Lab Assignment-4.2

Name : Saini.Deepthi

Hall ticket No. : 2403A52014

Batch No. : 24BTCAIAIB02

Course Title : Ai Assisted Coding

TASK -1

Description:

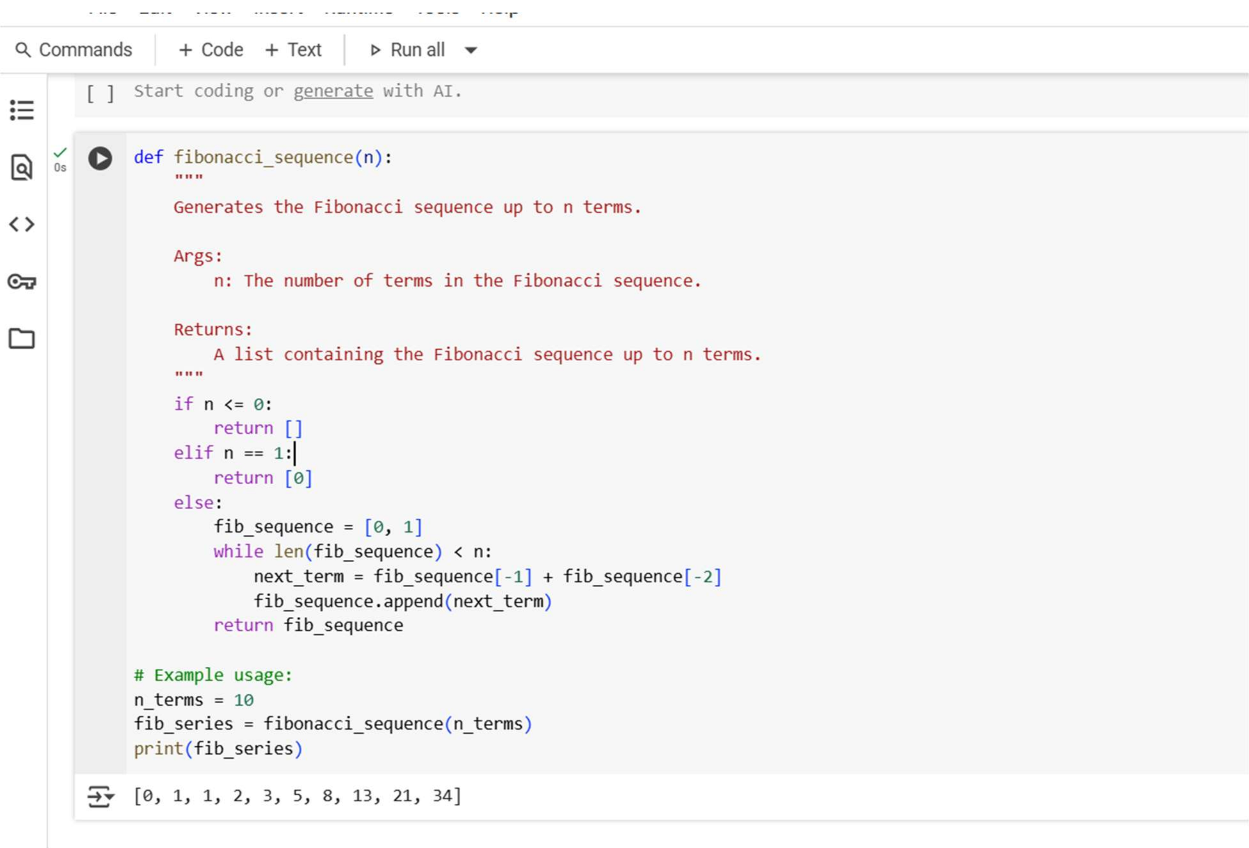
>>Zero-shot: Prompt AI with only the instruction — Write a Python function to generate the Fibonacci sequence up to n terms

Expected Output:

>> A working function without using any sample inputs/outputs

Prompt:

>> generate a code on write a Python function to generate the Fibonacci sequence up to n terms.



```python
def fibonacci_sequence(n):
    """
    Generates the Fibonacci sequence up to n terms.

    Args:
        n: The number of terms in the Fibonacci sequence.

    Returns:
        A list containing the Fibonacci sequence up to n terms.
    """
    if n <= 0:
        return []
    elif n == 1:
        return [0]
    else:
        fib_sequence = [0, 1]
        while len(fib_sequence) < n:
            next_term = fib_sequence[-1] + fib_sequence[-2]
            fib_sequence.append(next_term)
        return fib_sequence

# Example usage:
n_terms = 10
fib_series = fibonacci_sequence(n_terms)
print(fib_series)
```

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

Observation:

>>A simple observation from the output [0, 1, 1, 2, 3, 5, 8, 13, 21, 34] is that after the first two numbers (0 and 1), each subsequent number is the sum of the two preceding numbers (e.g., 1+2=3, 2+3=5, 3+5=8, and so on).

Explanation:

>> The Fibonacci sequence is a series of numbers where, after the first two numbers (which are usually 0 and 1), you get the next number by adding the two numbers before it. So, it goes 0, then 1, then 0+1=1, then 1+1=2, then 1+2=3, and so on: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34... It's like a simple pattern of adding the last two numbers to get the next one!

TASK -2

Description:

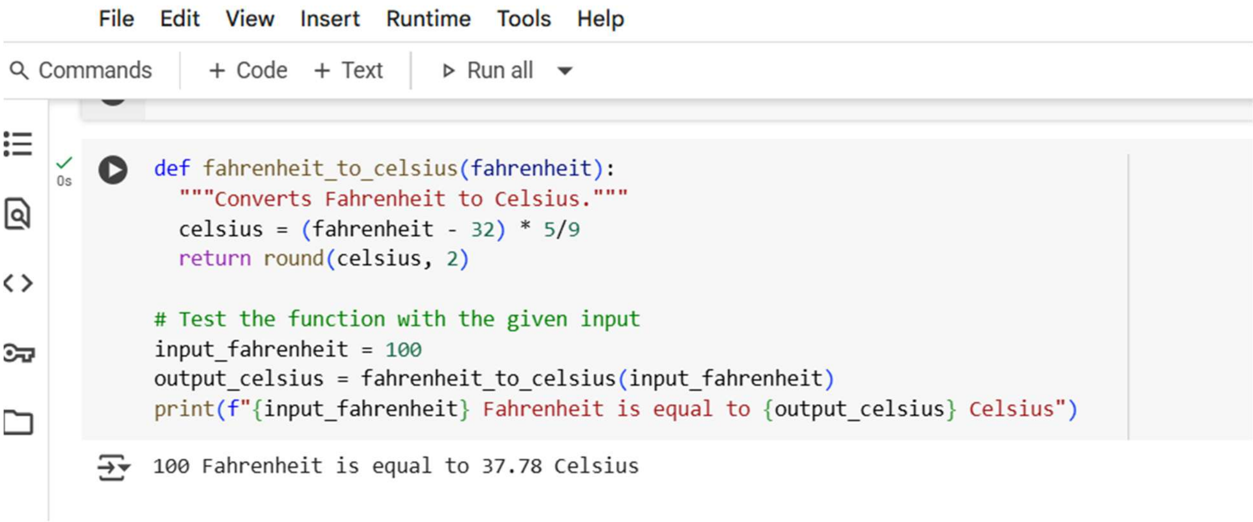
>>One-shot: Provide one example: Input: 100, Output: 37.78 to help AI generate a function that converts Fahrenheit to Celsius.

Expected Output:

>> A correct conversion function guided by the single example.

Prompt:

>>generate a function that converts Fahrenheit to Celsius with the given input and output Input: 100, Output: 37.78



```python
def fahrenheit_to_celsius(fahrenheit):
  """Converts Fahrenheit to Celsius."""
  celsius = (fahrenheit - 32) * 5/9
  return round(celsius, 2)

# Test the function with the given input
input_fahrenheit = 100
output_celsius = fahrenheit_to_celsius(input_fahrenheit)
print(f"{input_fahrenheit} Fahrenheit is equal to {output_celsius} Celsius")
```

```
100 Fahrenheit is equal to 37.78 Celsius
```

Observation:

>>The code defines a function called fahrenheit_to_celsius that takes a temperature in Fahrenheit as input and returns the equivalent temperature in Celsius, rounded to two decimal places. When the input is 100 Fahrenheit, the function correctly calculates and prints that it is equal to 37.78 Celsius.

Explanation:

>> This code converts Fahrenheit to Celsius. You give it a Fahrenheit temperature (like 100), and it calculates and shows you the Celsius equivalent (like 37.78).

TASK -3

Description:

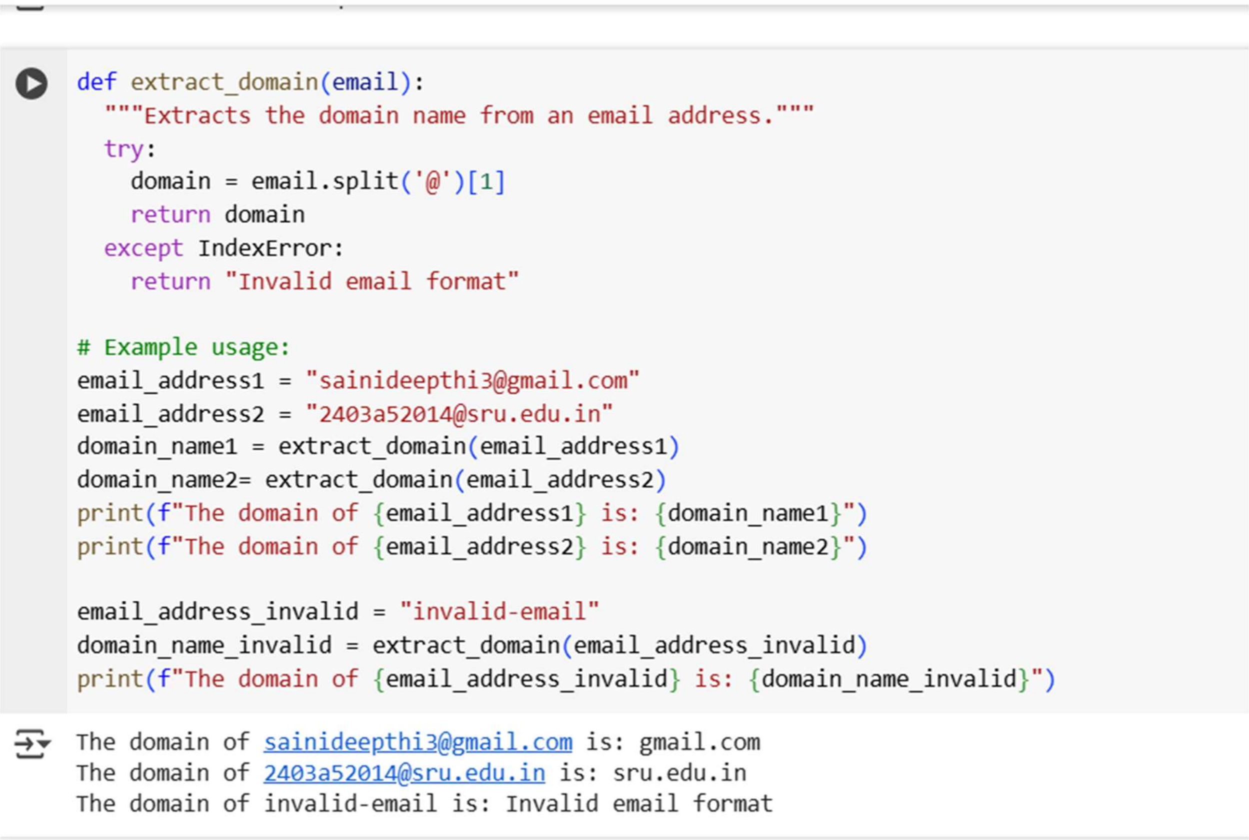
>>Few-shot: Give 2–3 examples to create a function that extracts the domain name from an email address.

Expected Output:

>>Accurate function that returns only the domain portion of an email (e.g., @gmail.com)

Prompt:

>> Write a Python function called extract_domain that takes an email address as input and returns the domain name.



```python
def extract_domain(email):
  """Extracts the domain name from an email address."""
  try:
    domain = email.split('@')[1]
    return domain
  except IndexError:
    return "Invalid email format"

# Example usage:
email_address1 = "sainideepthi3@gmail.com"
email_address2 = "2403a52014@sru.edu.in"
domain_name1 = extract_domain(email_address1)
domain_name2= extract_domain(email_address2)
print(f"The domain of {email_address1} is: {domain_name1}")
print(f"The domain of {email_address2} is: {domain_name2}")

email_address_invalid = "invalid-email"
domain_name_invalid = extract_domain(email_address_invalid)
print(f"The domain of {email_address_invalid} is: {domain_name_invalid}")
```

```
The domain of sainideepthi3@gmail.com is: gmail.com
The domain of 2403a52014@sru.edu.in is: sru.edu.in
The domain of invalid-email is: Invalid email format
```

Observation:

>>The code successfully extracts the domain name from valid email addresses by finding the text after the '@' symbol and handles invalid formats by returning None.

Explanation:

>> the code uses a pattern to find the "@" symbol in an email and then grabs everything that comes after it as the domain name. If there's no "@" followed by characters, it indicates an invalid format for this extraction purpose, and it returns None.

TASK -4

Description:

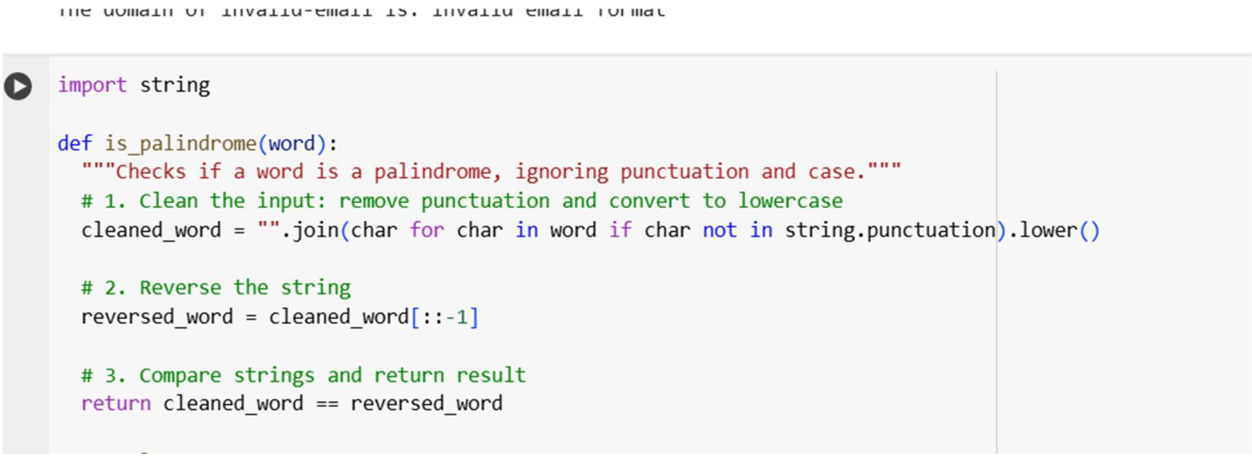
>> Compare zero-shot vs few-shot prompting for generating a function that checks whether a word is a palindrome, ignoring punctuation and case.

Expected Output:

>>Output comparison + student explanation on how examples helped the model.

Prompt1:

>>generating a function that checks whether a word is a palindrome, ignoring punctuation and case give a code.

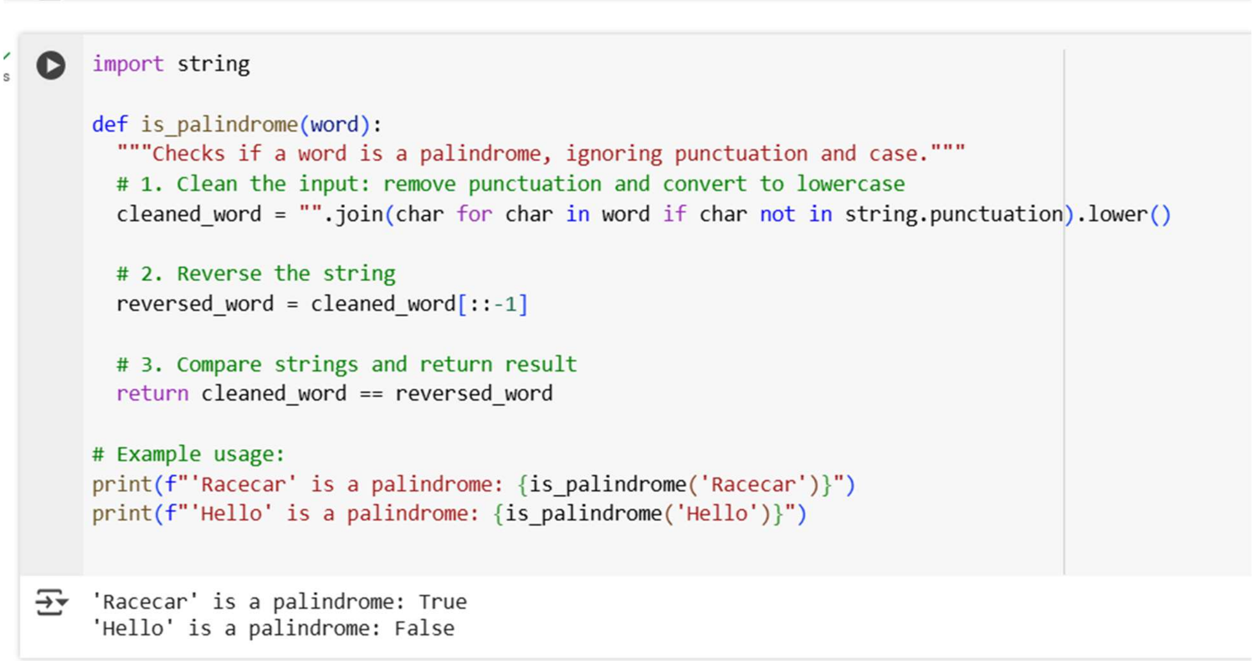


```
The domain of invalid-email is: Invalid email format
```

```python
import string

def is_palindrome(word):
  """Checks if a word is a palindrome, ignoring punctuation and case."""
  # 1. Clean the input: remove punctuation and convert to lowercase
  cleaned_word = "".join(char for char in word if char not in string.punctuation).lower()

  # 2. Reverse the string
  reversed_word = cleaned_word[::-1]

  # 3. Compare strings and return result
  return cleaned_word == reversed_word
```

Prompt2:

>> Generating a function that checks whether a word is a palindrome, ignoring punctuation and case by taking input output as racecar and hello.



```python
import string

def is_palindrome(word):
  """Checks if a word is a palindrome, ignoring punctuation and case."""
  # 1. Clean the input: remove punctuation and convert to lowercase
  cleaned_word = "".join(char for char in word if char not in string.punctuation).lower()

  # 2. Reverse the string
  reversed_word = cleaned_word[::-1]

  # 3. Compare strings and return result
  return cleaned_word == reversed_word

# Example usage:
print(f"'Racecar' is a palindrome: {is_palindrome('Racecar')}")
print(f"'Hello' is a palindrome: {is_palindrome('Hello')}")
```

```
'Racecar' is a palindrome: True
'Hello' is a palindrome: False
```

Observation:

>>This code checks if a word or phrase reads the same forwards and backward, even if you ignore spaces and punctuation and don't worry about uppercase or lowercase letters. It correctly identified ‘racecar' and 'Hello' as palindromes in the examples.

Explanation:

>> This code checks if a word or sentence is the same forwards and backward. It ignores things like commas, spaces, and whether letters are big or small. For example, "racecar" reads the same forwards and backward, so the code says it's a palindrome.

TASK -5

Description:

>> Use few-shot prompting with 3 sample inputs to generate a function that determines the maximum of three numbers without using the built-in max() function.

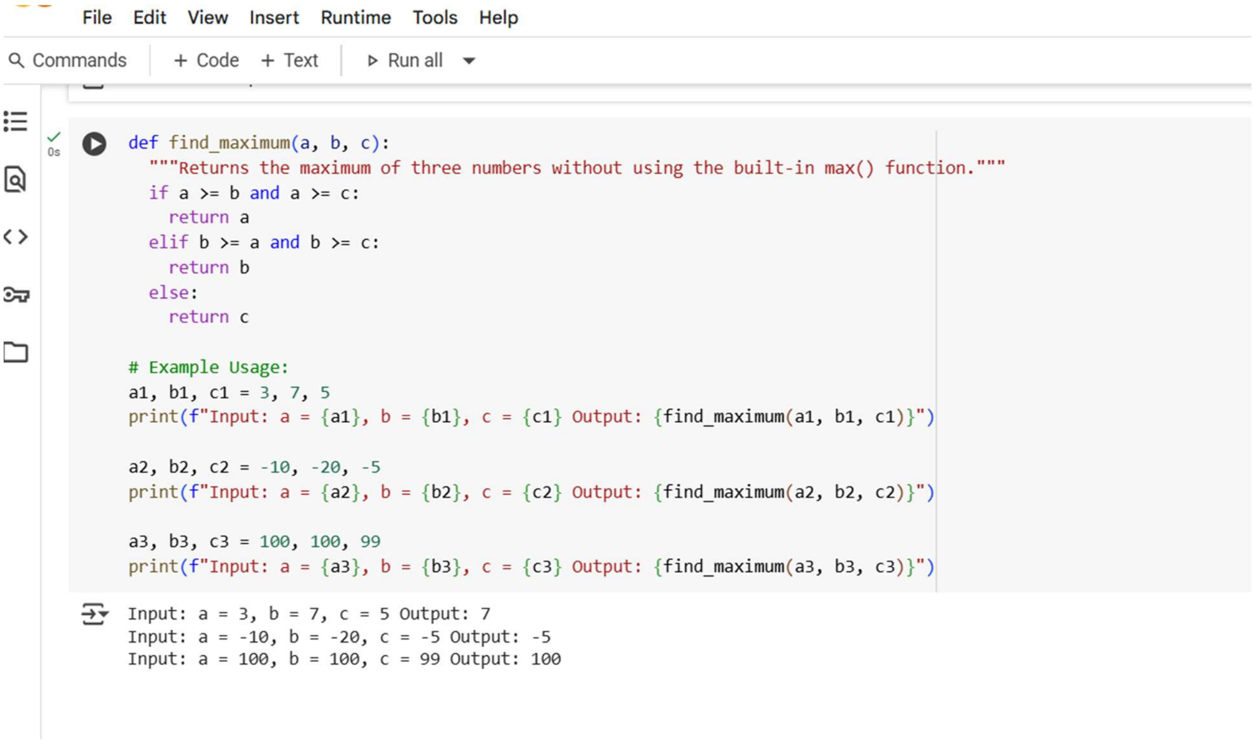
Expected Output:

>>A function that handles all cases with correct logic based on example patterns.

Prompt:

>> Write a Python function that returns the maximum of three numbers without using the built-in max() function. Example 1 Input: a = 3, b = 7, c = 5 Output: 7 Example 2 Input: a = -10, b = -20, c = -5 Output: -5 Example 3 Input: a = 100, b = 100, c = 99 Output: 100 give a

code



```python
def find_maximum(a, b, c):
  """Returns the maximum of three numbers without using the built-in max() function."""
  if a >= b and a >= c:
    return a
  elif b >= a and b >= c:
    return b
  else:
    return c

# Example Usage:
a1, b1, c1 = 3, 7, 5
print(f"Input: a = {a1}, b = {b1}, c = {c1} Output: {find_maximum(a1, b1, c1)}")

a2, b2, c2 = -10, -20, -5
print(f"Input: a = {a2}, b = {b2}, c = {c2} Output: {find_maximum(a2, b2, c2)}")

a3, b3, c3 = 100, 100, 99
print(f"Input: a = {a3}, b = {b3}, c = {c3} Output: {find_maximum(a3, b3, c3)}")
```

```
Input: a = 3, b = 7, c = 5 Output: 7
Input: a = -10, b = -20, c = -5 Output: -5
Input: a = 100, b = 100, c = 99 Output: 100
```

Observation:

>>This code looks at three numbers and finds the largest one by comparing them directly, without using the built-in max() tool. For instance, with the numbers 3, 7, and 5, it correctly identifies 7 as the biggest.

Explanation:

>>We want to find the biggest number among three: a, b, and c. So we compare them using if statements. First, we check if a is bigger than both b and c. If yes, then a is the biggest. If not, we check if b is bigger than both a and c. If yes, then b is the biggest. If neither of those is true, then c must be the biggest. This way, we find the largest number without using Python’s max() function