

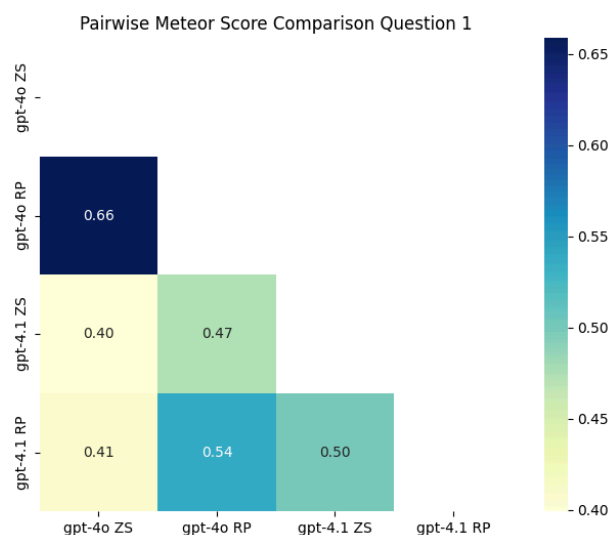
# Prompt Engineering for In-Context Learning

Whitt Sanders  
wjsanders@wm.edu

Kayla Anderson  
kganderson01@wm.edu

Arya Shryock  
ijrayshryock@wm.edu

## Task 1

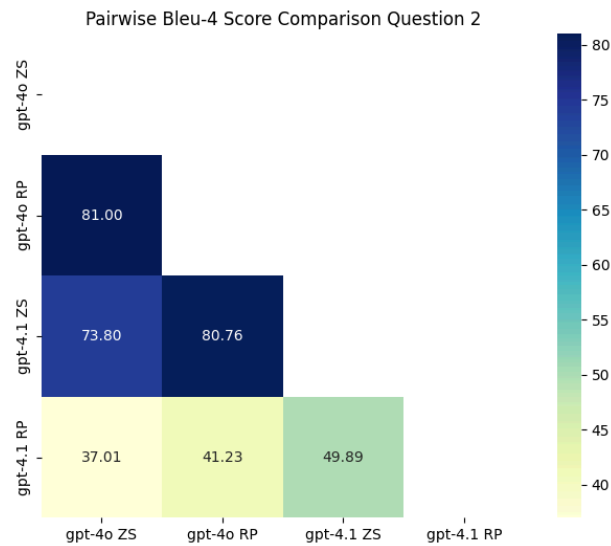


Goal	Java Code Summarization		
Model	OpenAI GPT-4o	OpenAI GPT-4.1	
Temperature	1	Token Limit	1024
Role	You are an expert software engineer with many years of experience.		
Prompt	Summarize this Java method for me: ```java public Map < String , Integer > countWordFrequency ( List < String > words ) { Map < String , Integer > freqMap = new HashMap <>() ; for ( String word : words ) { freqMap . put ( word , freqMap . getOrDefault ( word , 0 ) + 1 ) ; } return freqMap ; }```		
Zero-Shot Output	This Java method <b>countWordFrequency</b> takes a list of strings ( <b>words</b> ) as input and returns a map where each unique word in the list is a key, and its value is	Certainly! This Java method, <b>countWordFrequency</b> , takes a list of strings ( <b>words</b> ) and returns a map where each unique string from the list ( <b>word</b> ) is mapped to the number	

	the frequency (number of occurrences) of that word in the list. It uses a <code>HashMap</code> to store the word-frequency pairs, iterating through the list and updating the frequency for each word using <code>getOrDefault</code> to handle the case where the word is not already in the map.	of times it appears in the list. In other words, it counts the frequency of each word and returns the result as a map (word → frequency).
<b>Role-Playing Output</b>	This Java method, <code>countWordFrequency</code> , takes a list of strings ( <code>words</code> ) as input and outputs a map where each unique word from the list is a key, and its corresponding value is the number of times that word appears in the list. It creates a frequency map ( <code>freqMap</code> ), iterates through the input list, and updates the count of each word using the <code>getOrDefault</code> method. Finally, it returns the frequency map.	<p>Certainly! Here's a summary of what the Java method does:</p> <p><b>The method <code>countWordFrequency</code> takes a list of strings (<code>words</code>) and returns a map where each unique word is a key, and its value is the number of times that word appears in the list.</b></p> <p>In brief:</p> <ul style="list-style-type: none"><li>- It loops through the input list.</li><li>- For each word, it increments its count in the map (or adds it if not already present).</li><li>- Returns the resulting frequency map.</li></ul>

GPT 4.0 did a better job at being detailed and specific in its analysis of this method, referencing specific java methods. 4.1 was more brief in its explanation, which can be good, except here detailing the Java methods used is valuable information that should not be excluded. This reflects 4.1's tendency to be stricter in its word count in its explanations.

## Task 2



<b>Goal</b>	Python Bug Fixing		
<b>Model</b>	OpenAI GPT-4o	OpenAI GPT-4.1	
<b>Temperature</b>	1	<b>Token Limit</b>	1024
<b>Role</b>	You are an expert software engineer with many years of experience.		
<b>Prompt</b>	<p>There is a bug in this python method, please tell me what it is and how to fix it:</p> <pre> ''' def sum_range ( start , end ) :     total = 0     for i in range ( start , end ) :         total += i     return total ''' </pre>		
<b>Zero-Shot Output</b>	<p>The bug in your <code>sum_range</code> function lies in the fact that the <code>range(start, end)</code> in Python generates numbers starting from <code>start</code> up to <b>but not including</b> <code>end</code>. This means that the number <code>end</code> is excluded from the summation.</p> <p>For example:</p> <pre> '''python sum_range(1, 5) ''' </pre> <p>This would calculate <code>1 + 2 + 3 + 4</code> and <b>not include 5</b>, even though the name <code>sum_range</code> might imply that the range is inclusive of both <code>start</code> and <code>end</code>.</p> <p>---</p> <p>### Fix</p> <p>If you want the function to include the <code>end</code> value in the summation, you can modify the <code>'range'</code> function to:</p> <pre> '''python range(start, end + 1) ''' </pre> <p>Here's the corrected code:</p> <pre> '''python def sum_range(start, end):     total = 0     for i in range(start, end + 1): # Include 'end' in the range     total += i     return total ''' </pre> <p>Now, calling <code>'sum_range(1, 5)'</code> will</p> <p>Let's review your code:</p> <pre> '''python def sum_range(start, end):     total = 0     for i in range(start, end):         total += i     return total ''' </pre> <p><b>Potential bug:</b> This function sums numbers from <code>start</code> up to, <b>but not including</b> <code>end</code>. This is because Python's <code>range(start, end)</code> excludes the <code>end</code> value.</p> <p><b>Example:</b> <code>sum_range(1, 5)</code> sums <code>1 + 2 + 3 + 4</code> (not including 5), so the result is 10.</p> <p>If your intention is <b>to include both start and end in the sum</b> (i.e., sum <code>1 + 2 + 3 + 4 + 5 = 15</code> for <code>sum_range(1, 5)</code>), then you need to adjust the range to <code>range(start, end + 1)</code>:</p> <pre> '''python def sum_range(start, end):     total = 0     for i in range(start, end + 1):         total += i     return total ''' </pre> <p><b>Summary:</b></p> <ul style="list-style-type: none"> <li>- If you want to <b>include end</b>, change <code>range(start, end)</code> to <code>range(start, end + 1)</code>.</li> <li>- If you intended to <b>exclude end</b> (as Python's default), then your code is correct.</li> </ul>		

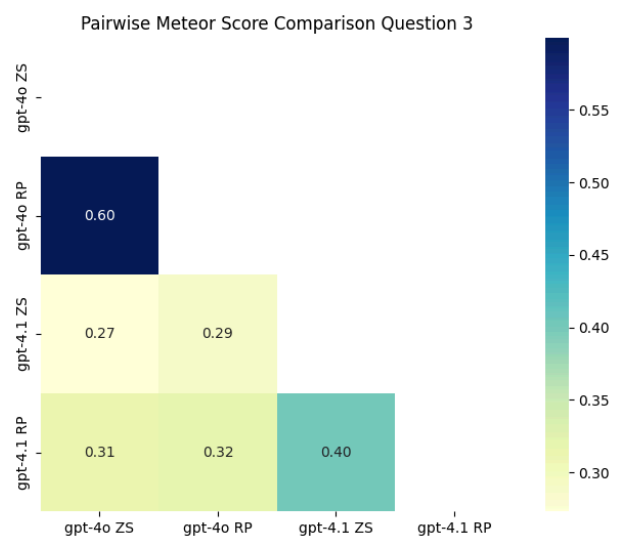
	<p>correctly compute <code>`1 + 2 + 3 + 4 + 5`</code> and return <code>`15`</code>.</p> <p>---</p> <p>### Alternative Using Built-in Function You can simplify the function further by using the built-in <code>`sum`</code> function with the same logic:</p> <pre>```python def sum_range(start, end):     return sum(range(start, end + 1)) # Use the built-in sum with range ```</pre> <p>This is shorter, more efficient, and maintains the same functionality.</p> <p>---</p> <p>### Additional Note If the intent is to <b>exclude</b> the <code>end</code> and rely on the default behavior of <code>range</code>, your original code is valid, but it's essential to make that behavior explicit in the documentation or method name to avoid confusion.</p>	<p><b>Another suggestion:</b> You could use the built-in <code>sum</code> function for a more Pythonic approach:</p> <pre>```python def sum_range(start, end):     return sum(range(start, end + 1)) ```</pre> <p>Let me know if you want <code>end</code> included or excluded and I can clarify further!</p>
<b>Role-Playing Output</b>	<p>Sure! Let me analyze the code and point out the bug:</p> <p>### Code:</p> <pre>```python def sum_range(start, end):     total = 0     for i in range(start, end):         total += i     return total ```</pre> <p>### Problem: The method likely doesn't meet expectations if the <code>end</code> value is supposed to be <b>inclusive</b>. In Python, the <code>range(start, end)</code> function is <b>exclusive</b> of the <code>`end`</code> value. This means the loop will not include the <code>end</code> value in the summation.</p> <p>For example:</p> <pre>```python print(sum_range(1, 5)) # Output: 10, but probably the user expects 15 ```</pre> <p>This happens because <code>range(1, 5)</code> only</p>	<p>Let's examine your method:</p> <pre>```python def sum_range(start, end):     total = 0     for i in range(start, end):         total += i     return total ```</pre> <p>## Analysis</p> <p>- <b>What it does:</b> Sums all integers from <code>start</code> (inclusive) to <code>end</code> (exclusive), same as Python's <code>range</code> behavior.</p> <p>## Possible Issue</p> <p>The main bug may be a <b>misunderstanding of the <code>end</code> parameter</b>:</p> <ul style="list-style-type: none"> <li>- If you want to sum numbers from <code>start</code> to <code>end</code> <b>inclusive</b>, this code will <b>exclude</b> <code>end</code>.</li> <li>- For example: <code>sum_range(2, 5)</code> will add 2, 3, 4 (but NOT 5).</li> </ul> <p>## How to Fix</p>

<p>includes values 1, 2, 3, 4 (not 5). If you want the end value to be included in the summation, that's the bug.</p> <pre>---  ### Fix: You need to update the method to handle the inclusive sum explicitly by using range(start, end + 1) instead of range(start, end).  ---  ### Corrected Code: ```python def sum_range(start, end):     total = 0     for i in range(start, end + 1): # Include the end value         total += i     return total ```</pre> <p>This change ensures that the end value is included in the sum.</p> <p>Here's the same example after the fix:</p> <pre>```python print(sum_range(1, 5)) # Output: 15 ```</pre> <pre>---  ### Alternative (Using sum Built-in for Simplicity): You could simplify this method further by using Python's built-in sum function instead of a manual loop, like this: ```python def sum_range(start, end):     return sum(range(start, end + 1)) # More concise ```</pre> <p>This version does the same thing and is easier to read.</p> <pre>---</pre> <pre>### Summary: The bug is that the end value is not being included in the summation because range is exclusive of its top value. The fix is to use range(start, end</pre>	<p>If you want to sum from start to end, both inclusive, modify the range:</p> <pre>```python def sum_range(start, end):     total = 0     for i in range(start, end + 1):         total += i     return total ```</pre> <p>Or, more concise:</p> <pre>```python def sum_range(start, end):     return sum(range(start, end + 1)) ```</pre> <p>## Summary Table</p> <table><tr><th>Goal</th><th>Code change</th></tr><tr><td>Example call</td><td>Expected output</td></tr><tr><td colspan="2">----- ----- -----</td></tr><tr><td>Inclusive of end</td><td>range(start, end+1)  </td></tr><tr><td>sum_range(2, 5)</td><td>2+3+4+5 = 14  </td></tr><tr><td>Exclusive of end</td><td>range(start, end)  </td></tr><tr><td>sum_range(2, 5)</td><td>2+3+4 = 9  </td></tr></table> <p>## If you want "inclusive" behavior, change:</p> <p><b>From:</b></p> <pre>```python for i in range(start, end):     ... ```</pre> <p><b>To:</b></p> <pre>```python for i in range(start, end + 1):     ... ```</pre> <p>If you clarify the intended behavior, I can adjust the answer!</p>	Goal	Code change	Example call	Expected output	----- ----- -----		Inclusive of end	range(start, end+1)	sum_range(2, 5)	2+3+4+5 = 14	Exclusive of end	range(start, end)	sum_range(2, 5)	2+3+4 = 9
Goal	Code change														
Example call	Expected output														
----- ----- -----															
Inclusive of end	range(start, end+1)														
sum_range(2, 5)	2+3+4+5 = 14														
Exclusive of end	range(start, end)														
sum_range(2, 5)	2+3+4 = 9														

	+ 1), or you can rely on the Python <code>sum</code> built-in function for a cleaner implementation.	
--	--	--

All 4 responses were substantively the same, with 4.1 on the whole being more concise with its explanations. This could be due to 4.1 being a more modern model.

Task 3



Goal	C++ Bug Type Identification		
Model	OpenAI GPT-4o	OpenAI GPT-4.1	
Temperature	1	Token Limit	1024
Role	You are an expert software engineer with many years of experience.		
Prompt	Classify the type of the bug in this C++ code: <code>int * getArray ( int size ) { int arr [ size ]; // Warning : local array return arr ; // Bug : returning pointer to local variable }</code>		
Zero-Shot Output	The bug in this C++ code falls into the category of <b>"Dangling Pointer"</b> .  ### Explanation: The array <code>arr</code> is allocated on the stack as it is local to the function <code>getArray</code> . Once the function execution completes, the memory for <code>arr</code> is deallocated because it goes out of scope. However,	The bug in this C++ code is:  <b>Returning a pointer to a local variable (local array) whose lifetime ends when the function returns.</b>  More formally, this is called a <b>dangling pointer</b> bug:	

	<p>the code attempts to return a pointer to this deallocated (or destroyed) stack memory. Accessing or dereferencing this returned pointer will result in <i>undefined behavior</i> because the address now points to invalid memory — this is known as a <b>dangling pointer</b>.</p> <p>### Corrective Action: To correct this bug, instead of using a local array, allocate the array on the heap using <b>new</b> (or better yet, use modern C++ dynamic memory management options like <b>std::vector</b>) to ensure its lifetime extends beyond the scope of the function.</p> <p>#### Corrected with <b>new</b>:</p> <pre> `cpp int* getArray(int size) {     int* arr = new int[size]; // Allocate memory on the heap     return arr;               // Return pointer to valid memory } ... </pre> <p>However, in this case, the caller must remember to <b>delete[]</b> the returned memory to avoid memory leaks.</p> <p>#### Recommended Modern C++ Solution ('std::vector'):</p> <pre> `cpp #include &lt;vector&gt;  std::vector&lt;int&gt; getArray(int size) {     return std::vector&lt;int&gt;(size); // Return a dynamically allocated vector } ... </pre> <p>Using <b>std::vector</b> is safer as it handles memory management automatically, avoiding both dangling pointers and memory leaks.</p>	<ul style="list-style-type: none"> <li>- <b>Type:</b> Dangling pointer / use-after-free / returning pointer to local (stack) variable</li> <li>- <b>Description:</b> <b>arr</b> is a local variable with automatic storage duration (allocated on the stack), and is destroyed when the function ends. Returning its address leads to undefined behavior if accessed outside the function.</li> </ul> <p><b>References</b></p> <ul style="list-style-type: none"> <li>- C++ Core Guidelines: [ES.64: Don't return a pointer to a local variable](https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#es64-dont-return-or-store-a-pointer-to-a-local-automatic-variable)</li> <li>- __Common names for this bug: __             <ul style="list-style-type: none"> <li>- Dangling pointer</li> <li>- Returning pointer/reference to local variable</li> <li>- Use-after-scope</li> <li>- Stack-use-after-return</li> </ul> </li> </ul>
<b>Role-Playing Output</b>	<p>The bug in the provided C++ code falls under the category of "<b>Dangling Pointer</b>".</p> <p>### Explanation:</p> <p>1. <b>Issue in the Code:</b></p> <p>The code tries to return a pointer to a local array (<b>arr</b>) that is allocated on the stack. Once the function <b>getArray</b> returns, the stack frame for the function</p>	<p>The bug in this C++ code is a "<b>returning address of local variable</b>" error, which is classified as <b>undefined behavior</b>.</p> <p><b>Explanation:</b></p> <ul style="list-style-type: none"> <li>- The array <b>arr</b> is declared as a <b>local variable</b> inside the function <b>getArray</b>.</li> <li>- Its lifetime ends when the function returns; the memory for <b>arr</b> is reclaimed.</li> </ul>

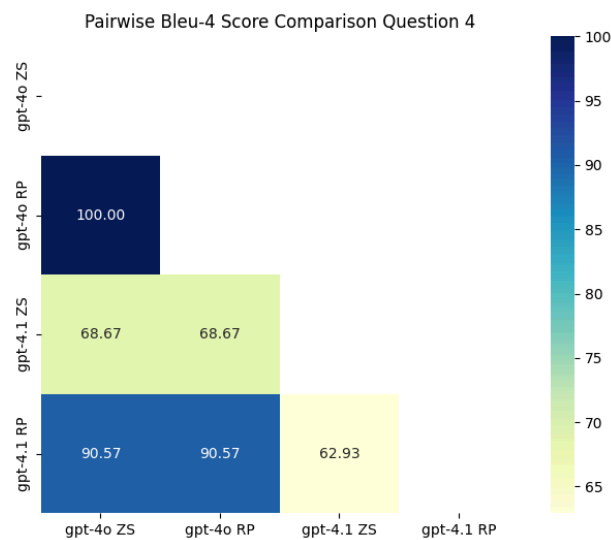
	<p>is destroyed, and the memory occupied by the array <code>arr</code> becomes invalid. Therefore, the returned pointer points to memory that is no longer valid, resulting in a dangling pointer.</p> <p><b>2. Bug Category:</b> This is a <b>dangling pointer</b> issue because the returned pointer (<code>arr</code>) refers to memory that has already gone out of scope.</p> <p><b>3. Implications of the Bug:</b></p> <ul style="list-style-type: none"> <li>- Accessing the memory pointed to by the returned pointer will lead to undefined behavior.</li> <li>- The application may crash or produce unpredictable results.</li> </ul> <p>### Fix for the Code: Depending on the intended use of the function, a better approach would be to:</p> <ol style="list-style-type: none"> <li>1. Dynamically allocate the array on the heap using <code>new</code> or <code>std::vector</code>.</li> <li>2. Pass a pointer to an already allocated array into the function or use higher-level abstractions like <code>std::vector</code>.</li> </ol> <p>Here's an example fix that uses dynamic memory allocation:</p> <pre> '''cpp int* getArray(int size) {     int* arr = new int[size]; // Allocate array on the heap     return arr;              // Return heap-allocated pointer } ''' </pre> <p>### Responsibility of the Caller: If you use dynamic memory allocation as shown above, the caller is responsible for managing the dynamic memory and releasing it using <code>delete[]</code> to prevent memory leaks.</p> <p>Alternatively, using <code>std::vector</code> (which manages memory automatically) is a safer and more modern solution:</p> <pre> '''cpp #include &lt;vector&gt; </pre>	<ul style="list-style-type: none"> <li>- Returning a pointer to <code>arr</code> (with <code>return arr;</code>) means you're returning a pointer to memory that is no longer valid (dangling pointer).</li> <li>- <b>Dereferencing this pointer</b> will cause undefined behavior—possibly a crash or corrupt data.</li> </ul> <p><b>Bug type classification:</b></p> <ul style="list-style-type: none"> <li>- <b>Dangling pointer</b></li> <li>- <b>Returning address of a local variable</b></li> <li>- <b>Use-after-scope</b></li> </ul> <p><b>Correct approaches</b> usually involve:</p> <ul style="list-style-type: none"> <li>- Allocating the array with <code>new</code> (dynamic allocation).</li> <li>- Using <code>std::vector&lt;int&gt;</code>.</li> <li>- Having the caller provide the array.</li> </ul> <p><b>Summary:</b> <b>Type of bug:</b> Returning address of local variable (dangling pointer/undefined behavior/use-after-scope).</p>
--	---	---



	<pre>std::vector&lt;int&gt; getArray(int size) {     return std::vector&lt;int&gt;(size);     // Return a vector, safe and efficient } ...</pre> <p>This is the preferred approach in modern C++ as it avoids manual memory management and issues like dangling pointers.</p>	
--	---	--

Here, there is a major difference between models. GPT 4.0 provided potential code fixes, which was not requested, while GPT 4.1 correctly only provided an explanation and classification of the bug. This could be explained by 4.0’s general trend of over explaining problems in its responses.

Task 4

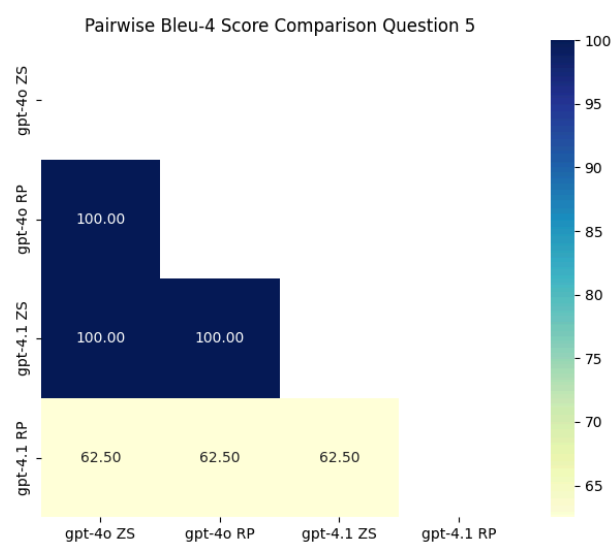


Goal	Python Method Generation		
Model	OpenAI GPT-4o	OpenAI GPT-4.1	
Temperature	1	Token Limit	1024
Role	You are an expert software engineer with many years of experience.		
Prompt	Generate a python method starting with <code>def is_valid_email(email):</code> that validates email addresses using a regex function. Given me only the code, no commentary.		
Zero-Shot	<code>python</code>	<code>python</code>	

<b>Output</b>	<pre>import re  def is_valid_email(email):     pattern =     r'^[a-zA-Z0-9_+-.]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-]+\.\$'</pre> <pre>        return re.match(pattern, email)     is not None     ...</pre>	<pre>import re  def is_valid_email(email):     pattern =     r'^[a-zA-Z0-9_+%+-.]+@[a-zA-Z0-9-]+\.[a-zA-Z]{2,}\$'</pre> <pre>        return bool(re.match(pattern, email))     ...</pre>
<b>Role-Playing Output</b>	<pre>```python import re  def is_valid_email(email):     pattern =     r'^[a-zA-Z0-9_+-.]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-]+\.\$'</pre> <pre>        return re.match(pattern, email)     is not None     ...</pre>	<pre>```python import re  def is_valid_email(email):     pattern =     r'^[a-zA-Z0-9_+-.]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-]+\.\$'</pre> <pre>        return re.match(pattern, email) is not     None     ...</pre>

In this task GPT 4o with zero shot and roleplay gave the same response, which was similar to GPT 4.1 with roleplay’s response. The outlier was GPT 4.1 with zeroshot, which adds a % sign for allowable characters in the email name and makes it so the domain is at least 2 characters. GPT 4.1 with zero shot is the most correct as email domains must have at least 2 characters, however while a percent sign can be valid for an email, it’s not always accepted.

Task 5

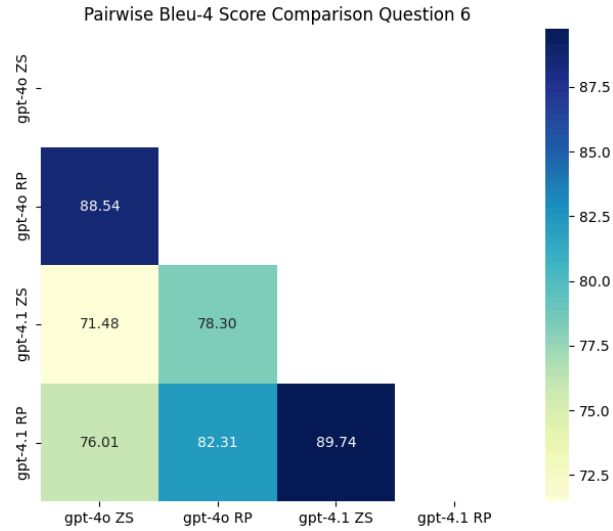


<b>Goal</b>	Python Code Completion
-------------	------------------------

<b>Model</b>	OpenAI GPT-4o	OpenAI GPT-4.1	
<b>Temperature</b>	1	<b>Token Limit</b>	1024
<b>Role</b>	You are an expert software engineer with many years of experience.		
<b>Prompt</b>	<p>I'm using Flask API in Python and need to create a '/greet/&lt;username&gt;' endpoint which returns a JSON greeting. Given me only the code, no commentary. Here is my code so far:</p> <pre> from flask import Flask , jsonify app = Flask ( __name__ ) @app . route ( '/ greet / &lt; username &gt; ' ) def greet ( username ) : # TODO : Return a JSON greeting pass </pre>		
<b>Zero-Shot Output</b>	<pre> python from flask import Flask, jsonify app = Flask(__name__) @app.route('/greet/&lt;username&gt;') def greet(username):     return jsonify({"message": f"Hello, {username}!"}) </pre>	<pre> python from flask import Flask, jsonify app = Flask(__name__)  @app.route('/greet/&lt;username&gt;') def greet(username):     return jsonify({"message": f"Hello, {username}!"}) </pre>	
<b>Role-Playing Output</b>	<pre> python from flask import Flask, jsonify app = Flask(__name__)  @app.route('/greet/&lt;username&gt;') def greet(username):     return jsonify({"message": f"Hello, {username}!"}) </pre>	<pre> python from flask import Flask, jsonify app = Flask(__name__)  @app.route('/greet/&lt;username&gt;') def greet(username):     return jsonify({'greeting': f"Hello, {username}!"})  if __name__ == '__main__':     app.run() </pre>	

All of the models except 4.1 roleplay gave the same response, 4.1 roleplay added main to run the app. A developer prompting a model would know whether they want the main section, so this isn't a significant difference.

## Task 6

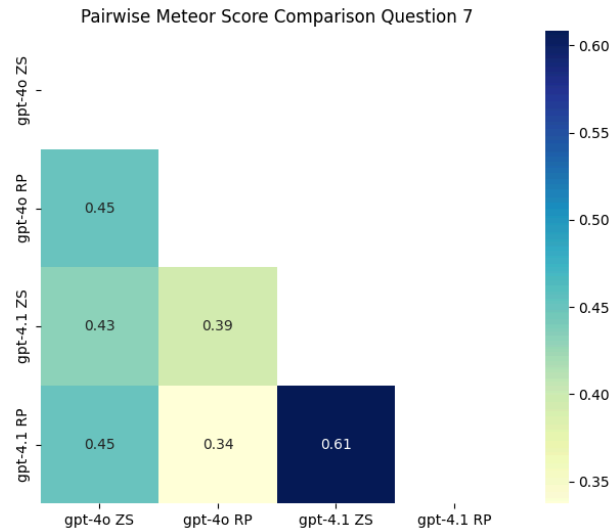


Goal	SQL Schema Generation		
Model	OpenAI GPT-4o	OpenAI GPT-4.1	
Temperature	1	Token Limit	1024
Role	You are an expert software engineer with many years of experience.		
Prompt	I'm trying to design an SQL schema for a review app containing users, books, and reviews. I have the following pseudocode, please generate me a schema. Given me only the code, no commentary. ``-- TODO : Design schema with appropriate keys and constraints -- Tables : users ( id , name ) , books ( id , title ) , reviews ( id , user_id , book_id , rating )``		
Zero-Shot Output	<pre> ```sql CREATE TABLE users (     id SERIAL PRIMARY KEY,     name VARCHAR(255) NOT NULL );  CREATE TABLE books (     id SERIAL PRIMARY KEY,     title VARCHAR(255) NOT NULL );  CREATE TABLE reviews (     id SERIAL PRIMARY KEY,     user_id INT NOT NULL,     book_id INT NOT NULL,     rating INT NOT NULL CHECK (rating &gt;= 1 AND rating &lt;= 5),     FOREIGN KEY (user_id) </pre>	<pre> ```sql CREATE TABLE users (     id SERIAL PRIMARY KEY,     name VARCHAR(255) NOT NULL );  CREATE TABLE books (     id SERIAL PRIMARY KEY,     title VARCHAR(255) NOT NULL );  CREATE TABLE reviews (     id SERIAL PRIMARY KEY,     user_id INTEGER NOT NULL,     book_id INTEGER NOT NULL,     rating INTEGER NOT NULL CHECK (rating &gt;= 1 AND rating &lt;= 5),     CONSTRAINT fk_user FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,     CONSTRAINT fk_book FOREIGN </pre>	

	REFERENCES users(id) ON DELETE CASCADE, FOREIGN KEY (book_id) REFERENCES books(id) ON DELETE CASCADE ); ...	KEY (book_id) REFERENCES books(id) ON DELETE CASCADE, CONSTRAINT unique_user_book_review UNIQUE (user_id, book_id) ); ...
<b>Role-Playing Output</b>	<pre> ```sql CREATE TABLE users (     id SERIAL PRIMARY KEY,     name VARCHAR(255) NOT NULL );  CREATE TABLE books (     id SERIAL PRIMARY KEY,     title VARCHAR(255) NOT NULL );  CREATE TABLE reviews (     id SERIAL PRIMARY KEY,     user_id INT NOT NULL,     book_id INT NOT NULL,     rating INT CHECK(rating &gt;= 1 AND rating &lt;= 5),     FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,     FOREIGN KEY (book_id) REFERENCES books(id) ON DELETE CASCADE,     UNIQUE (user_id, book_id) ); ``` </pre>	<pre> ```sql CREATE TABLE users (     id SERIAL PRIMARY KEY,     name VARCHAR(255) NOT NULL );  CREATE TABLE books (     id SERIAL PRIMARY KEY,     title VARCHAR(255) NOT NULL );  CREATE TABLE reviews (     id SERIAL PRIMARY KEY,     user_id INT NOT NULL,     book_id INT NOT NULL,     rating INT NOT NULL,     CONSTRAINT fk_user FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,     CONSTRAINT fk_book FOREIGN KEY (book_id) REFERENCES books(id) ON DELETE CASCADE,     CONSTRAINT unique_user_book UNIQUE (user_id, book_id),     CHECK (rating &gt;= 1 AND rating &lt;= 5) ); ``` </pre>

The output between the roleplay and zero shot responses were very similar to each other within each model. Differences in responses were seen between models, where GPT 4.1 had differences in its reviews table. This could be because creating an SQL schema is a standardized task, and the two models were working off slightly different standards for their schema generation.

## Task 7



Goal	Java Error Cause Analysis		
Model	OpenAI GPT-4o	OpenAI GPT-4.1	
Temperature	1	Token Limit	1024
Role	You are an expert software engineer with many years of experience.		
Prompt	Why am I getting null errors in this Java code: ``public int getLength ( String s ) { return s . length () ; // What if s is null ? }``		
Zero-Shot Output	<p>The null errors in your code are a result of trying to call the <code>length()</code> method on a <code>String</code> object that is <code>null</code>. When <code>s</code> is <code>null</code>, <code>s.length()</code> throws a <code>NullPointerException</code> because there is no valid <code>String</code> object on which the method can operate.</p> <p>Here's a breakdown of the issue:</p> <ul style="list-style-type: none"> <li>- The parameter <code>s</code> is of type <code>String</code>, but it could be <code>null</code>. In Java, <code>null</code> represents a reference that doesn't point to any object.</li> <li>- When you call a method (like <code>length()</code>) on a <code>null</code> reference, Java throws a <code>NullPointerException</code>, because there is no object to invoke the method on.</li> </ul> <p>### Example:</p> <pre>``java</pre> <p>You're getting <code>`null`</code> errors (specifically, a <code>**NullPointerException**</code>) in your Java code because <code>**you're calling a method (`length()`) on a variable that could be `null`**</code>.</p> <p>Let's look at your code:</p> <pre>``java public int getLength(String s) {     return s.length(); // What if s is null ? } ``</pre> <p>If you call <code>getLength(null)</code>, Java will try to execute <code>null.length()</code>, which is <b>illegal</b> because <code>null</code> doesn't refer to any object. This will throw a <code>NullPointerException</code>.</p> <p>### Example:</p> <pre>``java String myString = null; System.out.println(getLength(myString)); ``</pre>		

	<pre> getLength(null); // This will throw NullPointerException because s is null. '''  ### Solution:  To avoid <b>NullPointerException</b>, you can check if <b>s</b> is <b>null</b> before calling <b>s.length()</b>:  '''java public int getLength(String s) {     if (s == null) {         return 0; // or some other         default value     }     return s.length(); } '''  ### Explanation of the fix: 1. The <b>if (s == null)</b> condition checks if the input string `s` is null. 2. If <b>s</b> is null, you can return a default value (e.g., <b>0</b> or <b>-1</b>, based on your requirements) to ensure you don't try to access <b>s.length()</b>. 3. If <b>s</b> is not null, then <b>s.length()</b> can safely execute, and you return its result.  By adding this null check, you've addressed the possibility of handling <b>null</b> parameters gracefully and avoided runtime exceptions in your program.  ### Notes: - Assign meaningful behavior to null inputs based on your use case. For example, returning <b>0</b> might make sense for some cases (e.g., treating null as an empty string), while throwing an exception or returning a special value (<b>-1</b>, for example) might be better in other contexts. - If the method is not supposed to accept <b>null</b> as a valid input, you can explicitly throw an exception:  '''java public int getLength(String s) {     if (s == null) {         throw new IllegalArgumentExcep tion("String cannot be null");     } </pre>	<p><b>Output:</b></p> <pre> ''' Exception in thread "main" java.lang.NullPointerException '''  ---  ## How to fix it  You should check if <b>s</b> is <b>null</b> before calling <b>length()</b>, and decide what to do in that case (return <b>0</b>? Throw a custom exception?).  <b>Example:</b>  '''java public int getLength(String s) {     if (s == null) {         return 0; // or throw new         IllegalArgumentExcep tion("String cannot be null");     }     return s.length(); } '''  ---  <b>In summary:</b> You're getting null errors because you're calling a method on `null`, which isn't allowed in Java. Always check for <b>null</b> before calling methods on reference types! </pre>
--	--	--

	<pre> return s.length(); } ... </pre>	
<b>Role-Playing Output</b>	<p>The null errors you're experiencing in your Java code arise because of the possibility that the parameter <code>s</code>, which is an instance of <code>String</code>, might be <code>null</code> when the method <code>getLength</code> is called. Invoking any instance method (like <code>length()</code> in this case) on a <code>null</code> reference will result in a <code>NullPointerException</code> at runtime.</p> <p>Here's the breakdown of the problem:</p> <ol style="list-style-type: none"> <li><b>Method signature:</b> Your method accepts a <code>String</code> parameter <code>s</code>. There is no constraint in Java that prevents someone from passing <code>null</code> for this parameter.</li> <li><b><code>length()</code> method call:</b> The <code>length()</code> method of the <code>String</code> class is an instance method that operates on a <code>String</code> object. If you call it on <code>null</code>, Java will attempt to dereference the null pointer, resulting in a <code>NullPointerException</code>.</li> <li><b>What if <code>s</code> is <code>null</code>?</b> If someone calls <code>getLength(null)</code>, then in the line <code>return s.length();</code>, <code>s</code> is <code>null</code>. At this point, the program crashes with a <code>NullPointerException</code>.</li> </ol> <p>#### Solution</p> <p>To avoid <code>NullPointerException</code>, you should check whether <code>s</code> is <code>null</code> before calling any instance methods on it. Here's an updated, safe version of your code:</p> <pre> java public int getLength(String s) {     if (s == null) {         // Handle the null case         return 0; // Example: return 0 length for null strings     }     return s.length(); } ... </pre>	<p>You're getting <b>null errors</b> (specifically, a <code>NullPointerException</code>) because you are calling <code>s.length()</code> without first checking whether <code>s</code> is <code>null</code>. If <code>s</code> is <code>null</code>, this line:</p> <pre> java return s.length(); </pre> <p>will throw a <code>NullPointerException</code> at runtime, since you cannot call a method (<code>length()</code>) on a null reference.</p> <p>#### How to Fix</p> <p>You need to check if <code>s</code> is <code>null</code> before using it:</p> <pre> java public int getLength(String s) {     if (s == null) {         // Decide how you want to handle null: return 0 or throw an exception         return 0;         // Or: throw new IllegalArgumentException("String cannot be null");     }     return s.length(); } ... </pre> <p>#### Summary</p> <p><b><code>NullPointerException</code></b> is thrown because you tried to invoke a method on a <code>null</code> object. Always check for <code>null</code> before using variables that might be <code>null</code>.</p>



### Explanation of the Fix:

1. A **null** check (`if (s == null)`) is added to detect if the input is **null**.

2. In this example, I chose to return 0 for a **null** string. This behavior depends on your specific requirements.

**Other options include:**

- Throwing an exception to signal that a 'null' input is not allowed:

```
```java
    if (s == null) {
        throw new
IllegalArgumentException("Input string
cannot be null");
    }
```
```

- Returning a special value (e.g., `-1`) to indicate a **null** input.

- Handling the null case in a custom way based on your application's logic.

### Best Practices

**1. Always validate input arguments:**

Before processing any input, check for **null** or other invalid states where applicable. This prevents runtime errors like **NullPointerException**.

**2. Use annotations to indicate**

**nullability**: Modern development tools and libraries can help clarify which parameters can be **null**. For example:

- Use **@Nullable** to explicitly declare that a parameter can be 'null' and should be checked.

- Use **@NonNull** to declare a parameter must not be **null**.

Example:

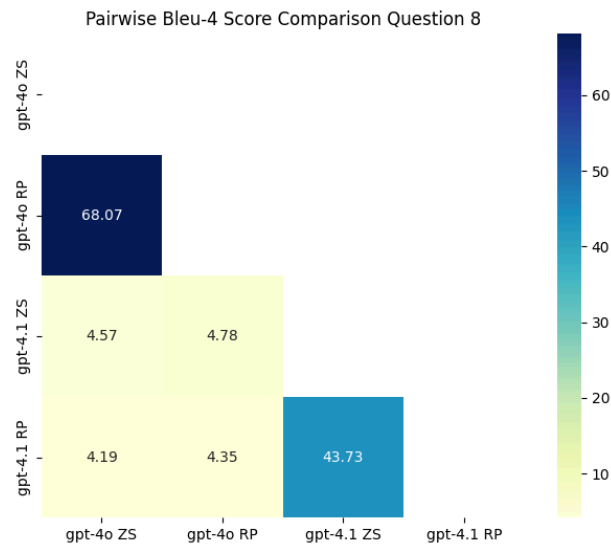
```
```java
public int getLength(@Nullable
String s) {
    if (s == null) {
        return 0;
    }
    return s.length();
}
```
```

**3. Leverage **Optional** where appropriate**: Java 8 introduced

|  |   |  |
|--|---|--|
|  | <p><b>Optional</b>, which can help you wrap nullable values in a more structured manner. Example:</p> <pre>```java public int getLength(Optional&lt;String&gt; s) {     return s.map(String::length).orElse(0); } ```</pre> <p>By incorporating null checks or annotations, you can make your code safer and more robust.</p> |  |
|--|---|--|

For this problem, GPT 4.1 did a much better job at explaining the issue in a concise manner, with GPT 4.0 frequently repeating information it had already explained. When given an expert role, 4.0’s explanation got longer, while 4.1’s explanation got shorter. This could be due to 4.0 interpreting its role as having to give as much detail as possible, while 4.1 interpreted the role as having to explain the issue in a concise manner.

Task 8



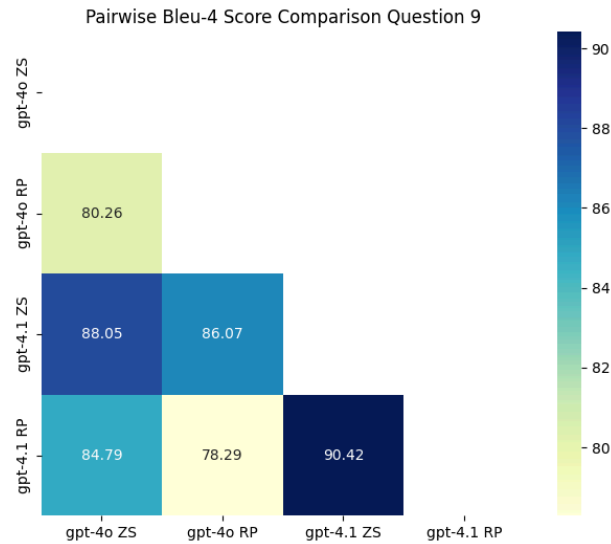
|             |                               |                |      |
|-------------|-------------------------------|----------------|------|
| Goal        | Improve the Python CVS parser |                |      |
| Model       | OpenAI GPT-4o                 | OpenAI GPT-4.1 |      |
| Temperature | 1                             | Token Limit    | 1024 |

|                            |  |  |
|----------------------------|--|--|
| <b>Role</b>                | You are an expert software engineer with many years of experience.   |  |
| <b>Prompt</b>              | Implement quoted fields support for the following python CSV parser, Given me only the code, no commentary: ```def parse_csv_line ( line ) : return line . split ( ' , ' ) # Incomplete : doesn 't handle quoted fields``` |  |
| <b>Zero-Shot Output</b>    | <pre>```python import csv  def parse_csv_line(line):     return next(csv.reader([line], skipinitialspace=True)) ```</pre>  | <pre>```python import re  def parse_csv_line(line):     pattern = re.compile(r" \s* ( "(?:[^\"] ")*" # Quoted field, supports escaped ""   [^\,]* # Unquoted field ) \s* (?:, \$) ", re.VERBOSE) fields = [] pos = 0 line_len = len(line) while pos &lt; line_len:     m = pattern.match(line, pos)     if not m:         break     field = m.group(1)     if field.startswith('"') and field.endswith('"'):         field = field[1:-1].replace('"', '')     fields.append(field)     pos = m.end() return fields ```</pre> |
| <b>Role-Playing Output</b> | <pre>```python import csv from io import StringIO  def parse_csv_line(line):     return next(csv.reader(StringIO(line))) ```</pre>   | <pre>```python import re  def parse_csv_line(line):     pattern = re.compile(r" \s* # Optional leading whitespace (?: # Start non-capturing group for either: " # Opening quote ( # Capture group for quoted field (?: [^\"]   " ) * # Any non-quote or escaped double quote ("" ) ) " # Closing quote   # OR ([^\,]* # Capture unquoted field (no commas) </pre>  |

|  |  |   |
|--|--|---|
|  |  | <pre>) \s*          # Optional trailing whitespace (?:, \$)      # Field ends with comma or string end ", re.VERBOSE)  fields = [] for match in pattern.finditer(line):     quoted, unquoted = match.group(1), match.group(2)     if quoted is not None:         fields.append(quoted.replace('"', ''))     else:         fields.append(unquoted) return fields '''</pre> |
|--|--|---|

In the Bleu-4 score analysis the GPT 4o zero-shot was most similar to the GPT 4o roleplay, this pattern was the same for the 4.1 model. The GPT 4o model is more concise than the GPT 4.1 model. The GPT 4o model relies on external libraries to handle the problem, while the 4.1 model handles the end cases within the code. I think the GPT 4o model achieved the goal without being too verbose.

## Task 9



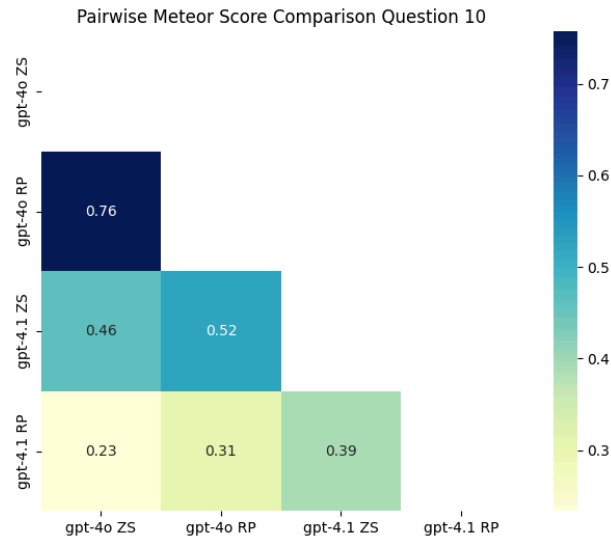
|             |                      |                |      |
|-------------|----------------------|----------------|------|
| Goal        | Ktor Code Generation |                |      |
| Model       | OpenAI GPT-4o        | OpenAI GPT-4.1 |      |
| Temperature | 1                    | Token Limit    | 1024 |

|                            |  |  |
|----------------------------|--|--|
| <b>Role</b>                | You are an expert software engineer with many years of experience.   |  |
| <b>Prompt</b>              | Use Ktor to convert this data class into a REST API, Given me only the code, no commentary: ```data class Product ( val id : Int , val name : String , val price : Double )<br>// TODO : Create GET and POST endpoints using Ktor```   |  |
| <b>Zero-Shot Output</b>    | <pre> ```kotlin import io.ktor.application.* import io.ktor.features.* import io.ktor.http.* import io.ktor.request.* import io.ktor.response.* import io.ktor.routing.* import io.ktor.server.engine.* import io.ktor.server.netty.*  data class Product(val id: Int, val name: String, val price: Double)  fun main() {     embeddedServer(Netty, port = 8080) {         install(ContentNegotiation) {             // Optional: Configure serializer if needed (e.g., Gson, Jackson, Kotlinx-serialization)         }          routing {             val products = mutableListOf&lt;Product&gt;()              get("/products") {                 call.respond(products)             }              post("/products") {                 val product = call.receive&lt;Product&gt;()                 products.add(product)                  call.respond(HttpStatusCode.Created, product)             }              }.start(wait = true)         }     } } ``` </pre> | <pre> ```kotlin import io.ktor.application.* import io.ktor.features.ContentNegotiation import io.ktor.http.* import io.ktor.request.* import io.ktor.response.* import io.ktor.routing.* import io.ktor.serialization.* import io.ktor.server.engine.* import io.ktor.server.netty.*  data class Product(val id: Int, val name: String, val price: Double)  fun main() {     val products = mutableListOf&lt;Product&gt;()      embeddedServer(Netty, port = 8080) {         install(ContentNegotiation) {             json()         }         routing {             get("/products") {                 call.respond(products)             }             post("/products") {                 val product = call.receive&lt;Product&gt;()                 products.add(product)                 call.respond(HttpStatusCode.Created, product)             }         }     }.start(wait = true) } ``` </pre> |
| <b>Role-Playing Output</b> | <pre> ```kotlin import io.ktor.application.* import io.ktor.http.* import io.ktor.request.* import io.ktor.response.* import io.ktor.routing.* </pre>  | <pre> ```kotlin import io.ktor.application.* import io.ktor.http.* import io.ktor.request.* import io.ktor.response.* import io.ktor.routing.* </pre>  |

|  |  |   |
|--|--|---|
|  | <pre> import io.ktor.server.engine.* import io.ktor.server.netty.*  data class Product(val id: Int, val name: String, val price: Double)  fun main() {     val products = mutableListOf&lt;Product&gt;()      embeddedServer(Netty, port = 8080) {         routing {             get("/products") {                 call.respond(products)             }              post("/products") {                 val product = call.receive&lt;Product&gt;()                 products.add(product)                  call.respond(HttpStatusCode.Created, product)             }         }     }.start(wait = true) } ... </pre> | <pre> import io.ktor.server.engine.* import io.ktor.server.netty.* import kotlinx.serialization.* import kotlinx.serialization.json.* import io.ktor.features.* import io.ktor.serialization.*  @Serializable data class Product(val id: Int, val name: String, val price: Double)  fun main() {     val products = mutableListOf&lt;Product&gt;()      embeddedServer(Netty, port = 8080) {         install(ContentNegotiation) {             json()         }         routing {             get("/products") {                 call.respond(products)             }             post("/products") {                 val product = call.receive&lt;Product&gt;()                 products.add(product)                 call.respond(HttpStatusCode.Created, product)             }         }     }.start(wait = true) } ... </pre> |
|--|--|---|

By the graph, both GPT 4.1 and the 4.0 Zeroshot response were quite similar, with the 4.0 roleplaying response showing the most difference when compared to the rest. It was also the most concise response, indicating GPT 4.0 may have interpreted its role in a way that led it to attempt to write more compact code, emulating what it would expect an expert to be able to do.

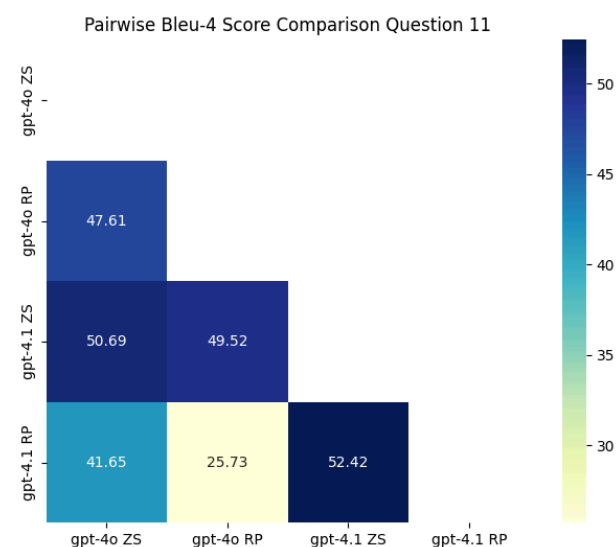
## Task 10



|                     |  |   |      |
|---------------------|--|---|------|
| Goal                | Python Code Summarization  |   |      |
| Model               | OpenAI GPT-4o  | OpenAI GPT-4.1  |      |
| Temperature         | 1  | Token Limit   | 1024 |
| Role                | You are an expert software engineer with many years of experience.   |   |      |
| Prompt              | Concisely summarize this python method: <code>def reverse_words ( sentence ) :<br/>return ' '.join ( sentence . split () [::-1])</code>  |   |      |
| Zero-Shot Output    | The <code>reverse_words</code> method takes a sentence, splits it into words, reverses their order, and joins them back into a string with spaces.   | The method <code>reverse_words</code> takes a sentence and returns a new sentence with the order of the words reversed. |      |
| Role-Playing Output | This Python method <code>reverse_words</code> takes a sentence as input, splits it into words, reverses the word order, and joins them back into a string with spaces. It effectively reverses the word order in the sentence. | This method reverses the order of words in a sentence.  |      |

There was not much of a difference in the actual content of each response, each model correctly summarized the method, but GPT 4.1 was better at being concise than GPT 4.0, likely due to it being a more recent model. Zero-shot worked better for GPT 4.0, while RP worked better for 4.1. This may be because 4.0 focused more on explaining in detail, while 4.1 focused more on being concise.

## Task 11



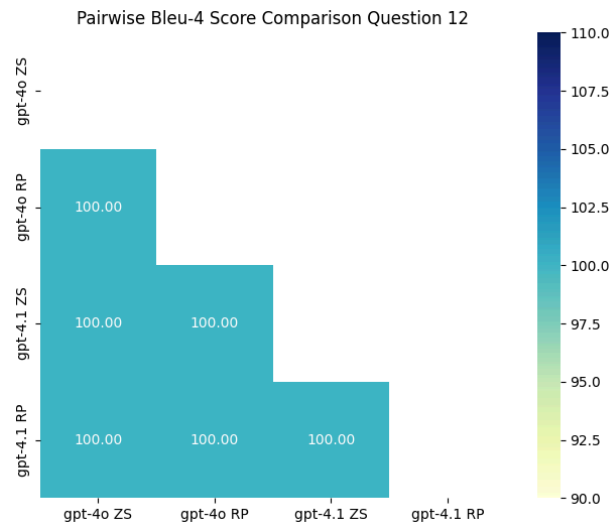
|                  |   |   |      |
|------------------|---|---|------|
| Goal             | Python Code Generation  |   |      |
| Model            | OpenAI GPT-4o   | OpenAI GPT-4.1  |      |
| Temperature      | 1   | Token Limit   | 1024 |
| Role             | You are an expert software engineer with many years of experience.  |   |      |
| Prompt           | Generate a python function that checks if a number is prime.  |   |      |
| Zero-Shot Output | <p>Here is a Python function that checks if a number is prime:</p> <pre> python def is_prime(n):     """         Check if a number is a prime number.          Args:             n (int): The number to check.          Returns:             bool: True if n is a prime number, False otherwise.     """     if n &lt;= 1:         return False # Numbers less than or equal to 1 are not prime     if n &lt;= 3:         return True # 2 and 3 are prime numbers     if n % 2 == 0 or n % 3 == 0:         return False # Exclude multiples of 2 and 3 </pre> | <p>Certainly! Here is a simple Python function to check if a number is prime:</p> <pre> python def is_prime(n):     """Check if a number is prime."""     if n &lt;= 1:         return False     if n &lt;= 3:         return True     if n % 2 == 0 or n % 3 == 0:         return False     i = 5     while i * i &lt;= n:         if n % i == 0 or n % (i + 2) == 0:             return False         i += 6     return True </pre> <p><b>Example usage:</b></p> <pre> python print(is_prime(2)) # True print(is_prime(15)) # False print(is_prime(17)) # True </pre> |      |



|                            |   |   |
|----------------------------|---|---|
|                            | <pre> # Check divisors from 5 to √n i = 5 while i * i &lt;= n:     if n % i == 0 or n % (i + 2) == 0:         return False         i += 6 # Increment by 6 (check divisors 5, 7, 11, 13, etc.)      return True ...  ### How it works: 1. If `n` is less than or equal to 1, it is not a prime number. 2. If `n` is 2 or 3, it is prime. 3. Numbers divisible by 2 or 3 are not prime. 4. For numbers greater than 3, we check divisors from 5 up to the square root of `n`, skipping even numbers and multiples of 3. Increment `i` by 6 to only test numbers of the form `6k ± 1`.  ### Example Usage: ```python print(is_prime(2))      # True print(is_prime(17))    # True print(is_prime(18))    # False print(is_prime(19))    # True print(is_prime(25))    # False print(is_prime(97))    # True ```  This implementation is efficient and works well for a wide range of inputs. </pre> | <p>Let me know if you need more explanation or a simpler version!</p>   |
| <b>Role-Playing Output</b> | <p>Sure! Here's an efficient Python function to check if a number is prime:</p> <pre> ```python def is_prime(number):     """     Check if a number is prime.      Args:         number (int): The number to check.      Returns:         bool: True if the number is prime, False otherwise.     """     if number &lt;= 1:         return False </pre>  | <p>Certainly! Here is a Python function that checks whether a given number is a prime number:</p> <pre> ```python def is_prime(n):     """Return True if n is a prime number, otherwise False."""     if n &lt;= 1:         return False     if n == 2:         return True     if n % 2 == 0:         return False     for i in range(3, int(n ** 0.5) + 1, 2):         if n % i == 0:             return False     return True ``` </pre> |

|  |  |   |
|--|--|---|
|  | <pre>         if number &lt;= 3:             return True # 2 and 3 are prime numbers         if number % 2 == 0 or number % 3 == 0:             return False          # Check for factors from 5 to √number         i = 5         while i * i &lt;= number:             if number % i == 0 or number % (i + 2) == 0:                 return False             i += 6          return True '''  ### Explanation: 1. <b>Handle Small Numbers:</b> Numbers less than or equal to 1 are not prime. Numbers 2 and 3 are prime. 2. <b>Divisibility Optimization:</b> Eliminate even numbers and multiples of 3, as these cannot be prime (except 2 and 3 themselves). 3. <b>Efficient Factor Checking:</b> Use the fact that a composite number must have a factor less than or equal to its square root. Increment checks in steps of 6 to test only potential prime factors (i.e., numbers of the form <math>6k \pm 1</math>).  ### Example Usage: '''python print(is_prime(7))      # True print(is_prime(10))    # False print(is_prime(13))    # True print(is_prime(1))     # False print(is_prime(101))   # True '''  This function is efficient and works well for reasonably large numbers. Let me know if you have any questions! </pre> | <p><b>Example usage:</b></p> <pre> '''python print(is_prime(7))  # True print(is_prime(18)) # False ''' </pre> <p>This function efficiently checks for primes by:</p> <ul style="list-style-type: none"> <li>- Excluding numbers <math>\leq 1</math> and even numbers <math>&gt; 2</math></li> <li>- Only checking divisibility up to the square root of <math>n</math></li> <li>- Only considering odd divisors (after checking for even numbers)</li> </ul> |
|--|--|---|

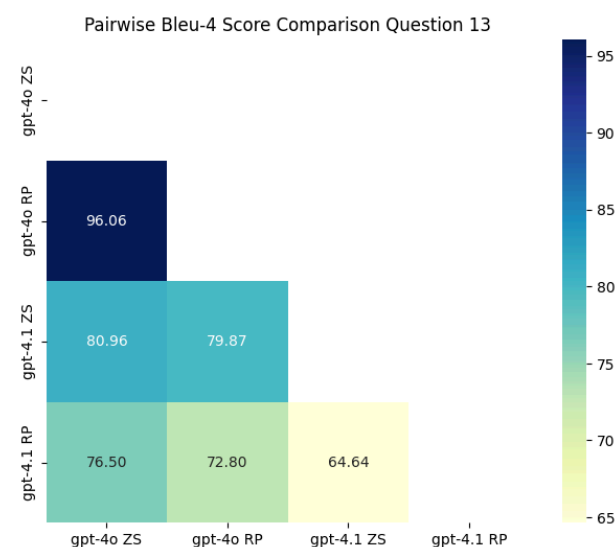
GPT 4o zero shot, roleplay and 4.1 zero shot are very much the same, the outlier is 4.1 roleplay. GPT 4.1 uses a different algorithm, which is less efficient; the others all check potential prime factors in the form of  $6k \pm 1$ .



|                            |   |  |      |
|----------------------------|---|--|------|
| <b>Goal</b>                | Python Bug Fixing   |  |      |
| <b>Model</b>               | OpenAI GPT-4o   | OpenAI GPT-4.1   |      |
| <b>Temperature</b>         | 1   | <b>Token Limit</b>   | 1024 |
| <b>Role</b>                | You are an expert software engineer with many years of experience.  |  |      |
| <b>Prompt</b>              | Fix the bug in this python method present when n=0, Given me only the code, no commentary.: ``def factorial ( n ) :<br>result = 1<br>for i in range ( 1 , n ) :<br>result *= i<br>return result`` |  |      |
| <b>Zero-Shot Output</b>    | ``python<br>def factorial(n):<br>result = 1<br>for i in range(1, n + 1):<br>result *= i<br>return result<br>``  | ``python<br>def factorial(n):<br>result = 1<br>for i in range(1, n + 1):<br>result *= i<br>return result<br>`` |      |
| <b>Role-Playing Output</b> | ``python<br>def factorial(n):<br>result = 1<br>for i in range(1, n + 1):<br>result *= i<br>return result<br>``  | ``python<br>def factorial(n):<br>result = 1<br>for i in range(1, n + 1):<br>result *= i<br>return result<br>`` |      |

All of the model's returned the same exact answer. This is a common enough problem with a straightforward solution, so I'm not surprised all the answers are the same.

## Task 13

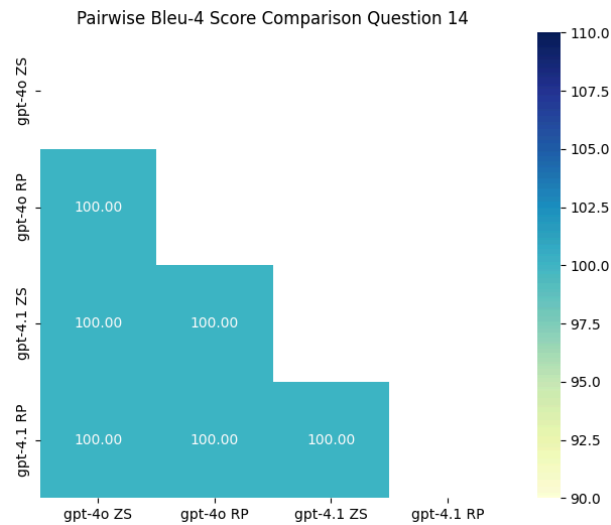


|                  |   |  |      |
|------------------|---|--|------|
| Goal             | C Code Completion   |  |      |
| Model            | OpenAI GPT-4o   | OpenAI GPT-4.1   |      |
| Temperature      | 1   | Token Limit  | 1024 |
| Role             | You are an expert software engineer with many years of experience.  |  |      |
| Prompt           | I'm trying to implement a linked list in C, and need to implement delete by value. Given me only the code, no commentary. I have this code already: ``struct Node {<br>int data ;<br>struct Node * next ;<br>};<br>void deleteNode ( struct Node ** head , int key ) {<br>// TODO : Implement node deletion<br>}``  |  |      |
| Zero-Shot Output | <pre>``c void deleteNode(struct Node **head, int key) {     struct Node *temp = *head,     *prev = NULL;      if (temp != NULL &amp;&amp; temp-&gt;data == key) {         *head = temp-&gt;next;         free(temp);         return;     }      while (temp != NULL &amp;&amp; temp-&gt;data != key) {         prev = temp;         temp = temp-&gt;next;     } }</pre> | <pre>``c void deleteNode(struct Node **head, int key) {     struct Node* temp = *head, *prev = NULL;     while (temp != NULL &amp;&amp; temp-&gt;data == key) {         *head = temp-&gt;next;         free(temp);         temp = *head;     }     while (temp != NULL) {         while (temp != NULL &amp;&amp; temp-&gt;data != key) {             prev = temp;             temp = temp-&gt;next;         }         if (temp == NULL) return;         prev-&gt;next = temp-&gt;next;         free(temp);     } }</pre> |      |

|                            |   |  |
|----------------------------|---|--|
|                            | <pre> if (temp == NULL) return;  prev-&gt;next = temp-&gt;next; free(temp); } ... </pre>  | <pre> temp = prev-&gt;next; } } ... </pre>   |
| <b>Role-Playing Output</b> | <pre> ```c void deleteNode(struct Node** head, int key) {     struct Node* temp = *head;     struct Node* prev = NULL;      if (temp != NULL &amp;&amp; temp-&gt;data == key) {         *head = temp-&gt;next;         free(temp);         return;     }      while (temp != NULL &amp;&amp; temp-&gt;data != key) {         prev = temp;         temp = temp-&gt;next;     }      if (temp == NULL) return;      prev-&gt;next = temp-&gt;next;     free(temp); } ... </pre> | <pre> ```c void deleteNode(struct Node **head, int key) {     struct Node *temp = *head, *prev = NULL;     while (temp != NULL &amp;&amp; temp-&gt;data != key) {         prev = temp;         temp = temp-&gt;next;     }     if (temp == NULL) return;     if (prev == NULL) *head = temp-&gt;next;     else prev-&gt;next = temp-&gt;next;     free(temp); } ... </pre> |

In the Bleu-4 analysis GPT 4o zero shot and GPT 4o roleplay are the most similar to each other, and surprisingly they are both fairly similar to GPT 4.1 zero shot. The difference is GPT 4.1 roleplays response deletes all with the value, and the others delete only one node. Depending on what you want either could work, the prompt isn't specific in its requirements. Giving the role of expert software engineer may be why the GPT 4.1 model inferred the intended behavior was to delete all with the value.

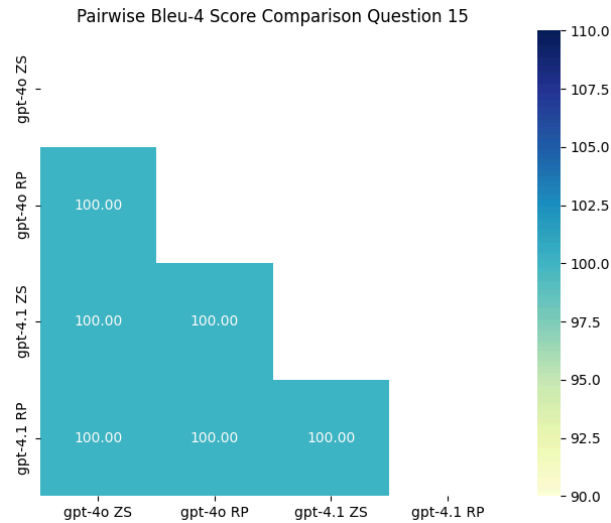
## Task 14



|                     |   |   |      |
|---------------------|---|---|------|
| Goal                | Python Code Generation  |   |      |
| Model               | OpenAI GPT-4o   | OpenAI GPT-4.1  |      |
| Temperature         | 1   | Token Limit   | 1024 |
| Role                | You are an expert software engineer with many years of experience.  |   |      |
| Prompt              | Generate a recursive fibonacci function from the pseudocode, Given me only the code, no commentary. ```def fibonacci ( n ) :<br># TODO : Base cases and recursive call<br>pass``` |   |      |
| Zero-Shot Output    | ```python<br>def fibonacci(n):<br>if n <= 1:<br>return n<br>return fibonacci(n - 1) +<br>fibonacci(n - 2)<br>```  | ```python<br>def fibonacci(n):<br>if n <= 1:<br>return n<br>return fibonacci(n - 1) + fibonacci(n - 2)<br>``` |      |
| Role-Playing Output | ```python<br>def fibonacci(n):<br>if n <= 1:<br>return n<br>return fibonacci(n - 1) +<br>fibonacci(n - 2)<br>```  | ```python<br>def fibonacci(n):<br>if n <= 1:<br>return n<br>return fibonacci(n - 1) + fibonacci(n - 2)<br>``` |      |

All of the models returned the same answer, which isn't surprising as fibonacci is a commonly used coding teaching problem.

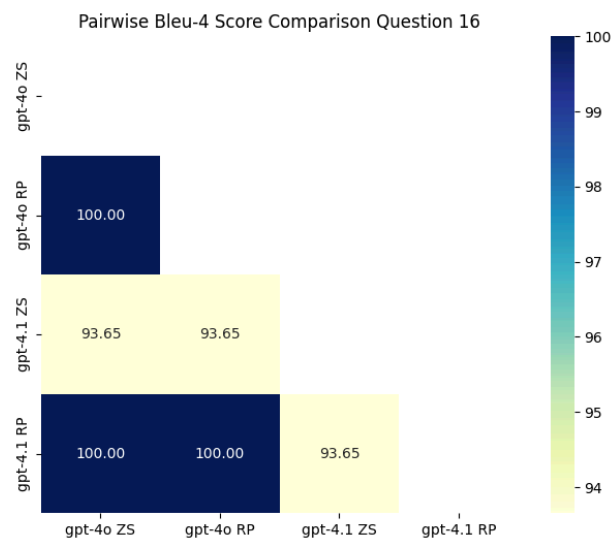
## Task 15



|                     |  |  |      |
|---------------------|--|--|------|
| Goal                | Python Code Completion   |  |      |
| Model               | OpenAI GPT-4o  | OpenAI GPT-4.1   |      |
| Temperature         | 1  | Token Limit  | 1024 |
| Role                | You are an expert software engineer with many years of experience.   |  |      |
| Prompt              | Complete this constructor in python, Given me only the code, no commentary: ``class Person :<br>def __init__( self ) :<br># TODO : Add name , age , and optional email<br>pass`` |  |      |
| Zero-Shot Output    | ``python<br>class Person:<br>def __init__(self, name, age,<br>email=None):<br>self.name = name<br>self.age = age<br>self.email = email<br>``                                     | ``python<br>class Person:<br>def __init__(self, name, age, email=None):<br>self.name = name<br>self.age = age<br>self.email = email<br>``    |      |
| Role-Playing Output | ``python<br>class Person:<br>def __init__(self, name, age,<br>email=None):<br>self.name = name<br>self.age = age<br>self.email = email<br>``                                     | ``python<br>class Person:<br>def __init__(self, name, age,<br>email=None):<br>self.name = name<br>self.age = age<br>self.email = email<br>`` |      |

In this case all the models returned the same output. This is a fairly simple coding problem and the models all completed it correctly.

Task 16



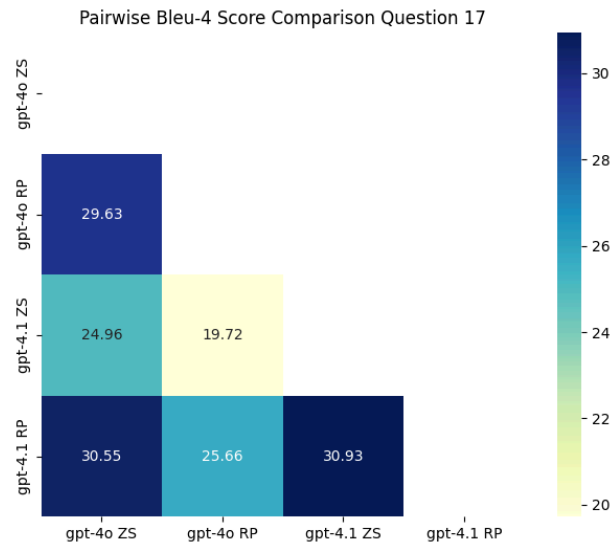
|                  |   |   |  |
|------------------|---|---|--|
| Goal             | Java Code Completion  |   |  |
| Model            | OpenAI GPT-4o   | OpenAI GPT-4.1  |  |
| Temperature      | 1   | Token Limit   |  |
| Role             | You are an expert software engineer with many years of experience.  |   |  |
| Prompt           | I am working on a binary search implementation in Java, can you complete this implementation for me, Given me only the code, no commentary: ```public int binarySearch ( int [] arr , int target ) { int left = 0 , right = arr . length - 1; while ( left <= right ) { int mid = ( left + right ) / 2; // TODO : Compare and adjust bounds } return -1; }``` |   |  |
| Zero-Shot Output | ```java public int binarySearch(int[] arr, int target) { int left = 0, right = arr.length - 1; while (left <= right) { int mid = left + (right - left) / 2; if (arr[mid] == target) { return mid; } else if (arr[mid] < target) { left = mid + 1; } else { right = mid - 1; } } return -1; }  | ```java public int binarySearch(int[] arr, int target) { int left = 0, right = arr.length - 1; while (left <= right) { int mid = (left + right) / 2; if (arr[mid] == target) { return mid; } else if (arr[mid] < target) { left = mid + 1; } else { right = mid - 1; } } return -1; } |  |



|                            |   |   |
|----------------------------|---|---|
|                            | ```   |   |
| <b>Role-Playing Output</b> | <pre>```java public int binarySearch(int[] arr, int target) {     int left = 0, right = arr.length - 1;     while (left &lt;= right) {         int mid = left + (right - left) / 2;         if (arr[mid] == target) {             return mid;         } else if (arr[mid] &lt; target) {             left = mid + 1;         } else {             right = mid - 1;         }     }     return -1; } ```</pre> | <pre>```java public int binarySearch(int[] arr, int target) {     int left = 0, right = arr.length - 1;     while (left &lt;= right) {         int mid = left + (right - left) / 2;         if (arr[mid] == target) {             return mid;         } else if (arr[mid] &lt; target) {             left = mid + 1;         } else {             right = mid - 1;         }     }     return -1; } ```</pre> |

All 4 responses are nearly identical, indicating that both models were able to find a common solution to the problem, likely due to its simplicity.

## Task 17



|                    |  |                    |      |
|--------------------|--|--------------------|------|
| <b>Goal</b>        | Resolve inconsistency between function name and logic:                                     |                    |      |
| <b>Model</b>       | OpenAI GPT-4o  | OpenAI GPT-4.1     |      |
| <b>Temperature</b> | 1  | <b>Token Limit</b> | 1024 |
| <b>Role</b>        | You are an expert software engineer with many years of experience.                         |                    |      |
| <b>Prompt</b>      | This C++ function does not do what I expect based on the function name, why is this? ```// |                    |      |

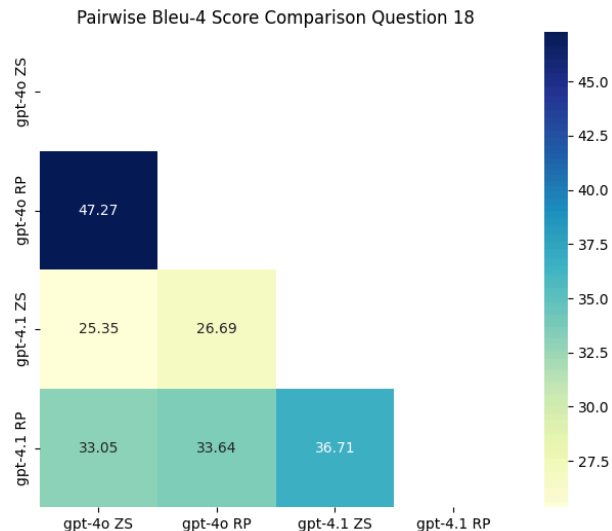
|                         |   |  |
|-------------------------|---|--|
|                         | <p>Supposed to return true if x is even</p> <pre>bool isOdd ( int x ) { return x % 2 == 0; // Logic contradicts function name }'''</pre>  |  |
| <b>Zero-Shot Output</b> | <p>The issue here is indeed with the <b>naming of the function</b> and the <b>logic inside the function</b> not matching. The function is named <b>isOdd</b>, which implies it should return <b>true</b> for odd numbers. However, the current logic within the function checks for <b>even</b> numbers instead of odd ones (i.e., <b>x % 2 == 0</b> evaluates to <b>true</b> for even numbers).</p> <p>Here is a breakdown of why this is happening:</p> <ol style="list-style-type: none"> <li><b>Function Name:</b> The name <b>isOdd</b> suggests the function will return <b>true</b> when the input <b>x</b> is odd.</li> <li><b>Logic Inside:</b> The condition <b>x % 2 == 0</b> checks if <b>x</b> is divisible by 2 without a remainder. This condition is true for even numbers, not odd ones.</li> <li><b>Contradiction:</b> Since the logic does not align with the function name, the function ends up doing the opposite of what its name suggests.</li> </ol> <p>### How to Fix This</p> <p>To resolve the confusion, you should align the function's name and its logic. You have two options:</p> <p>---</p> <p><b>Option 1: Fix the Function Name</b></p> <p>Change the function name to <b>isEven</b>, because the logic checks for even numbers.</p> <pre>'''cpp // Correctly named function to check if x is even bool isEven(int x) {     return x % 2 == 0; }'''</pre> <p>---</p> <p><b>Option 2: Fix the Logic</b></p> | <p>Great observation! The issue here is a classic mismatch between the <b>function's name</b> and its <b>implementation logic</b>.</p> <p>### Let's break it down:</p> <ul style="list-style-type: none"> <li>- <b>Function Name:</b> <b>isOdd</b> <ul style="list-style-type: none"> <li>- This name implies the function <b>should return true if x is odd</b>.</li> </ul> </li> <li>- <b>Current Implementation:</b> <pre>'''cpp return x % 2 == 0; '''</pre> <ul style="list-style-type: none"> <li>- This returns <b>true if x is even</b>, because numbers divisible by 2 (those with a remainder 0 after dividing by 2) are even.</li> </ul> </li> </ul> <p>### Why the mismatch?</p> <p>The code says "return true if x is even," but the name says "isOdd." This contradiction causes confusion and bugs.</p> <p>### Fixing the Code</p> <p>To match the function name, you should check if <b>x is odd</b>:</p> <pre>'''cpp bool isOdd(int x) {     return x % 2 != 0; }'''</pre> <p>Alternatively, <b>if you actually wanted to test for evenness</b>, rename the function:</p> <pre>'''cpp bool isEven(int x) {     return x % 2 == 0; }'''</pre> <p>---</p> <p><b>Summary:</b></p> <ul style="list-style-type: none"> <li>- The implementation checks for evenness, contradicting the function name <b>isOdd</b>.</li> <li>- To align implementation with the name, change</li> </ul> |

|                     |   |  |
|---------------------|---|--|
|                     | <p>Keep the function name <code>isOdd</code>, but change the logic to correctly reflect the intention of checking for odd numbers.</p> <pre> '''cpp // Correct logic for checking if x is odd bool isOdd(int x) {     return x % 2 != 0; } '''  ---</pre> <p>### Key Takeaway<br/>It's important to ensure that the <b>name of a function</b> and its <b>logic</b> are consistent with each other. Otherwise, it can lead to confusion and bugs. In this case, the mismatch between the name <code>isOdd</code> and the condition <code>x % 2 == 0</code> caused the function to do something opposite to what its name suggests.</p>   | <p>to <code>x % 2 != 0</code>.</p> <ul style="list-style-type: none"> <li>- Always keep function names and their logic consistent!</li> </ul>  |
| Role-Playing Output | <p>The issue lies in a <b>mismatch between the function's name and its implementation logic</b>. The function name, <code>isOdd</code>, implies that the function should check whether a number is odd. However, the implementation logic (<code>x % 2 == 0</code>) checks if the number <code>x</code> is even, which contradicts the function's name.</p> <p>### Problem Details</p> <ul style="list-style-type: none"> <li>- <b>Function Name:</b> <code>isOdd</code> suggests it should return <code>true</code> when the input is an odd number.</li> <li>- <b>Implementation Logic:</b> The expression <code>x % 2 == 0</code> evaluates to <code>true</code> if <code>x</code> is even, not odd.</li> </ul> <p>### Why This Happens?</p> <p>This appears to be a <b>naming and semantic mistake</b>. Either:</p> <ol style="list-style-type: none"> <li>1. The <b>name</b> of the function is wrong and should reflect "checking for evenness."</li> <li>2. The <b>logic</b> inside the function is wrong and should align with the name to check for oddness.</li> </ol> <p>### How to Fix It?</p> <p>You need to either change the <b>function name</b> or the <b>logic</b> so that the function is consistent and does what a developer</p> | <p>You are <b>absolutely correct</b> the function's <b>name and logic are mismatched</b>. Let's break it down:</p> <p>### The Code:</p> <pre> '''cpp // Supposed to return true if x is even bool isOdd ( int x ) {     return x % 2 == 0; // Logic contradicts function name } '''  ### Analysis: <ul style="list-style-type: none"> <li>- <b>Function Name:</b> <code>isOdd</code><br/><i>Implies that it should return <code>true</code> if <code>x</code> is odd.</i></li> <li>- <b>Function Logic:</b> <code>return x % 2 == 0;</code><br/><i>This condition returns <code>true</code> when <code>x</code> is even, not odd.</i></li> </ul> <p>### Why is this a problem?</p> <ul style="list-style-type: none"> <li>- There is a <b>contradiction</b>:<br/>The logic checks if <code>x</code> is <b>even</b> (i.e., remainder is zero), but the name says "odd."</li> <li>- This <b>confuses anyone using your code</b> and can cause subtle bugs if not noticed.</li> </ul> <p>---</p> </pre> |

|   | <p>expects based on its name.</p> <p>##### Option 1: Fix the Function Name<br/>If the intention is to check if the number is <b>even</b>, rename the function to <b>isEven</b>:</p> <pre>```cpp // Check if x is even bool isEven(int x) {     return x % 2 == 0; // Correctly reflects the function's purpose } ```</pre> <p>##### Option 2: Fix the Logic<br/>If the intention is to check if the number is <b>odd</b>, fix the logic to:</p> <pre>```cpp // Check if x is odd bool isOdd(int x) {     return x % 2 != 0; // Correctly reflects the function's purpose } ```</pre> <p>#### Why Does This Matter?<br/>Naming functions accurately and writing clear, logical code is crucial for <b>readability</b>, <b>maintainability</b>, and <b>avoiding bugs</b>. When someone else (or even your future self) looks at your code, they should be able to deduce the function's purpose.</p> <p>For example:</p> <pre>```cpp std::cout &lt;&lt; isOdd(2); // Expected: false, Actual: true (with current incorrect implementation) std::cout &lt;&lt; isOdd(3); // Expected: true, Actual: false (with current incorrect implementation) ```</pre> <p>This discrepancy can lead to confusing behavior that's difficult to debug.</p> <p>Always ensure that:</p> <ol style="list-style-type: none"><li>1. The <b>function's name</b> is descriptive and accurately reflects its purpose.</li><li>2. The <b>logic</b> inside the function matches the intended behavior implied by the name.</li></ol> | <p>## How to fix?</p> <p><b>Option 1: Fix the function name</b> (if the logic is correct):</p> <pre>```cpp bool isEven(int x) {     return x % 2 == 0; } ```</pre> <p><b>Option 2: Fix the function logic</b> (if the name is what you want):</p> <pre>```cpp bool isOdd(int x) {     return x % 2 != 0;     // or: return (x &amp; 1) != 0; // bitwise, sometimes more performant } ```</pre> <p>---</p> <p>## Summary Table:</p> <table><tr><th colspan="3">  Desired Functionality   Function Name   Logic</th></tr><tr><td colspan="3"> ----- ----- ----- </td></tr><tr><td>----- </td><td></td><td></td></tr><tr><td>  Return true if even  </td><td>isEven</td><td>  x % 2 == 0</td></tr><tr><td> </td><td></td><td></td></tr><tr><td>  Return true if odd  </td><td>isOdd</td><td>  x % 2 != 0</td></tr><tr><td>or (x &amp; 1) != 0  </td><td></td><td></td></tr></table> <p>---</p> <p>## Recommendation</p> <p><b>Always make sure your function name matches the logic inside!</b> This avoids confusion and errors for you and others who maintain your code.</p> | Desired Functionality   Function Name   Logic |  |  | ----- ----- ----- |  |  | ----- |  |  | Return true if even | isEven | x % 2 == 0 |  |  |  | Return true if odd | isOdd | x % 2 != 0 | or (x & 1) != 0 |  |  |
|---|--|--|---|--|--|-------------------|--|--|-------|--|--|---------------------|--------|------------|--|--|--|--------------------|-------|------------|-----------------|--|--|
| Desired Functionality   Function Name   Logic |  |  |   |  |  |                   |  |  |       |  |  |                     |        |            |  |  |  |                    |       |            |                 |  |  |
| ----- ----- -----                             |  |  |   |  |  |                   |  |  |       |  |  |                     |        |            |  |  |  |                    |       |            |                 |  |  |
| -----   |  |  |   |  |  |                   |  |  |       |  |  |                     |        |            |  |  |  |                    |       |            |                 |  |  |
| Return true if even                           | isEven   | x % 2 == 0   |   |  |  |                   |  |  |       |  |  |                     |        |            |  |  |  |                    |       |            |                 |  |  |
|   |  |  |   |  |  |                   |  |  |       |  |  |                     |        |            |  |  |  |                    |       |            |                 |  |  |
| Return true if odd                            | isOdd  | x % 2 != 0   |   |  |  |                   |  |  |       |  |  |                     |        |            |  |  |  |                    |       |            |                 |  |  |
| or (x & 1) != 0                               |  |  |   |  |  |                   |  |  |       |  |  |                     |        |            |  |  |  |                    |       |            |                 |  |  |

All 4 responses correctly identified the mismatch between function name and logic, without much substantive difference between responses. GPT 4.1 was overall easier to read, however, since its responses were less wordy. This could be because GPT 4.1 is a more recent model.

## Task 18



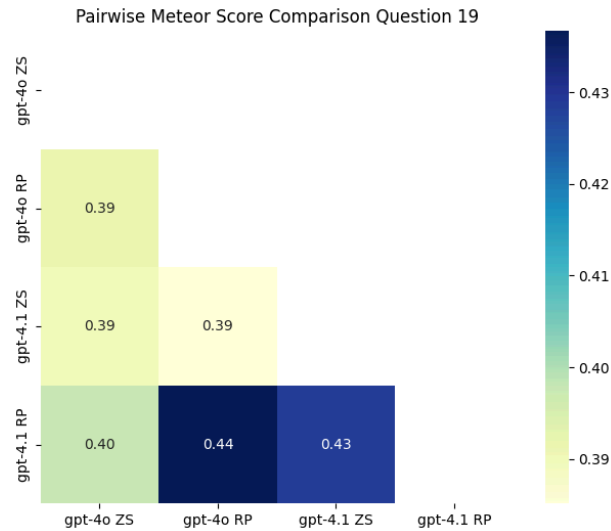
|                  |   |   |      |
|------------------|---|---|------|
| Goal             | Identify and fix the bug  |   |      |
| Model            | OpenAI GPT-4o   | OpenAI GPT-4.1  |      |
| Temperature      | 1   | Token Limit   | 1024 |
| Role             | You are an expert software engineer with many years of experience.  |   |      |
| Prompt           | I am running into a bug with the following JavaScript function: <code>function isEven ( n ) { return n % 2; // Returns 1 or 0 , not true / false }`</code> . Can you identify what the problem is and provide a solution?   |   |      |
| Zero-Shot Output | <p>Certainly! The problem with your <code>isEven</code> function lies in the return value. It uses the <code>n % 2</code> operation to calculate whether the number <code>n</code> is even or odd. However, this operation returns <code>0</code> if the number is even, and <code>1</code> if the number is odd. These values (<code>0</code> and <code>1</code>) are numeric, not a boolean (<code>true</code> or <code>false</code>), which is likely not what you intended.</p> <p>To fix this issue, you need to return a boolean (<code>true</code> for even numbers and <code>false</code> for odd numbers). Here's how you can adjust your function:</p> <p>### Solution:</p> | <p>Yes! Let's examine your function:</p> <pre>```js function isEven(n) {   return n % 2; // Returns 1 or 0, not true / false } ```</pre> <p><b>Problem:</b><br/>This function returns <code>0</code> for even numbers and <code>1</code> for odd numbers because of how the modulo operator (<code>%</code>) works. In JavaScript, <code>0</code> is falsy and <code>1</code> is truthy, but strictly the function does <i>not</i> return a boolean <code>true</code> or <code>false</code> as its name suggests.</p> <p>For example:</p> |      |

|                            |  |  |
|----------------------------|--|--|
|                            | <pre> '''javascript function isEven(n) {   return n % 2 === 0; // Returns true for even numbers and false for odd numbers } '''  ### Explanation: - <code>n % 2 === 0</code> checks if the remainder when <code>n</code> is divided by 2 is equal to 0. If it is, the number is even, and the expression evaluates to <code>true</code>. - If the remainder is not 0, the number is odd, and the expression evaluates to <code>false</code>.  ### Example Usage: '''javascript console.log(isEven(4)); // true console.log(isEven(7)); // false '''  This version of the function ensures you're working with proper boolean values, which is likely the expected behavior. </pre> | <p>- <code>isEven(4)</code> returns <code>0</code> (falsy, not the boolean <code>true</code>)</p> <p>- <code>isEven(5)</code> returns <code>1</code> (truthy, not the boolean <code>false</code>)</p> <p>While this works in <code>if</code> statements due to JavaScript's type coercion, it's not semantically correct and can cause confusion or bugs, especially when strict boolean checks are required.</p> <p>---</p> <p><b>Solution:</b><br/>Modify the function to return a boolean value. You can do this by comparing the result of <code>n % 2</code> to <code>0</code>:</p> <pre> '''js function isEven(n) {   return n % 2 === 0; } ''' </pre> <p>Now:</p> <p>- <code>isEven(4)</code> returns <code>true</code></p> <p>- <code>isEven(5)</code> returns <code>false</code></p> <p>---</p> <p><b>Summary:</b><br/>Replace your function with:</p> <pre> '''js function isEven(n) {   return n % 2 === 0; } ''' </pre> <p>This always returns a boolean, which is clearer and less error-prone!</p> |
| <b>Role-Playing Output</b> | <p>Certainly! The problem with your <code>isEven</code> function is that it calculates the remainder of <code>n</code> divided by <code>2</code> (using <code>%</code>) and directly returns it. This will result in the function returning <code>1</code> for odd numbers and <code>0</code> for even numbers, rather than returning <code>true</code> or <code>false</code>, which is typically expected for Boolean predicates like <code>isEven</code>.</p> <p>### Fix:</p> <p>The return value should be explicitly converted to a Boolean value. In JavaScript, the expression <code>n % 2 ===</code></p>  | <p><b>Problem:</b><br/>Your <code>isEven</code> function currently returns <code>n % 2</code>, which will give <code>0</code> for even numbers and <code>1</code> for odd numbers. In JavaScript, the values <code>0</code> and <code>1</code> are not strictly equal to boolean <code>true</code> or <code>false</code>. Therefore:</p> <p>- <code>isEven(2)</code> returns <code>0</code>, not <code>true</code>.</p> <p>- <code>isEven(3)</code> returns <code>1</code>, not <code>false</code>.</p> <p>This can cause bugs if your code expects <code>isEven</code> to return a boolean.</p> <p>---</p>  |

|  |   |  |
|--|---|--|
|  | <p>0` evaluates to `true` for even numbers and `false` for odd numbers, as `n % 2` equals `0` for even numbers.</p> <p>Here's the corrected function:</p> <pre>```javascript function isEven(n) {   return n % 2 === 0; // Returns true for even, false for odd } ```</pre> <p>### Explanation:</p> <ul style="list-style-type: none"> <li>- The `%` operator gives the remainder of `n` divided by `2`.</li> <li>- When `n` is even, `n % 2` equals `0`.</li> <li>- When `n` is odd, `n % 2` equals `1`.</li> <li>- By comparing `n % 2 === 0`, the function explicitly checks for evenness and returns `true` or `false`.</li> </ul> <p>### Example Usage:</p> <pre>```javascript console.log(isEven(4)); // true console.log(isEven(7)); // false ```</pre> <p>This will now work as expected!</p> | <p><b>Solution:</b><br/>Return a boolean value by comparing the result of <code>n % 2</code> to 0:</p> <pre>```javascript function isEven(n) {   return n % 2 === 0; } ```</pre> <p>---</p> <p><b>Usage Example:</b></p> <pre>```javascript console.log(isEven(2)); // true console.log(isEven(3)); // false ```</pre> <p><b>Summary:</b><br/>Return a boolean with <code>n % 2 === 0</code> instead of just <code>n % 2</code>.</p> |
|--|---|--|

GPT 4.1 did a much better job being concise with its explanations, and giving the models a role made both of them more concise as well. All 4 responses provided essentially the same recommendation for bug fixing, but I found 4.1's RP response to be the easiest to follow. This could be because this model is a more recent model, and it being given a role incentivizes a shorter answer with less unnecessary fluff.

## Task 19



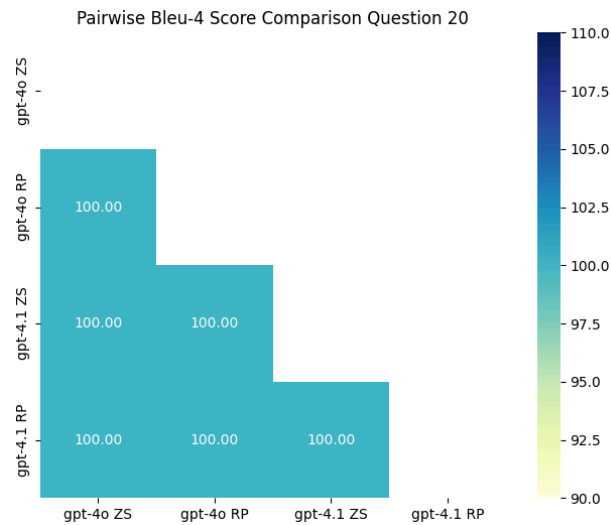
|                  |   |   |      |
|------------------|---|---|------|
| Goal             | Decompose the high-level comment/summary into logical steps.  |   |      |
| Model            | OpenAI GPT-4o   | OpenAI GPT-4.1  |      |
| Temperature      | 1   | Token Limit   | 1024 |
| Role             | You are an expert software engineer with many years of experience.  |   |      |
| Prompt           | Decompose the high-level comment/summary into logical steps in this C++ method: ```// Function that validates an input , calculates square , and returns result<br>int process ( int x ) {<br>if ( x < 0) return -1;<br>return x * x ;<br>}```  |   |      |
| Zero-Shot Output | <p>Certainly! Let's break down the high-level functionality of the provided C++ method <code>process()</code> into logical steps:</p> <pre>```cpp // Function that validates an input, calculates square, and returns result int process(int x) {     if (x &lt; 0) return -1;     return x * x; } ```</pre> <p>### Decomposed Logical Steps:</p> <ol style="list-style-type: none"> <li><b>Input Validation:</b> <ul style="list-style-type: none"> <li>- The function checks if the input value <code>x</code> is less than 0 (<code>x &lt; 0</code>).</li> <li>- If the condition is true, the function returns <code>-1</code> immediately, which likely</li> </ul> </li> </ol> | <p>Certainly! Let's break down the high-level summary (i.e., "validates an input , calculates square , and returns result") into logical steps that map to the actual C++ code:</p> <p>Original code:</p> <pre>```cpp // Function that validates an input , calculates square , and returns result int process(int x) {     if (x &lt; 0) return -1;     return x * x; } ```</pre> <p><b>Decomposition into Logical Steps:</b></p> <ol style="list-style-type: none"> <li><b>Validate the input:</b> <ul style="list-style-type: none"> <li>- Check if the input <code>x</code> is negative (<code>x &lt; 0</code>).</li> <li>- If so, terminate the function early and return</li> </ul> </li> </ol> |      |



|   | <p>serves as an error signal since negative inputs aren't valid for this operation.</p> <p><b>2. Calculation:</b></p> <ul style="list-style-type: none"><li>- If the input value <code>x</code> is valid (i.e., not less than 0), the function proceeds to calculate the square of the number (<code>x * x</code>).</li></ul> <p><b>3. Result Return:</b></p> <ul style="list-style-type: none"><li>- The calculated value (<code>x * x</code>) is returned as the output of the function.</li></ul> <p>### Full Flow of Execution:</p> <ul style="list-style-type: none"><li>- <b>Input a number <code>x</code>.</b></li><li>- <b>Check if <code>x</code> is less than 0.</b><ul style="list-style-type: none"><li>- If <code>true</code>, return <code>-1</code>.</li><li>- If <code>false</code>, proceed to the next step.</li></ul></li><li>- <b>Calculate the square of <code>x</code> (<code>x * x</code>).</b></li><li>- <b>Return the result of the calculation.</b></li></ul> <p>### Example Usages:</p> <ul style="list-style-type: none"><li>- <code>process(5)</code> Valid input; returns <code>'25'</code>.</li><li>- <code>process(-3)</code> Invalid input; returns <code>'-1'</code>.</li></ul> <p>This ensures that the function only operates on non-negative inputs and handles edge cases properly.</p> | <p>the error value <code>-1</code>.</p> <p><b>2. Calculate the square:</b></p> <ul style="list-style-type: none"><li>- If the input passes the validation (i.e., <code>x &gt;= 0</code>), compute the square of the input (<code>x * x</code>).</li></ul> <p><b>3. Return the result:</b></p> <ul style="list-style-type: none"><li>- Return the computed square as the result of the function.</li></ul> <p>---</p> <p><b>Step-by-step breakdown mapping code to logical steps:</b></p> <p><b>1. Input validation:</b></p> <pre>```cpp if (x &lt; 0) return -1; ```</pre> <ul style="list-style-type: none"><li>- "If the input is negative, return -1 to indicate an invalid input."</li></ul> <p><b>2. Calculation:</b></p> <pre>```cpp return x * x; ```</pre> <ul style="list-style-type: none"><li>- "Otherwise, compute and return the square of <code>x</code>."</li></ul> <p>---</p> <p><b>Summary Table:</b></p> <table><tr><th>Step</th><th>Purpose</th></tr><tr><td></td><td>Code</td></tr><tr><td colspan="2">----- -----</td></tr><tr><td colspan="2">----- -----</td></tr><tr><td>1. Validate the input (invalid) numbers</td><td>Handle negative if (<code>x &lt; 0</code>) return <code>-1</code>;</td></tr><tr><td>2. Calculate the square of valid input</td><td>Compute result for <code>return x * x</code>;</td></tr></table> <p>Let me know if you want this annotated in the code itself!</p> | Step | Purpose |  | Code | ----- ----- |  | ----- ----- |  | 1. Validate the input (invalid) numbers | Handle negative if ( <code>x &lt; 0</code> ) return <code>-1</code> ; | 2. Calculate the square of valid input | Compute result for <code>return x * x</code> ; |
|---|---|---|------|---------|--|------|-------------|--|-------------|--|---|---|--|--|
| Step                                    | Purpose   |   |      |         |  |      |             |  |             |  |   |   |  |  |
|   | Code  |   |      |         |  |      |             |  |             |  |   |   |  |  |
| ----- -----                             |   |   |      |         |  |      |             |  |             |  |   |   |  |  |
| ----- -----                             |   |   |      |         |  |      |             |  |             |  |   |   |  |  |
| 1. Validate the input (invalid) numbers | Handle negative if ( <code>x &lt; 0</code> ) return <code>-1</code> ;   |   |      |         |  |      |             |  |             |  |   |   |  |  |
| 2. Calculate the square of valid input  | Compute result for <code>return x * x</code> ;  |   |      |         |  |      |             |  |             |  |   |   |  |  |
| <b>Role-Playing Output</b>              | <p>Certainly! The high-level summary described in the comment can be broken down into the following logical steps, which align with the C++ method implementation:</p> <p><b>1. Accept an integer input:</b></p>  | <p>Certainly! Here's the decomposition of the high-level summary comment into logical steps for the given C++ method:</p> <p><b>High-level summary:</b></p> <p>"Function that validates an input, calculates square, and returns result"</p>  |      |         |  |      |             |  |             |  |   |   |  |  |

|  |  |  |
|--|--|--|
|  | <p>- The function accepts an integer input parameter <code>x</code>.</p> <p><b>2. Validate the input:</b></p> <ul style="list-style-type: none"> <li>- Check if the input <code>x</code> is less than 0 (negative).</li> </ul> <p><b>3. Return a failure code for invalid input:</b></p> <ul style="list-style-type: none"> <li>- If the input is negative (<code>x &lt; 0</code>), return <code>-1</code> to indicate an error or invalid input.</li> </ul> <p><b>4. Calculate the square of the input:</b></p> <ul style="list-style-type: none"> <li>- If the input is valid (non-negative), calculate the square of <code>x</code> by multiplying <code>x</code> by itself (<code>x * x</code>).</li> </ul> <p><b>5. Return the result:</b></p> <ul style="list-style-type: none"> <li>- Return the computed square of <code>x</code> if the input was valid.</li> </ul> <p>To summarize:</p> <ol style="list-style-type: none"> <li>1. Input validation is performed.</li> <li>2. If invalid, return <code>-1</code>.</li> <li>3. If valid, compute and return <code>x * x</code>.</li> </ol> | <p><b>Logical steps:</b></p> <p><b>1. Input Validation:</b></p> <ul style="list-style-type: none"> <li>- Check if the input <code>x</code> is less than 0 (i.e., negative).</li> <li>- If <code>x</code> is negative, return <code>-1</code> as an error code.</li> </ul> <p><b>2. Calculation:</b></p> <ul style="list-style-type: none"> <li>- If the input is valid (not negative), compute the square of <code>x</code>.</li> </ul> <p><b>3. Return Result:</b></p> <ul style="list-style-type: none"> <li>- Return the computed square result (or the error code if validation failed).</li> </ul> <p>---</p> <p><b>Mapped to code:</b></p> <pre> '''cpp int process(int x) {     // Step 1: Input Validation     if (x &lt; 0)         return -1; // Invalid input      // Step 2: Calculation     // Step 3: Return Result     return x * x; } ''' </pre> <p><b>Each logical step explained:</b></p> <ul style="list-style-type: none"> <li>- <b>Step 1:</b> Check for invalid input and handle the error.</li> <li>- <b>Step 2:</b> Perform the required calculation for valid input.</li> <li>- <b>Step 3:</b> Return the outcome (either error or result of calculation).</li> </ul> |
|--|--|--|

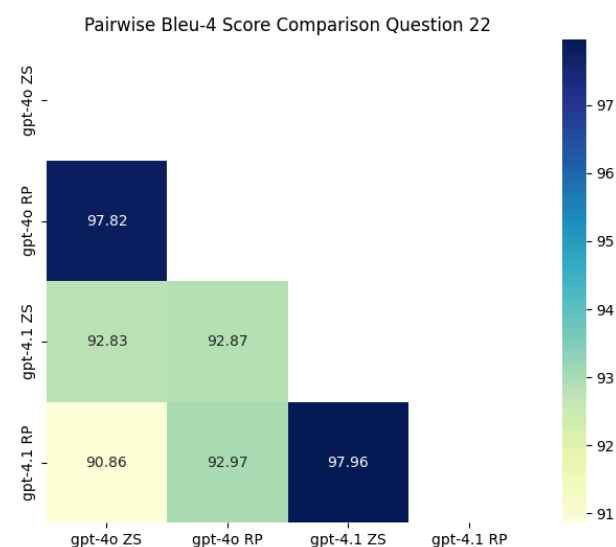
All 4 responses did a similar job breaking down and explaining the method, but 4.1 was overall slightly more detailed in its breakdown. This could be due to its more modern age. The roleplay responses did a better job explaining the code more concisely, which makes it arguably better as its result will be easier to read and understand. This may be driven by a desire to emulate an expert's ability to explain concepts in as few words as possible.



|                     |  |  |      |
|---------------------|--|--|------|
| Goal                | Complete the function based on intent.   |  |      |
| Model               | OpenAI GPT-4o  | OpenAI GPT-4.1   |      |
| Temperature         | 1  | Token Limit  | 1024 |
| Role                | You are an expert software engineer with many years of experience.   |  |      |
| Prompt              | Complete the python function based on intent, Given me only the code, no commentary:<br><pre> def calculate_average ( scores ) :     total = 0     # TODO : Complete to return average     pass </pre> |  |      |
| Zero-Shot Output    | <pre> python def calculate_average(scores):     total = sum(scores)     return total / len(scores) if scores else 0 </pre>   | <pre> python def calculate_average(scores):     total = sum(scores)     return total / len(scores) if scores else 0 </pre> |      |
| Role-Playing Output | <pre> python def calculate_average(scores):     total = sum(scores)     return total / len(scores) if scores else 0 </pre>   | <pre> python def calculate_average(scores):     total = sum(scores)     return total / len(scores) if scores else 0 </pre> |      |

Every one of these responses is the same, indicating that the models found a common solution to the problem. This could be due to the simplicity of the problem.

## Task 21



|                         |   |   |      |
|-------------------------|---|---|------|
| <b>Goal</b>             | Analyze the following utility script. Identify any potential logic issues or design flaws. Then, refactor it for better readability, correctness, and safety.   |   |      |
| <b>Model</b>            | OpenAI GPT-4o   | OpenAI GPT-4.1  |      |
| <b>Temperature</b>      | 1   | <b>Token Limit</b>  | 1024 |
| <b>Role</b>             | You are an expert software engineer with many years of experience.  |   |      |
| <b>Prompt</b>           | Analyze the following python utility script. Identify any potential logic issues or design flaws. Then, refactor it for better readability, correctness, and safety: ````# utils . py - Script to parse and summarize numeric CSV files           import csv           def read_csv ( filepath ) :               with open ( filepath , 'r ' ) as f :                   return [ row for row in csv . reader ( f ) ]           def summarize_column ( data , index ) :               values = [ float ( row [ index ] ) for row in data [ 1 : ] ] # skip header               total = sum ( values )               avg = total / len ( values )               return total , avg           def main () :               filepath = ' data . csv '               data = read_csv ( filepath )               total , avg = summarize_column ( data , 1 )               print ( ' Total : ' , total )               print ( ' Average : ' , avg )           if __name__ == ' __main__ ' :               main ()```` |   |      |
| <b>Zero-Shot Output</b> | The provided Python script has several logic issues, design flaws, and opportunities for improvement. Let's analyze the script step by step and refactor it for better readability, correctness, and safety.  | Let's break down the issues first, then provide a refactored version.<br><br>---<br><br><b>## 1. Issues in the utility script</b> |      |

|  |   |   |
|--|---|---|
|  | <pre> ---  ### Identified Issues  1. <b>File Path Hardcoding:</b> - The <code>filepath</code> variable in <code>main()</code> is hardcoded to <code>'data.csv'</code>, which includes unnecessary spaces, making it error-prone. The path should either be passed as an argument or specified more clearly.  2. <b>Lack of Input Validation:</b> - No handling of non-numeric values in the CSV file. If a non-numeric value is encountered in the column, the code will raise a <code>ValueError</code>. - The script doesn't check if the header exists or whether the specified <code>index</code> in <code>summarize_column</code> is valid.  3. <b>Magic Number for Column Index:</b> - The column index is <code>1</code>, hardcoded directly in the function calls without explanation.  4. <b>Error Handling:</b> - No error handling for file operations (e.g., missing file, permission issues) or for operations on the CSV data (e.g., empty files, invalid formats).  5. <b>Poor Readability:</b> - Inconsistent spacing (e.g., <code>'data.csv '</code>) and lack of comments explaining key portions of the code. - Function and variable names could be more descriptive (<code>read_csv</code>, <code>summarize_column</code>, and <code>data</code> are too generic).  6. <b>Duplicated Code:</b> - The slicing <code>[1:]</code> for skipping the header in <code>summarize_column</code> assumes a header exists but does not confirm it. - Repeatedly constructs the <code>values</code> list without encapsulation.  7. <b>Modularity and Extensibility:</b> - This script only works with one column (index 1) and doesn't provide ways to select other columns dynamically. - It's not flexible for use with different </pre> | <pre> ### A. Syntax/Formatting Issues - Inconsistent and extraneous spaces inside parentheses and around arguments. - Filenames and <code>__name__</code> variable are written with extra spaces: <code>'data.csv '</code>, <code>'__main__ '</code>. - Function names and code lack conventional PEP-8 spacing/indentation. - No docstrings for functions. - The script is not robust to errors (e.g., file not found, missing column, non-numeric data).  ### B. Logic Issues - The script assumes column 1 has all numeric values, and header is only in row 0. - Does not handle empty rows, missing columns, or conversion errors (<code>ValueError</code>). - No configurability for the column to summarize, input file, or delimiter. - Hard-coding <code>data.csv</code> and column index.  ### C. Safety/Design Flaws - Does not close the file if an exception occurs before the end of <code>with</code>. - No command-line argument parsing, so can't choose CSV or column on execution. - Does not use exceptions or error handling for bad input or missing files. - Prints total and average with extra spaces. - Code logic for skipping header is implicit; better to make it explicit. - Only works with comma-delimited files, no way to configure. - No type checks for empty data or short files.  ---  ## 2. Refactored Version  Here's a more robust, clearer version with docstrings, error handling, style fixes, configurability, and sanity checking:  ```python import csv import sys from typing import List, Tuple  def read_csv(filepath: str, delimiter: str = ',') -&gt; List[List[str]]:     """Read a CSV file and return its contents as a list of lists."""     try:         with open(filepath, 'r', newline='') as f:             reader = csv.reader(f, </pre> |
|--|---|---|

|  |   |   |
|--|---|---|
|  | <p>datasets without manually modifying the code.</p> <p>---</p> <p>### Refactored Code</p> <p>Below is the improved version of the script with fixes for the above issues:</p> <pre> '''python # utils.py - Script to parse and summarize numeric CSV files import csv import sys  def read_csv(filepath):     """     Reads a CSV file and returns     the data as a list of rows.      Args:     filepath (str): Path to the CSV     file.      Returns:     list: List of rows (each row is a     list of strings).     """     try:         with open(filepath, 'r',         newline="", encoding='utf-8') as f:             reader = csv.reader(f)             return list(reader)     except FileNotFoundError:         raise     FileNotFoundError(f'File not found:     {filepath}')     except PermissionError:         raise PermissionError(f'No     permission to read the file: {filepath}')     except Exception as e:         raise RuntimeError(f'Error     reading file: {filepath}. Details: {e}')  def summarize_column(data, index):     """     Summarizes a single column     of numeric data.      Args:     data (list): List of rows (each     row is a list of strings).     index (int): The index of the     column to summarize.</pre> | <pre> delimiter=delimiter)     return [row for row in reader if row] #     skip empty rows     except FileNotFoundError:         print(f'Error: File '{filepath}' not         found.")         sys.exit(1)  def summarize_column(data: List[List[str]], index: int) -&gt; Tuple[float, float]:     """     Summarize (sum and mean) the numeric     values in a specified column index.     The first row is assumed to be the header     and is skipped.     """     if not data or len(data) &lt; 2:         raise ValueError("CSV file is empty or         has no data rows.")     values = []     for row_num, row in enumerate(data[1:],     2): # data[1:] skips header, start row numbering at     2         try:             values.append(float(row[index]))         except IndexError:             print(f'Warning: Row {row_num} does             not have column {index}. Skipped.")         except ValueError:             print(f'Warning: Non-numeric value in             row {row_num} column {index}: '{row[index]}'.             Skipped.")         if not values:             raise ValueError("No valid numeric data             found in selected column.")         total = sum(values)         avg = total / len(values)         return total, avg  def main():     import argparse     parser =     argparse.ArgumentParser(description="Parse and     summarize a column from a numeric CSV file.")     parser.add_argument('filepath',     help="Path to the CSV file.")     parser.add_argument('-c', '--column',     type=int, default=1, help="Column index to     summarize (0-based, default: 1).")     parser.add_argument('--delimiter',     type=str, default=',', help="CSV delimiter     (default: ',').")     args = parser.parse_args()      data = read_csv(args.filepath,</pre> |
|--|---|---|

|  |  |  |
|--|--|--|
|  | <p>Returns:<br/>tuple: Total and average of the numeric column.</p> <pre> """     if not data or len(data) &lt;= 1:         raise ValueError("The provided data is empty or only contains a header.")      try:         # Extract values from the specified column, skipping the header         values = []         for row in data[1:]: # Skip the header (row 0)             if index &gt;= len(row):                 raise IndexError(f"Column index {index} out of range for row: {row}")          values.append(float(row[index]))          if not values:             raise ValueError("The specified column is empty.")          total = sum(values)         avg = total / len(values)         return total, avg     except ValueError:         raise ValueError(f"Non-numeric value encountered in column {index}.")     except IndexError as e:         raise IndexError(f"Index error: {e}")  def main(filepath, column_index):     """         Main function to read a CSV file and summarize a specific column.          Args:         filepath (str): Path to the CSV file.         column_index (int): The column index to summarize.     """     try:         data = read_csv(filepath)         total, avg = summarize_column(data, column_index)         print(f"Total: {total}") </pre> | <pre> delimiter=args.delimiter)         header = data[0] if data else []         try:             total, avg = summarize_column(data, args.column)             header_name = header[args.column] if header and len(header) &gt; args.column else f"Column {args.column}"             print(f"Summary for column '{header_name}':")             print(f"  Total: {total}")             print(f"  Average: {avg}")         except Exception as e:             print(f"Error: {e}")             sys.exit(1)  if __name__ == '__main__':     main() ''' ---</pre> <p><b>## Key Improvements</b></p> <ul style="list-style-type: none"> <li>- <b>PEP-8 styling:</b> Proper spacing, naming, and formatting.</li> <li>- <b>Error handling:</b> Handles file-not-found, missing columns, and non-numeric data.</li> <li>- <b>Command-line interface:</b> More flexible and user friendly.</li> <li>- <b>Docstrings and type hints:</b> Clear documentation and type checking.</li> <li>- <b>Robustness:</b> Won't crash on bad input, and warns user on bad rows.</li> <li>- <b>Usability:</b> User can specify file, column, and delimiter.</li> </ul> |
|--|--|--|

|                            |   |  |
|----------------------------|---|--|
|                            | <pre> print(f"Average: {avg}") except Exception as e: print(f"Error: {e}", file=sys.stderr)  if __name__ == '__main__': import argparse  # Use argparse for command-line interface parser = argparse.ArgumentParser(description=" Summarize a numeric column in a CSV file.") parser.add_argument('filepath', help="Path to the CSV file.") </pre>  |  |
| <b>Role-Playing Output</b> | <p>Here's an in-depth analysis of the provided Python script, highlighting potential issues and design flaws, along with a refactored version of the script for improved readability, correctness, and robustness.</p> <p>---</p> <p><b>### Analysis:</b></p> <p><b>1. File Handling:</b></p> <ul style="list-style-type: none"> <li>- File paths like <code>'data . csv'</code> might lead to unexpected errors due to unintended spaces. This could cause the script to fail at runtime if invoked with the provided code.</li> </ul> <p><b>2. Error Handling:</b></p> <ul style="list-style-type: none"> <li>- The script has no error handling. If the file is missing, unreadable, the CSV is improperly formatted, or column indices are out of range, it will throw unhandled exceptions.</li> <li>- Conversion of column values to <code>'float'</code> may fail if non-numeric data is encountered.</li> </ul> <p><b>3. Hardcoded Filepath:</b></p> <ul style="list-style-type: none"> <li>- The hardcoded filepath <code>'data . csv'</code> makes this utility less reusable and inflexible for general-purpose processing.</li> </ul> <p><b>4. Readable Design:</b></p> <ul style="list-style-type: none"> <li>- Excessive spacing in the code (e.g., <code>'__main__'</code> and <code>'data . csv'</code>) hampers</li> </ul> | <p>Let's systematically analyze, identify issues, and refactor the script:</p> <p>---</p> <p><b>## 1. Analysis &amp; Issues</b></p> <p><b>### A. Formatting &amp; Style (PEP8)</b></p> <ul style="list-style-type: none"> <li>- Inconsistent and excessive whitespace (e.g., <code>sum ( values )</code> instead of <code>sum(values)</code>).</li> <li>- Filepath with weird spaces (<code>'data . csv '</code>) likely unintended.</li> <li>- Indentation is inconsistent.</li> <li>- Spaces inside function calls and brackets.</li> <li>- Missing shebang line and module docstring for a utility script.</li> </ul> <p><b>### B. Logic and Design Flaws</b></p> <ul style="list-style-type: none"> <li>- <b>File Opening:</b> <ul style="list-style-type: none"> <li>- Hardcoded file name.</li> <li>- No error handling (missing file, I/O errors).</li> </ul> </li> <li>- <b>CSV Reading:</b> <ul style="list-style-type: none"> <li>- Doesn't handle quoted values or dialect selection.</li> <li>- Assumes header exists but doesn't validate.</li> </ul> </li> <li>- <b>Data Parsing:</b> <ul style="list-style-type: none"> <li>- Assumes all data in target column is float-convertible.</li> <li>- Skips header via slicing (<code>data[1:]</code>), which is okay, but no validation.</li> <li>- No handling for empty lines, missing or malformed entries.</li> </ul> </li> <li>- <b>Index Boundaries:</b> <ul style="list-style-type: none"> <li>- No check for enough columns in every row.</li> </ul> </li> <li>- <b>Summarize Column:</b> <ul style="list-style-type: none"> <li>- Division by zero if data empty (should handle).</li> </ul> </li> </ul> |



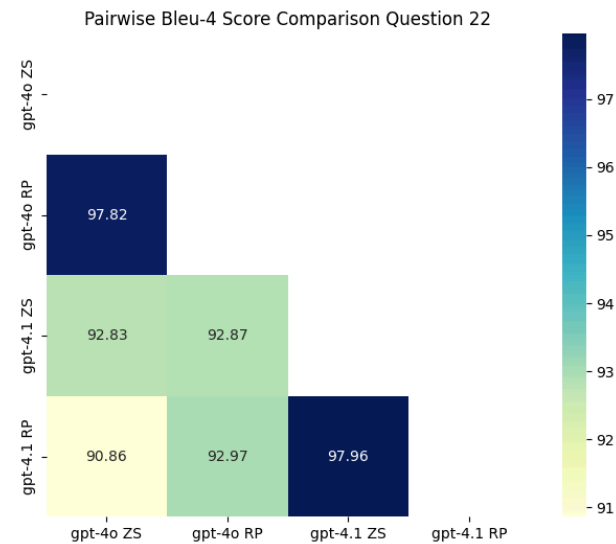
|  |  |   |
|--|--|---|
|  | <p>readability.</p> <ul style="list-style-type: none"> <li>- Meaningful function and variable names would improve clarity.</li> </ul> <p><b>5. Header Handling:</b></p> <ul style="list-style-type: none"> <li>- The function skips the header row when reading the CSV, but this behavior is implicit and not configurable. It is unclear whether the header is assumed.</li> </ul> <p><b>6. Resource Management:</b></p> <ul style="list-style-type: none"> <li>- The script does not explicitly close the file handle. Although using <code>with open()</code> handles it implicitly, it is worth noting.</li> </ul> <p><b>7. Scalability:</b></p> <ul style="list-style-type: none"> <li>- The <code>summarize_column</code> function computes only the total and average but lacks extensibility for other operations (e.g., min, max).</li> </ul> <p><b>8. Documentation:</b></p> <ul style="list-style-type: none"> <li>- The script lacks comments and docstrings, making it harder to understand or maintain.</li> </ul> <p>---</p> <p><b>### Refactored Version:</b></p> <p>Here's the refactored script addressing the above issues:</p> <pre> ``python import csv from typing import List, Tuple, Optional  def read_csv(filepath: str) -&gt; List[List[str]]:     """     Reads a CSV file and returns its content as a list of rows.     Each row is represented as a list of strings.      Args:         filepath (str): Path to the CSV file.      Returns:         List[List[str]]: Content of the CSV file.     """ </pre> | <p><b>- Print Formatting:</b></p> <ul style="list-style-type: none"> <li>- Awkward spacing (' <b>Total</b> : ').</li> </ul> <p><b>- Entrypoint:</b></p> <ul style="list-style-type: none"> <li>- <code>if __name__ == '__main__':</code> has spaces and may not execute.</li> </ul> <p>---</p> <p><b>## 2. Refactored Version</b></p> <pre> ``python """ utils.py - Utility to parse and summarize numeric CSV files """  import csv from typing import List, Tuple  def read_csv(filepath: str) -&gt; List[List[str]]:     """Reads a CSV file and returns a list of rows."""     try:         with open(filepath, newline="", encoding='utf-8') as f:             reader = csv.reader(f)             data = list(reader)             if not data:                 raise ValueError("CSV file is empty.")             return data     except FileNotFoundError:         raise FileNotFoundError(f"File not found: {filepath}")     except Exception as exc:         raise RuntimeError(f"Error reading CSV file: {exc}")  def summarize_column(data: List[List[str]], index: int) -&gt; Tuple[float, float]:     """     Summarizes the values in the specified column (by index).      Returns:         total (float): Sum of all values         avg (float): Average of all values     """     if len(data) &lt; 2:         raise ValueError("CSV must contain a header and at least one data row.")      values = []     for row_num, row in enumerate(data[1:], </pre> |
|--|--|---|

|  |   |  |
|--|---|--|
|  | <pre> """ try: with open(filepath, 'r') as f: reader = csv.reader(f) return [row for row in reader] except FileNotFoundError: raise FileNotFoundError(f"File not found: {filepath}") except Exception as e: raise RuntimeError(f"An error occurred while reading the file: {e}")  def summarize_column(data: List[List[str]], index: int) -&gt; Tuple[Optional[float], Optional[float]]: """  Summarizes a specified numeric column in the CSV data by computing the total and average.  Args: data (List[List[str]]): The content of the CSV file as a list of rows. index (int): The index of the numeric column to summarize.  Returns: Tuple[Optional[float], Optional[float]]: A tuple containing the total and average of the column. """  try: # Skip the header row and extract the column values values = [float(row[index]) for row in data[1:] if row[index].strip()] if not values: return None, None # Return None if no numeric values are found total = sum(values) avg = total / len(values) return total, avg except IndexError: raise ValueError(f"Index {index} is out of range for the given data.") except ValueError: raise ValueError(f"Non-numeric data encountered in column {index}.") except Exception as e: raise RuntimeError(f"An error occurred while summarizing the </pre> | <pre> start=2): # Skip header try: value_str = row[index] value = float(value_str) values.append(value) except IndexError: raise ValueError(f"Row {row_num} does not have column index {index}. Row: {row}") except ValueError: raise ValueError(f"Could not convert row {row_num} column {index} to float: '{row[index]}'")  if not values: raise ValueError("No data found in the selected column.")  total = sum(values) avg = total / len(values) return total, avg  def main(): filepath = 'data.csv' # Use sensible default or accept as an argument try: data = read_csv(filepath) total, avg = summarize_column(data, 1) # Column 1 (second column) print(f"Total: {total}") print(f"Average: {avg}") except Exception as e: print(f"Error: {e}")  if __name__ == '__main__': main() ''' ---</pre> <p><b>## 3. Summary of Improvements</b></p> <ul style="list-style-type: none"> <li>- <b>Strict adherence to PEP8</b> (formatting, function naming, variable spacing).</li> <li>- <b>Robust error handling</b>: messages for file issues and data errors.</li> <li>- <b>Explicit encoding</b> for CSV reading.</li> <li>- <b>Input validation</b>: warns if the file is empty or rows are malformed.</li> <li>- <b>Informative exceptions</b> for easier debugging.</li> <li>- <b>Fixed entrypoint</b>: <code>if __name__ == '__main__':</code> (no spaces).</li> <li>- <b>Cleaner print statements</b> and better messaging.</li> <li>- <b>Type annotations</b> for clarity.</li> </ul> |
|--|---|--|

|  |   |   |
|--|---|---|
|  | <pre> column: {e}")  def main():     """     Entry point for the script.     Reads a CSV file, summarizes     a specific column, and prints the     results.     """     filepath = 'data.csv' # Update     this to match the desired CSV file     try:         data = read_csv(filepath)         if not data or len(data) &lt; 2:             print("The CSV file is empty             or does not contain sufficient data.")         return          # Customize the column index         as needed (e.g., 1 for the second         column)         column_index = 1         total, avg =         summarize_column(data,         column_index)          if total is None or avg is None:             print(f"No numeric data found             in column {column_index}.")         else:             print(f"Total: {total}")             print(f"Average: {avg}")         except Exception as e:             print(f"Error: {e}")  if __name__ == "__main__":     main() ...  ---  ### Explanation of Improvements:  1. **Filepath Handling:</pre> | <pre> ---</pre> <p><b>### Optional Enhancements</b></p> <ul style="list-style-type: none"> <li>- Accept a filename and column index via <b>argparse</b> for more flexibility.</li> <li>- Add logging instead of print for production scripts.</li> <li>- Add tests.</li> </ul> <p>Let me know if you'd like to see any of those enhancements!</p> |
|--|---|---|

All four responses did a similar job in identifying flaws and providing refactored code, with the largest similarities being between model's responses and not strategies. This could be because of the more concise nature of 4.1's responses being readily apparent here, due to the length of the responses.

## Task 22



|                  |  |   |      |
|------------------|--|---|------|
| Goal             | The goal is to clean each line, remove punctuation, and count word frequencies correctly.  |   |      |
| Model            | OpenAI GPT-4o  | OpenAI GPT-4.1  |      |
| Temperature      | 1  | Token Limit   | 1024 |
| Role             | You are an expert software engineer with many years of experience.   |   |      |
| Prompt           | <p>How would you complete the following python file script with the goal of cleaning each line, removing punctuation, and counting word frequencies? ``# file_processor . py - Incomplete script for processing text files</p> <pre>import string def load_file ( filepath ) :     with open ( filepath , 'r ' ) as f :         return f . readlines () def clean_line ( line ) :     # TODO : Remove punctuation and make lowercase     pass def count_words ( lines ) :     word_counts = {}     for line in lines :         clean = clean_line ( line )         for word in clean . split () :             word_counts [ word ] = word_counts . get ( word , 0 ) + 1     return word_counts def main () :     filepath = ' input . txt '     lines = load_file ( filepath )     counts = count_words ( lines )     for word , count in sorted ( counts . items () ) :         print ( f' { word } : { count } ' ) if __name__ == ' __main__ ' :     main ()``</pre> |   |      |
| Zero-Shot Output | ``python<br>import string  | ``python<br># file_processor.py - Incomplete script for |      |

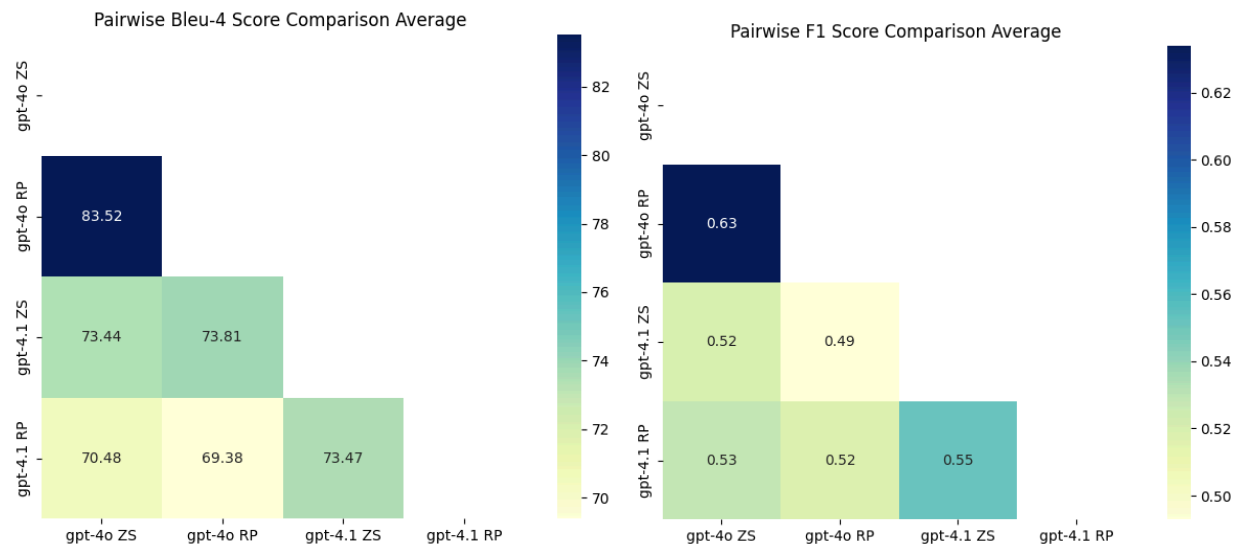
|                            |  |   |
|----------------------------|--|---|
|                            | <pre> def load_file(filepath):     with open(filepath, 'r') as f:         return f.readlines()  def clean_line(line):     return line.translate(str.maketrans(", ", string.punctuation)).lower()  def count_words(lines):     word_counts = {}     for line in lines:         clean = clean_line(line)         for word in clean.split():             word_counts[word] = word_counts.get(word, 0) + 1     return word_counts  def main():     filepath = 'input.txt'     lines = load_file(filepath)     counts = count_words(lines)     for word, count in sorted(counts.items()):         print(f'{word}: {count}')  if __name__ == '__main__':     main() ''' </pre> | <pre> processing text files import string  def load_file(filepath):     with open(filepath, 'r') as f:         return f.readlines()  def clean_line(line):     return line.translate(str.maketrans(", ", string.punctuation)).lower()  def count_words(lines):     word_counts = {}     for line in lines:         clean = clean_line(line)         for word in clean.split():             word_counts[word] = word_counts.get(word, 0) + 1     return word_counts  def main():     filepath = 'input.txt'     lines = load_file(filepath)     counts = count_words(lines)     for word, count in sorted(counts.items()):         print(f'{word}: {count}')  if __name__ == '__main__':     main() ''' </pre> |
| <b>Role-Playing Output</b> | <pre> '''python import string  def load_file(filepath):     with open(filepath, 'r') as f:         return f.readlines()  def clean_line(line):     return line.translate(str.maketrans(", ", string.punctuation)).lower().strip()  def count_words(lines):     word_counts = {}     for line in lines:         clean = clean_line(line)         for word in clean.split():             word_counts[word] = word_counts.get(word, 0) + 1     return word_counts  def main():     filepath = 'input.txt' </pre>  | <pre> '''python # file_processor.py - Incomplete script for processing text files import string  def load_file(filepath):     with open(filepath, 'r') as f:         return f.readlines()  def clean_line(line):     return line.translate(str.maketrans(", ", string.punctuation)).lower().strip()  def count_words(lines):     word_counts = {}     for line in lines:         clean = clean_line(line)         for word in clean.split():             word_counts[word] = word_counts.get(word, 0) + 1     return word_counts  def main(): </pre>  |

|  |  |   |
|--|--|---|
|  | <pre>lines = load_file(filepath) counts = count_words(lines) for word, count in sorted(counts.items()):     print(f'{word}: {count}')  if __name__ == '__main__':     main()``</pre> | <pre>filepath = 'input.txt' lines = load_file(filepath) counts = count_words(lines) for word, count in sorted(counts.items()):     print(f'{word}: {count}')  if __name__ == '__main__':     main()``</pre> |
|--|--|---|

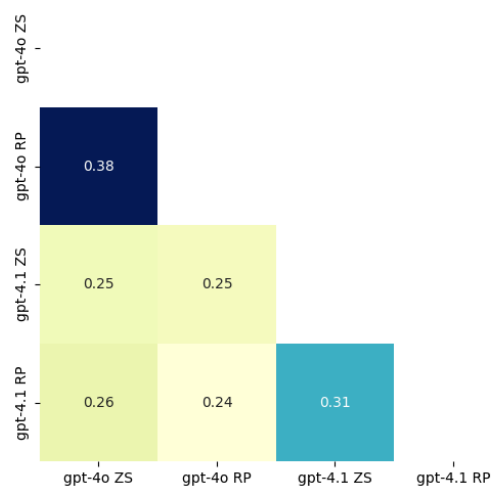
Every response here is nearly identical, evidencing that there is a common solution the models were able to find and report.

Conclusion:

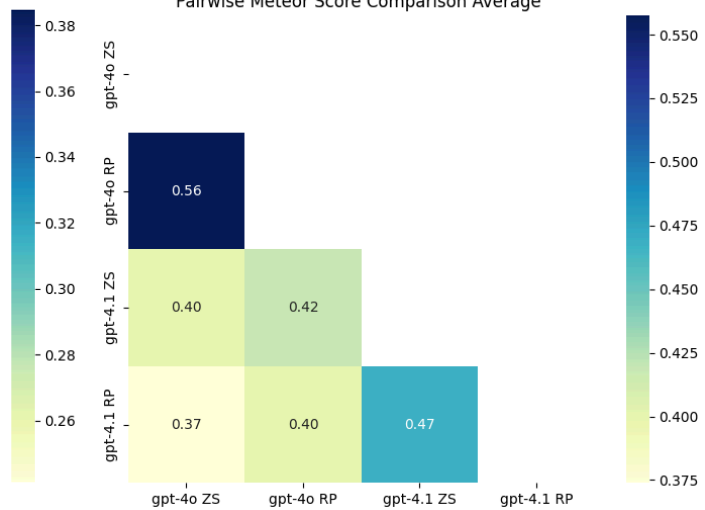
There was on average a greater difference between models than between strategies



Pairwise Rouge Score Comparison Average



Pairwise Meteor Score Comparison Average



Pairwise Bleu Score Comparison Average

