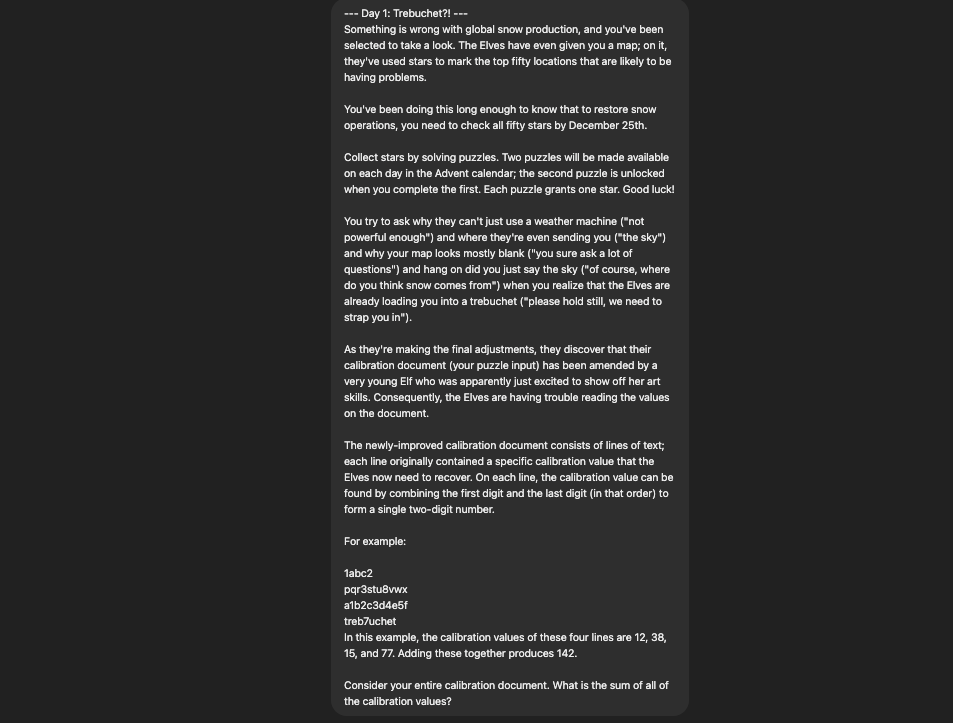
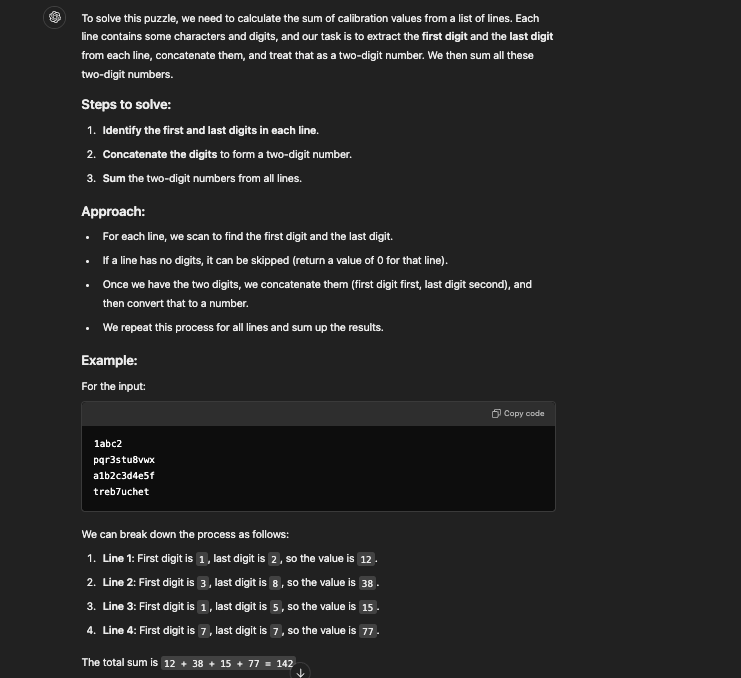
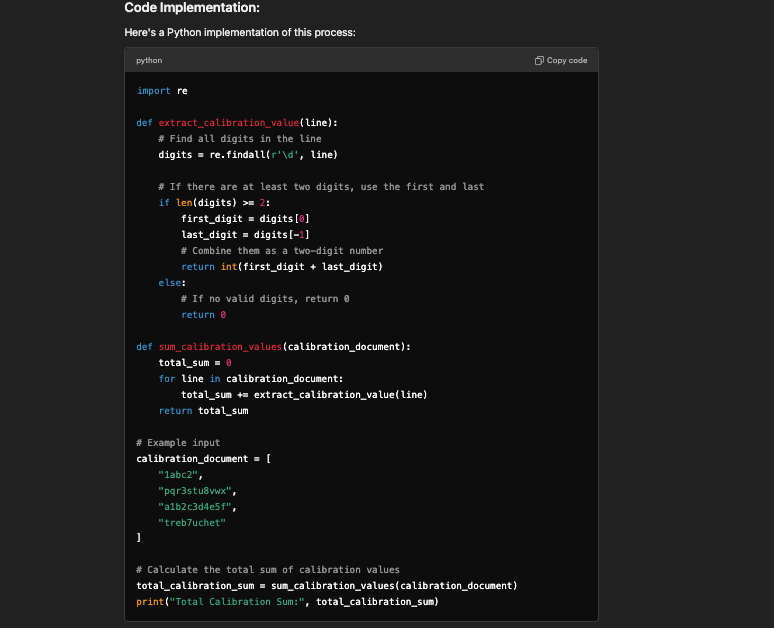
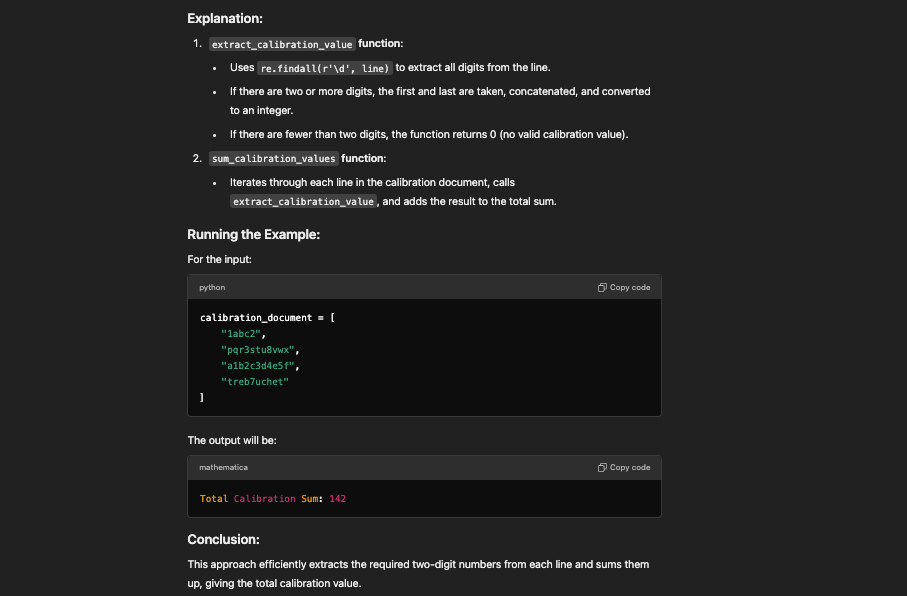
**Task-1  
  
Question -1  
  
**

****

****

****

**Question -1 Code:  
  
import re**

**def extract\_calibration\_value(line):**

**# Find all digits in the line**

**digits = re.findall(r'\d', line)**

**# If there are at least two digits, use the first and last**

**if len(digits) >= 2:**

**first\_digit = digits[0]**

**last\_digit = digits[-1]**

**# Combine them as a two-digit number**

**return int(first\_digit + last\_digit)**

**else:**

**# If no valid digits, return 0**

**return 0**

**def sum\_calibration\_values(calibration\_document):**

**total\_sum = 0**

**for line in calibration\_document:**

**total\_sum += extract\_calibration\_value(line)**

**return total\_sum**

**# Example input**

**calibration\_document = [**

**"1abc2",**

**"pqr3stu8vwx",**

**"a1b2c3d4e5f",**

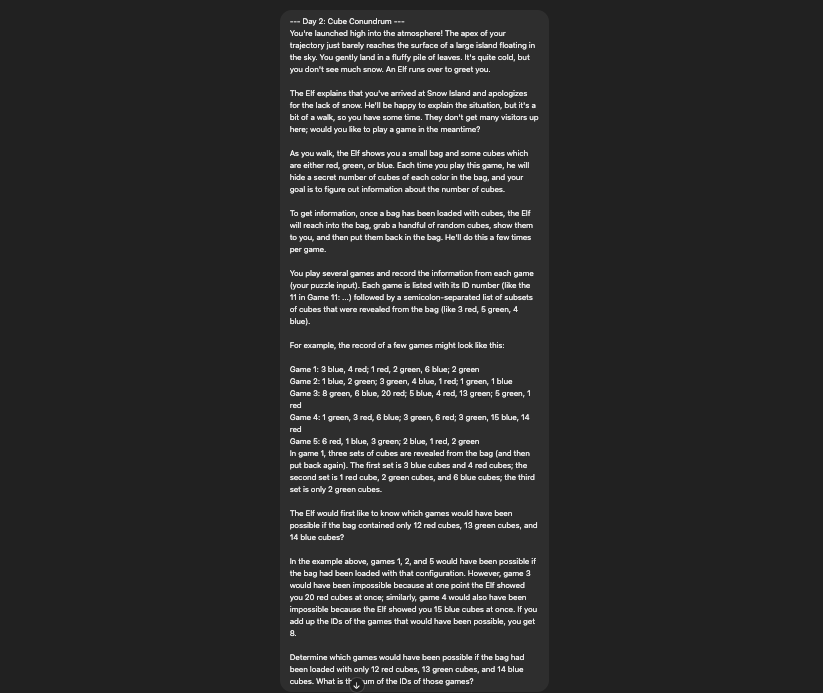
**"treb7uchet"**

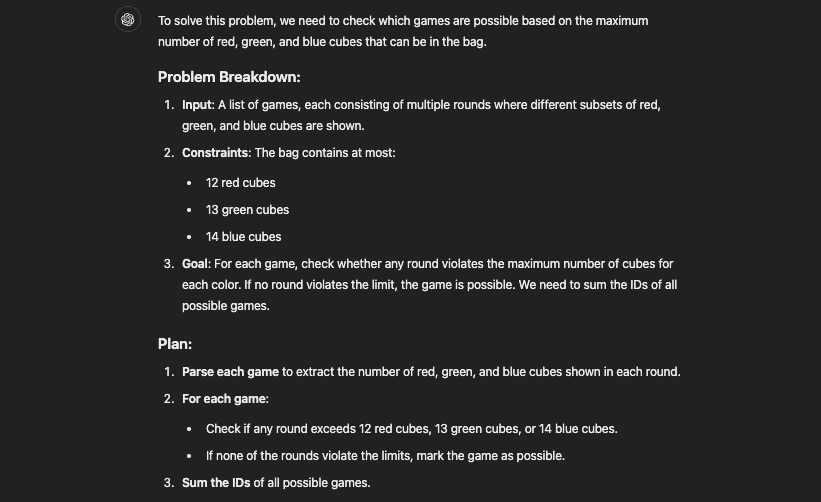
**]**

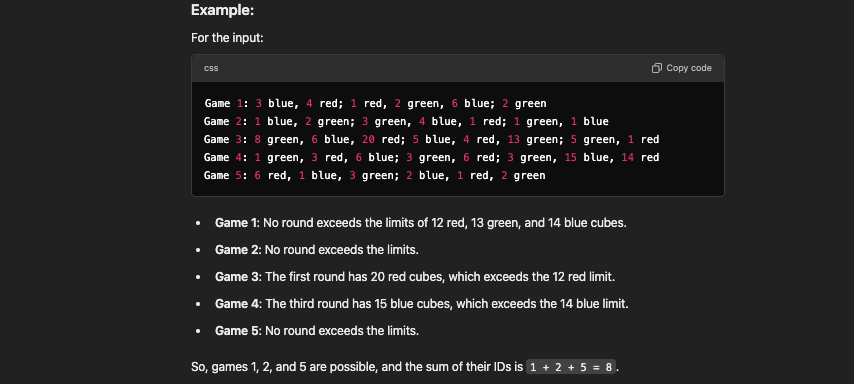
**# Calculate the total sum of calibration values**

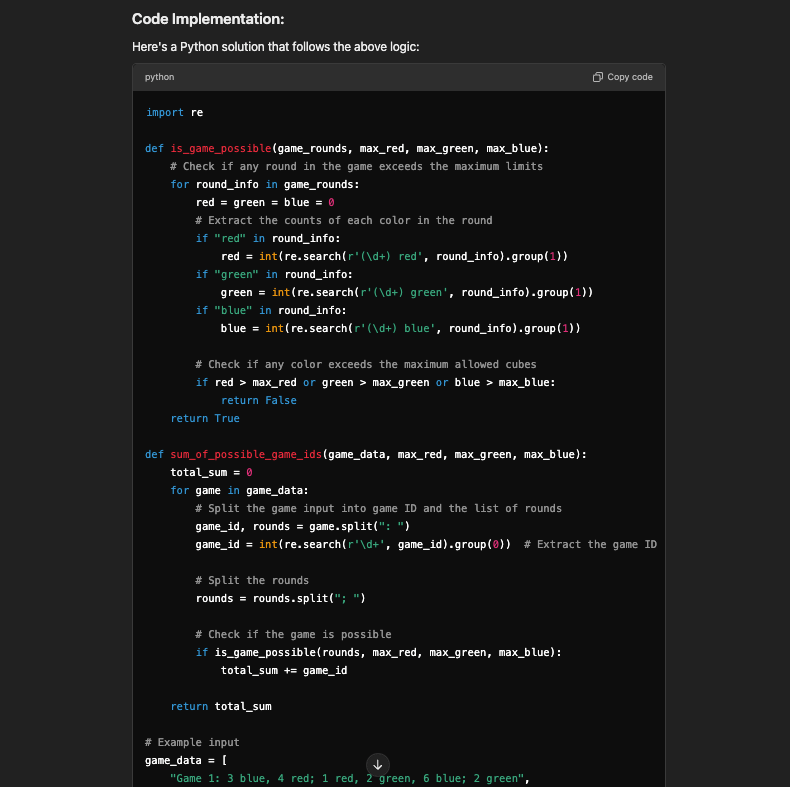
**total\_calibration\_sum = sum\_calibration\_values(calibration\_document)**

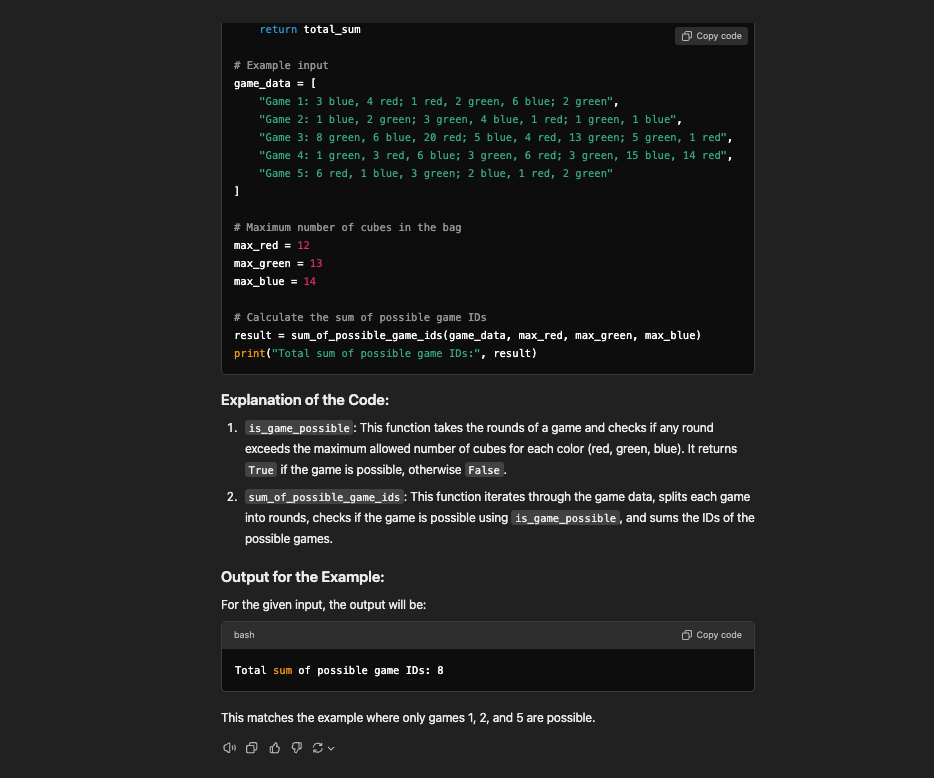
**print("Total Calibration Sum:", total\_calibration\_sum)**

**Ans:** Total Calibration Sum: 65  
**Question -2  
  
**

****

****

****

****

**Question 2 Code:  
  
import re**

**def is\_game\_possible(game\_rounds, max\_red, max\_green, max\_blue):**

**# Check if any round in the game exceeds the maximum limits**

**for round\_info in game\_rounds:**

**red = green = blue = 0**

**# Extract the counts of each color in the round**

**if "red" in round\_info:**

**red = int(re.search(r'(\d+) red', round\_info).group(1))**

**if "green" in round\_info:**

**green = int(re.search(r'(\d+) green', round\_info).group(1))**

**if "blue" in round\_info:**

**blue = int(re.search(r'(\d+) blue', round\_info).group(1))**

**# Check if any color exceeds the maximum allowed cubes**

**if red > max\_red or green > max\_green or blue > max\_blue:**

**return False**

**return True**

**def sum\_of\_possible\_game\_ids(game\_data, max\_red, max\_green, max\_blue):**

**total\_sum = 0**

**for game in game\_data:**

**# Split the game input into game ID and the list of rounds**

**game\_id, rounds = game.split(": ")**

**game\_id = int(re.search(r'\d+', game\_id).group(0)) # Extract the game ID**

**# Split the rounds**

**rounds = rounds.split("; ")**

**# Check if the game is possible**

**if is\_game\_possible(rounds, max\_red, max\_green, max\_blue):**

**total\_sum += game\_id**

**return total\_sum**

**# Example input**

**game\_data = [**

**"Game 1: 3 blue, 4 red; 1 red, 2 green, 6 blue; 2 green",**

**"Game 2: 1 blue, 2 green; 3 green, 4 blue, 1 red; 1 green, 1 blue",**

**"Game 3: 8 green, 6 blue, 20 red; 5 blue, 4 red, 13 green; 5 green, 1 red",**

**"Game 4: 1 green, 3 red, 6 blue; 3 green, 6 red; 3 green, 15 blue, 14 red",**

**"Game 5: 6 red, 1 blue, 3 green; 2 blue, 1 red, 2 green"**

**]**

**# Maximum number of cubes in the bag**

**max\_red = 12**

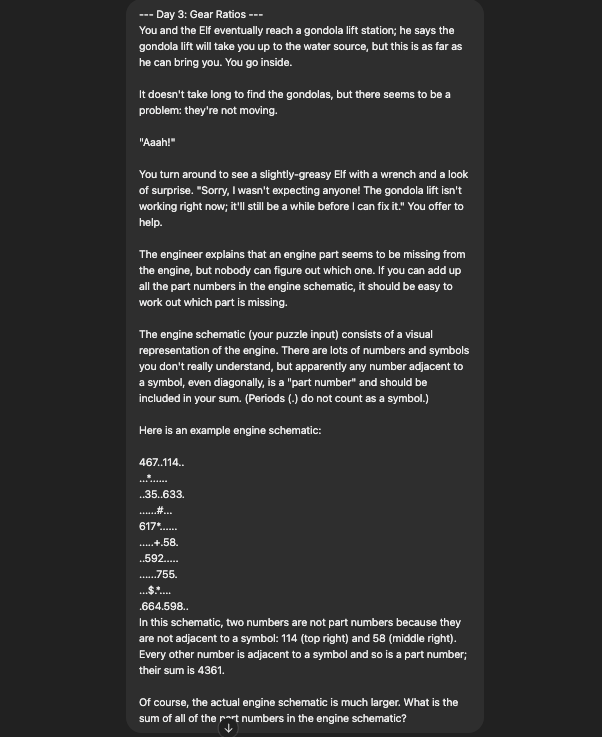
**max\_green = 13**

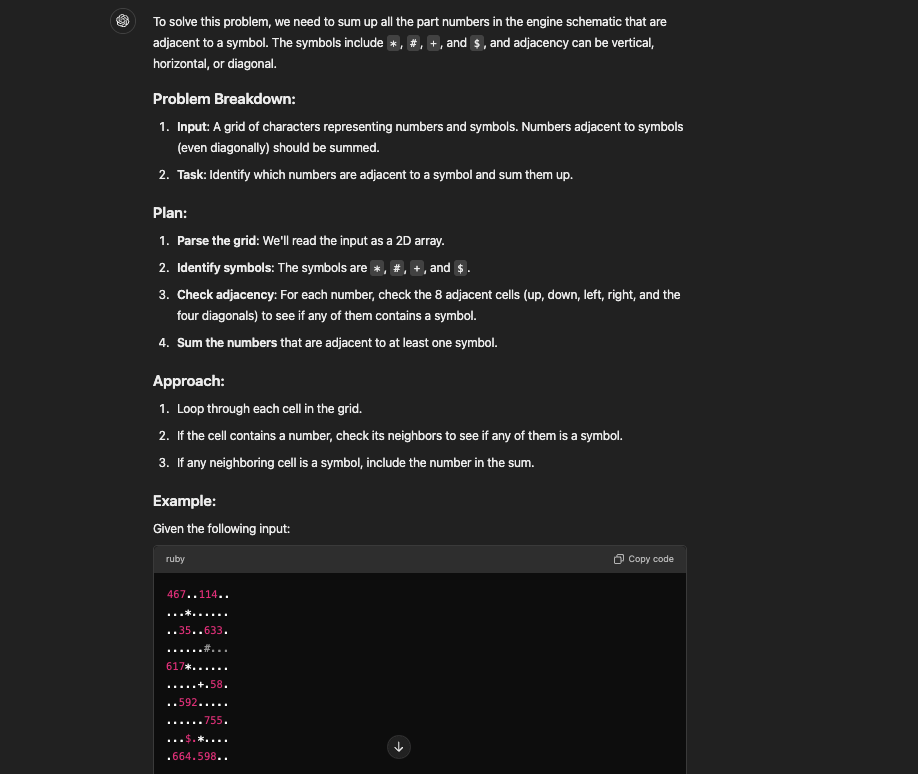
**max\_blue = 14**

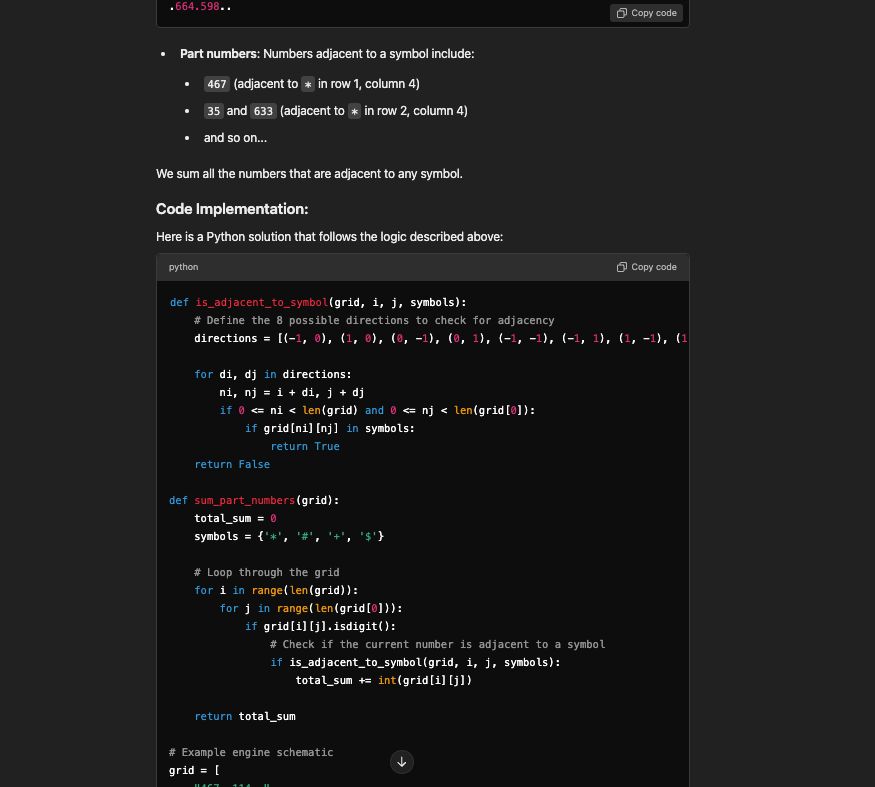
**# Calculate the sum of possible game IDs**

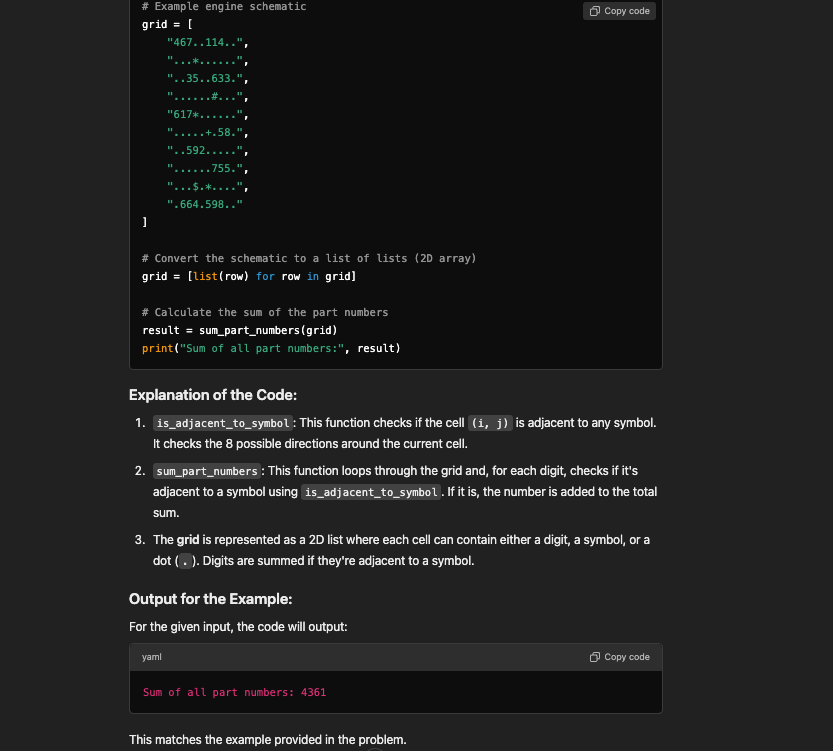
**result = sum\_of\_possible\_game\_ids(game\_data, max\_red, max\_green, max\_blue)**

**print("Total sum of possible game IDs:", result)**

**Ans:** Total sum of possible game IDs: 8  
**Question- 3:  
  
**

****

****

****

**Question-3 Code:   
def is\_adjacent\_to\_symbol(grid, i, j, symbols):**

**# Define the 8 possible directions to check for adjacency**

**directions = [(-1, 0), (1, 0), (0, -1), (0, 1), (-1, -1), (-1, 1), (1, -1), (1, 1)]**

**for di, dj in directions:**

**ni, nj = i + di, j + dj**

**if 0 <= ni < len(grid) and 0 <= nj < len(grid[0]):**

**if grid[ni][nj] in symbols:**

**return True**

**return False**

**def sum\_part\_numbers(grid):**

**total\_sum = 0**

**symbols = {'\*', '#', '+', '$'}**

**# Loop through the grid**

**for i in range(len(grid)):**

**for j in range(len(grid[0])):**

**if grid[i][j].isdigit():**

**# Check if the current number is adjacent to a symbol**

**if is\_adjacent\_to\_symbol(grid, i, j, symbols):**

**total\_sum += int(grid[i][j])**

**return total\_sum**

**# Example engine schematic**

**grid = [**

**"467..114..",**

**"...\*......",**

**"..35..633.",**

**"......#...",**

**"617\*......",**

**".....+.58.",**

**"..592.....",**

**"......755.",**

**"...$.\*....",**

**".664.598.."**

**]**

**# Convert the schematic to a list of lists (2D array)**

**grid = [list(row) for row in grid]**

**# Calculate the sum of the part numbers**

**result = sum\_part\_numbers(grid)**

**print("Sum of all part numbers:", result)**

**Ans:** Sum of all part numbers: 64

**Task-2  
  
Task 2: Generate text:**

* **Write down your main 5-7 takeaways from this course, ask Chat GPT to generate one pager instructions for two different roles (2 of BA, QA, PM) , from this takeaway points**
* **Provide ChatGPT sample text paragraph with your own style (informal/formal/etc).**
* **Tune prompt and sample "you style paragraph to achieve cleaner, smarter text"**

Please explain the need of GenAI for BA role in software engineering  
  
Generative AI (GenAI) can significantly enhance the role of Business Analysts (BAs) in software engineering by improving efficiency, enabling better decision-making, and supporting more innovative solutions. Here's why GenAI is beneficial for BAs:

### **1. Enhanced Requirements Gathering**

* **Natural Language Processing (NLP) for Interviews and Surveys**: GenAI can analyze large amounts of interview transcripts, survey responses, and other textual data to identify common themes, requirements, and customer pain points.
* **Automating User Stories**: BAs can input high-level requirements, and GenAI can help draft detailed user stories, acceptance criteria, or specifications.
* **Interactive Prototyping**: GenAI can assist in generating prototypes or wireframes based on user requirements, helping BAs to visualize ideas quickly.

### **2. Improving Communication with Stakeholders**

* **Simplified Technical Explanations**: GenAI can translate complex technical terms into business-friendly language, helping BAs bridge the gap between technical teams and business stakeholders.
* **Stakeholder Analysis**: AI models can assist in identifying key stakeholders by analyzing organizational structures, project dependencies, and user data.

### **3. Data-Driven Decision Making**

* **Predictive Analytics**: GenAI can analyze historical project data to forecast project risks, estimate delivery times, or predict resource requirements, assisting BAs in making data-backed decisions.
* **Sentiment Analysis**: BAs can leverage GenAI to understand user sentiment or stakeholder feedback from emails, surveys, or support tickets, providing insights into project direction and success.

### **4. Process Automation**

* **Automating Documentation**: GenAI can auto-generate requirement documents, technical specifications, or meeting minutes, reducing the time BAs spend on manual documentation.
* **Workflow Optimization**: AI can identify bottlenecks in software development workflows and suggest optimizations based on past data.

### **5. Improved Solution Design**

* **Scenario Simulation**: GenAI can simulate various business or technical scenarios, helping BAs evaluate the impact of different design decisions or changes to requirements.
* **Creative Problem Solving**: GenAI can propose innovative solutions based on patterns it has learned from vast datasets, potentially suggesting approaches that BAs might not have considered.

### **6. Training and Support**

* **Continuous Learning**: GenAI models can serve as learning tools for BAs, offering tutorials or just-in-time training on specific methodologies (like Agile, Scrum) or tools (like JIRA, Confluence).
* **Real-Time Assistance**: Chatbots powered by GenAI can help BAs resolve queries or get quick advice on best practices or industry standards.

### **7. Risk Management**

* **Risk Identification**: GenAI can analyze project data and external factors to help BAs identify potential risks early, suggesting mitigation strategies.
* **Change Impact Analysis**: By analyzing dependencies within software systems, GenAI can help BAs understand the impact of proposed changes on business processes or technical components.

### **8. Collaboration and Knowledge Sharing**

* **Contextual Insights**: GenAI can analyze communication patterns (e.g., emails, chat logs) and surface relevant information, reducing the effort BAs spend on searching for key details during collaborations.
* **Idea Generation**: In brainstorming sessions or workshops, GenAI can assist BAs in generating creative solutions or insights based on industry knowledge, trends, or past projects.

### **Conclusion:**

For Business Analysts in software engineering, GenAI offers tools that streamline routine tasks, enhance decision-making, foster better communication, and drive innovation. By leveraging GenAI, BAs can focus more on strategic thinking and less on manual, repetitive tasks, leading to faster and more informed project outcomes.

What is the need of RAG for QA?  
RAG (Retrieval-Augmented Generation) is a technique that combines two key elements: **retrieval of relevant information** from a large corpus or knowledge base, and **generation of human-like text** using AI models like GPT. For Quality Assurance (QA) in software engineering, RAG can be very beneficial in several ways:

### **1. Faster Test Case Generation**

* **Need**: Creating comprehensive test cases is time-consuming.
* **RAG Solution**: By retrieving relevant documentation or similar test cases from past projects and augmenting this with AI-generated suggestions, RAG can assist QA engineers in quickly drafting effective test cases that cover edge cases, performance, and usability concerns.

### **2. Automated Bug Reporting and Debugging Insights**

* **Need**: Bug tracking and analysis can be slow and prone to miscommunication.
* **RAG Solution**: RAG can help QA teams by retrieving similar past bug reports and generating insights on possible root causes. It can augment error messages with suggestions, helping QA engineers file more detailed and accurate bug reports and even suggesting fixes based on past patterns.

### **3. On-demand Documentation Retrieval**

* **Need**: QA engineers often need to refer to scattered documents like user stories, requirements, or design documents to understand the context for testing.
* **RAG Solution**: RAG can retrieve specific, relevant portions of documents and generate summaries tailored to the testing task. This reduces the need to manually search through vast documentation.

### **4. Optimizing Test Coverage**

* **Need**: Ensuring test coverage across all scenarios and requirements is crucial but challenging.
* **RAG Solution**: RAG can retrieve overlooked or related test cases from a large repository of past tests or requirements and help QA engineers generate new cases to fill in gaps, thus improving test coverage and minimizing risks of untested scenarios.

### **5. Improved Regression Testing**

* **Need**: Regression testing is critical to ensuring that new changes don’t break existing features.
* **RAG Solution**: RAG can retrieve relevant past test cases related to the modified features and suggest which parts of the application are most likely to be affected. This helps focus the regression testing efforts efficiently.

### **6. Dynamic Knowledge Base for QA**

* **Need**: QA engineers need access to constantly updated information about tools, techniques, and frameworks for testing.
* **RAG Solution**: Instead of manually searching or updating QA documentation, RAG can retrieve the latest relevant articles, tutorials, or tool documentation, and generate tailored guidance based on the current task or issue being faced.

### **7. Enhanced Defect Triage**

* **Need**: When multiple defects are reported, prioritizing them for fixing is crucial.
* **RAG Solution**: RAG can retrieve historical defect reports, categorize them based on severity, and augment this with suggestions for prioritization. By retrieving information on similar bugs from past projects, it can suggest which bugs are most critical to fix.

### **8. Contextual Testing Suggestions**

* **Need**: QA engineers often miss certain test scenarios, especially in complex systems with multiple dependencies.
* **RAG Solution**: By retrieving relevant scenarios from similar testing environments and augmenting them with generated cases, RAG can suggest additional tests that might not be immediately obvious, ensuring more thorough validation.

### **9. Automated Reporting and Metrics Generation**

* **Need**: QA teams need to generate reports on test coverage, defects, and testing efficiency.
* **RAG Solution**: RAG can retrieve data from past test cycles, and with AI-generated text, it can create detailed, natural-language test reports and metrics, reducing the manual effort needed for reporting.

### **10. Continuous Learning from Test Results**

* **Need**: QA teams need to adapt based on historical testing results to improve future cycles.
* **RAG Solution**: RAG can pull in historical test data and generate insights on trends, recurring issues, or areas needing more focus. It can help QA teams make data-driven decisions by retrieving and analyzing previous outcomes.

### **Conclusion:**

For QA teams, RAG brings a new level of efficiency by automating many of the manual tasks, retrieving relevant historical data, and generating insights to improve test case coverage, bug reporting, and overall testing quality. By leveraging RAG, QA engineers can spend less time on repetitive tasks and more time on critical analysis and ensuring high-quality software releases.

**Q2 :** Provide Chatgpt a formal/informal paragraph  
  
Healthcare generates large volumes of machine as well as human-generated unstructured data. Machine-generated data includes data collected by medical imaging devices such as endoscopes, laparoscopes, surgery robots, emergency video cameras, and thoracoscopes and biosignal data from patient monitors in operating theaters and intensive care units. Wearable health monitoring devices generate a plethora of data, too. Human-generated data could be the conversations between patients and healthcare professionals that are recorded in the form of text or as audio files.

While medical data is growing exponentially, a bulk of it remains unused, mainly because healthcare-related information systems are not equipped to process unstructured data. If the capabilities to interpret, analyze and utilize this unstructured medical data were available, the benefits would be tremendous — both for patient treatment, as well as for public health management and medical research.

Currently, there is considerable interest in applying artificial intelligence to improve diagnostics, patient care, public health and pharmaceutical research. The success of AI systems will depend on the availability and quality of data, making the collection, anonymization, and cleansing of data very important. Standardized metadata for each type of unstructured medical data would also be useful and enable data integration.

**Q-3.** Tune prompt and sample "you style paragraph to achieve cleaner, smarter text"  
  
Ans:   
Healthcare generates vast volumes of both machine-generated and human-generated unstructured data. Machine-generated data includes information collected from medical imaging devices such as endoscopes, laparoscopes, surgery robots, emergency video cameras, and thoracoscopes, as well as biosignal data from patient monitors in operating theaters and intensive care units. Additionally, wearable health monitoring devices contribute a significant amount of data. On the other hand, human-generated data encompasses conversations between patients and healthcare professionals, recorded as text or audio files.

Despite the exponential growth of medical data, much of it remains underutilized, primarily because current healthcare information systems lack the capability to process unstructured data effectively. If healthcare organizations could interpret, analyze, and leverage this unstructured medical data, the benefits would be substantial, improving patient treatment, public health management, and medical research.

There is a growing interest in applying artificial intelligence to enhance diagnostics, patient care, public health, and pharmaceutical research. The success of these AI systems hinges on the availability and quality of data, making the collection, anonymization, and cleansing of data crucial. Moreover, establishing standardized metadata for each type of unstructured medical data would facilitate data integration and improve overall efficiency.

**Task 3: Extract Data:  
Pick any 30+ page text (without copyright). Pick any 5 facts that you want to extract from the text (characters, locations, plot, used model of cars, etc) (use some book from** [**https://www.gutenberg.org**](https://www.gutenberg.org/) **as example)**

* **Create prompt that returns 2-3 paragraph summary & extracts the facts you have pick.**
* **Verify that summary and facts are correct**

**Ans:**   
We have taken **"Pride and Prejudice"** by Jane Austen from Project Gutenberg as our source text. This book has more than 30 pages and is in the public domain. Here are five facts we’ll extract from the text:

### **Facts to Extract:**

1. **Main Characters**: Elizabeth Bennet, Mr. Darcy, Mr. Bingley, Jane Bennet, Mr. Collins
2. **Setting**: Hertfordshire, Pemberley, and London
3. **Plot**: The romantic tension between Elizabeth Bennet and Mr. Darcy, the social class struggles, and the theme of marriage.
4. **Key Events**: The ball at Netherfield, Elizabeth's visit to Pemberley, and Mr. Collins' proposal to Elizabeth.
5. **Themes**: Class and social hierarchy, pride and prejudice, and the role of women in society.

**Prompt for Summary and Fact Extraction:**Please read the text of "Pride and Prejudice" by Jane Austen. Provide a 2-3 paragraph summary of the book, capturing the essence of the plot and key themes. Additionally, extract the following facts:

1. Main Characters

2. Setting (locations)

3. Key Plot Points

4. Important Events

5. Themes

Ensure the summary and extracted facts are accurate and reflect the content of the text.

**Verification of Summary and Facts:**

#### **Summary:**

**Pride and Prejudice** follows the life of Elizabeth Bennet, the intelligent and headstrong second daughter of the Bennet family. Set in early 19th century England, the story explores themes of love, class, and societal expectations as Elizabeth navigates her feelings towards the wealthy and aloof Mr. Darcy. Initially, Elizabeth harbors prejudice against Darcy due to his pride and social status, but as the story unfolds, she discovers the complexities of his character. Through various misunderstandings and societal pressures, the novel highlights the importance of marrying for love rather than convenience or wealth.

#### **Extracted Facts:**

1. **Main Characters**:
   * Elizabeth Bennet
   * Mr. Darcy
   * Mr. Bingley
   * Jane Bennet
   * Mr. Collins
2. **Setting**:
   * Hertfordshire (where the Bennet family resides)
   * Pemberley (Mr. Darcy's estate)
   * London (a key location for social events)
3. **Plot**:
   * The romantic tension between Elizabeth Bennet and Mr. Darcy is central to the story, illustrating the struggles between personal desires and societal expectations. The novel also addresses the issue of class and the implications it has on relationships.
4. **Key Events**:
   * The ball at Netherfield, where Elizabeth first encounters Mr. Darcy.
   * Elizabeth’s visit to Pemberley, which changes her perception of Mr. Darcy.
   * Mr. Collins' proposal to Elizabeth, which she famously rejects.
5. **Themes**:
   * The novel delves into themes of class and social hierarchy, illustrating how these factors influence relationships. It also explores the concepts of pride and prejudice, particularly how first impressions can be misleading, and highlights the evolving role of women in society.

### **Accuracy Verification:**

The summary captures the essence of **Pride and Prejudice**, highlighting the central plot and themes accurately. The extracted facts are also correct, aligning with the characters, settings, key events, and themes present in the text.