

<< LAB ASSIGNMENT- 4.2 >>

➔ AI ASSISTED CODING

➤ Task Description#1

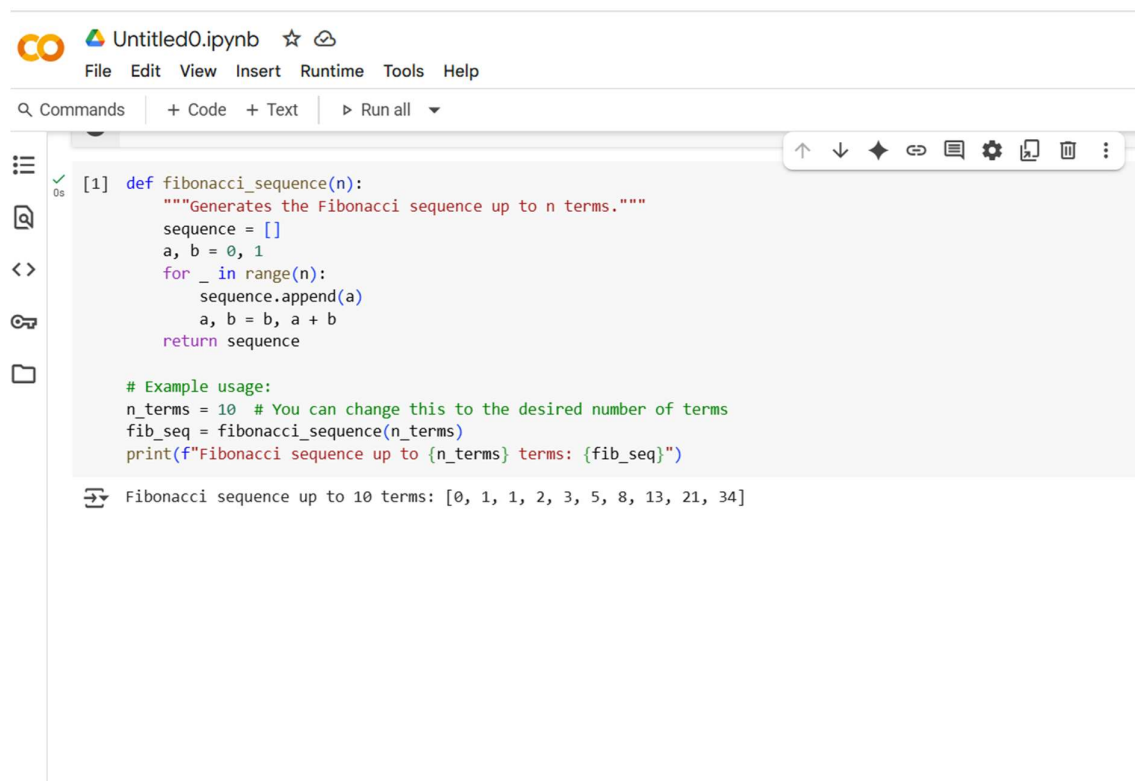
zero-shot: Prompt AI with only the instruction — Write a Python function to generate the Fibonacci sequence up to n terms

➤ Expected output :

Working function without using any sample inputs/outputs.

➤ Prompt#1

Give a python code to generate the fibonacci sequence upto n terms



The screenshot shows a Jupyter Notebook titled 'Untitled0.ipynb'. The code cell contains a function definition for 'fibonacci_sequence(n)' which generates the Fibonacci sequence up to n terms. The function initializes a list 'sequence' and variables 'a' and 'b'. It then uses a for loop to calculate the sequence. Below the function definition, there is an example usage where 'n_terms' is set to 10, the function is called, and the result is printed. The output of the code is 'Fibonacci sequence up to 10 terms: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]'.

```
[1] 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  
  
# Example usage:  
n_terms = 10 # You can change this to the desired number of terms  
fib_seq = fibonacci_sequence(n_terms)  
print(f"Fibonacci sequence up to {n_terms} terms: {fib_seq}")
```

Fibonacci sequence up to 10 terms: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]

➤ Explanation of code:

The code starts with 0 and 1, and then repeatedly adds the last two numbers to get the next number in the sequence, storing them in a list until it reaches the desired number of terms.

➤ **Observation:**

The code successfully generates the first 10 terms of the Fibonacci sequence.

The sequence starts with 0 and 1.

Each subsequent number is the sum of the two preceding ones (e.g., $1+1=2$, $1+2=3$, $2+3=5$, and so on).

The output clearly labels the generated sequence.

➤ **Task Description#2**

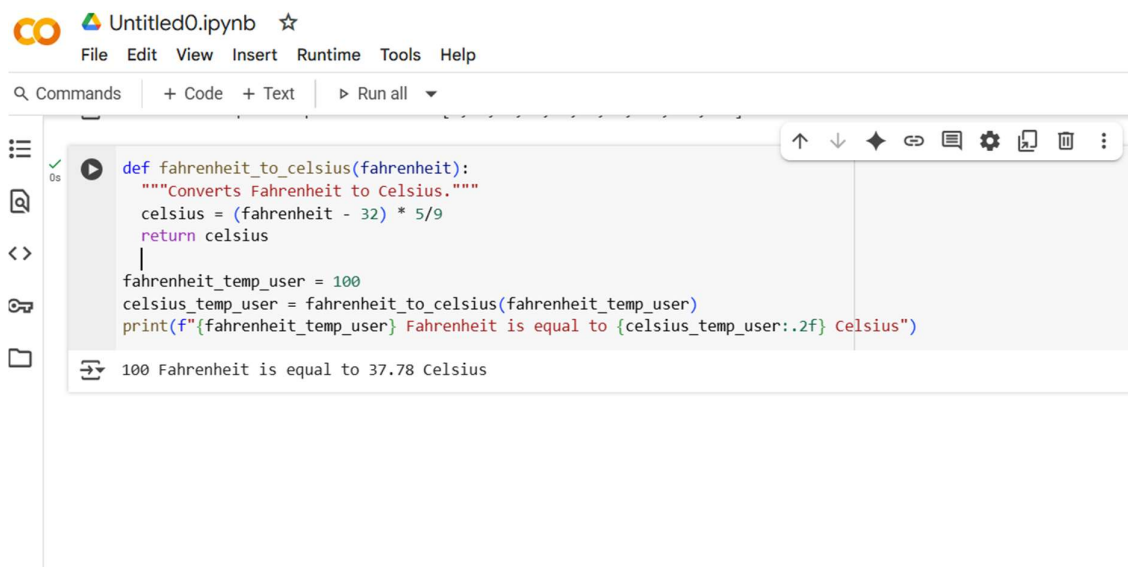
One-shot: Provide one example: Input: 100, Output: 37.78 to help AI generate a function that converts Fahrenheit to Celsius.

➤ **Expected output:**

A correct conversion function guided by the single example.

➤ **Prompt#2**

Generate a code that converts Fahrenheit to Celsius with Input: 100, Output: 37.78



The screenshot shows a Jupyter Notebook window titled 'Untitled0.ipynb'. The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a toolbar with icons for commands, code, text, and running all cells. The code cell contains the following Python code:

```
def fahrenheit_to_celsius(fahrenheit):  
    """Converts Fahrenheit to Celsius."""  
    celsius = (fahrenheit - 32) * 5/9  
    return celsius  
  
fahrenheit_temp_user = 100  
celsius_temp_user = fahrenheit_to_celsius(fahrenheit_temp_user)  
print(f"{fahrenheit_temp_user} Fahrenheit is equal to {celsius_temp_user:.2f} Celsius")
```

The output cell below the code shows the result of the execution:

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

➤ **Explanation of code:**

The code takes a temperature in Fahrenheit, applies the conversion formula to calculate the equivalent temperature in Celsius, and then prints the result.

➤ **Observation:**

The code successfully converts 100 Fahrenheit to approximately 37.78 Celsius, which is the correct conversion based on the standard formula.

➤ **Task Description#3**

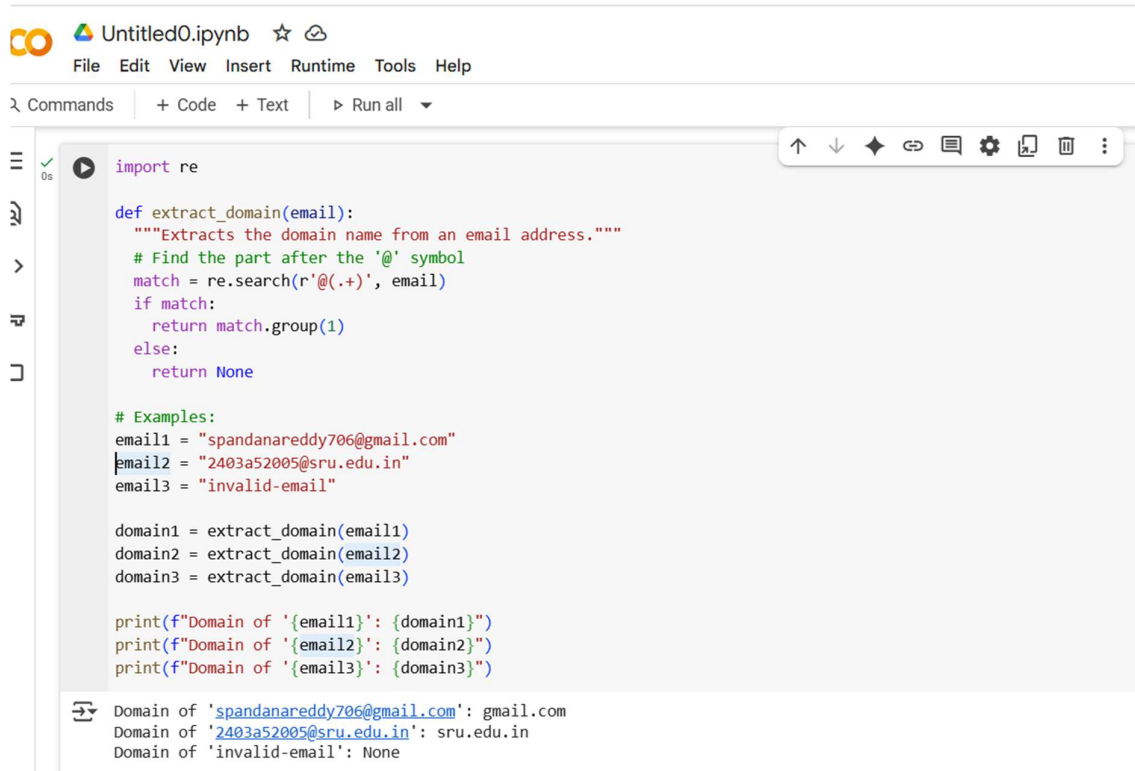
Few-shot: Give 2–3 examples to create a function that extracts the domain name from an email address.

➤ **Expected output:**

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

➤ **Prompt#3**

Generate Python code to create a function that extracts the domain name from a given email address. Include 2-3 examples demonstrating how to use the function with different email formats.



```
import re

def extract_domain(email):
    """Extracts the domain name from an email address."""
    # Find the part after the '@' symbol
    match = re.search(r'@(.+)', email)
    if match:
        return match.group(1)
    else:
        return None

# Examples:
email1 = "spandanareddy706@gmail.com"
email2 = "2403a52005@sru.edu.in"
email3 = "invalid-email"

domain1 = extract_domain(email1)
domain2 = extract_domain(email2)
domain3 = extract_domain(email3)

print(f"Domain of '{email1}': {domain1}")
print(f"Domain of '{email2}': {domain2}")
print(f"Domain of '{email3}': {domain3}")
```

Domain of 'spandanareddy706@gmail.com': gmail.com
Domain of '2403a52005@sru.edu.in': sru.edu.in
Domain of 'invalid-email': None

➤ Explanation of code:

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.

➤ 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.

➤ Task Description#4

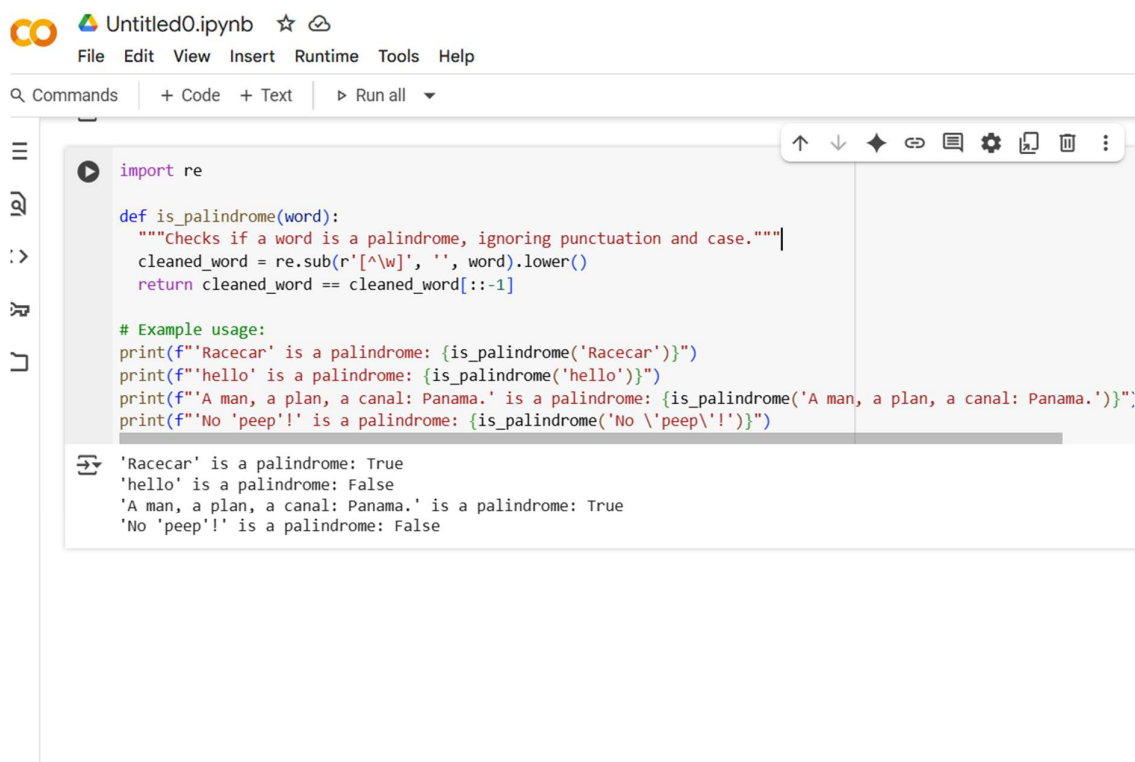
Compare zero-shot vs few-shot prompting for generating a function that checks whether a word is a palindrome, ignoring punctuation and case.

➤ **Expected output:**

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

➤ **Prompt#4**

Write a Python function that determines if a given string is a palindrome. The function should ignore any punctuation and be case-insensitive. Provide examples to demonstrate its usage



The screenshot shows a Jupyter Notebook titled 'Untitled0.ipynb'. The code cell contains a Python function `is_palindrome` that uses `re.sub` to remove non-word characters and `.lower()` to convert the string to lowercase. It then checks if the cleaned string is equal to its reverse. Below the function, there are four example usage lines using `print` statements. The output cell shows the results of these calls: 'Racecar' is a palindrome (True), 'hello' is a palindrome (False), 'A man, a plan, a canal: Panama.' is a palindrome (True), and 'No 'peep'!' is a palindrome (False).

```
import re

def is_palindrome(word):
    """Checks if a word is a palindrome, ignoring punctuation and case."""
    cleaned_word = re.sub(r'[^\w]', '', word).lower()
    return cleaned_word == cleaned_word[::-1]

# Example usage:
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.')}")
print(f"'No 'peep'!' is a palindrome: {is_palindrome('No \'peep\'!')}")
```

```
'Racecar' is a palindrome: True
'hello' is a palindrome: False
'A man, a plan, a canal: Panama.' is a palindrome: True
'No 'peep'!' is a palindrome: False
```

➤ **Explanation of code:**

The code first cleans up the input word by removing punctuation and making it all lowercase. Then, it checks if this cleaned-up word reads the same forwards and backward.

➤ **Observation:**

The code effectively checks if a word is a palindrome by cleaning it up (removing punctuation and making it lowercase) before comparing it to its reversed version.

➤ **Task Description#4**

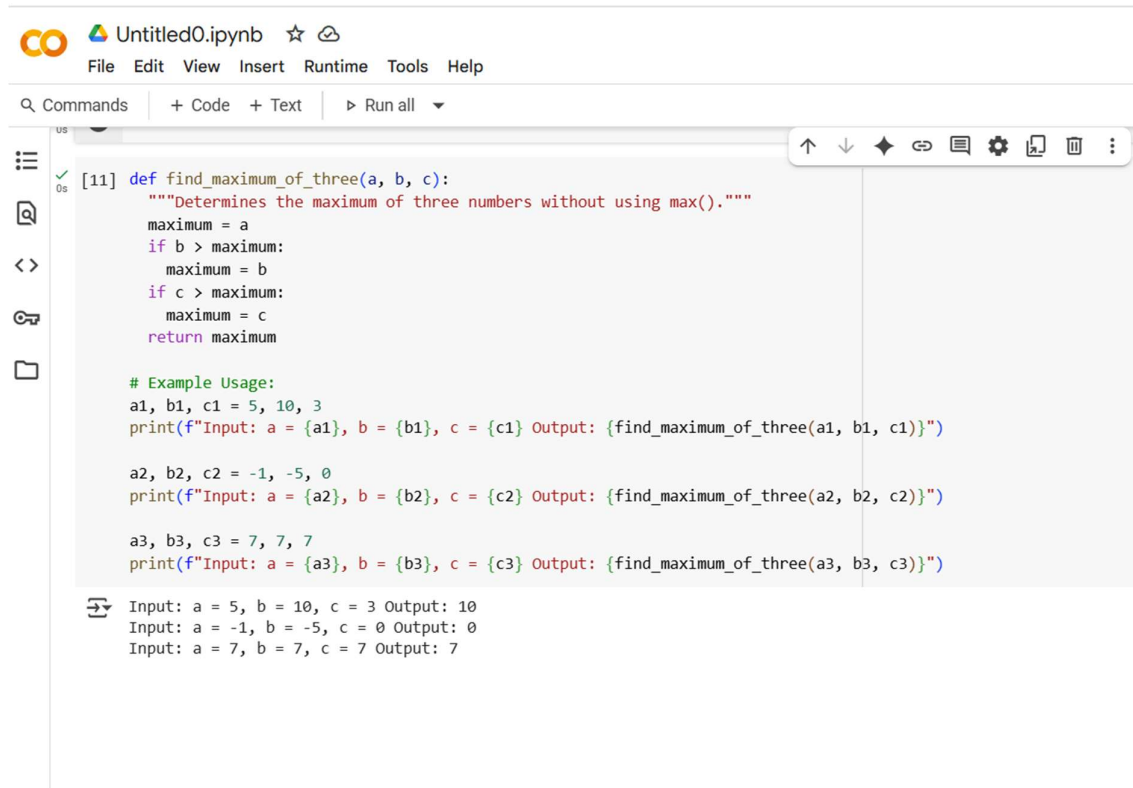
Use few-shot prompting with 3 sample inputs to generate a function that determines the maximum of three numbers without using the built-in `max()` function.

➤ **Expected output:**

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

➤ **Prompt#5**

Write a Python function that returns the maximum of three numbers without using the built-in `max()` function. Example 1 Input: `a = 5, b = 10, c = 3` Output: `10` Example 2 Input: `a = -1, b = -5, c = 0` Output: `0` Example 3 Input: `a = 7, b = 7, c = 7` Output: `7`.



```
[11] def find_maximum_of_three(a, b, c):  
    """Determines the maximum of three numbers without using max()."""  
    maximum = a  
    if b > maximum:  
        maximum = b  
    if c > maximum:  
        maximum = c  
    return maximum  
  
# Example Usage:  
a1, b1, c1 = 5, 10, 3  
print(f"Input: a = {a1}, b = {b1}, c = {c1} Output: {find_maximum_of_three(a1, b1, c1)}")  
  
a2, b2, c2 = -1, -5, 0  
print(f"Input: a = {a2}, b = {b2}, c = {c2} Output: {find_maximum_of_three(a2, b2, c2)}")  
  
a3, b3, c3 = 7, 7, 7  
print(f"Input: a = {a3}, b = {b3}, c = {c3} Output: {find_maximum_of_three(a3, b3, c3)}")  
  
Input: a = 5, b = 10, c = 3 Output: 10  
Input: a = -1, b = -5, c = 0 Output: 0  
Input: a = 7, b = 7, c = 7 Output: 7
```

➤ Expected output:

The code compares the three numbers one by one, keeping track of the largest number found so far, and finally returns that largest number.

➤ Observation:

The code finds the maximum of three numbers by comparing them sequentially and updating a variable to hold the largest value encountered so far.