**LAB ASSIGNMENT-4.2**

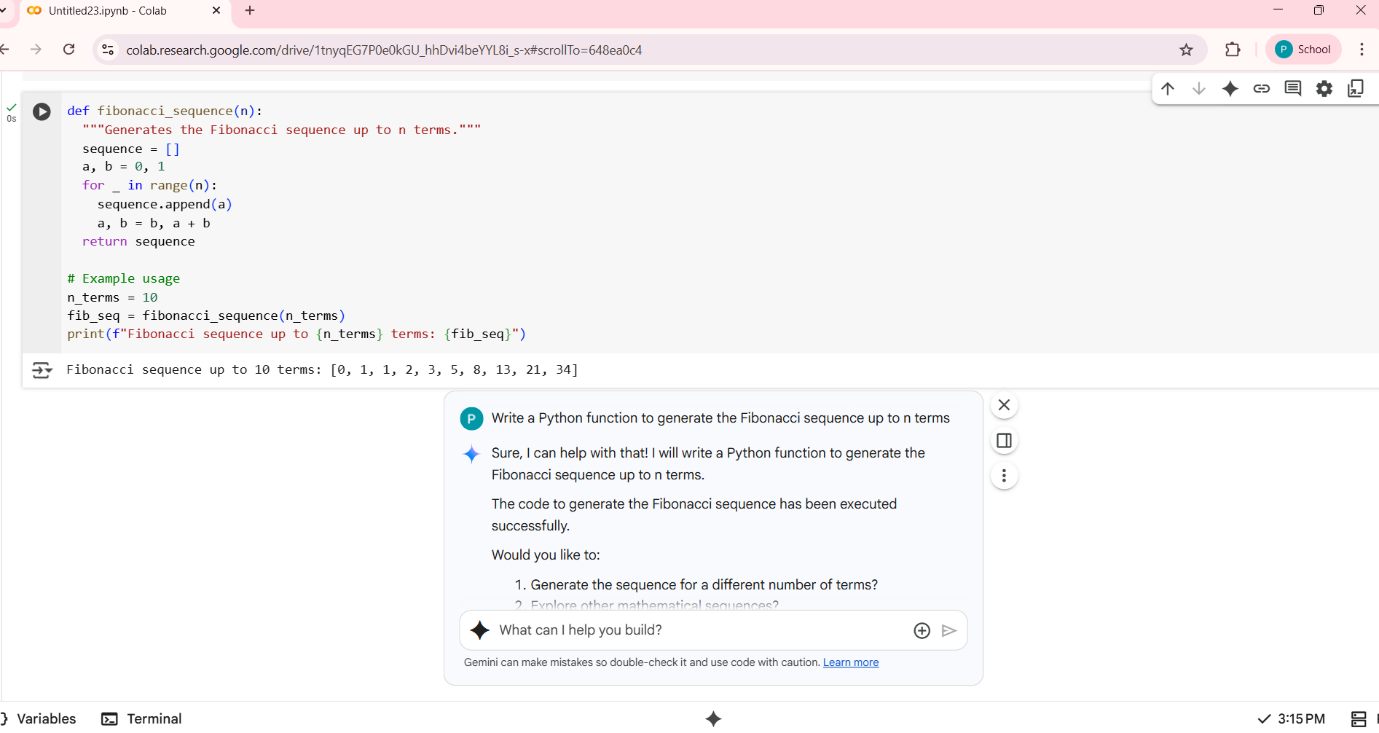
**Name:**P.Navya

**HTNO:**2403a51331

**Batch:**24BTCAICSB13

**Assignment-4.2**

**Task-1:**

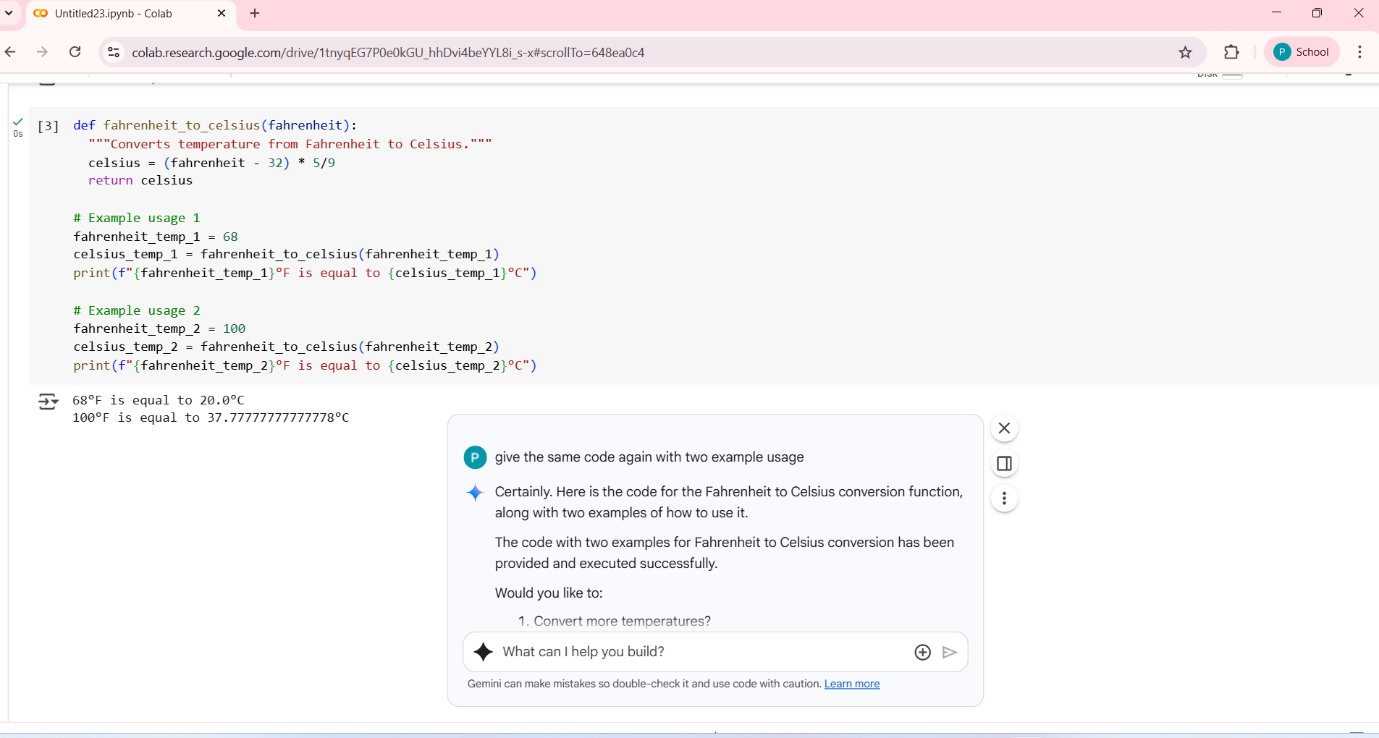
* Zero-shot: Prompt AI with only the instruction — Write a Python function to generate the Fibonacci sequence up to n terms
* 

**Explanation:**

* **def fibonacci\_sequence(n)::** This line defines the function named fibonacci\_sequence that takes one argument, n, which represents the number of terms in the sequence you want to generate.
* **"""Generates the Fibonacci sequence up to n terms.""":** This is a docstring, which explains what the function does.
* **sequence = []:** This initializes an empty list called sequence. This list will store the generated Fibonacci numbers.
* **a, b = 0, 1:** This initializes two variables, a and b, with the first two numbers of the Fibonacci sequence, which are 0 and 1.
* **for \_ in range(n**):: This starts a for loop that will iterate n times. The \_ is used as a variable name when you don't need to use the loop counter within the loop.
* **sequence.append(a):** In each iteration, the current value of a (which represents the next Fibonacci number) is added to the sequence list**.**
* **a, b = b, a + b:** This is the core of the Fibonacci logic. It updates the values of a and b for the next iteration. The new a becomes the current b, and the new b becomes the sum of the current a and b.
* **return sequence:** After the loop finishes, the function returns the sequence list containing the generated Fibonacci numbers.

**Task-2:**

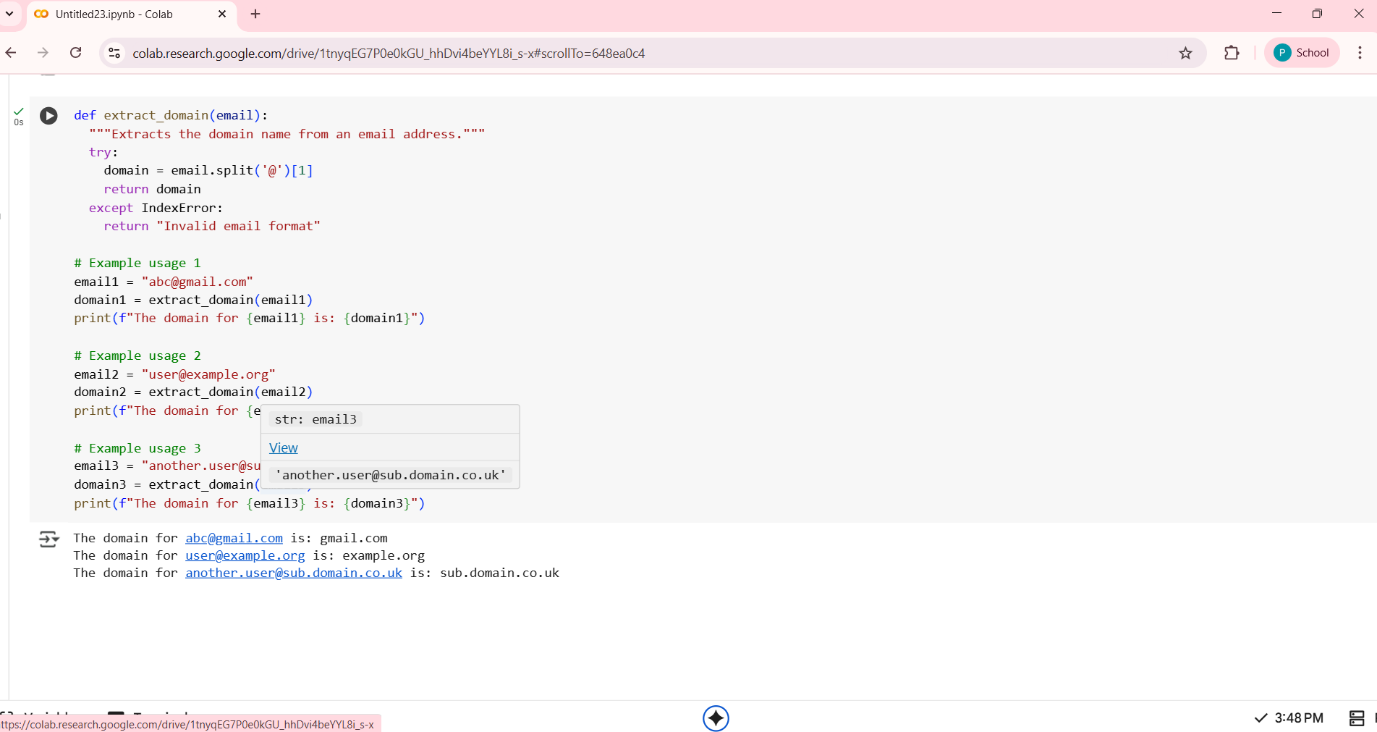
* One-shot: Provide one example: Input: 100, Output: 37.78 to help AI generate a function that converts Fahrenheit to Celsius.



**Explanation:**

* **def fahrenheit\_to\_celsius(fahrenheit):**: This line defines a function named fahrenheit\_to\_celsius that takes one argument, fahrenheit, which is the temperature in degrees Fahrenheit that you want to convert.
* **"""Converts temperature from Fahrenheit to Celsius."""**: This is a docstring explaining what the function does.
* **celsius = (fahrenheit - 32) \* 5/9**: This is the core of the conversion. It applies the standard formula to convert Fahrenheit to Celsius: subtract 32 from the Fahrenheit temperature and then multiply the result by 5/9. The result is stored in the celsius variable.
* **return celsius**: The function returns the calculated Celsius temperature.

**Task-3:**

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

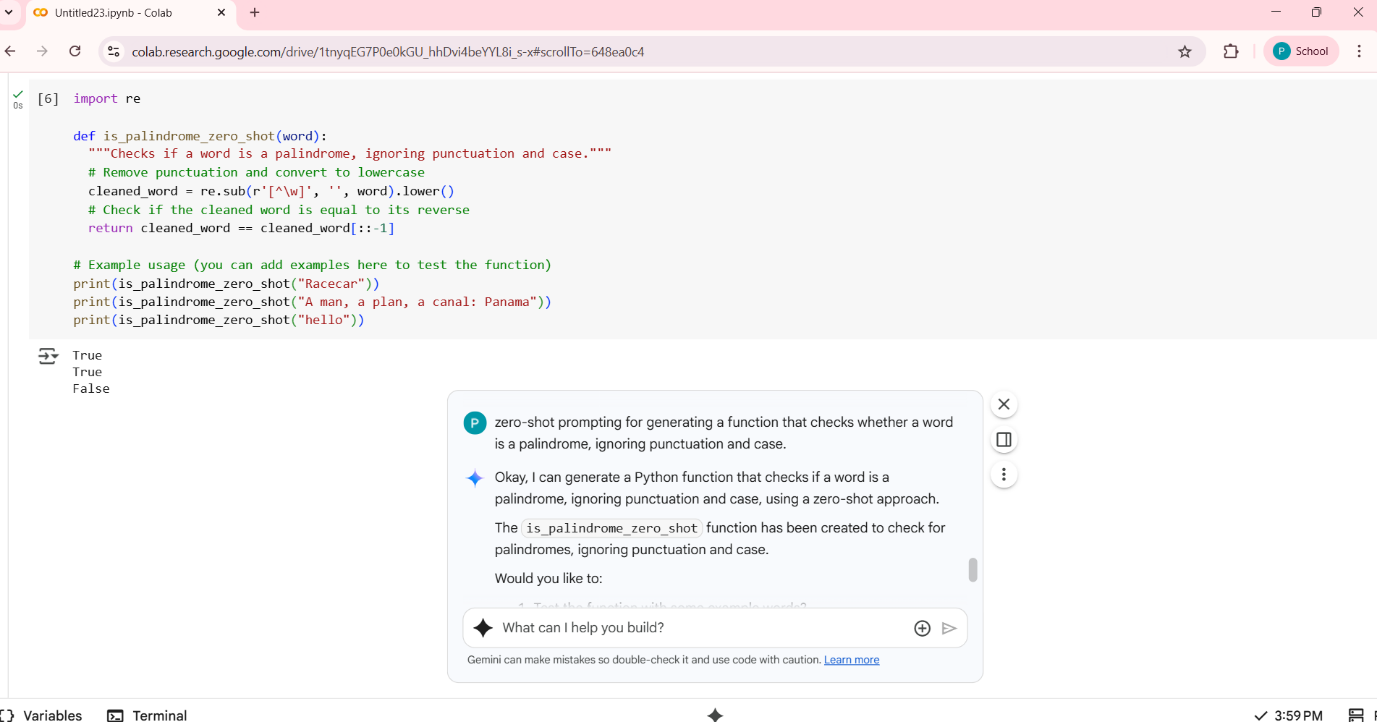
**Explanation:**

* **def extract\_domain(email):**: This line defines a function named extract\_domain that takes one argument, email, which is the email address as a string.
* **"""Extracts the domain name from an email address."""**: This is a docstring explaining the function's purpose.
* **try:**: This starts a try block, which is used for error handling. Code within this block is attempted, and if an error occurs, the code in the except block is executed.
* **domain = email.split('@')[1]**: This is the core logic for extracting the domain.
  + email.split('@') splits the email string into a list of substrings using the "@" symbol as the delimiter. For example, "abc@gmail.com".split('@') would result in ['abc', 'gmail.com'].
  + [1] accesses the element at index 1 of the resulting list, which is the part after the "@" symbol (the domain name).
  + The extracted domain is stored in the domain variable.
* **return domain**: If the split and indexing are successful, the function returns the extracted domain.
* **except IndexError:**: This is the start of the except block, which catches an IndexError. An IndexError will occur if the split('@') operation does not produce a list with at least two elements (i.e., if there is no "@" symbol in the email address).
* **return "Invalid email format"**: If an IndexError occurs (meaning the email format is invalid because it lacks an "@" symbol), the function returns the string "Invalid email format"

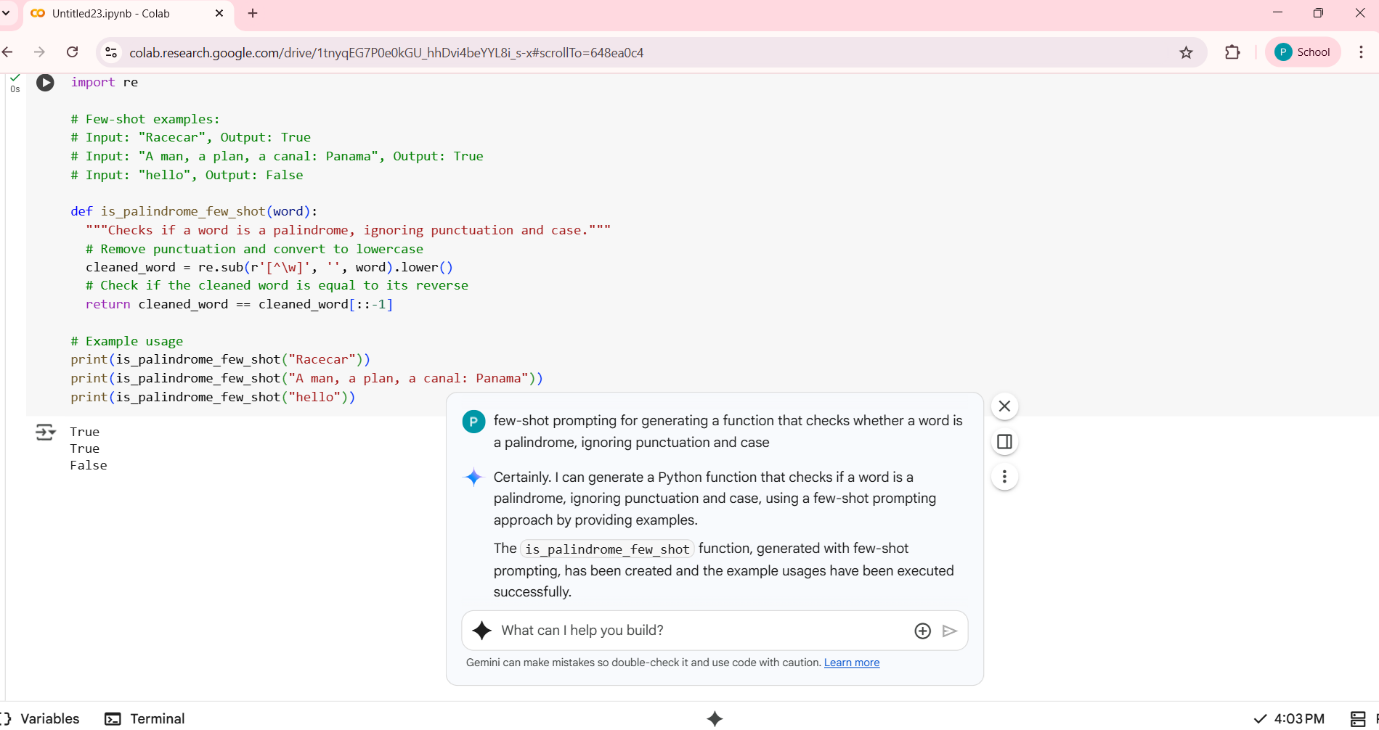
**Task-4**

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

**Zero-shot:**

****

**Few-shot:**

****

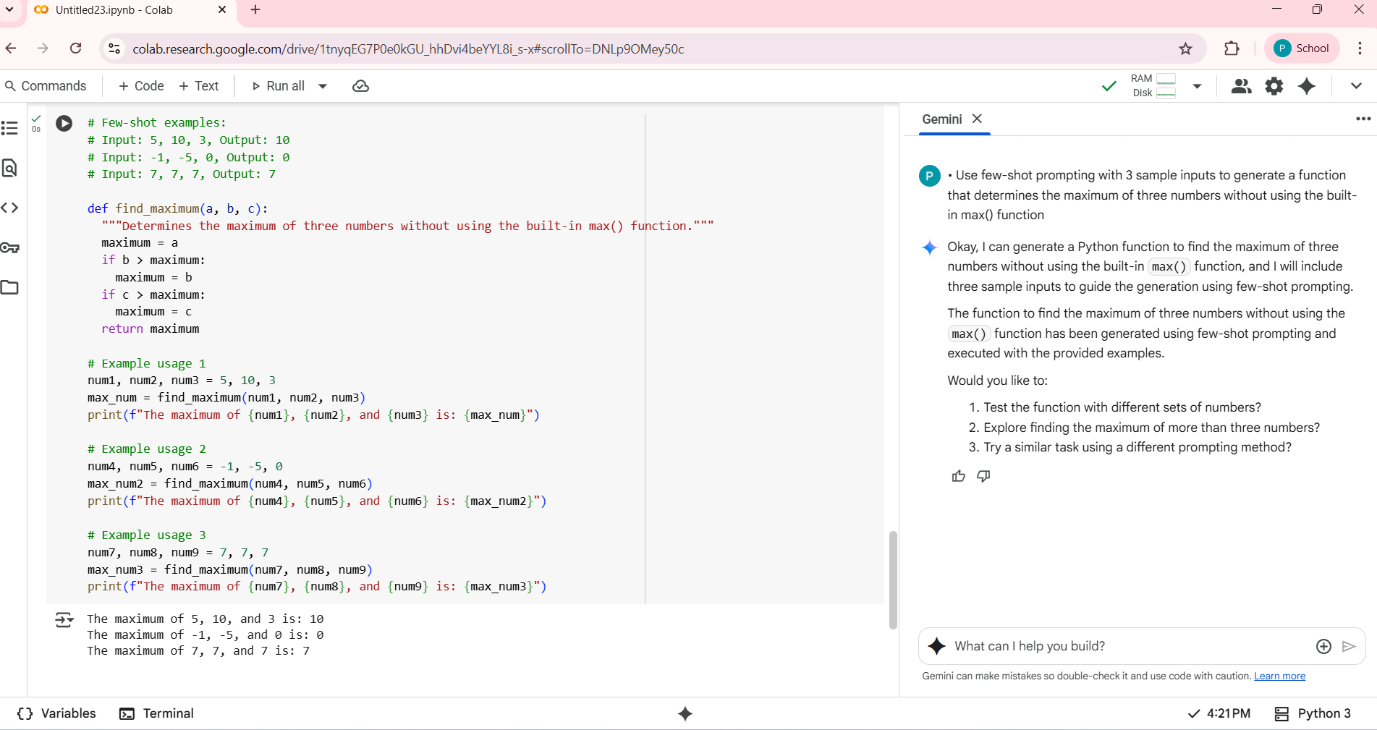
**Comparision:**

| **Feature** | **Zero-Shot Prompting** | **Few-Shot Prompting** |
| --- | --- | --- |
| Examples Provided | No examples of input/output are given. | A few examples of input/output are given to guide the model. |
| Guidance Level | Relies solely on the model's pre-training and understanding of the task description. | Provides explicit examples to demonstrate the desired behavior and output format. |
| Complexity of Task | More suitable for simpler, well-defined tasks where the expected output is unambiguous. | Can be more effective for complex or nuanced tasks where examples help clarify the requirements. |
| Outcome (in this case) | Generated a correct function for checking palindromes, ignoring punctuation and case. | Generated the same correct function for checking palindromes, ignoring punctuation and case, as the task was relatively simple. |
| Code Generated | is\_palindrome\_zero\_shot function | is\_palindrome\_few\_shot function (identical code to zero-shot in this case) |

**Explanation:**

* The two functions to check for palindromes, one using a zero-shot prompt (is\_palindrome\_zero\_shot) and one using a few-shot prompt (is\_palindrome\_few\_shot).
* In this particular case, the code generated for both functions and examples is the same

**Task-5:**

* **Prompt:** 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**.**
* 

**Explanation:**

* **def find\_maximum(a, b, c)::** This line defines a function named find\_maximum that takes three arguments: a, b, and c, which are the three numbers you want to compare**.**
* **"""Determines the maximum of three numbers without using the built-in max() function.""":** This is a docstring explaining the function's purpose.
* **maximum = a:** This line initializes a variable called maximum and assumes that the first number a is the maximum.
* **if b > maximum::** This is a conditional statement. It checks if the second number b is greater than the current value of maximum.
* **maximum = b:** If b is indeed greater than maximum, the value of maximum is updated to b.
* **if c > maximum:: This is another conditional statement. It checks if the third number c is greater than the current value of maximum.**
* **maximum = c: If c is greater than the current value of maximum, the value of maximum is updated to c.**
* **return maximum: After checking both b and c against the current maximum, the function returns the final value of maximum, which will be the largest of the three input numbers.**