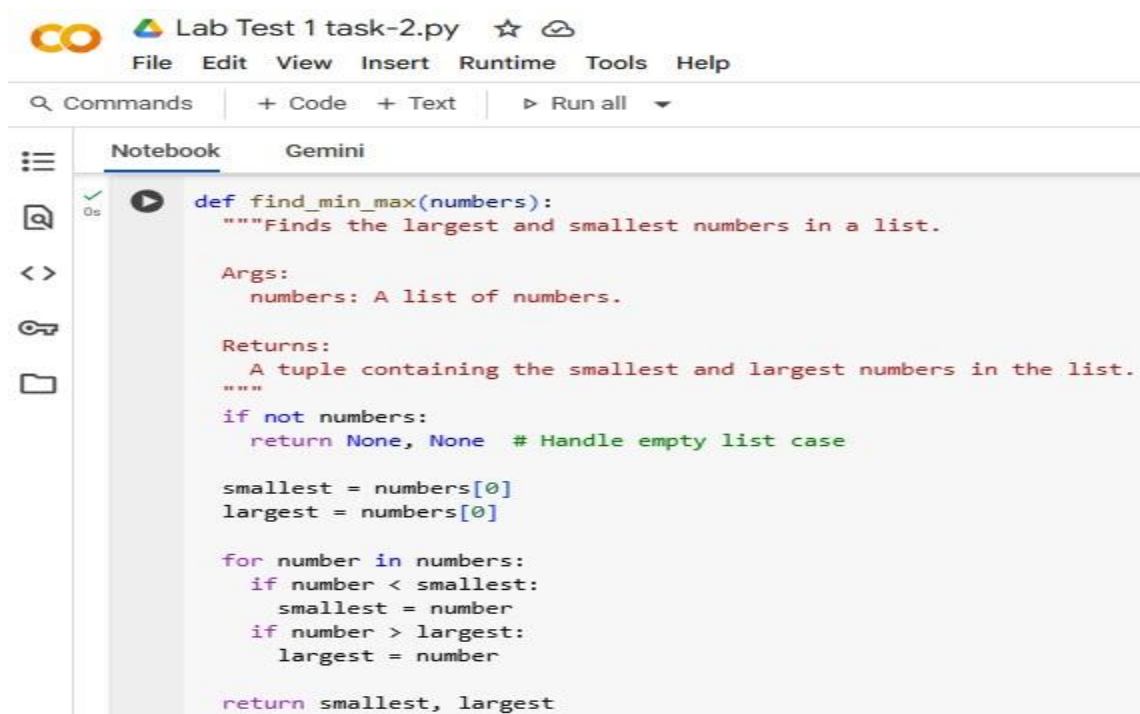
Question-2:

Task-2:

Prompt:

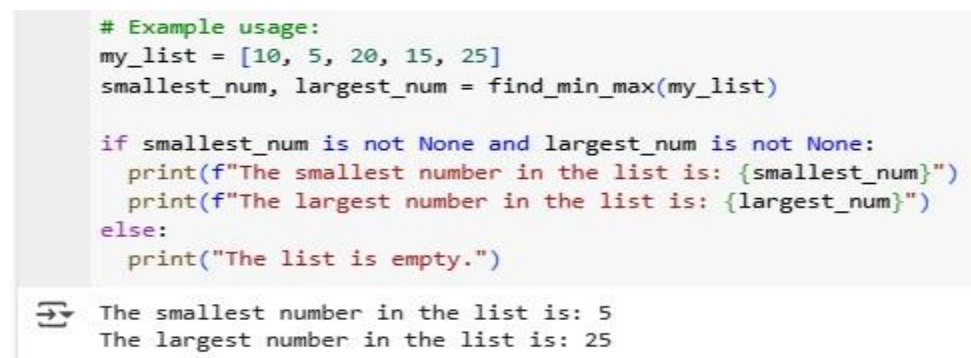
#write a python program that find largest and smallest numbers in given list.

Code with Output:



The screenshot shows a Jupyter Notebook titled "Lab Test 1 task-2.py". The interface includes a menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". Below the menu is a toolbar with "Commands", "+ Code", "+ Text", and "Run all". The notebook has two tabs: "Notebook" and "Gemini". The "Notebook" tab is active, displaying a Python function `find_min_max`. The function has a docstring that describes its purpose and arguments. It handles an empty list by returning `None, None`. For a non-empty list, it initializes `smallest` and `largest` to the first element and then iterates through the list to find the actual minimum and maximum values.

```
def find_min_max(numbers):  
    """Finds the largest and smallest numbers in a list.  
  
    Args:  
        numbers: A list of numbers.  
  
    Returns:  
        A tuple containing the smallest and largest numbers in the list.  
    """  
    if not numbers:  
        return None, None # Handle empty list case  
  
    smallest = numbers[0]  
    largest = numbers[0]  
  
    for number in numbers:  
        if number < smallest:  
            smallest = number  
        if number > largest:  
            largest = number  
  
    return smallest, largest
```



The screenshot shows the execution of the code. It starts with an example usage where a list `my_list = [10, 5, 20, 15, 25]` is passed to the `find_min_max` function. The function returns a tuple of the smallest and largest numbers. The code then checks if both values are not `None` and prints them. The output shows that the smallest number is 5 and the largest number is 25.

```
# Example usage:  
my_list = [10, 5, 20, 15, 25]  
smallest_num, largest_num = find_min_max(my_list)  
  
if smallest_num is not None and largest_num is not None:  
    print(f"The smallest number in the list is: {smallest_num}")  
    print(f"The largest number in the list is: {largest_num}")  
else:  
    print("The list is empty.")
```

The smallest number in the list is: 5
The largest number in the list is: 25

Explanation:

- `def find_min_max(numbers):` : This line defines the function named `find_min_max` which accepts one argument, `numbers`, which is expected to be a list.
- `if not numbers:` : This checks if the input list `numbers` is empty.
- `return None, None` : If the list is empty, the function returns `None` for both the smallest and largest numbers.
- `smallest = numbers[0] and largest = numbers[0]` : If the list is not empty, the first element of the list is initially assumed to be both the smallest and largest number.
- `for number in numbers:` : This loop iterates through each `number` in the input `numbers` list.
- `if number < smallest:` : Inside the loop, this checks if the current `number` is smaller than the current `smallest` value. If it is, `smallest` is updated to the current `number`.
- `if number > largest:` : This checks if the current `number` is larger than the current `largest` value. If it is, `largest` is updated to the current `number`.
- `return smallest, largest` : After the loop finishes, the function returns the final `smallest` and `largest` values found as a tuple.
- `my_list = [10, 5, 20, 15, 25]` : This line creates a sample list called `my_list`.
- `smallest_num, largest_num = find_min_max(my_list)` : This line calls the `find_min_max` function with `my_list` and assigns the returned smallest and largest values to the variables `smallest_num` and `largest_num`.
- `if smallest_num is not None and largest_num is not None:` : This checks if the returned values are not `None` (meaning the input list was not empty).
- `print(...)` : These lines print the smallest and largest numbers found.
- `else: print(...)` : This handles the case where the input list was empty and prints a message indicating that.