

COURSE : AI ASSISTED CODING

NAME : G.SAI SPOORTHY

HT.NO : 2403A52290

BATCH : 01

Task Description#1

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

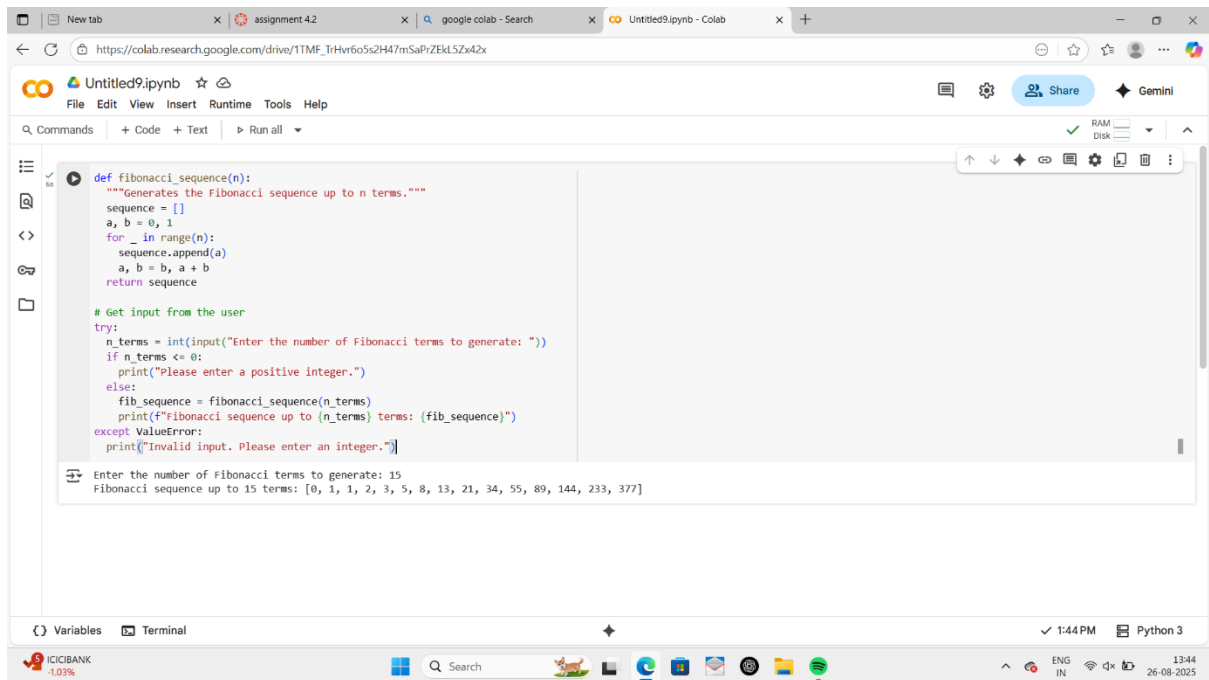
PROMPT:

write a code in Python function to generate the Fibonacci sequence up to n term with input is given by user.

Expected Output#1

- A working function without using any sample inputs/outputs

CODE :



The screenshot shows a Google Colab notebook titled 'Untitled9.ipynb'. The code defines a function `fibonacci_sequence(n)` that generates a Fibonacci sequence up to `n` terms. It then prompts the user to enter the number of terms, handles invalid input (non-integer or negative), and prints the resulting sequence. The output shows the sequence for 15 terms: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377].

```
def fibonacci_sequence(n):
    """Generates the Fibonacci sequence up to n terms."""
    sequence = []
    a, b = 0, 1
    for _ in range(n):
        sequence.append(a)
        a, b = b, a + b
    return sequence

# Get input from the user
try:
    n_terms = int(input("Enter the number of Fibonacci terms to generate: "))
    if n_terms <= 0:
        print("Please enter a positive integer.")
    else:
        fib_sequence = fibonacci_sequence(n_terms)
        print(f"Fibonacci sequence up to {n_terms} terms: {fib_sequence}")
except ValueError:
    print("Invalid input. Please enter an integer.")
```

Enter the number of Fibonacci terms to generate: 15
Fibonacci sequence up to 15 terms: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377]

OBSERVATION:

generate_fibonacci_sequence that generates the Fibonacci sequence up to a specified number of terms. It then prompts the user to enter the desired number of terms, handles potential invalid input (non-integer or negative numbers), and prints the resulting Fibonacci sequence

Task Description#2

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

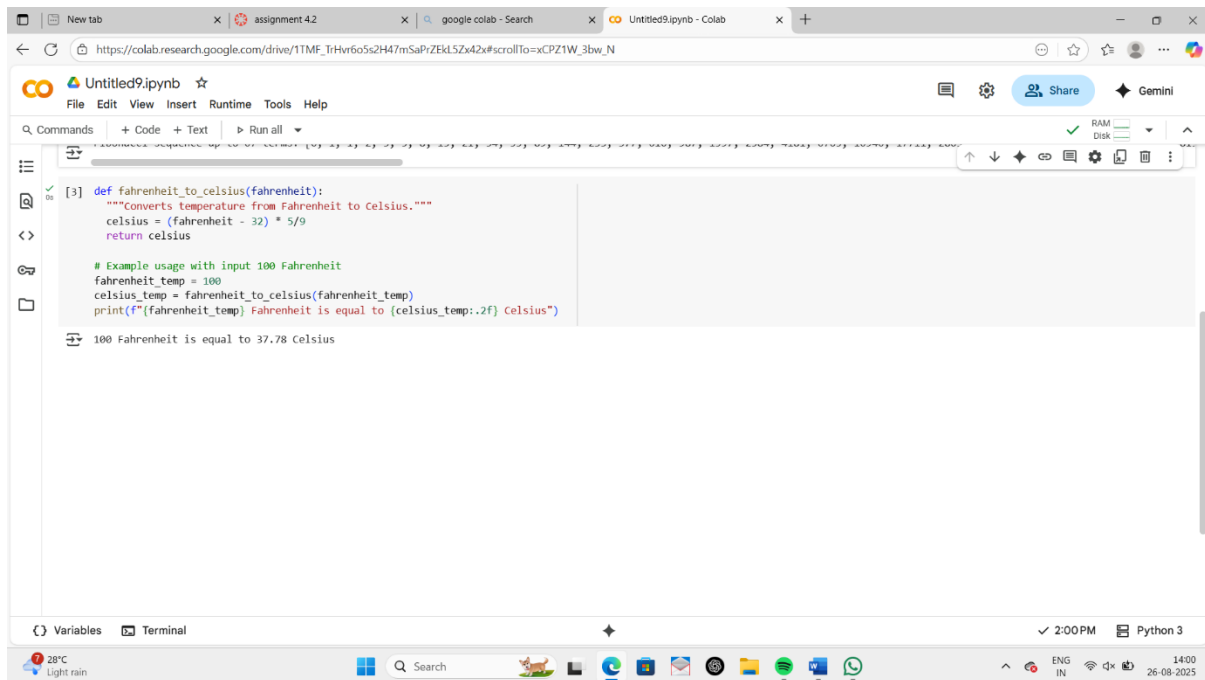
Expected Output#2

- A correct conversion function guided by the single example

PROMPT:

Generate a function that converts Fahrenheit to Celsius take input as 100 and output as 37.78

CODE:



The screenshot shows a Google Colab notebook titled 'Untitled9.ipynb'. The code cell contains a function to convert Fahrenheit to Celsius. The function is defined as follows:

```
[3] def fahrenheit_to_celsius(fahrenheit):  
    """Converts temperature from Fahrenheit to Celsius."""  
    celsius = (fahrenheit - 32) * 5/9  
    return celsius  
  
# Example usage with input 100 Fahrenheit  
fahrenheit_temp = 100  
celsius_temp = fahrenheit_to_celsius(fahrenheit_temp)  
print(f"{fahrenheit_temp} Fahrenheit is equal to {celsius_temp:.2f} Celsius")
```

The output of the code is:

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

OBSERVATION:

This code defines a function to convert Fahrenheit to Celsius and prompts the user for the temperature in Fahrenheit.

Task Description#3

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

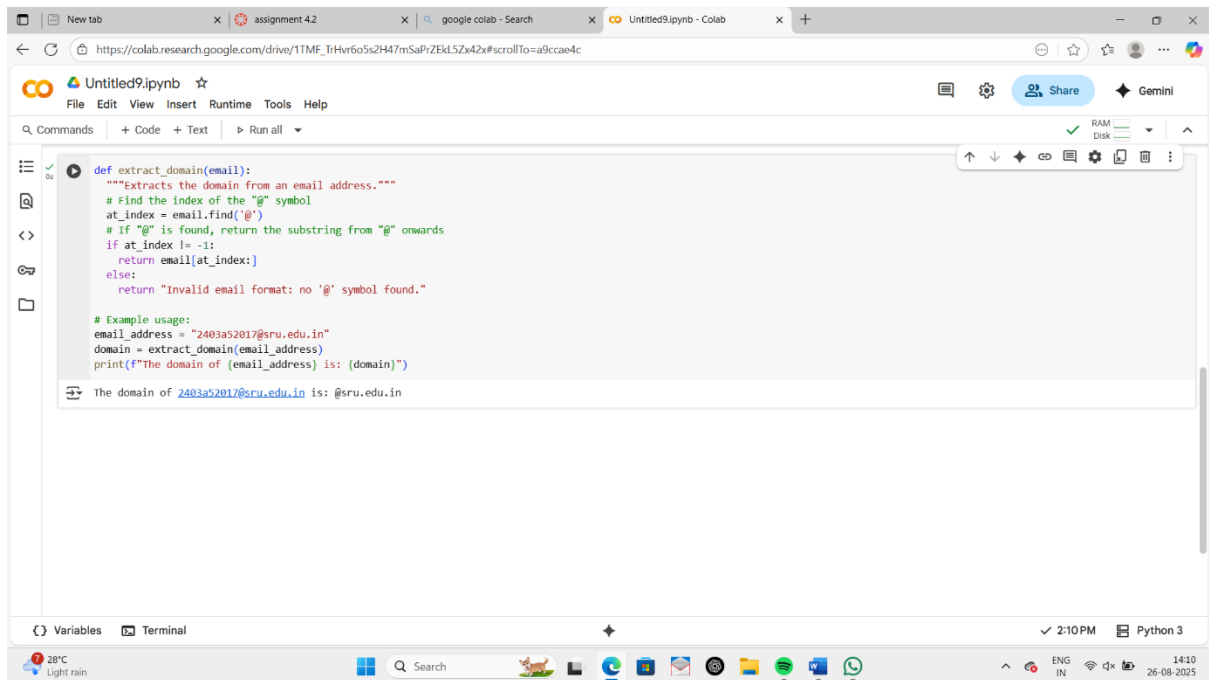
Expected Output#3

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

Prompt:

Generate a code to extract the domain from the email address take input as 2403a52017@sru.edu.in and output=@sru.edu.in

Code:



The screenshot shows a Google Colab notebook with a single code cell. The code defines a function `extract_domain(email)` that takes an email address as input. It uses `email.find('@')` to find the index of the '@' symbol. If the symbol is found, it returns the substring from that index onwards. If not found, it returns "Invalid email format: no '@' symbol found." An example usage is provided: `email_address = "2403a52017@sru.edu.in"`, `domain = extract_domain(email_address)`, and `print(f"The domain of {email_address} is: {domain}")`. The output of the cell shows the result: "The domain of 2403a52017@sru.edu.in is: @sru.edu.in".

```
def extract_domain(email):  
    """Extracts the domain from an email address."""  
    # Find the index of the "@" symbol  
    at_index = email.find('@')  
    # If "@" is found, return the substring from "@" onwards  
    if at_index != -1:  
        return email[at_index:]  
    else:  
        return "Invalid email format: no '@' symbol found."  
  
# Example usage:  
email_address = "2403a52017@sru.edu.in"  
domain = extract_domain(email_address)  
print(f"The domain of {email_address} is: {domain}")  
  
The domain of 2403a52017@sru.edu.in is: @sru.edu.in
```

OBSERVATION:

The code in cell defines function `extract_domain` that takes an email address as input. It attempts to split the email address at the "@" symbol and returns the part after the "@" with a "@" prefix to represent the domain. It also includes error handling using a try-except block to catch `IndexError` if the input is not a valid email format (i.e., it doesn't contain an "@" symbol), in which case it returns "Invalid email format".

Task Description#4

- 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#4

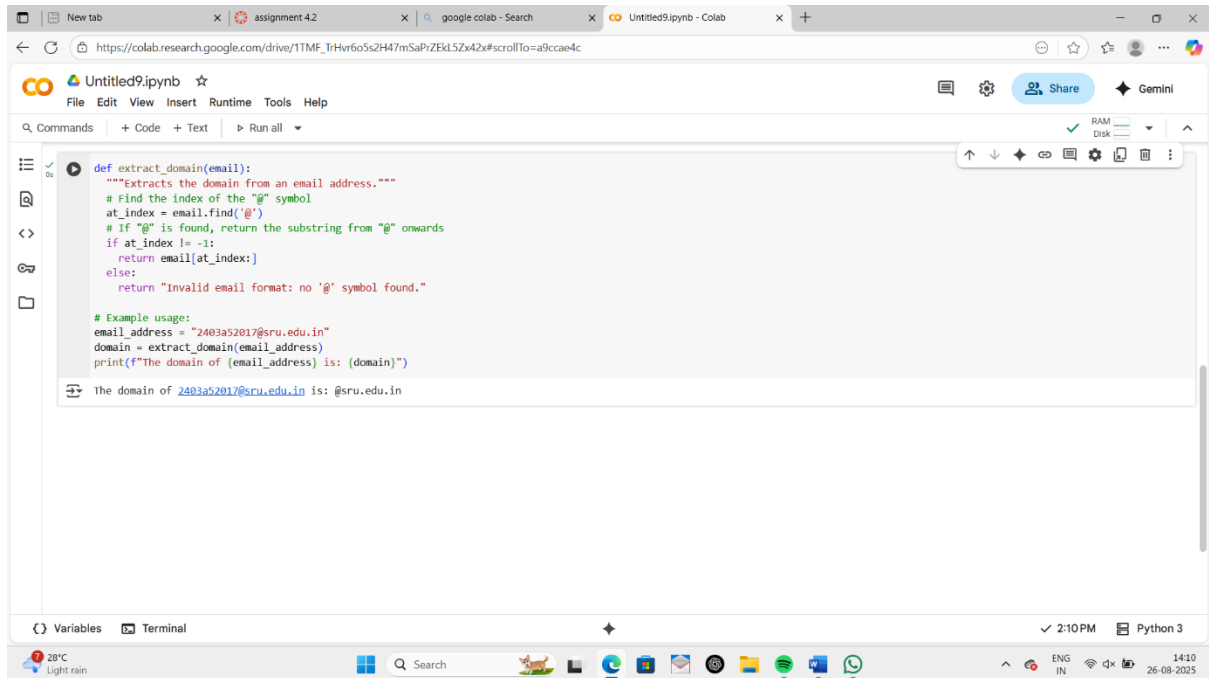
- Output comparison + student explanation on how examples helped the mode

PROMPT:

ZEROSHOT:

generate a function that checks whether a word is a palindrome, ignoring punctuation and case?
with user input?

Code



The screenshot shows a Google Colab notebook titled 'Untitled9.ipynb'. The code defines a function `extract_domain(email)` that extracts the domain from an email address. The function uses `email.find('@')` to find the index of the '@' symbol. If found, it returns the substring from that index onwards. If not found, it returns an error message. An example usage is provided, showing the domain of '2403a52017@sru.edu.in' is '@sru.edu.in'.

```
def extract_domain(email):  
    """Extracts the domain from an email address."""  
    # Find the index of the "@" symbol  
    at_index = email.find('@')  
    # If "@" is found, return the substring from "@" onwards  
    if at_index != -1:  
        return email[at_index:]  
    else:  
        return "Invalid email format: no '@' symbol found."  
  
# Example usage:  
email_address = "2403a52017@sru.edu.in"  
domain = extract_domain(email_address)  
print(f"The domain of {email_address} is: {domain}")  
  
The domain of 2403a52017@sru.edu.in is: @sru.edu.in
```

Few shot:

Prompt:

Generate a code to check whether input is palindrome or not by user given input is mom and output as mom

Code:

The first screenshot shows the initial code in a Google Colab notebook. The code defines a function `is_palindrome_alt(word)` that checks if a word is a palindrome by removing punctuation and converting to lowercase. It uses two pointers, `left` and `right`, to compare characters from both ends of the string. The code also includes a user input prompt and a print statement to display the result.

```
import string

def is_palindrome_alt(word):
    """
    Checks if a word is a palindrome, ignoring punctuation and case using a different approach.

    Args:
        word: The word to check.

    Returns:
        True if the word is a palindrome, False otherwise.
    """
    # Remove punctuation and convert to lowercase
    cleaned_word = ''.join(char for char in word if char not in string.punctuation).lower()

    # Use two pointers to check for palindrome
    left = 0
    right = len(cleaned_word) - 1

    while left < right:
        if cleaned_word[left] != cleaned_word[right]:
            return False
        left += 1
        right -= 1
    return True

# Get input from the user
word = input("Enter a word to check if it is a palindrome: ")

if is_palindrome_alt(word):
    print(f"'{word}' is a nallindrome.")
```

The second screenshot shows the same notebook after execution. The user has entered "mom" as input, and the output is "'mom' is a palindrome." The code has been modified to include a print statement for the "else" case, which is not visible in the first screenshot.

```
word = input("Enter a word to check if it is a palindrome: ")

if is_palindrome_alt(word):
    print(f"'{word}' is a palindrome.")
else:
    print(f"'{word}' is not a palindrome.")
```

Enter a word to check if it is a palindrome: mom
'mom' is a palindrome.

Observation

"Looking at palindrome functions again, I see how the examples provided helped clarify the requirements. The examples like mom specifically showed that punctuation and case should be ignored. If only a simple example like 'level' was given, the code might not have included the steps to remove punctuation and convert to lowercase. This shows how giving a few varied examples (few-

shot) is really useful for making sure the code handles different situations correctly, compared to just one simple example (one-shot)."

Task Description#5

- 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.

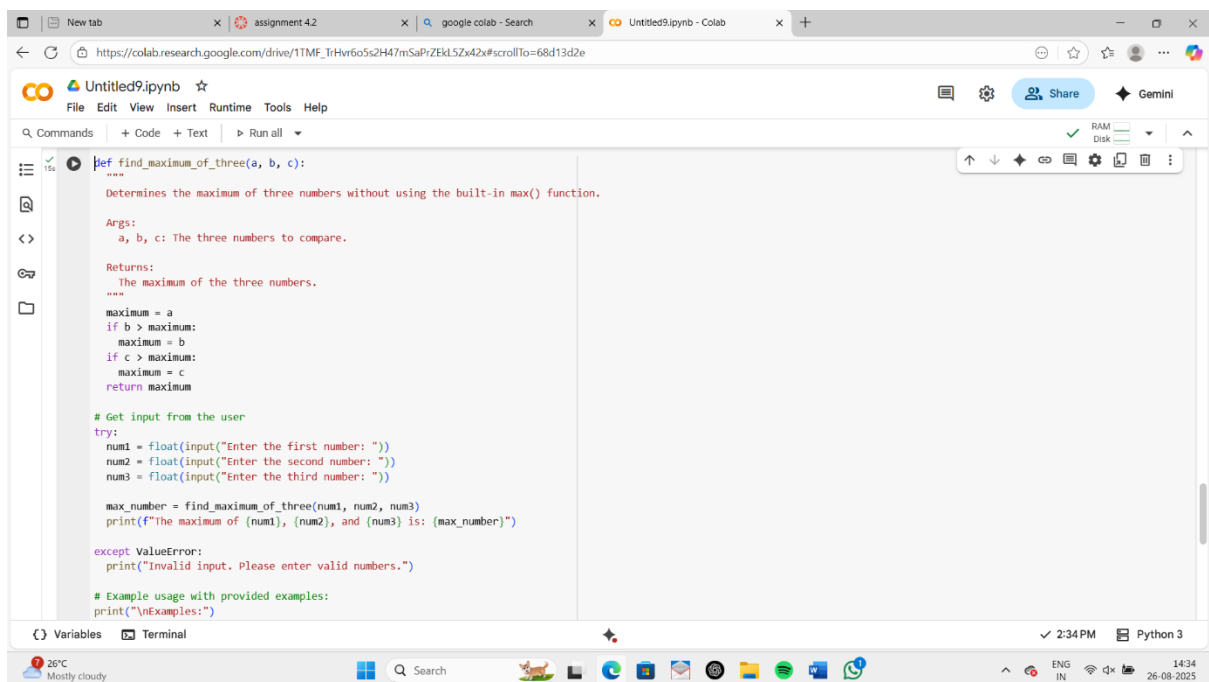
Expected Output#5

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

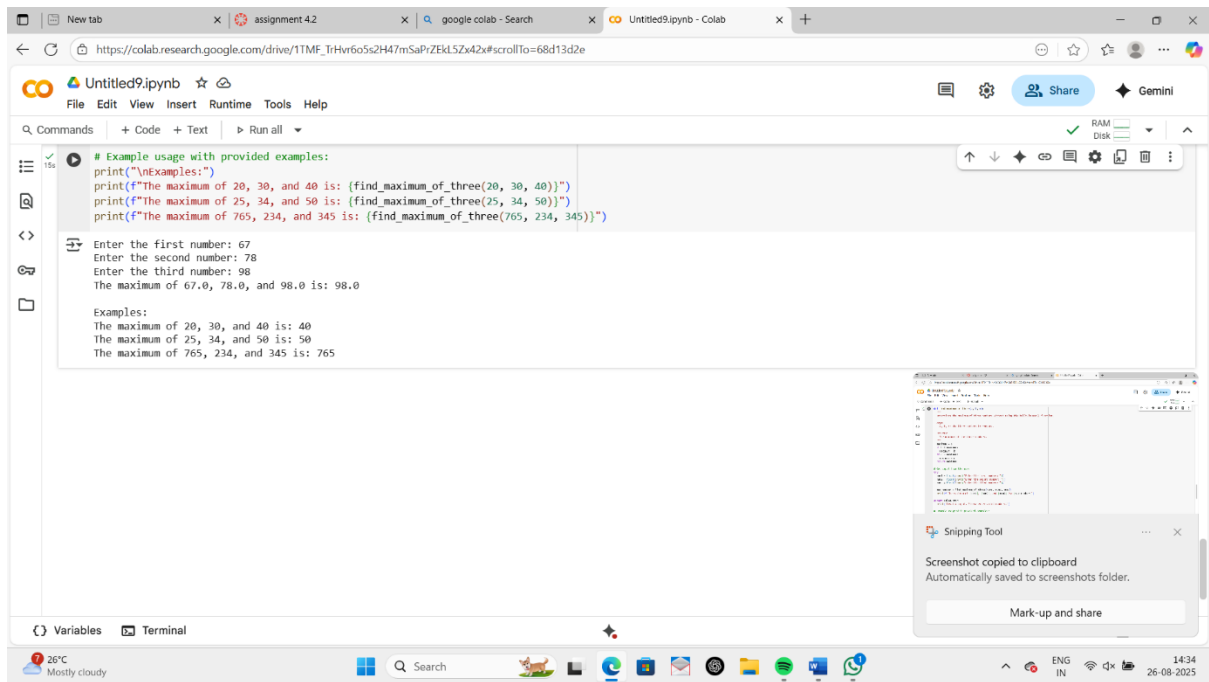
Prompt

create a python function that determines the maximum of three numbers without using the built-in max() function with user given input? ex1: a=20,b=30,c=40;output=40
ex2:a=25,b=34,c=50;output=50 ex3:a=765,b=234,c=345;output=765?

Code



```
def find_maximum_of_three(a, b, c):  
    """  
    Determines the maximum of three numbers without using the built-in max() function.  
  
    Args:  
        a, b, c: The three numbers to compare.  
  
    Returns:  
        The maximum of the three numbers.  
    """  
    maximum = a  
    if b > maximum:  
        maximum = b  
    if c > maximum:  
        maximum = c  
    return maximum  
  
# Get input from the user  
try:  
    num1 = float(input("Enter the first number: "))  
    num2 = float(input("Enter the second number: "))  
    num3 = float(input("Enter the third number: "))  
  
    max_number = find_maximum_of_three(num1, num2, num3)  
    print(f"The maximum of {num1}, {num2}, and {num3} is: {max_number}")  
  
except ValueError:  
    print("Invalid input. Please enter valid numbers.")  
  
# Example usage with provided examples:  
print("\nExamples:")
```



Observation

The code in cell defines a function `find_maximum_of_three` that takes three numbers as input. It initializes a variable `maximum` with the value of the first number. Then, it compares the second number with the current maximum and updates maximum if the second number is greater. Finally, it compares the third number with the current maximum and updates it again if the third number is greater. The function returns the final maximum value. The code also includes a part that prompts the user for three numbers and calls this function to find and print the maximum, with error handling for non-numeric input.