

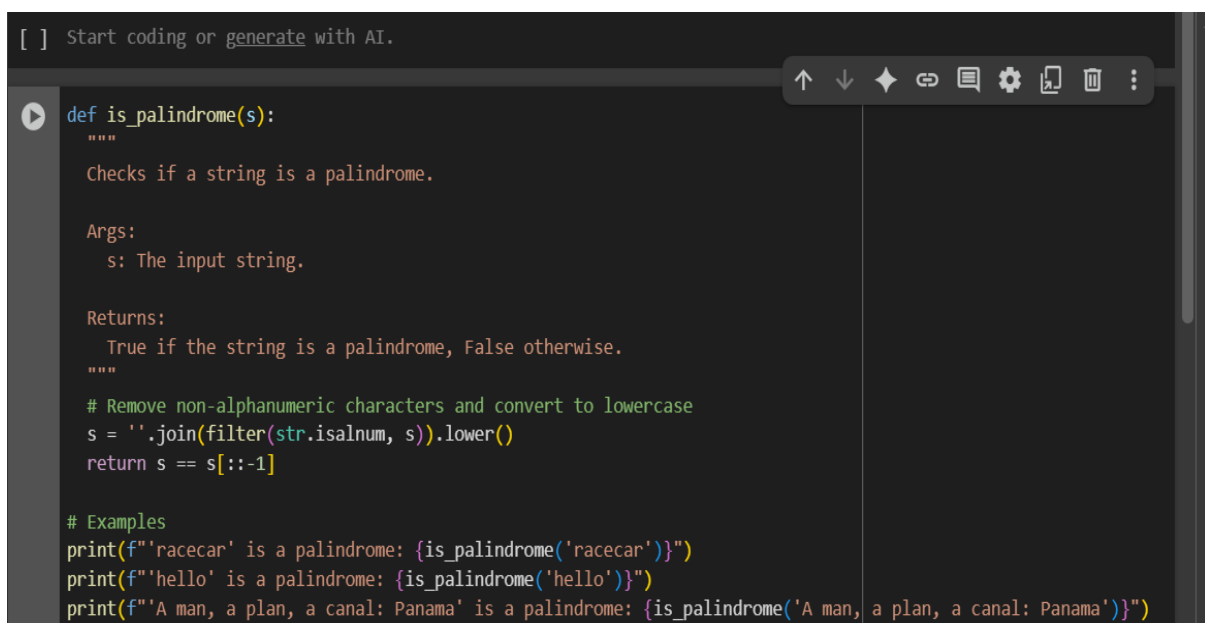
AI ASSISTED CODING

LAB EXAM – 1

TASK-1:

Write a python program to find whether the given number is palindrome or not.

CODE SCREENSHOTS:



```
[ ] Start coding or generate with AI.

def is_palindrome(s):
    """
    Checks if a string is a palindrome.

    Args:
        s: The input string.

    Returns:
        True if the string is a palindrome, False otherwise.
    """
    # Remove non-alphanumeric characters and convert to lowercase
    s = ''.join(filter(str.isalnum, s)).lower()
    return s == s[::-1]

# Examples
print(f"'racecar' is a palindrome: {is_palindrome('racecar')}")
print(f"'hello' is a palindrome: {is_palindrome('hello')}")
print(f"'A man, a plan, a canal: Panama' is a palindrome: {is_palindrome('A man, a plan, a canal: Panama')}")
```

OUTPUT:

```
'racecar' is a palindrome: True
'hello' is a palindrome: False
'A man, a plan, a canal: Panama' is a palindrome: True
```

EXPLANATION:

J Please explain this code:

```
<> def is_palindrome(s):
```

◆ No problem! This code defines a function called `is_palindrome` that checks if a given string is a palindrome, ignoring case and non-alphanumeric characters.

Here's a breakdown:

1. `def is_palindrome(s):` : This line defines a function named `is_palindrome` that takes one argument, `s`, which is the string to be checked.
2. `s = ''.join(filter(str.isalnum, s)).lower()` : This is the core of the preprocessing step:
 - `filter(str.isalnum, s)` : This filters the input string `s`, keeping only characters for which `str.isalnum()` returns `True` (i.e., alphanumeric characters).
 - `''.join(...)` : This joins the filtered characters back into a single string.
 - `.lower()` : This converts the resulting string to lowercase. This makes the check case-insensitive.
3. `return s == s[::-1]` : This line checks if the processed string `s` is equal to its reverse (`s[::-1]`).
 - `s[::-1]` is a slicing technique in Python that creates a reversed copy of the string.
 - If the string is the same forwards and backward, the function returns `True`, indicating it's a palindrome. Otherwise, it returns `False`.

The lines after the function definition are examples that demonstrate how to use the `is_palindrome` function with different strings and print the results.



TASK-2:

Write a python program to find the largest and the smallest numbers by user provided list and ask gemini to give step by step explanation of code.

CODE:

```
def find_largest_smallest(numbers):  
    """  
    Finds the largest and smallest numbers in a list.  
  
    Args:  
        numbers: A list of numbers.  
  
    Returns:  
        A tuple containing the largest and smallest numbers, or None if the list is empty.  
    """  
    if not numbers:  
        return None, None  
  
    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: ")
```

```
        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(x) for x 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_num, smallest_num = find_largest_smallest(numbers_list)  
  
if largest_num is not None and smallest_num is not None:  
    print(f"The largest number is: {largest_num}")  
    print(f"The smallest number is: {smallest_num}")  
else:  
    print("No numbers were entered.")
```

OUTPUT:

```
➦ Enter a list of numbers separated by spaces: 45 56 92
The largest number is: 92.0
The smallest number is: 45.0
```

EXPLANATION:

Gemini X

J Please explain this code:

```
<> def find_largest_smallest(numbers):
```

✦ This code defines a function `find_largest_smallest` that takes a list of numbers as input and returns the largest and smallest numbers in the list.

Here's a breakdown of the code:

1. `def find_largest_smallest(numbers):` : This defines a function named `find_largest_smallest` that accepts one argument, `numbers`, which
2. `if not numbers:` : This checks if the input list `numbers` is empty.
3. `return None, None` : If the list is empty, the function returns `None` for both the largest and smallest numbers.
4. `largest = numbers[0]` and `smallest = numbers[0]` : If the list is not empty, the first element of the list is initially assumed to be both the largest
5. `for number in numbers:` : This loop iterates through each `number` in the input `numbers` list.
6. `if number > largest:` : Inside the loop, this checks if the current `number` is greater than the current `largest` number found so far.
7. `largest = number` : If the current `number` is greater, it updates the `largest` variable to the current `number`.
8. `if number < smallest:` : This checks if the current `number` is smaller than the current `smallest` number found so far.
9. `smallest = number` : If the current `number` is smaller, it updates the `smallest` variable to the current `number`.
10. `return largest, smallest` : After the loop finishes, the function returns a tuple containing the final `largest` and `smallest` numbers found.
11. `input_string = input("Enter a list of numbers separated by spaces: ")` : This line prompts the user to enter a string of numbers separated by spaces and stores it in the `input_string` variable.

10. `return largest, smallest` : After the loop finishes, the function returns a tuple containing the final `largest` and `smallest` numbers found.
11. `input_string = input("Enter a list of numbers separated by spaces: ")` : This line prompts the user to enter a string of numbers separated by spaces and stores it in the `input_string` variable.
12. `try: ... except ValueError:` : This is a `try-except` block to handle potential errors.
13. `numbers_list = [float(x) for x in input_string.split()]` : This line splits the `input_string` by spaces and converts each part to a floating-point number, storing them in the `numbers_list`.
14. `print("Invalid input. Please enter numbers separated by spaces.")` and `numbers_list = []` : If a `ValueError` occurs during the conversion (e.g., the user enters non-numeric text), this block prints an error message and sets `numbers_list` to an empty list.
15. `largest_num, smallest_num = find_largest_smallest(numbers_list)` : This calls the `find_largest_smallest` function with the `numbers_list` and assigns the returned largest and smallest numbers to `largest_num` and `smallest_num` respectively.
16. `if largest_num is not None and smallest_num is not None:` : This checks if the `find_largest_smallest` function returned valid numbers (i.e., the input list was not empty).
17. `print(f"The largest number is: {largest_num}")` and `print(f"The smallest number is: {smallest_num}")` : If valid numbers were found, these lines print the largest and smallest numbers using an f-string for formatting.
18. `else:` : This is the block executed if the input list was empty.
19. `print("No numbers were entered.")` : This line is printed if no numbers were entered by the user.

Let me know if you'd like a more detailed explanation of any specific part!



