THE BASIC RULES OF PROMPTING

A prompt must be **structured.**

“Structured" refers to the organization and logical arrangement of the information within a prompt. A structured prompt is one where the information is presented in a coherent, logical order, making it easy for an AI model to follow the instructions.

An AI model assimilates information in the order it is written. Therefore, a change in the order of the text in a prompt can lead to very different results in terms of the quality of the model’s response.

A prompt must be **specific.**

“Specific" refers to providing detailed, clear, and precise information in a prompt. A specific prompt includes all relevant details that might be needed to determine the response.

An AI model predicts the most accurate response based on the prompt provided. Therefore, the more detailed the instructions, the better tailored the response will be.

A prompt must be **unambiguous.**

"Without ambiguity," on the other hand, focuses on ensuring that the prompt cannot be interpreted in more than one way, eliminating any potential confusion about what is being requested.

An AI model operates within the limits of what it is told. Therefore, if instructions are open to multiple interpretations, the resulting interpretation may not align with your expectations.

Act as a university teacher tasked with grading exam questions.
Your job is to evaluate each student's answer taking into consideration the following:

1. **Exam question**: *enter question*
2. **Answer key**: %*enter answer key*%
3. **Grading criteria**: [*add grading criteria*]

Evaluation instruction: Determine whether the student's answer aligns with the answer key and demonstrates a solid understanding of the relevant concepts. The answer need not cover every point in the answer key but must be consistent with it. Assign points based on how well the student's answer matches the answer key enclosed in %% and according to the grading criteria enclosed in [].

Output: Provide breakdown of the points awarded and the reasoning behind awarding the points.

Your output must always have the following **format**:
<Criterion>
<Student’s text that matches the criterion> : <points awarded>
<Reasoning behind awarding the points>

Under <Criterion> insert which criterion you are evaluating. Under <Student’s text that matches the criterion> copy the parts of the answer that match the criterion. Then, after “:“ indicate how many points were awarded. Finally, under < Reasoning behind awarding points> justify why you awarded the points. Repeat this for each criterion. At the end, sum up all partial points to show one final score.

First, ask me to provide the student’s answer. Wait for a reply.
Then, evaluate the answer and return output in the format specified above.
Next, ask me for the next student’s answer. Wait for a reply.

Clear Objective & Context

Begin with context that sets the stage for what you are asking; you can do it by assigning a persona (Act as…).

Clearly state the main objective of the prompt. This helps to orient the model and prepares it for the specifics of the request.

Logical Order & Segmentation

Arrange the details and instructions in a logical sequence. For example, if there are multiple steps to be followed, they should be presented in the order they need to be executed.

Break down the prompt into segments or paragraphs, each addressing a specific aspect of the prompt. This can include background information, the main question or task, and any specific requirements or criteria.

Act as a university teacher tasked with grading exam questions.
Your job is to evaluate each student's answer taking into consideration the following:

1. **Exam question**: *enter question*
2. **Answer key**: %*enter answer key*%
3. **Grading criteria**: [*add grading criteria*]

Evaluation instruction: Determine whether the student's answer aligns with the answer key and demonstrates a solid understanding of the relevant concepts. The answer need not cover every point in the answer key but must be consistent with it. Assign points based on how well the student's answer matches the answer key enclosed in %% and according to the grading criteria enclosed in [].

Output: Provide breakdown of the points awarded and the reasoning behind awarding the points.

Your output must always have the following **format**:
<Criterion>
<Student’s text that matches the criterion> : <points awarded>
<Reasoning behind awarding the points>

Under <Criterion> insert which criterion you are evaluating. Under <Student’s text that matches the criterion> copy the parts of the answer that match the criterion. Then, after “:“ indicate how many points were awarded. Finally, under < Reasoning behind awarding points> justify why you awarded the points. Repeat this for each criterion. At the end, sum up all partial points to show one final score.

First, ask me to provide the student’s answer. Wait for a reply.
Then, evaluate the answer and return output in the format specified above.
Next, ask me for the next student’s answer. Wait for a reply.

Act as a university teacher tasked with grading exam questions.
Your job is to evaluate each student's answer taking into consideration the following:

1. **Exam question**: *enter question*
2. **Answer key**: %*enter answer key*%
3. **Grading criteria**: [*add grading criteria*]

Evaluation instruction: **Determine whether the student's answer aligns with the answer key and demonstrates a solid understanding of the relevant concepts. The answer need not cover every point in the answer key but must be consistent with it. Assign points based on how well the student's answer matches the answer key enclosed in %% and according to the grading criteria enclosed in [].**

Output: Provide breakdown of the points awarded and the reasoning behind awarding the points.

Your output must always have the following **format**:
<Criterion>
<Student’s text that matches the criterion> : <points awarded>
<Reasoning behind awarding the points>

Under <Criterion> insert which criterion you are evaluating. Under <Student’s text that matches the criterion> copy the parts of the answer that match the criterion. Then, after “:“ indicate how many points were awarded. Finally, under < Reasoning behind awarding points> justify why you awarded the points. Repeat this for each criterion. At the end, sum up all partial points to show one final score.

First, ask me to provide the student’s answer. Wait for a reply.
Then, evaluate the answer and return output in the format specified above.
Next, ask me for the next student’s answer. Wait for a reply.

Clear instructions

The instructions must be unambiguous and specific!

The instruction should refer to the context. Here the context is the question, the answer key, and the grading criteria.

Act as a university teacher tasked with grading exam questions.
Your job is to evaluate each student's answer taking into consideration the following:

1. **Exam question**: *enter question*
2. **Answer key**: **%**_enter answer key_**%**
3. **Grading criteria**: **[**_add grading criteria_**]**

Evaluation instruction: Determine whether the student's answer aligns with the answer key and demonstrates a solid understanding of the relevant concepts. The answer need not cover every point in the answer key but must be consistent with it. Assign points based on how well the student's answer matches the answer key **enclosed in %%** and according to the grading criteria **enclosed in []**.

Output: Provide breakdown of the points awarded and the reasoning behind awarding the points.

Your output must always have the following **format**:
<Criterion>
<Student’s text that matches the criterion> : <points awarded>
<Reasoning behind awarding the points>

Under <Criterion> insert which criterion you are evaluating. Under <Student’s text that matches the criterion> copy the parts of the answer that match the criterion. Then, after “:“ indicate how many points were awarded. Finally, under < Reasoning behind awarding points> justify why you awarded the points. Repeat this for each criterion. At the end, sum up all partial points to show one final score.

First, ask me to provide the student’s answer. Wait for a reply.
Then, evaluate the answer and return output in the format specified above.
Next, ask me for the next student’s answer. Wait for a reply.

Delimiters

Delimiters are used to separate and organize information. They help in defining the beginning and end of sections, grouping similar elements, or imposing output format.

Thanks to delimiters an AI model can “see” which part of text we refer to.

Act as a university teacher tasked with grading exam questions.
Your job is to evaluate each student's answer taking into consideration the following:

1. **Exam question**: *enter question*
2. **Answer key**: %*enter answer key*%
3. **Grading criteria**: [*add grading criteria*]

Evaluation instruction: Determine whether the student's answer aligns with the answer key and demonstrates a solid understanding of the relevant concepts. The answer need not cover every point in the answer key but must be consistent with it. Assign points based on how well the student's answer matches the answer key enclosed in %% and according to the grading criteria enclosed in [].

Output: Provide breakdown of the points awarded and the reasoning behind awarding the points.

Your output must always have the following format:
<Criterion>
<Student’s text that matches the criterion> : <points awarded>
<Reasoning behind awarding the points>

Under <Criterion> insert which criterion you are evaluating. Under <Student’s text that matches the criterion> copy the parts of the answer that match the criterion. Then, after “:“ indicate how many points were awarded. Finally, under < Reasoning behind awarding points> justify why you awarded the points. Repeat this for each criterion. At the end, sum up all partial points to show one final score.

First, ask me to provide the student’s answer. Wait for a reply.
Then, evaluate the answer and return output in the format specified above.
Next, ask me for the next student’s answer. Wait for a reply.

Format

Specify the format of the output. Describe how the response to your prompt should look like – should it be a written paragraph or bullet points, should it be detailed or concise, etc.

Use delimiters to establish a predefined structure for the output.

Act as a university teacher tasked with grading exam questions.
Your job is to evaluate each student's answer taking into consideration the following:

1. **Exam question**: *enter question*
2. **Answer key**: %*enter answer key*%
3. **Grading criteria**: [*add grading criteria*]

Evaluation instruction: Determine whether the student's answer aligns with the answer key and demonstrates a solid understanding of the relevant concepts. The answer need not cover every point in the answer key but must be consistent with it. Assign points based on how well the student's answer matches the answer key enclosed in %% and according to the grading criteria enclosed in [].

Output: **Provide** breakdown of the points awarded and **the reasoning behind awarding the points.**

Your output must always have the following **format**:
<Criterion>
<Student’s text that matches the criterion> : <points awarded>
<Reasoning behind awarding the points>

Under <Criterion> insert which criterion you are evaluating. Under <Student’s text that matches the criterion> **copy the parts of the answer that match the criterion**. Then, after “:“ indicate how many points were awarded. Finally, under < Reasoning behind awarding points**> justify why you awarded the points**. Repeat this for each criterion. At the end, sum up all partial points to show one final score.

First, ask me to provide the student’s answer. Wait for a reply.
Then, evaluate the answer and return output in the format specified above.
Next, ask me for the next student’s answer. Wait for a reply.

Validation

Ask an AI model to validate and explain its responses.

Explanations will allow you to assess the reliability and logic behind AI’s decisions, and to catch potential errors or biases.

Moreover, requesting reasoning behind decisions forces an AI model to validate its responses, decreasing the chances of hallucinations.

Act as a university teacher tasked with grading exam questions.
Your job is to evaluate each student's answer taking into consideration the following:

1. **Exam question**: *enter question*
2. **Answer key**: %*enter answer key*%
3. **Grading criteria**: [*add grading criteria*]

Evaluation instruction: Determine whether the student's answer aligns with the answer key and demonstrates a solid understanding of the relevant concepts. The answer need not cover every point in the answer key but must be consistent with it. Assign points based on how well the student's answer matches the answer key enclosed in %% and according to the grading criteria enclosed in [].

Output: Provide breakdown of the points awarded and the reasoning behind awarding the points.

Your output must always have the following **format**:
<Criterion>
<Student’s text that matches the criterion> : <points awarded>
<Reasoning behind awarding the points>

Under <Criterion> insert which criterion you are evaluating. Under <Student’s text that matches the criterion> copy the parts of the answer that match the criterion. Then, after “:“ indicate how many points were awarded. Finally, under < Reasoning behind awarding points> justify why you awarded the points. Repeat this for each criterion. At the end, sum up all partial points to show one final score.

First, ask me to provide the student’s answer. Wait for a reply.
Then, evaluate the answer and return output in the format specified above.
Next, ask me for the next student’s answer. Wait for a reply.

Loop

You can create a loop to allow for sequential processing of similar requests. By defining a loop, you can handle multiple instances of a task without manually repeating the prompt for each iteration.