

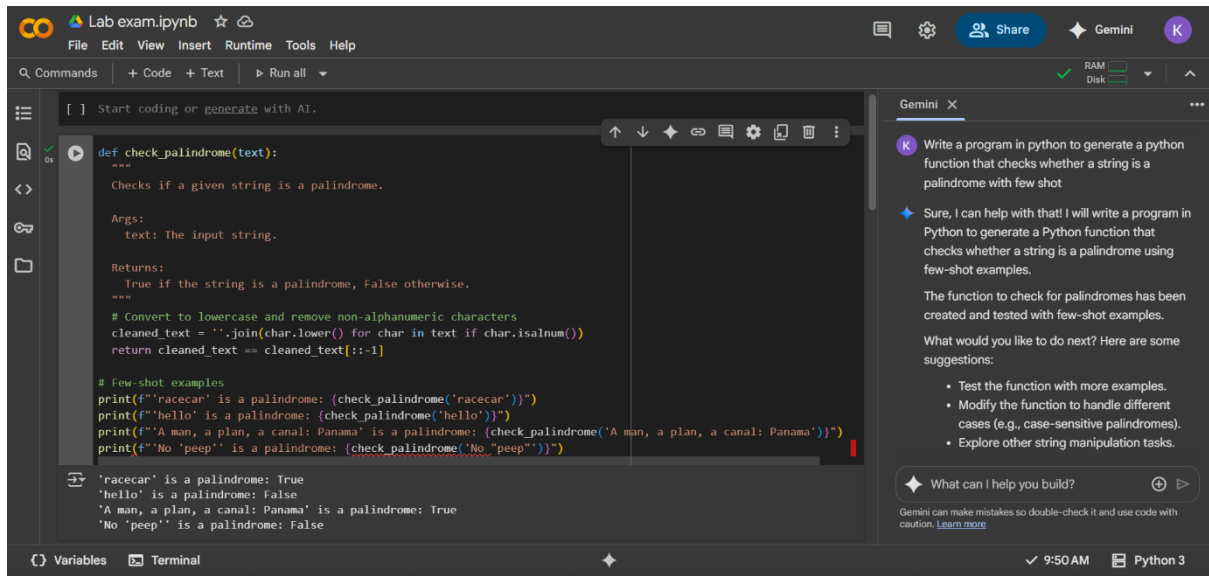
Lab Exam-1

2403a52132

Task 1:

Write a program in python to generate a python function that checks whether a string is a palindrome with few shot.

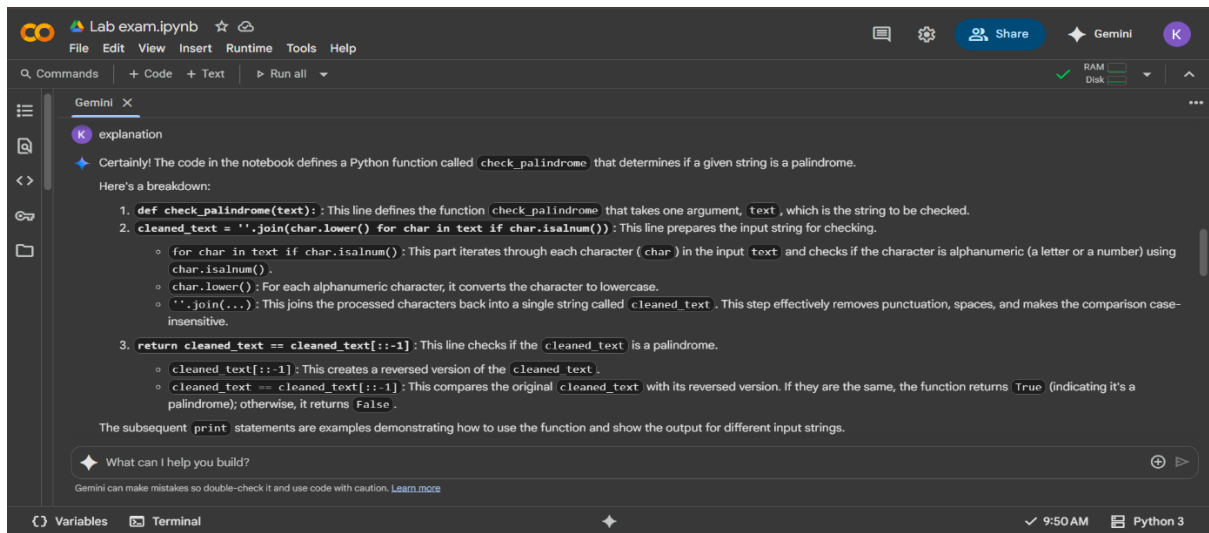
Code and output:



The screenshot shows a Jupyter Notebook interface with a dark theme. The notebook is titled "Lab exam.ipynb". The code cell contains a Python function `check_palindrome(text)` that checks if a string is a palindrome. The function converts the string to lowercase, removes non-alphanumeric characters, and compares the string with its reverse. It includes several print statements for testing with "racecar", "hello", "A man, a plan, a canal: Panama", and "No 'peep'". The output cell shows the results of these tests: "racecar" is a palindrome (True), "hello" is not (False), "A man, a plan, a canal: Panama" is (True), and "No 'peep'" is not (False). The right sidebar shows a Gemini chat window with a prompt to write a program to generate a python function that checks whether a string is a palindrome with few shot, and a response explaining the function and providing suggestions for further exploration.

```
def check_palindrome(text):  
    """  
    Checks if a given string is a palindrome.  
  
    Args:  
        text: The input string.  
  
    Returns:  
        True if the string is a palindrome, False otherwise.  
    """  
    # Convert to lowercase and remove non-alphanumeric characters  
    cleaned_text = ''.join(char.lower() for char in text if char.isalnum())  
    return cleaned_text == cleaned_text[::-1]  
  
# Few-shot examples  
print(f'"racecar" is a palindrome: {check_palindrome("racecar")}")  
print(f'"hello" is a palindrome: {check_palindrome("hello")}")  
print(f'"A man, a plan, a canal: Panama" is a palindrome: {check_palindrome("A man, a plan, a canal: Panama")}")  
print(f'"No \'peep\' is a palindrome: {check_palindrome("No \'peep\'")}")  
  
'racecar' is a palindrome: True  
'hello' is a palindrome: False  
'A man, a plan, a canal: Panama' is a palindrome: True  
'No 'peep' is a palindrome: False
```

Explanation:



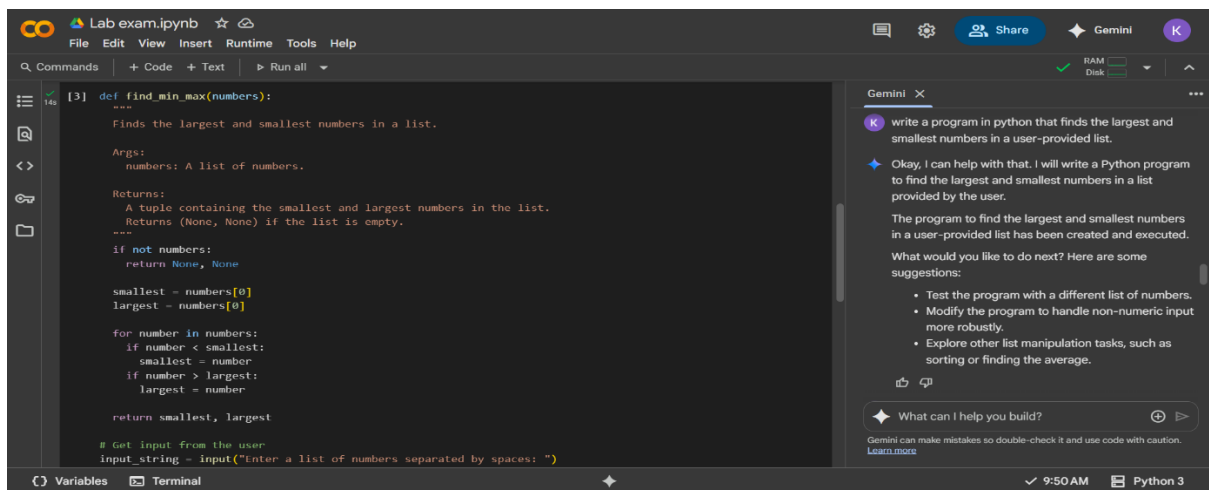
The screenshot shows a Jupyter Notebook interface with a dark theme. The notebook is titled "Lab exam.ipynb". The code cell contains a Python function `check_palindrome(text)` that checks if a string is a palindrome. The function converts the string to lowercase, removes non-alphanumeric characters, and compares the string with its reverse. It includes several print statements for testing with "racecar", "hello", "A man, a plan, a canal: Panama", and "No 'peep'". The output cell shows the results of these tests: "racecar" is a palindrome (True), "hello" is not (False), "A man, a plan, a canal: Panama" is (True), and "No 'peep'" is not (False). The right sidebar shows a Gemini chat window with a prompt to write a program to generate a python function that checks whether a string is a palindrome with few shot, and a response explaining the function and providing suggestions for further exploration.

```
def check_palindrome(text):  
    """  
    Checks if a given string is a palindrome.  
  
    Args:  
        text: The input string.  
  
    Returns:  
        True if the string is a palindrome, False otherwise.  
    """  
    # Convert to lowercase and remove non-alphanumeric characters  
    cleaned_text = ''.join(char.lower() for char in text if char.isalnum())  
    return cleaned_text == cleaned_text[::-1]  
  
# Few-shot examples  
print(f'"racecar" is a palindrome: {check_palindrome("racecar")}")  
print(f'"hello" is a palindrome: {check_palindrome("hello")}")  
print(f'"A man, a plan, a canal: Panama" is a palindrome: {check_palindrome("A man, a plan, a canal: Panama")}")  
print(f'"No \'peep\' is a palindrome: {check_palindrome("No \'peep\'")}")  
  
'racecar' is a palindrome: True  
'hello' is a palindrome: False  
'A man, a plan, a canal: Panama' is a palindrome: True  
'No 'peep' is a palindrome: False
```

Task 2:

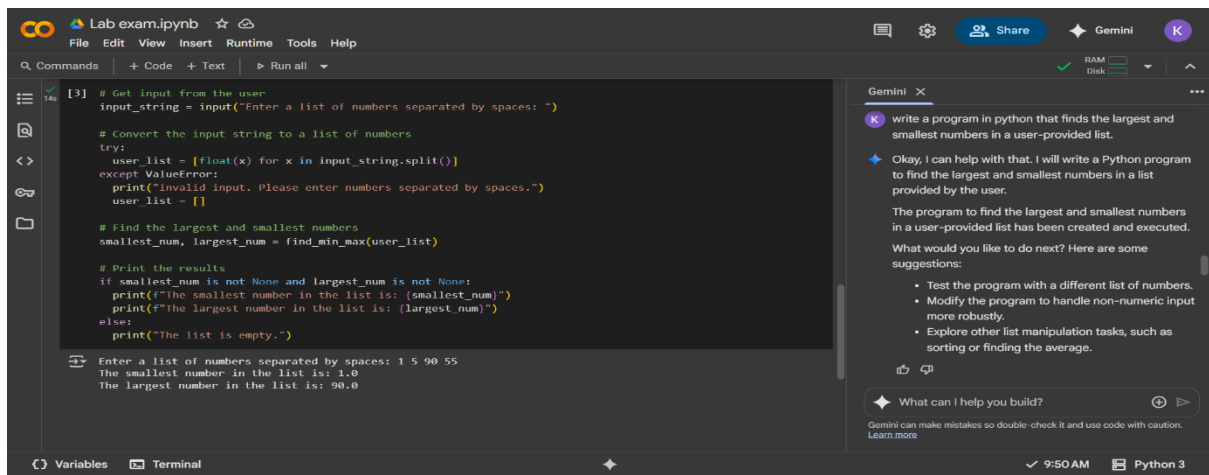
write a program in python that finds the largest and smallest numbers in a user-provided list.

Code and Output:



The screenshot shows a Jupyter Notebook with a single code cell. The code defines a function `find_min_max` that takes a list of numbers and returns a tuple of the minimum and maximum values. It also includes a prompt for user input to test the function.

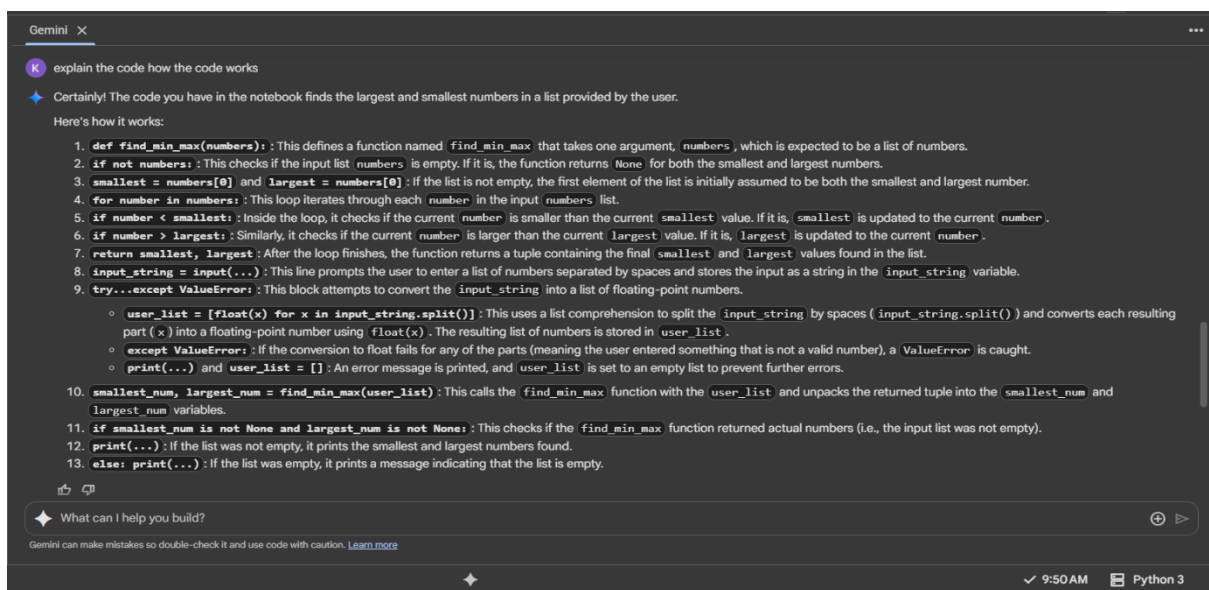
```
[3] def find_min_max(numbers):  
    """  
    Finds the largest and smallest numbers in a list.  
  
    Args:  
        numbers: A list of numbers.  
  
    Returns:  
        A tuple containing the smallest and largest numbers in the list.  
        Returns (None, None) if the list is empty.  
    """  
    if not numbers:  
        return None, None  
  
    smallest = numbers[0]  
    largest = numbers[0]  
  
    for number in numbers:  
        if number < smallest:  
            smallest = number  
        if number > largest:  
            largest = number  
  
    return smallest, largest  
  
# Get input from the user  
input_string = input("Enter a list of numbers separated by spaces: ")
```



The screenshot shows the same Jupyter Notebook with the code cell executed. The output displays the user input and the results of the function.

```
[3] # 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:  
    user_list = [float(x) for x in input_string.split()]  
except ValueError:  
    print("Invalid input. Please enter numbers separated by spaces.")  
    user_list = []  
  
# Find the largest and smallest numbers  
smallest_num, largest_num = find_min_max(user_list)  
  
# Print the results  
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.")  
  
Enter a list of numbers separated by spaces: 1 5 90 55  
The smallest number in the list is: 1.0  
The largest number in the list is: 90.0
```

Explanation:



The screenshot shows a Gemini chat interface with a user prompt asking for an explanation of the code. The Gemini response provides a detailed breakdown of the code's logic, line by line.

explain the code how the code works

Certainly! The code you have in the notebook finds the largest and smallest numbers in a list provided by the user. Here's how it works:

- `def find_min_max(numbers):` : This defines a function named `find_min_max` that takes one argument, `numbers`, which is expected to be a list of numbers.
- `if not numbers:` : This checks if the input list `numbers` is empty. If it is, 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, it 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:` : Similarly, it 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 a tuple containing the final `smallest` and `largest` values found in the list.
- `input_string = input(...)` : This line prompts the user to enter a list of numbers separated by spaces and stores the input as a string in the `input_string` variable.
- `try...except ValueError:` : This block attempts to convert the `input_string` into a list of floating-point numbers.
 - `user_list = [float(x) for x in input_string.split()]` : This uses a list comprehension to split the `input_string` by spaces (`input_string.split()`) and converts each resulting part (`x`) into a floating-point number using `float(x)`. The resulting list of numbers is stored in `user_list`.
 - `except ValueError:` : If the conversion to float fails for any of the parts (meaning the user entered something that is not a valid number), a `ValueError` is caught.
 - `print(...)` and `user_list = []` : An error message is printed, and `user_list` is set to an empty list to prevent further errors.
- `smallest_num, largest_num = find_min_max(user_list)` : This calls the `find_min_max` function with the `user_list` and unpacks the returned tuple into the `smallest_num` and `largest_num` variables.
- `if smallest_num is not None and largest_num is not None:` : This checks if the `find_min_max` function returned actual numbers (i.e., the input list was not empty).
- `print(...)` : If the list was not empty, it prints the smallest and largest numbers found.
- `else: print(...)` : If the list was empty, it prints a message indicating that the list is empty.