LAB- EXAM -1

1STQUESION:

WRITE A COMMENT PROMPT WITH 2-3 EXAMPLE'S ASKING COPILOT/GEMINI TO GENERATE A PYTHON FUNCTION THAT CHECK'S WETHER A STRING IS A PALINDROME

PROMPT:

WRITE A PYTHON FUNCTION THAT CHECKS WETHER A GIVEN STRING IS PALINDROME OR NOT IF IS AN PALINDROME IT HAS TO GIVE TRUE AND IF NOT IT HAS TO GIVE FALSE FOR YOUR REFERNCE/EXAMPLE'S WHICH MIGHT HELP;

Madam-true,Racecare-true,Hello-false.Then docstring/explain this code line-by-line

CODE:

```
def is_palindrome(s):
  """
  Checks if a string is a palindrome, ignoring case and spaces.

  Args:
    s: The input string.

  Returns:
    True if the string is a palindrome, False otherwise.
  """
  # Remove spaces and convert to lowercase
  s = s.replace(" ", "").lower()
  # Check if the string is equal to its reverse
  return s == s[::-1]

# Examples
print(f"'madam' is a palindrome: {is_palindrome('madam')}")
print(f"'racecar' is a palindrome: {is_palindrome('racecar')}")
print(f"'hello' is a palindrome: {is_palindrome('hello')}")
print(f"'Madam' is a palindrome: {is_palindrome('Madam')}") # Test case
for case insensitivity
print(f"'Race car' is a palindrome: {is_palindrome('Race car')}") #
Test case for ignoring spaces
```

OUTPUT:

```
'madam' is a palindrome: True
'racecar' is a palindrome: True
'hello' is a palindrome: False
'Madam' is a palindrome: True
'Race car' is a palindrome: True
```

CODE EXPLAINATION LINE-BY-LINE:

- def is_palindrome(s): : This line defines a function named is_palindrome that takes one argument, s.

- """ ... """ : This is the docstring. It provides a summary of what the function does, explains its arguments (Args), and describes what it returns (Returns). This is good practice for documenting your code.
- s = s.replace(" ", "").lower(): This line prepares the input string s for the palindrome check.
    - s.replace(" ", ""): This removes all spaces from the string.
    - .lower(): This converts the entire string to lowercase. This makes the check case-insensitive (e.g., "Madam" will be treated the same as "madam").
- return s == s[::-1]: This line performs the actual palindrome check and returns the result.
    - s[::-1]: This is a slicing technique in Python that creates a reversed version of the string s.
    - s == s[::-1]: This compares the processed string s (without spaces and in lowercase) with its reversed version. If they are the same, it means the original string (ignoring spaces and case) was a palindrome, and the expression evaluates to True. Otherwise, it evaluates to False.

2ND-QUESTION:

ASK GOOGLE/GEMINI TO GENERATE A PYTHON PROGRAMM THAT FIND'S THE LARGEST AND SMALLEST NUMBERSN AN USER PROVIDED LIST.THEN ASK GEMINI TO EXPLAIN HOW THE CODE WORK'S

PROMT:

WRITE A PYTHON FUNCTION TO FIND THE LARGEST AND SMALLEST NUMBER IN AN USER PROVIDED LIST,THEN AFTER THAT EXPLAIN THIS CODE COMPLETELY

CODE:

```
def find_min_max(numbers):
  """
  Finds the largest and smallest number in a list.

  Args:
    numbers: A list of numbers.

  Returns:
    A tuple containing the largest and smallest number in the list.
    Returns (None, 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

# Example usage:
my_list = [10, 5, 20, 2, 15]
largest_num, smallest_num = find_min_max(my_list)

if largest_num is not None and smallest_num is not None:
  print(f"Largest number: {largest_num}")
  print(f"Smallest number: {smallest_num}")
else:
  print("The list is empty.")
```

OUTPUT:

```
Largest number: 20
Smallest number: 2
```

CODE-EXPLAINATION:

1. ***Function Definition: The code defines a function called find_min_max that takes one argument, numbers, which is expected to be a list.***
2. ***Empty List Check: It first checks if the input list numbers is empty. If it is, it returns None for both the largest and smallest numbers.***
3. ***Initialization: If the list is not empty, it initializes two variables, largest and smallest, with the first element of the list.***
4. ***Iteration: It then loops through each number in the input list.***
5. ***Comparison and Update: Inside the loop, it compares the current number with the current largest and smallest values. If the number is greater than largest, largest is updated to the current number. Similarly, if the number is less than smallest, smallest is updated.***
6. ***Return Values: After iterating through all the numbers in the list, the function returns a tuple containing the final largest and smallest values found.***
7. ***Example Usage: The code then demonstrates how to use the function with an example list my_list. It calls the function, stores the returned values in largest_num and smallest_num, and finally prints the results.***