AI ASSISTED CODING

LAB ASSIGNMENT - 3.2

ACHINA MANASA

2403A53043

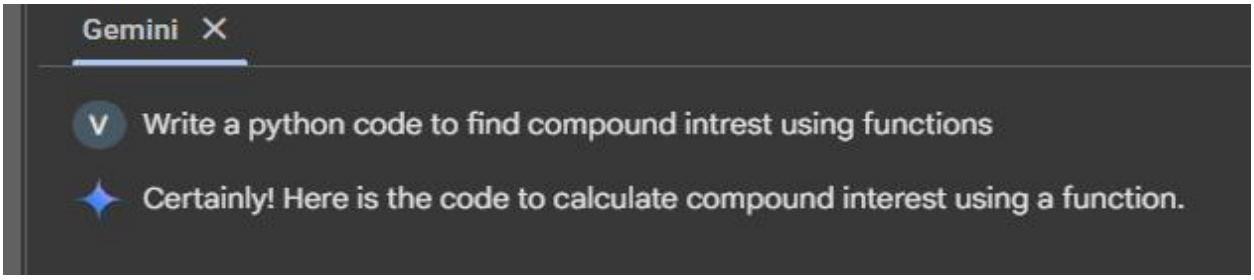
24BTCAICYB02

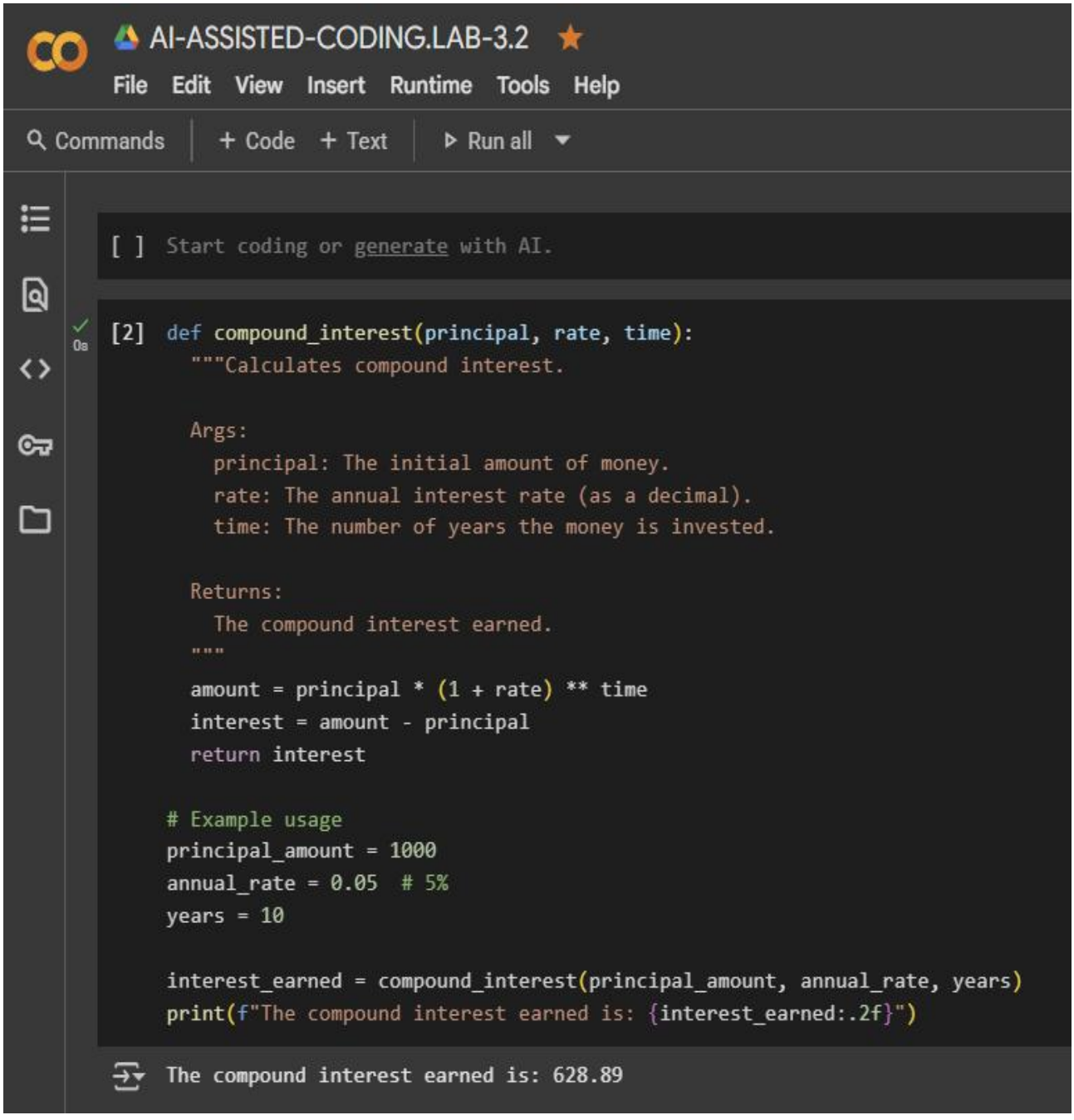
Task Description#1

- Ask AI to write a function to calculate compound interest, starting with only the function name. Then add a docstring, then input-output example.

Expected Output#1

- Comparison of AI-generated code styles.



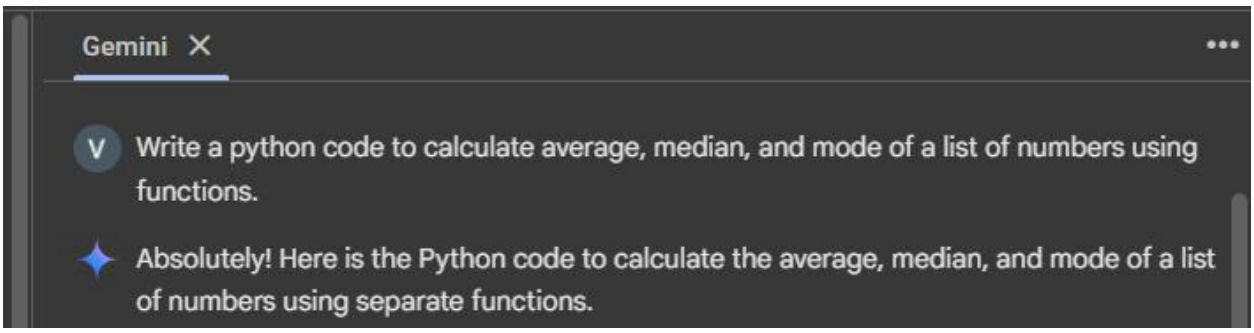

AI-ASSISTED-CODING.LAB-3.2
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
[ ] Start coding or generate with AI.
[2] 0s
def compound_interest(principal, rate, time):
"""Calculates compound interest.
Args:
principal: The initial amount of money.
rate: The annual interest rate (as a decimal).
time: The number of years the money is invested.
Returns:
The compound interest earned.
"""
amount = principal * (1 + rate) ** time
interest = amount - principal
return interest
# Example usage
principal_amount = 1000
annual_rate = 0.05 # 5%
years = 10
interest_earned = compound_interest(principal_amount, annual_rate, years)
print(f"The compound interest earned is: {interest_earned:.2f}")
The compound interest earned is: 628.89

Task Description#2

- Do math stuff, then refine it to: # Write a function to calculate average, median, and mode of a list of numbers.

Expected Output#2

- AI-generated function evolves from unclear to accurate multi-statistical operation.



AI-ASSISTED-CODING.LAB-3.2

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

```python
from collections import Counter

def calculate_average(numbers):
  """Calculates the average of a list of numbers."""
  return sum(numbers) / len(numbers)

def calculate_median(numbers):
  """Calculates the median of a list of numbers."""
  sorted_numbers = sorted(numbers)
  n = len(sorted_numbers)
  mid_index = n // 2
  if n % 2 == 0:
    return (sorted_numbers[mid_index - 1] + sorted_numbers[mid_index]) / 2
  else:
    return sorted_numbers[mid_index]

def calculate_mode(numbers):
  """Calculates the mode of a list of numbers."""
  count = Counter(numbers)
  max_count = max(count.values())
  mode = [num for num, c in count.items() if c == max_count]
  return mode

# Example usage
data = [1, 2, 3, 4, 5, 5, 6, 6, 6, 7]

average = calculate_average(data)
median = calculate_median(data)
mode = calculate_mode(data)

print(f"The data is: {data}")
print(f"The average is: {average}")
print(f"The median is: {median}")
print(f"The mode is: {mode}")
```

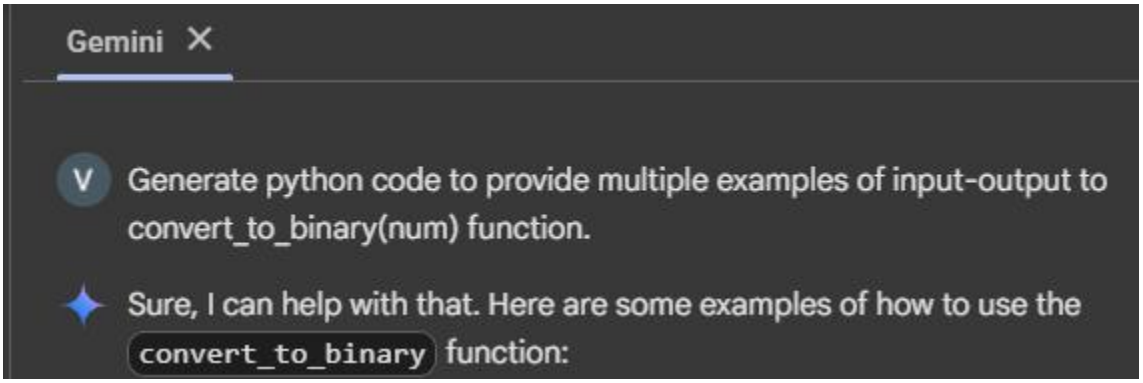
```
The data is: [1, 2, 3, 4, 5, 5, 6, 6, 6, 7]
The average is: 4.5
The median is: 5.0
The mode is: [6]
```

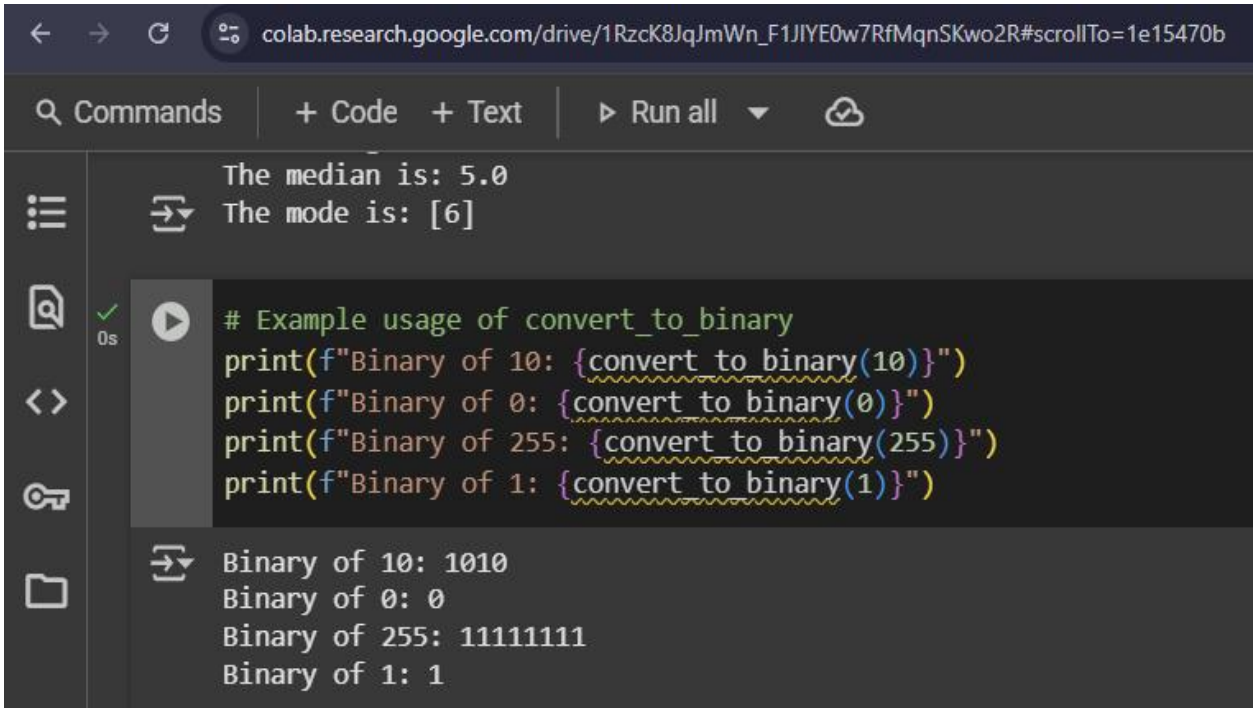
Task Description#3

- Provide multiple examples of input-output to the AI for convert_to_binary(num) function. Observe how AI uses few-shot prompting to generalize.

Expected Output#3

- Enhanced AI output with clearer prompts .



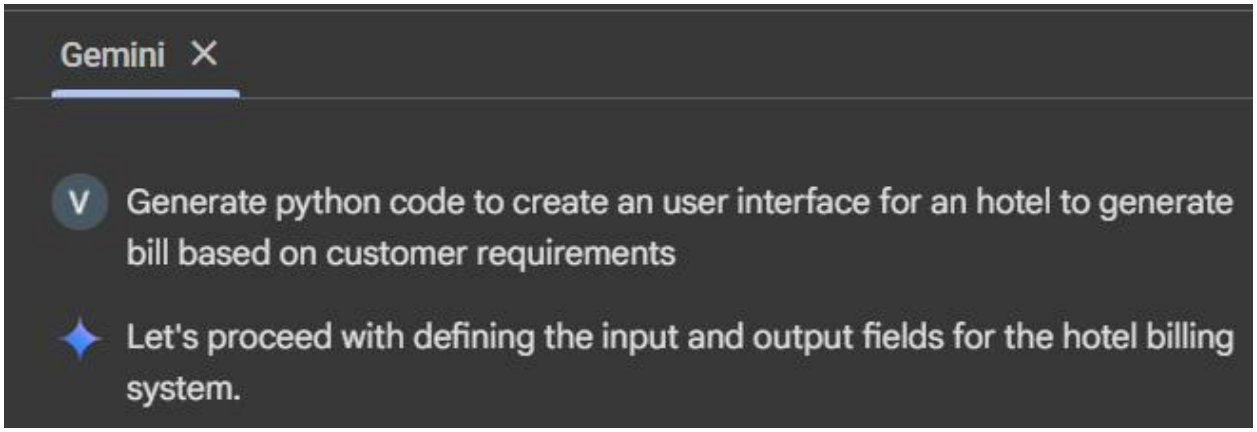


Task Description#4

- Create an user interface for an hotel to generate bill based on customer requirements.

Expected Output#4

- Consistent functions with shared logic.



colab.research.google.com/drive/1RzcK8JqJmWn_F1JIYE0w7RfMqnSKwo2R#scrollTo=d559d110

Commands | + Code | + Text | Run all

Reasoning: Enumerate the required input and output fields for an hotel billing system, categorize them, and specify their data types and the level of detail for the output as per the instructions.

```
# Input fields
inputs = {
    "customer_name": {"category": "input", "type": "text"},
    "room_type": {"category": "input", "type": "text"}, # e.g., Single, Double, Suite
    "number_of_nights": {"category": "input", "type": "number"},
    "check_in_date": {"category": "input", "type": "date"},
    "check_out_date": {"category": "input", "type": "date"},
    "services_used": {"category": "input", "type": "list of items"}, # e.g., restaurant, laundry, minibar
    "service_charges": {"category": "input", "type": "dictionary of numbers"}, # e.g., {"restaurant": 50.00, "laundry": 20.00}
    "room_rate_per_night": {"category": "input", "type": "number"}
}

# Output fields
outputs = {
    "bill_amount": {"category": "output", "type": "number"}, # Grand total
    "itemized_breakdown": {"category": "output", "type": "dictionary of numbers"} # e.g., {"room": 100.00, "restaurant": 50.00, "laundry": 20.00}
}

# Level of detail for output: The bill should show both a grand total and an itemized list of charges.

print("Required Input Fields:")
for field, details in inputs.items():
    print(f"- {field}: Category - {details['category']}, Type - {details['type']}")

print("\nRequired Output Fields:")
for field, details in outputs.items():
    print(f"- {field}: Category - {details['category']}, Type - {details['type']}")

print("\nOutput Detail Level: Grand total and itemized breakdown.")
```

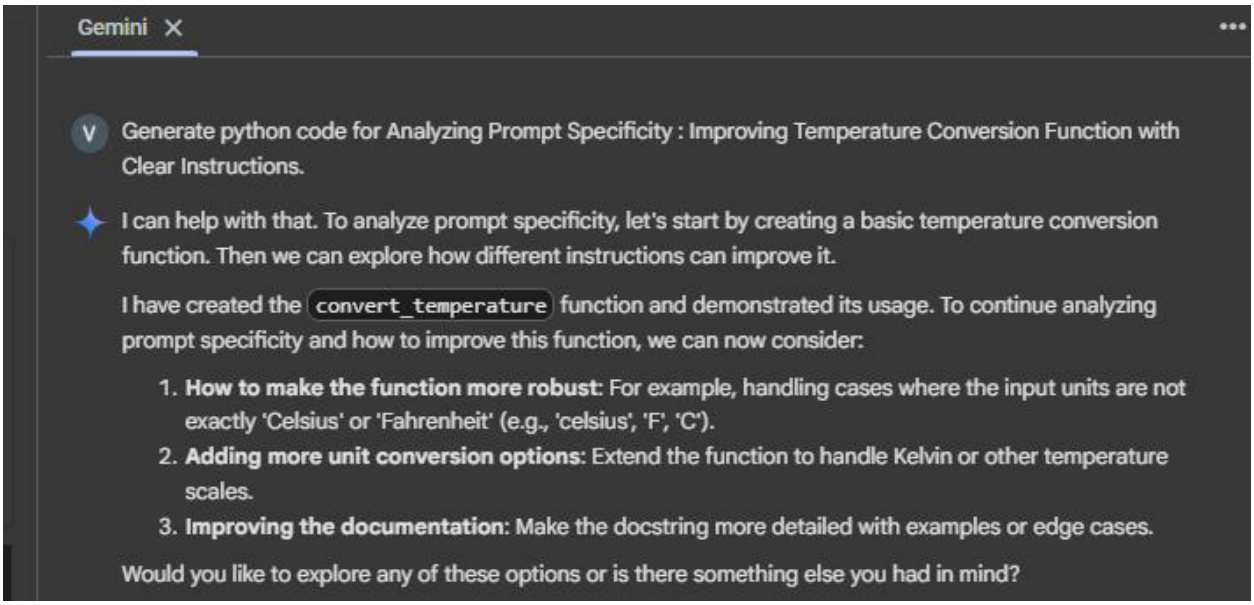
```
Required Input Fields:
- customer_name: Category - input, Type - text
- room_type: Category - input, Type - text
- number_of_nights: Category - input, Type - number
- check_in_date: Category - input, Type - date
- check_out_date: Category - input, Type - date
- services_used: Category - input, Type - list of items
- service_charges: Category - input, Type - dictionary of numbers
- room_rate_per_night: Category - input, Type - number

Required Output Fields:
- bill_amount: Category - output, Type - number
- itemized_breakdown: Category - output, Type - dictionary of numbers

Output Detail Level: Grand total and itemized breakdown.
```

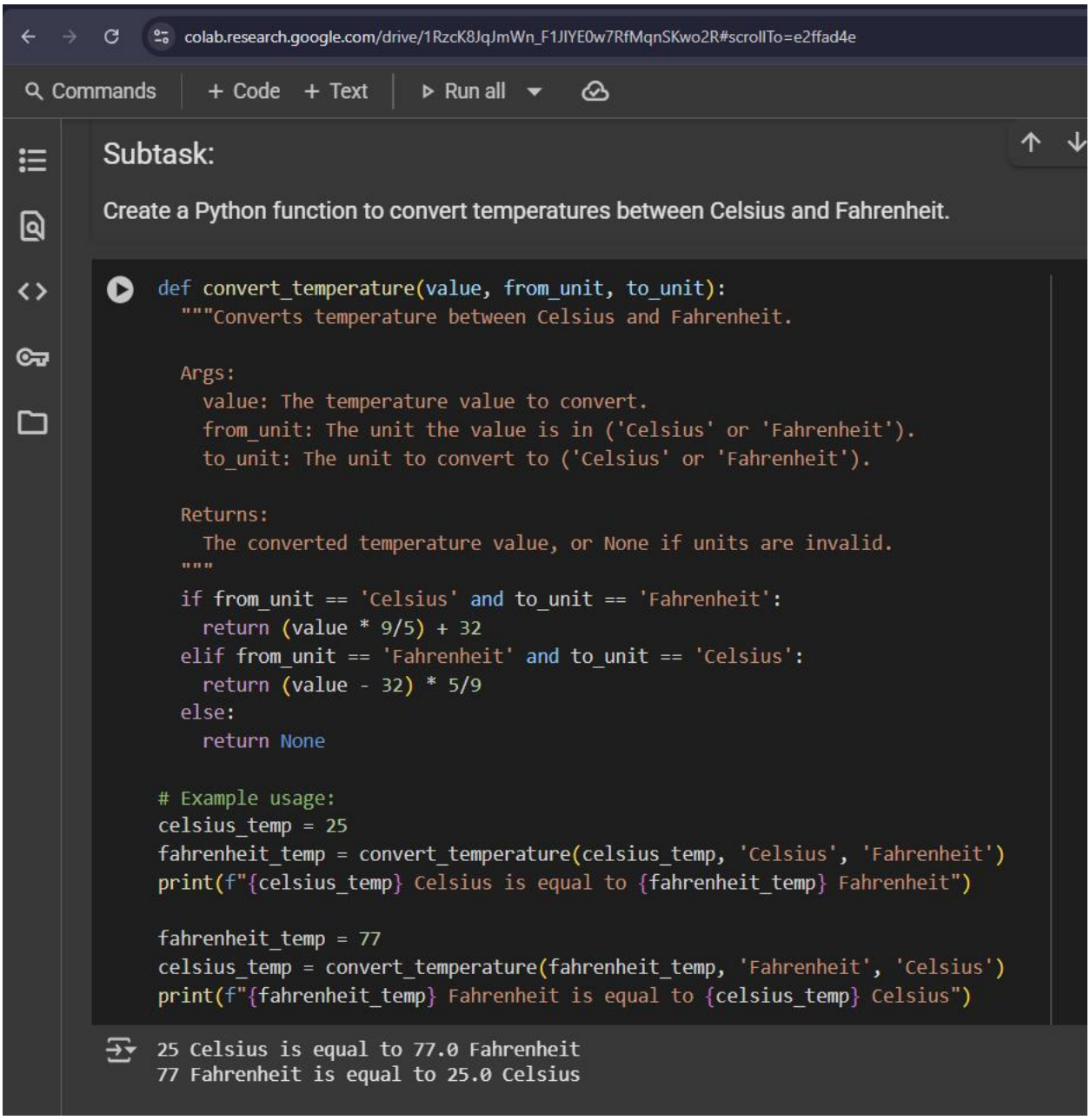
Task Description#5

- Analyzing Prompt Specificity: Improving Temperature Conversion Function with Clear Instructions.

Expected Output#5

- Code quality difference analysis for various prompts.




colab.research.google.com/drive/1RzcK8JqJmWn_F1JIYE0w7RfMqnSKwo2R#scrollTo=e2ffad4e
Commands
+ Code
+ Text
Run all
Subtask:
Create a Python function to convert temperatures between Celsius and Fahrenheit.
def convert_temperature(value, from_unit, to_unit):
"""Converts temperature between Celsius and Fahrenheit.
Args:
value: The temperature value to convert.
from_unit: The unit the value is in ('Celsius' or 'Fahrenheit').
to_unit: The unit to convert to ('Celsius' or 'Fahrenheit').
Returns:
The converted temperature value, or None if units are invalid.
"""
if from_unit == 'Celsius' and to_unit == 'Fahrenheit':
return (value * 9/5) + 32
elif from_unit == 'Fahrenheit' and to_unit == 'Celsius':
return (value - 32) * 5/9
else:
return None
# Example usage:
celsius_temp = 25
fahrenheit_temp = convert_temperature(celsius_temp, 'Celsius', 'Fahrenheit')
print(f"{celsius_temp} Celsius is equal to {fahrenheit_temp} Fahrenheit")
fahrenheit_temp = 77
celsius_temp = convert_temperature(fahrenheit_temp, 'Fahrenheit', 'Celsius')
print(f"{fahrenheit_temp} Fahrenheit is equal to {celsius_temp} Celsius")
25 Celsius is equal to 77.0 Fahrenheit
77 Fahrenheit is equal to 25.0 Celsius