Table of Contents

[Question 1– Lambda Function 3](#_Toc181818238)

[Codestral-Mamba 3](#_Toc181818239)

[Granite-code:8b-instruct 3](#_Toc181818240)

[Granite-code:8b-dense 3](#_Toc181818241)

[Llama3.1 4](#_Toc181818242)

[Starcoder2: 4](#_Toc181818243)

[Question 2 – Quicksort Code 5](#_Toc181818244)

[Codestral-Mamba 5](#_Toc181818245)

[Granite-code:8b-instruct 5](#_Toc181818246)

[Granite-code:8b-dense 5](#_Toc181818247)

[Llama3.1 6](#_Toc181818248)

[Starcoder2: 6](#_Toc181818249)

[Question 3 – Constructor/Destructor in Java 7](#_Toc181818250)

[Codestral-Mamba 7](#_Toc181818251)

[Granite-code:8b-instruct 7](#_Toc181818252)

[Granite-code:8b-dense 7](#_Toc181818253)

[Llama3.1 8](#_Toc181818254)

[Starcoder2: 8](#_Toc181818255)

[Question 4 – Binary Search Code 9](#_Toc181818256)

[Codestral-Mamba 9](#_Toc181818257)

[Granite-code:8b-instruct 9](#_Toc181818258)

[Granite-code:8b-dense 9](#_Toc181818259)

[Llama3.1 10](#_Toc181818260)

[Starcoder2: 10](#_Toc181818261)

[Question 5 – Debugging the Code 11](#_Toc181818262)

[Codestral-Mamba 11](#_Toc181818263)

[Granite-code:8b-instruct 11](#_Toc181818264)

[Granite-code:8b-dense 11](#_Toc181818265)

[Llama3.1 12](#_Toc181818266)

[Starcoder2: 12](#_Toc181818267)

[Question 6 – Port the code to Java 14](#_Toc181818268)

[Codestral-Mamba 14](#_Toc181818269)

[Granite-code:8b-instruct 14](#_Toc181818270)

[Granite-code:8b-dense 14](#_Toc181818271)

[Llama3.1 15](#_Toc181818272)

[Starcoder2: 15](#_Toc181818273)

[Question 7 – Refactor Into Quarkus 16](#_Toc181818274)

[Codestral-Mamba 16](#_Toc181818275)

[Granite-code:8b-instruct 16](#_Toc181818276)

[Granite-code:8b-dense 16](#_Toc181818277)

[Llama3.1 16](#_Toc181818278)

[Starcoder2: 17](#_Toc181818279)

[Question 8 – Optimization Of Code 18](#_Toc181818280)

[Codestral-Mamba 18](#_Toc181818281)

[Granite-code:8b-instruct 18](#_Toc181818282)

[Granite-code:8b-dense 18](#_Toc181818283)

[Llama3.1 19](#_Toc181818284)

[Starcoder2: 19](#_Toc181818285)

[Question 9 – Documentation Of Code 20](#_Toc181818286)

[Question 10 – Unit Test of Code 21](#_Toc181818287)

[Codestral-Mamba 21](#_Toc181818288)

[Granite-code:8b-instruct 21](#_Toc181818289)

[Granite-code:8b-dense 22](#_Toc181818290)

[Llama3.1 22](#_Toc181818291)

[Starcoder2: 22](#_Toc181818292)

# How To Analyze This Document?

This document contains several test cases ran on five different models - Llama 3.2, granite-8b-instruct-128k, granite-8b-dense, Mistral Mamba and Starcoder2.

The test cases cover different scenarios and topics ranging from a simple theoretical question to documenting a code, from generating a code to debugging the code.

The text in red means that it was a problematic behavior by the model, while the text in green means that the behavior was good.

# Question 1– Lambda Function

Query - What is Lambda function in python?

### Codestral-Mamba

###### Execution 1-3: PASS

Format used in all three executions: Definition -> Usage -> Two examples. Results of all three execution is similar. Simple definition with well explained examples.

### Granite-code:8b-instruct -> Rank

###### Execution 1: PASS

Format used: Definition -> Usage Syntax -> Uses of Lambda function. Only gives generic usage example and not a real-world example.

###### Execution 2: PASS

Format used: Definition -> Usage Syntax -> Example with explanation. Improvement over the first result.

###### Execution 3: PASS

Only definition is given. No example. First two results were better.

### Granite-code:8b-dense

###### Execution 1: PASS

Similar result as granite-code:8b 3rd result. Also gives usage syntax.

###### Execution 2: PASS

Better result. Format: Definition -> Usage syntax -> Example. But no explanation of the example was provided.

###### Execution 3: PASS

Same result as execution 2.

### Llama3.1

###### Execution 1: PASS

Format used: Definition -> Usage Syntax -> Example -> Use Cases -> Best Practices -> Real-world Examples -> Explanation of examples.

###### Execution 2: PASS

Format used: Definition -> Usage Syntax -> Example 1-3 -> Advantages -> Best practices.

###### Execution 3: PASS

Format used: Definition -> Usage Syntax -> Examples use cases -> Advantages.

Everything is well written, but it might be an overkill for some users as it takes a lot of time to print this information for such a simple question.

Starcoder2:

FAIL => Only prints one line: A lambda function is a small anonymous function.

# Question 2 – Quicksort Code

Query - Write a code for quick-sort algorithm in Python.

### Codestral-Mamba

###### Execution 1: PASS

Prints a full undocumented python code with brief explanation of the code.

###### Execution 2: PASS

Prints an undocumented python code which is different from the first execution. It uses list comprehension instead of fully writing the loops. Also writes a brief explanation of the code.

###### Execution 3: PASS

Similar result as execution 2.

### Granite-code:8b-instruct

###### Execution 1: PASS

Prints a full undocumented python code with brief explanation of the code.

Execution 2: FAIL

Prints just the quick-sort function using without main function or how to use the function. Gives a good explanation of the algorithm.

Failure Reason: No main/calling function provided.

###### Execution 3: FAIL

Prints the quick-sort function and partition function. Does not print the main function again.

Failure Reason: No main/calling function provided.

We can conclude that execution 2 and 3 are not usable directly. This means that they must write the main function which was not required in the output of execution 1.

### Granite-code:8b-dense

###### Execution 1: FAIL

Prints a list comprehension version of quicksort function with a brief explanation. Does not print the function calling line or the main function.

Failure Reason: No main/calling function provided.

###### Execution 2: FAIL

Same result as execution 1. It also does not print the main function.

Failure Reason: No main/calling function provided.

###### Execution 3: FAIL

Again, same result but it also writes the time complexity of quick-sort algorithm.

Failure Reason: No main/calling function provided.

### Llama3.1

###### Execution 1: PASS

The output contains -> A well-documented quicksort function with usage code, the explanation of how quicksort works and advantages of quicksort.

###### Execution 2: PASS

Adds more detail to the previous result like time complexity and best practices to use the code.

Execution 3: PASS

Similar result as previous result but with slightly more explanation.

Starcoder2:  
  
FAIL => Explains quick-sort algorithm in a couple of lines. Does not write a single line of code.

# Question 3 – Constructor/Destructor in Java

Query - How do you declare constructors and destructors in Java?

### Codestral-Mamba

###### Execution 1: PASS

Gives simple definition of constructors and destructors, along with an example.

###### Execution 2: PASS

Similar result as execution 1. Also adds the syntax for finalize () method.

###### Execution 3: PASS

Similar result as execution 2 with slight modification in the code.

### Granite-code:8b-instruct

###### Execution 1: PASS

Gives definition of constructors and destructors with example. Also explain how we can write constructor with parameters. Explains finalize () method as well.

###### Execution 2: FAIL

Gives not so good result as the first execution. Ignores the parameterized constructor example and the code to create finalize () method. It also explains destructor in a paragraph form which might be unreadable for some users.

Failure Reason: It ignores some key details and examples that are necessary to explain constructors and destructors.

###### Execution 3: PASS

The result is like the result of first execution in the sense that it gives definition and code for both constructors and destructors.

### Granite-code:8b-dense

###### Execution 1: PASS

Gives simple definition of constructors and destructors, along with usage syntax of constructor and finalize () function.

###### Execution 2: PASS

Similar result as execution 1, just adds a little more explanation.

###### Execution 3: PASS

Slightly improved version of first two executions but nothing different in the format of the answer.

### Llama3.1

###### Execution 1: PASS

Gives a detailed explanation about constructors and destructors, their basic syntax, the rules to use them, and a lot of examples.

###### Execution 2: FAIL

Gives incomplete output -> it starts explaining the types of constructors and explains a few of them but stops printing abruptly.

Failure Reason: Incomplete code.

###### Execution 3: PASS

Explains constructors, destructors and types of constructors. It also explains a few unnecessary things like difference between constructors and methods, which could have been ignored.

Starcoder2:

FAIL => Writes the definition of constructors in a couple of lines. Does not write a single line of code

# Question 4 – Binary Search Code

Query – Write a code for Binary Search along with the documentation of the code.

### Codestral-Mamba

###### Execution 1: PASS

Prints a documented binary search code with example and brief explanation.

###### Execution 2: PASS

Prints a documented binary search code like execution 1 with additional details in the explanation.

###### Execution 3: PASS

Prints a binary search code with more detailed documentation and more detailed explanation of the code.

### Granite-code:8b-instruct

###### Execution 1: FAIL

Prints a well-documented binary search function along with an explanation. But it does not print the main function or the instructions of how to call the function.

Failure Reason: No main/calling function provided.

###### Execution 2: FAIL

Prints a well-documented binary search function with line-by-line explanation of the code. It also does not print the main function.

Failure Reason: No main/calling function provided.

###### Execution 3: FAIL

The result is just like the result of the first execution. It also gives a documented code with brief explanation but does not print the main function.

Failure Reason: No main/calling function provided.

### Granite-code:8b-dense

###### Execution 1: FAIL

Prints a documented binary search function along with an explanation. But it does not print the main function or the instructions of how to call the function.

Failure Reason: No main/calling function provided.

###### Execution 2: FAIL

Prints a documented binary search function with more explanation than the first execution. But it still does not print the main function or the instructions of how to call the function.

Failure Reason: No main/calling function provided.

###### Execution 3: FAIL

Same result as execution 2.

Failure Reason: No main/calling function provided.

### Llama3.1

###### Execution 1: PASS

Prints full code of binary search with a lot of documentation and explanation. It also prints the time complexity of binary search.

###### Execution 2: PASS

Prints a documented Java code for binary search along with explanation of the code. It also explains the time and space complexity of the code.

###### Execution 3: PASS

Similar result as execution 2, but this time the code is in Python.

Starcoder2:

FAIL => Explains binary search in a small paragraph. Does not write code.

# Question 5 – Debugging the Code

Query - Fix the bugs in the code: @documentation\_with\_bugs.py

The result for question 5 can be perceived as least bad to worst. In that case, we can look at the detailed result as:

### Codestral-Mamba

###### Execution 1: PASS

Best result out of all. Corrects all the bugs in the code. Does not explain all the bugs but correct all of them.

###### Execution 2: FAIL

It does not fully fix the code as it left a bug at the end of the code.

Failure Reason: Does not fix the code.

###### Execution 3: FAIL

Fixed all the bugs in the code, but the explanation which is provided is unrelated to the bugs fixed, i.e., the bug which was fixed was different, and the explanation provided was different.

Failure Reason: Incorrect explanation provided.

### Granite-code:8b-instruct

###### Execution 1-3: FAIL

Could only debug the first class and does not provide output of the second class and main function.

Failure Reason: Incomplete code.

### Granite-code:8b-dense

###### Execution 1: FAIL

Does not read the full code either but reads a few more line and fixes one more bug than 8b-instruct. In one bug, it removes the buggy line instead of fixing it in the code.

Failure Reason: Incomplete code.

###### Execution 2-3: FAIL

Reads a few more lines and fixes a couple more bugs but is still not able to read and finish the full code.

Failure Reason: Incomplete code.

### Llama3.1

###### Execution 1: PASS

Fixes all the bugs. Also adds some exception handling to the code.

###### Execution 2: PASS

Fixes all the bugs, also adds some documentation to the code.

###### Execution 3: PASS

Fixes all the bugs, also does some unnecessary code changes like changing the name of the function. Does not do documentation and error handling in this case.

NOTE: In all the three scenarios of Llama, it did not provide a satisfactory explanation. It gave details of what it added to the code (like error handling) but does not provide good explanation of the bug fixes.

Starcoder2:  
  
FAIL => Simply returns the error from the execution, instead of fixing the bugs.

The following table contains a summary of how the models performed. Each cell contains the total number of bugs fixed by the model out of the total 10 bugs.

**Summary Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model Name-> | Granite-code:8b | Granite-dense:8b-dense | Llama3.1:8b | Codestral-mamba:7b | Starcoder2:7b |
| Execution 1 | 3/10 | 4/10# | 10/10 | 10/10 | 0/10 |
| Execution 2 | 3/10 | 6/10 | 10/10 | 9/10 | - |
| Execution 3 | 3/10 | 6/10 | 10/10 | 10/10\* | - |

# It removed one line containing bug instead of fixing it

\*- Bugs were removed in the code but incorrect explanation/ explanation for irrelevant sections was given

# Question 6 – Port the code to Java

Query - Port the code @documentation.py to java

### Codestral-Mamba

###### Execution 1: FAIL

Converts the code and gives a brief explanation. But the converted Java code is broken and does not work as intended.

Failure Reason: Output code has errors.

###### Execution 2: PASS

Converts the code correctly but does not give any explanation.

###### Execution 3: FAIL

Converts the code into multiple classes which needs to be created as separate files. This can be an undesirable behavior.

Failure Reason: Does not convert the code into one file.

### Granite-code:8b-instruct

###### Execution 1: FAIL

Creates two public classes from the original code. Does not import any library/class. Also does not convert the main function. Gives a brief description of the code.

Failure Reason: Incomplete code.

###### Execution 2: FAIL

This time the classes are not public, but it still does not import any library, nor does it convert the main function. Gives a detailed description of the conversion.

Failure Reason: Incomplete code.

Execution 3: FAIL

Again, makes the classes as public and does not import any library. But this time it converts the main function.

Failure Reason: Incomplete code.

ALL three executions give an unusable code.

### Granite-code:8b-dense

###### Execution 1: FAIL

Prints incomplete code (prints both classes but incomplete main function -> prints only a few cases of switch statement). Also makes both classes as public.

Failure Reason: Incomplete code.

###### Execution 2: FAIL

Prints incomplete code again but this time prints a few more cases of the switch statement. Does not make the classes public in this case.

Failure Reason: Incomplete code.

###### Execution 3: FAIL

Similar code length read (incomplete) in this case as well. Again, makes both classes as public.

Failure Reason: Incomplete code.

### Llama3.1

###### Execution 1: FAIL

Converts full code into Java but leaves out some print statement. If those print statements were there, then it would have been a perfect conversion. It also gives the explanation of how functions and objects in Python were mapped to Java.

Failure Reason: Incomplete code.

###### Execution 2: FAIL

Incorrectly converts the Python code into multiple public classes.

Pros: Tells in the beginning how to use those classes in separate files.

Failure Reason: Output code has errors.

###### Execution 3: PASS

Finally converts the full code accurately. Gives a brief description about the functions and objects used.

NOTE: In all three executions, the model changed the names of a few methods and variables. This is an undesirable behavior as it might break the dependency of the code on some other files.

Starcoder2:

FAIL Prints a few lines of gibberish code.

# Question 7 – Refactor Into Quarkus

Query - Refactor the above generated code to quarkus.

### Codestral-Mamba

###### Execution 1:

###### Execution 2:

###### Execution 3:

### Granite-code:8b-instruct

Execution 1: FAIL

Same output is printed again and again -> Basically it gets stuck in an infinite loop.

Execution 2: FAIL

Does not convert the main function.  
Failure Reason: Incomplete code.

Execution 3:

### Granite-code:8b-dense

Execution 1:

Execution 2:

Execution 3:

### Llama3.1

Execution 1:

Execution 2:

Execution 3:

Starcoder2:

FAIL Prints a few lines of gibberish code.

# Question 8 – Optimization Of Code

Query – Optimize the code @code\_to\_optimize.py.

[The input code was an unoptimized version of bubble sort]

### Codestral-Mamba

###### Execution 1: PASS

Converts the code into quick-sort and explains that quick-sort has better time complexity than bubble sort.

###### Execution 2: PASS

Same behavior has the first execution, but with more explanation.

###### Execution 3: PASS

Again, converts into quick sort. This time it uses list comprehension instead of the full code.

### Granite-code:8b-instruct

###### Execution 1: PASS

Converts the code to give a one-line function which uses the inbuilt sorting function of Python. It also lists out the advantages of using the inbuilt function.

###### Execution 2: FAIL

This execution converts the unoptimized bubble sort into an optimized bubble sort. Explains how this version reduces the number of comparisons. But it does not print a main function or a calling function.

Failure Reason: Incomplete code.

Execution 3: FAIL

Gives a broken code which does not sort the array. This result is not usable.

Failure Reason: Output code has errors.

### Granite-code:8b-dense

###### Execution 1: PASS

Converts the code to use the inbuilt sorting function of Python.

###### Execution 2: PASS

Uses the inbuilt list.sort() method of python.

###### Execution 3: FAIL

Gives multiple ways to improve the sorting algorithm. The function given in first and second method does not work -> it does not sort the array.

The third method is an implementation of quicksort with a syntax error. After fixing the error manually, the code works.

### Llama3.1

###### Execution 1: PASS

This execution converts the unoptimized bubble sort into an optimized bubble sort. It also explains the optimization done by the model, like reducing number of comparisons.

UNDESIRABLE BEHAVIOUR: It changed the name of the function given as input.

###### Execution 2: FAIL

It returns a similar output as execution 1, but this time the code does not work. The output algorithm does not sort the input array.

Failure Reason: Incorrect code.

###### Execution 3: PASS

It optimizes the input code by just a slight margin. It also explains how the in-built “sort” function of Python is a better choice to use.

Starcoder2:

Prints a few lines of gibberish code.

# Question 9 – Documentation Of Code

Query – Write documentation comment of the code @documentation.py

VIEW DOCUMENTATION RESULTS ANALYSIS IN TO SEPARATE FOLDER.

# Question 10 – Unit Test of Code

Query -Create Unit test for the code @documentation.py

### Codestral-Mamba

###### Execution 1: PASS

Give unit test of the code using unittest library of python. Runs 5 test case and explains the output code.

###### Execution 2: PASS

Similar code and output as 1st execution.

###### Execution 3: PASS

Creates a separate file for each subclass to unit-test the code and explains how to run these files.

### Granite-code:8b-instruct

###### Execution 1: FAIL

Prints the same result twice.

The output code runs 4 test cases, and it briefly explains how the test cases work.

Execution 2: FAIL

It does not create the result for the full code i.e., it ignores the second class. This behavior is like its behavior in many other cases.

Execution 3: FAIL

It gives output which is unrelated to the prompt given.

Prompt to the model: Create unit test for the code @documentation.py

Output is related to converting the input code to quarkus.

### Granite-code:8b-dense

###### Execution 1: FAIL

Does not even read the first class.

###### Execution 2: FAIL

Does not read the second class.

###### Execution 3: FAIL

Does not read the main function.

### Llama3.1

###### Execution 1: FAIL

Gives incorrect code for testing. The output code has attribute error.

###### Execution 2: PASS

Creates testing classes for both the input classes and explains where the code will fail.

###### Execution 3: PASS

Gives modular code to test the two classes separately in separate files.

Starcoder2:

FAIL: Prints a few lines of gibberish code.