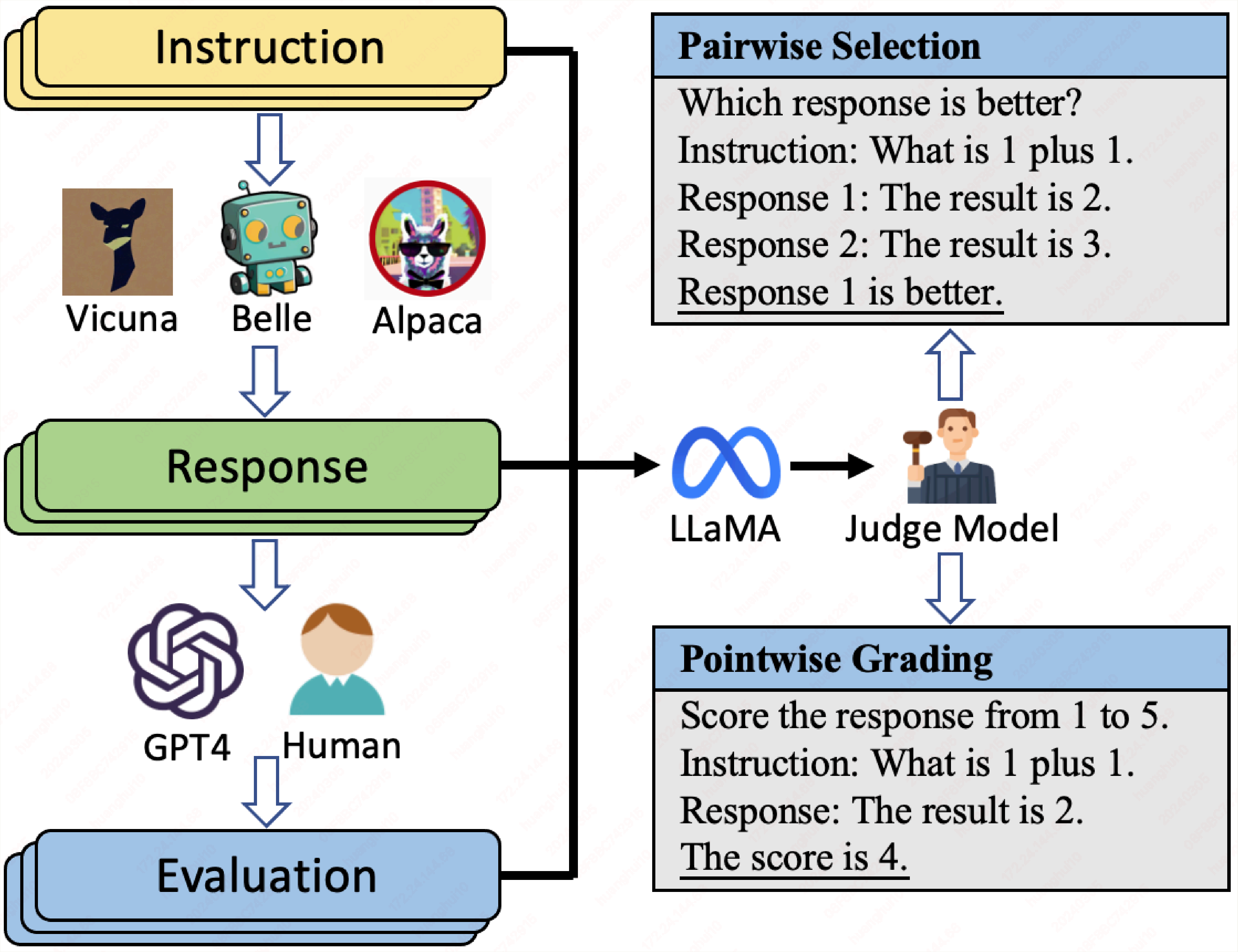
Practical Guide to LLM-as-a-Judge Evaluation



Pairwise Selection

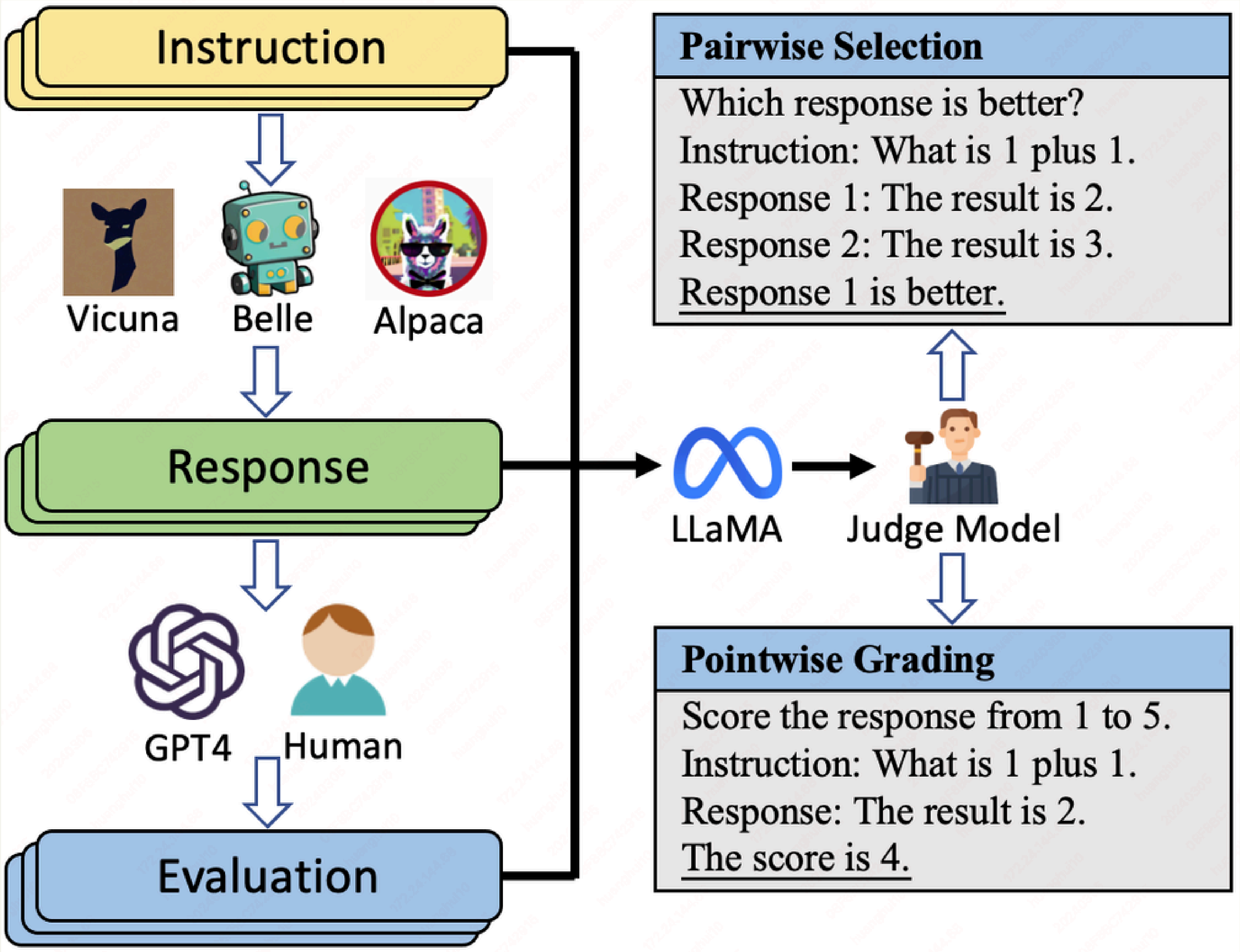
Which response is better?
Instruction: What is 1 plus 1.
Response 1: The result is 2.
Response 2: The result is 3.
Response 1 is better.

Pointwise Grading

Score the response from 1 to 5.
Instruction: What is 1 plus 1.
Response: The result is 2.
The score is 4.

Dipanjan (DJ)

What is LLM-as-a-Judge?



Pairwise Selection

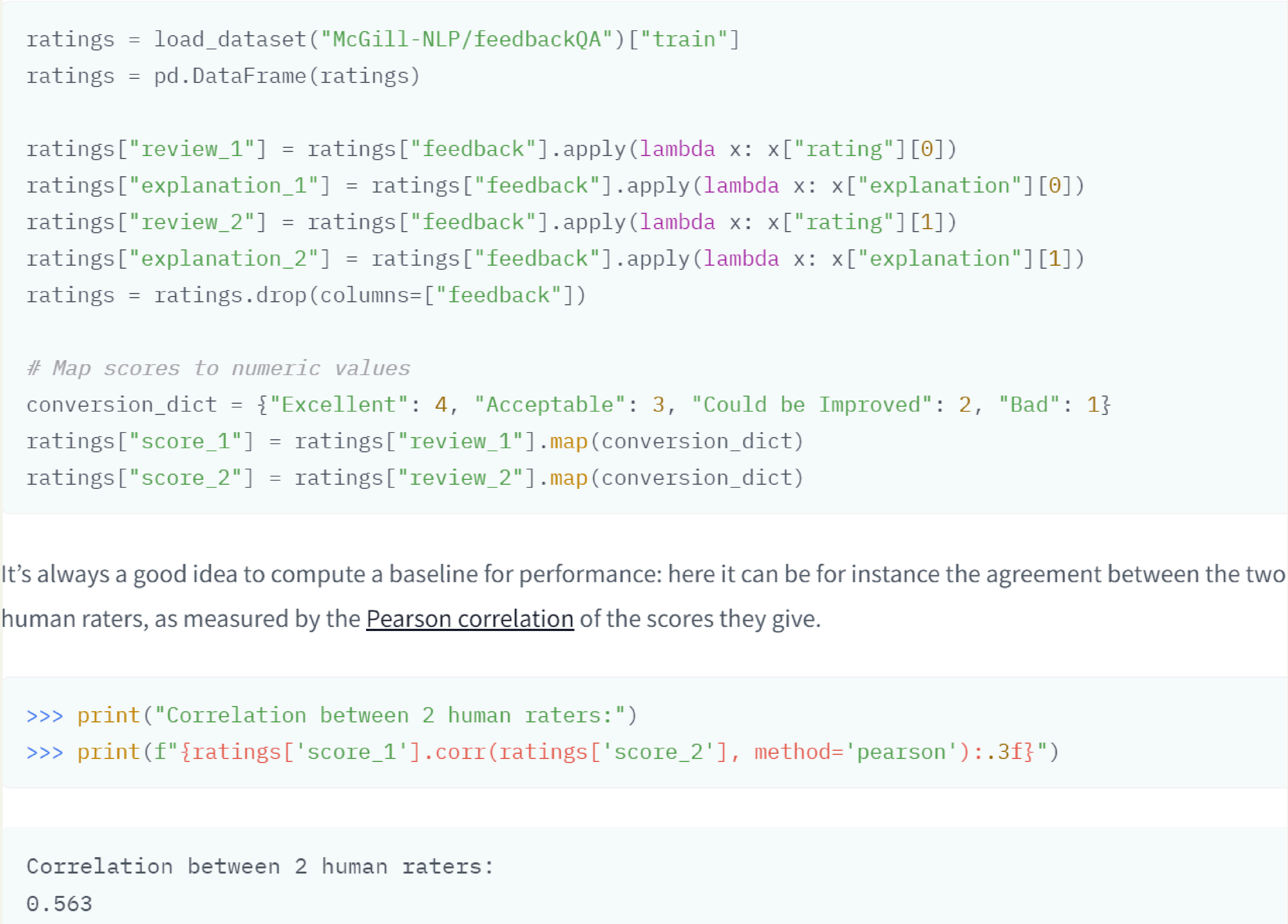
Which response is better?
Instruction: What is 1 plus 1.
Response 1: The result is 2.
Response 2: The result is 3.
Response 1 is better.

Pointwise Grading

Score the response from 1 to 5.
Instruction: What is 1 plus 1.
Response: The result is 2.
The score is 4.

- **Evaluating LLM responses are hard as they are unstructured**
- **Traditional evaluation metrics are based on similarity like BLEU, ROUGE etc.**
- **Traditional metrics become difficult to evaluate especially when we can have a lot of variation in responses**
- **LLM-as-a-Judge enables us to specific a strict set of criteria as prompts to reason and evaluate another LLM's responses**
- **It is recommended to use a powerful LLM for evaluation**

Step 1 - Eval Dataset Prep



```
ratings = load_dataset("McGill-NLP/feedbackQA")["train"]
ratings = pd.DataFrame(ratings)

ratings["review_1"] = ratings["feedback"].apply(lambda x: x["rating"][0])
ratings["explanation_1"] = ratings["feedback"].apply(lambda x: x["explanation"][0])
ratings["review_2"] = ratings["feedback"].apply(lambda x: x["rating"][1])
ratings["explanation_2"] = ratings["feedback"].apply(lambda x: x["explanation"][1])
ratings = ratings.drop(columns=["feedback"])

# Map scores to numeric values
conversion_dict = {"Excellent": 4, "Acceptable": 3, "Could be Improved": 2, "Bad": 1}
ratings["score_1"] = ratings["review_1"].map(conversion_dict)
ratings["score_2"] = ratings["review_2"].map(conversion_dict)
```

It’s always a good idea to compute a baseline for performance: here it can be for instance the agreement between the two human raters, as measured by the Pearson correlation of the scores they give.

```
>>> print("Correlation between 2 human raters:")
>>> print(f"{ratings['score_1'].corr(ratings['score_2'], method='pearson'):.3f}")
```

```
Correlation between 2 human raters:
0.563
```

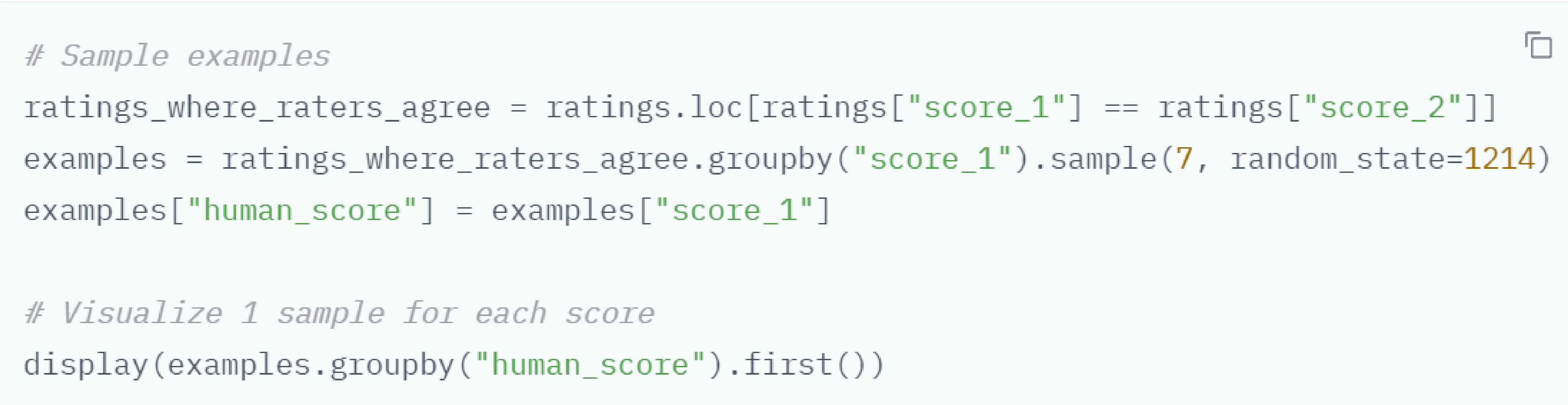
- **The dataset used here is feedbackQA, which contains 2 human evaluations and scores for each question/answer couple**

- **Looks like there are problems in the ground truth itself, something you need to be careful if you are using multiple human graders to create the initial eval dataset**

- **Various ways of handling this in terms of taking averages, removing disagreement examples, re-evaluation etc.**

Source: https://huggingface.co/learn/cookbook/en/llm_judge

Step 2 - Correct Dataset



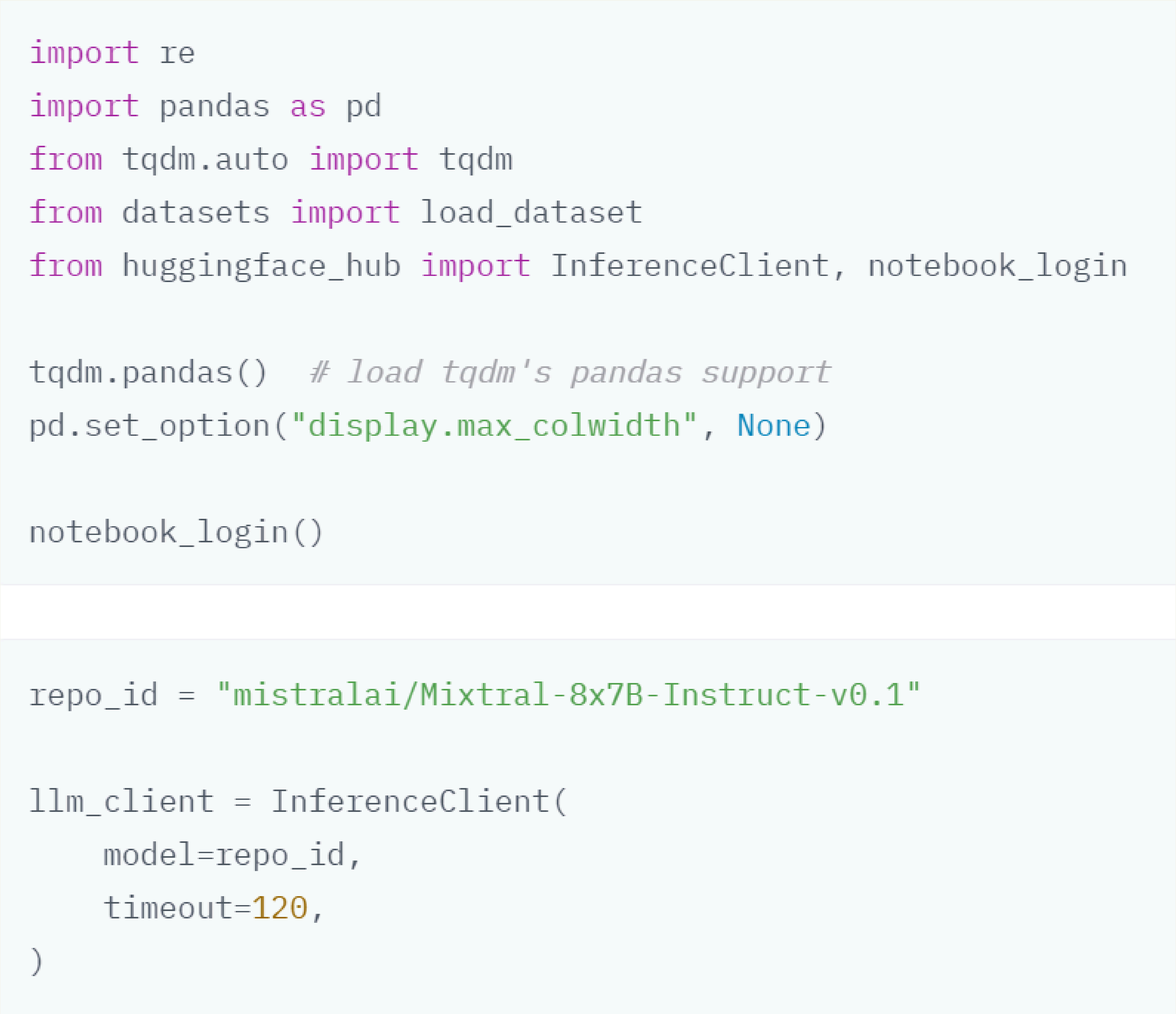
```python
# Sample examples
ratings_where_raters_agree = ratings.loc[ratings["score_1"] == ratings["score_2"]]
examples = ratings_where_raters_agree.groupby("score_1").sample(7, random_state=1214)
examples["human_score"] = examples["score_1"]

# Visualize 1 sample for each score
display(examples.groupby("human_score").first())
```

- **As we saw, the correlation between 2 human raters is not that good so we need to correct our dataset**

- **To make things simple, we only keep examples where the 2 human reviewers are in agreement**

Source: https://huggingface.co/learn/cookbook/en/llm_judge

Step 3 - Create our LLM judge



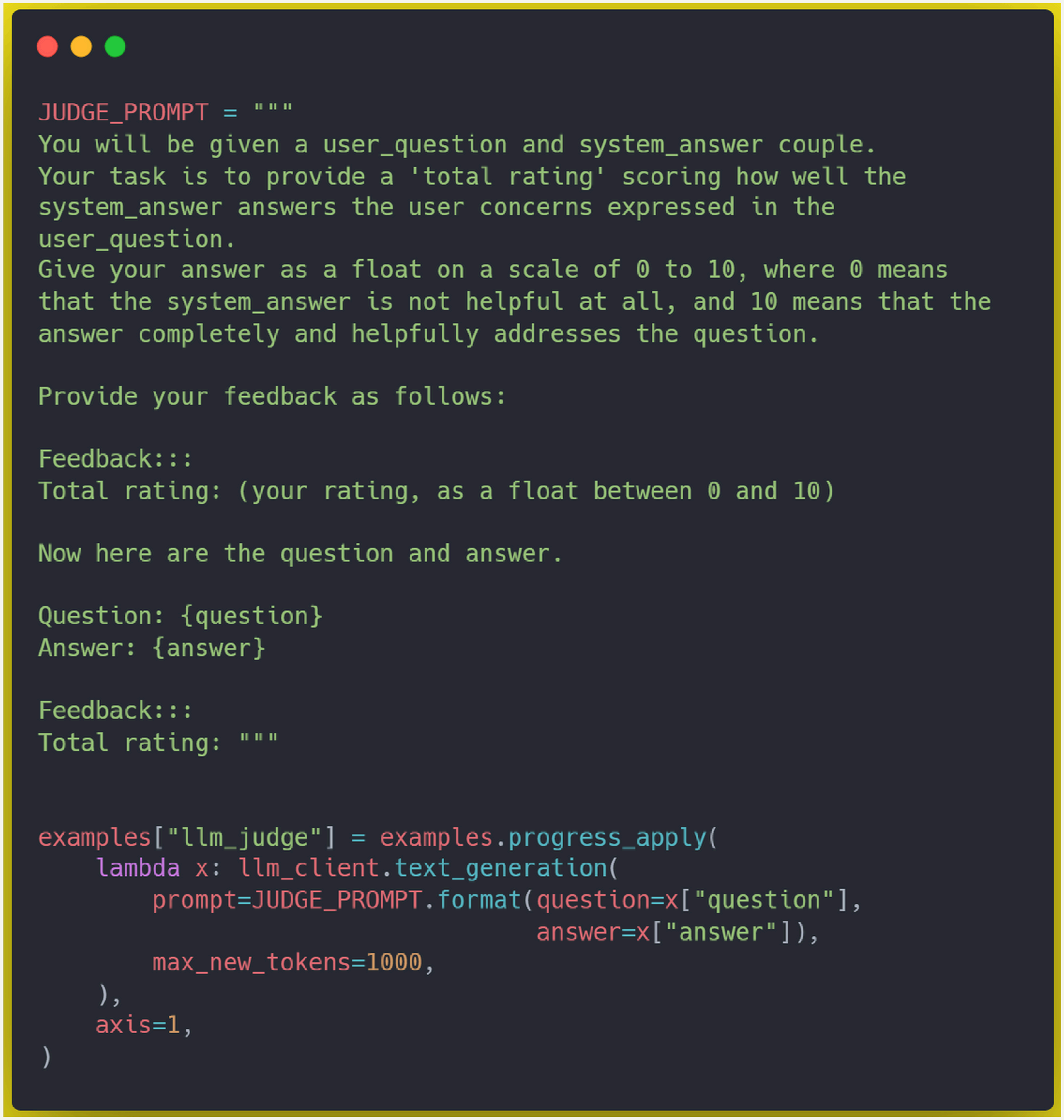
```python
import re
import pandas as pd
from tqdm.auto import tqdm
from datasets import load_dataset
from huggingface_hub import InferenceClient, notebook_login

tqdm.pandas()  # load tqdm's pandas support
pd.set_option("display.max_colwidth", None)

notebook_login()
```

```python
repo_id = "mistralai/Mixtral-8x7B-Instruct-v0.1"

llm_client = InferenceClient(
    model=repo_id,
    timeout=120,
)
```

- **We load up a Mistral AI Mixtral 7B LLM to use as a Judge**

Source: https://huggingface.co/learn/cookbook/en/llm_judge

Step 3 - Create our LLM judge

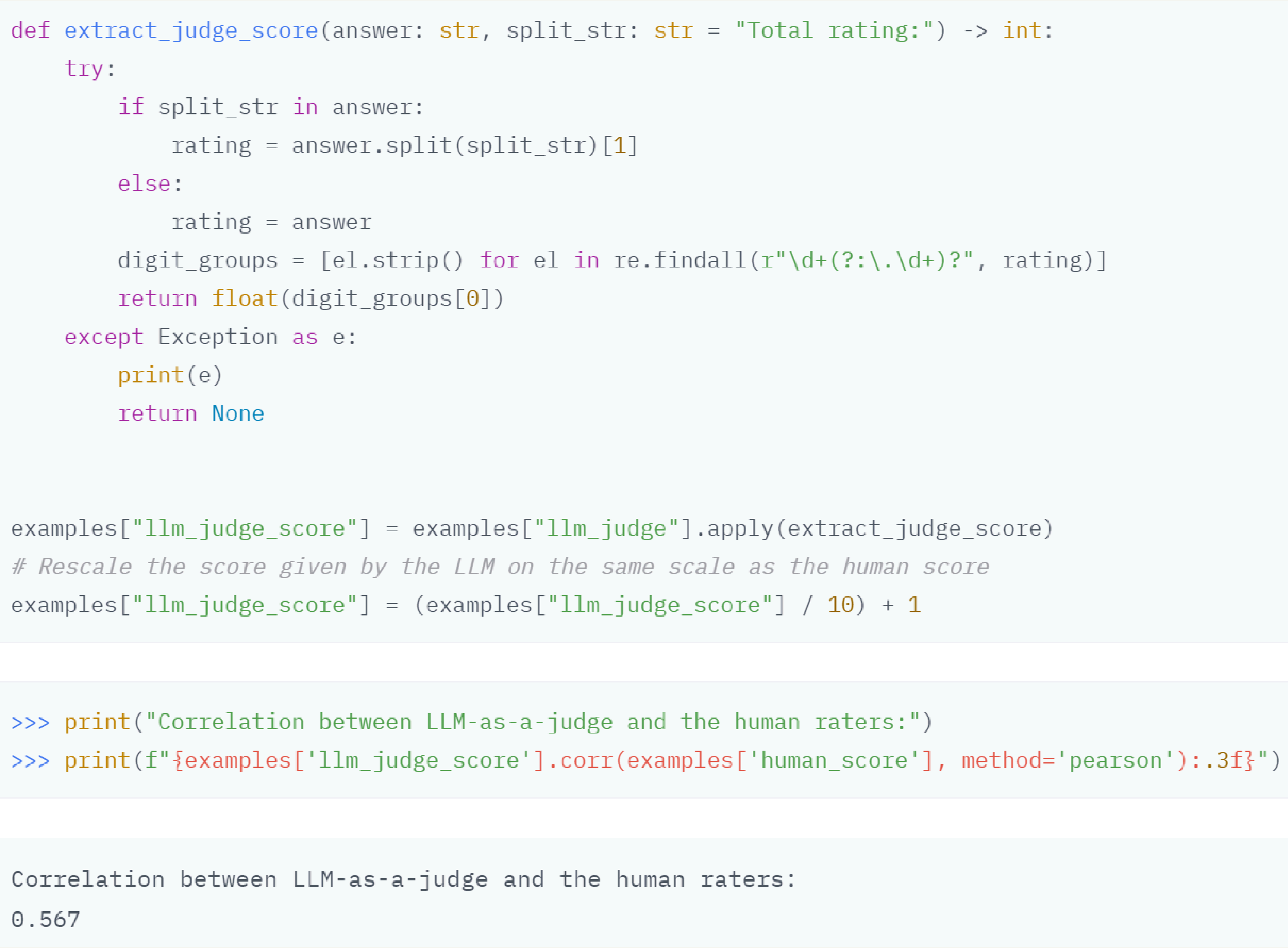


```python
JUDGE_PROMPT = """
You will be given a user_question and system_answer couple.
Your task is to provide a 'total rating' scoring how well the
system_answer answers the user concerns expressed in the
user_question.
Give your answer as a float on a scale of 0 to 10, where 0 means
that the system_answer is not helpful at all, and 10 means that the
answer completely and helpfully addresses the question.

Provide your feedback as follows:

Feedback:::
Total rating: (your rating, as a float between 0 and 10)

Now here are the question and answer.

Question: {question}
Answer: {answer}

Feedback:::
Total rating: """


examples["llm_judge"] = examples.progress_apply(
    lambda x: llm_client.text_generation(
        prompt=JUDGE_PROMPT.format(question=x["question"],
                                   answer=x["answer"]),
        max_new_tokens=1000,
    ),
    axis=1,
)
```

- **We build our LLM judge with a basic prompt, containing these elements:**
  - task description
  - scale description: minimum, maximum, value types (float here)
  - explanation of the output format
  - a beginning of an answer, to take the LLM by the hand as far as we can
- **We run this prompt then for each QA pair**

Source: https://huggingface.co/learn/cookbook/en/llm_judge

Step 3 - Create our LLM judge



```python
def extract_judge_score(answer: str, split_str: str = "Total rating:") -> int:
    try:
        if split_str in answer:
            rating = answer.split(split_str)[1]
        else:
            rating = answer
        digit_groups = [el.strip() for el in re.findall(r"\d+(?:\.\d+)?", rating)]
        return float(digit_groups[0])
    except Exception as e:
        print(e)
        return None


examples["llm_judge_score"] = examples["llm_judge"].apply(extract_judge_score)
# Rescale the score given by the LLM on the same scale as the human score
examples["llm_judge_score"] = (examples["llm_judge_score"] / 10) + 1
```

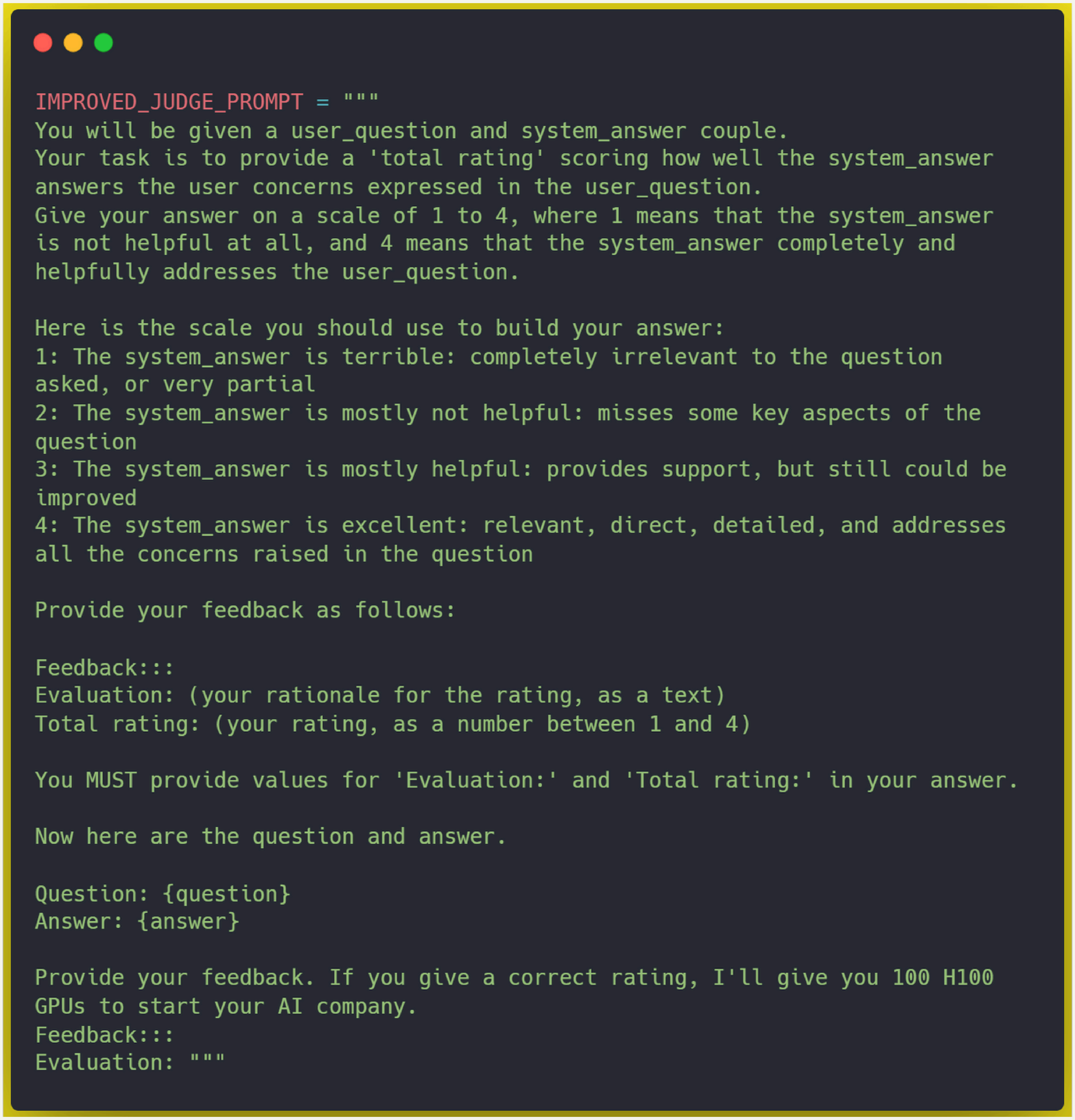
```
>>> print("Correlation between LLM-as-a-judge and the human raters:")
>>> print(f"{examples['llm_judge_score'].corr(examples['human_score'], method='pearson'):.3f}")
```

```
Correlation between LLM-as-a-judge and the human raters:
0.567
```

- **In this step we extract out the scores from the LLM Judge, we could use a powerful LLM and even get a more structured score in JSON also making it easier**
- **Comparing the scores between the LLM Judge and Human Graders we get a correlation of 56.7% which is not bad, but how can we improve it?**

Source: https://huggingface.co/learn/cookbook/en/llm_judge

Step 4- Improve LLM judge

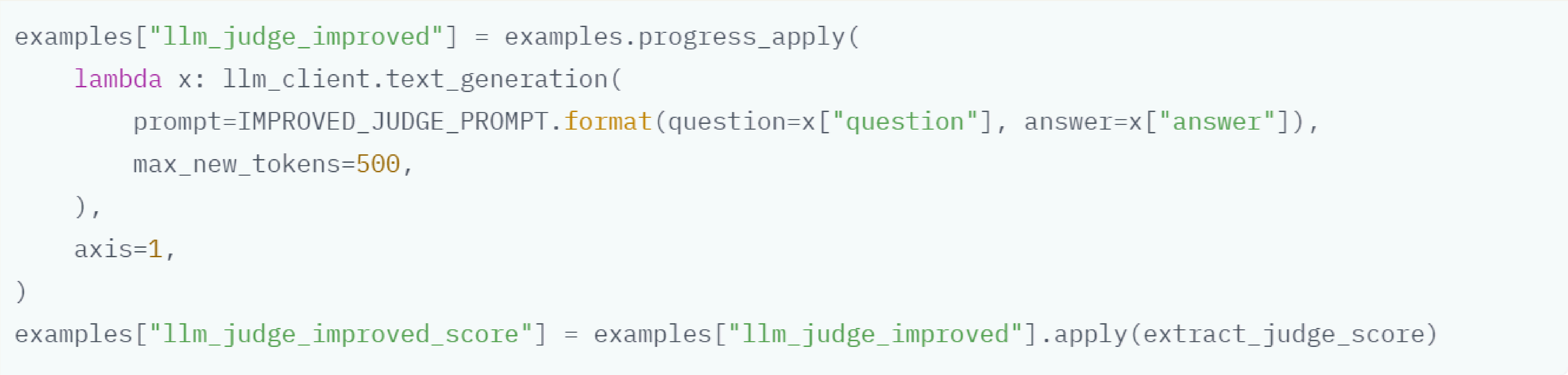


```
IMPROVED_JUDGE_PROMPT = """
You will be given a user_question and system_answer couple.
Your task is to provide a 'total rating' scoring how well the system_answer
answers the user concerns expressed in the user_question.
Give your answer on a scale of 1 to 4, where 1 means that the system_answer
is not helpful at all, and 4 means that the system_answer completely and
helpfully addresses the user_question.

Here is the scale you should use to build your answer:
1: The system_answer is terrible: completely irrelevant to the question
asked, or very partial
2: The system_answer is mostly not helpful: misses some key aspects of the
question
3: The system_answer is mostly helpful: provides support, but still could be
improved
4: The system_answer is excellent: relevant, direct, detailed, and addresses
all the concerns raised in the question

Provide your feedback as follows:

Feedback:::
Evaluation: (your rationale for the rating, as a text)
Total rating: (your rating, as a number between 1 and 4)

You MUST provide values for 'Evaluation:' and 'Total rating:' in your answer.

Now here are the question and answer.

Question: {question}
Answer: {answer}

Provide your feedback. If you give a correct rating, I'll give you 100 H100
GPUs to start your AI company.
Feedback:::
Evaluation: """
```

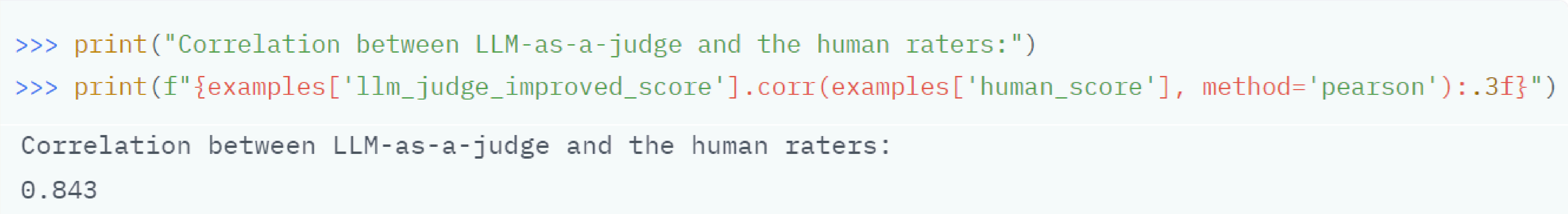
- ⏳ **Leave more time for thought by adding an Evaluation field before the final answer.**
- 🔢 **Use a small integer scale like 1-4 or 1-5 instead of a large float scale as we had previously.**
- 👩‍🏫 **Provide an indicative scale for guidance.**
- **We even add a carrot to motivate the LLM!**

Source: https://huggingface.co/learn/cookbook/en/llm_judge

Step 4- Improve LLM judge



```
examples["llm_judge_improved"] = examples.progress_apply(
    lambda x: llm_client.text_generation(
        prompt=IMPROVED_JUDGE_PROMPT.format(question=x["question"], answer=x["answer"]),
        max_new_tokens=500,
    ),
    axis=1,
)
examples["llm_judge_improved_score"] = examples["llm_judge_improved"].apply(extract_judge_score)
```



```
>>> print("Correlation between LLM-as-a-judge and the human raters:")
>>> print(f"{examples['llm_judge_improved_score'].corr(examples['human_score'], method='pearson'):.3f}")
```

```
Correlation between LLM-as-a-judge and the human raters:
0.843
```

- **The correlation was improved by nearly 30% with only a few tweaks to the prompt**

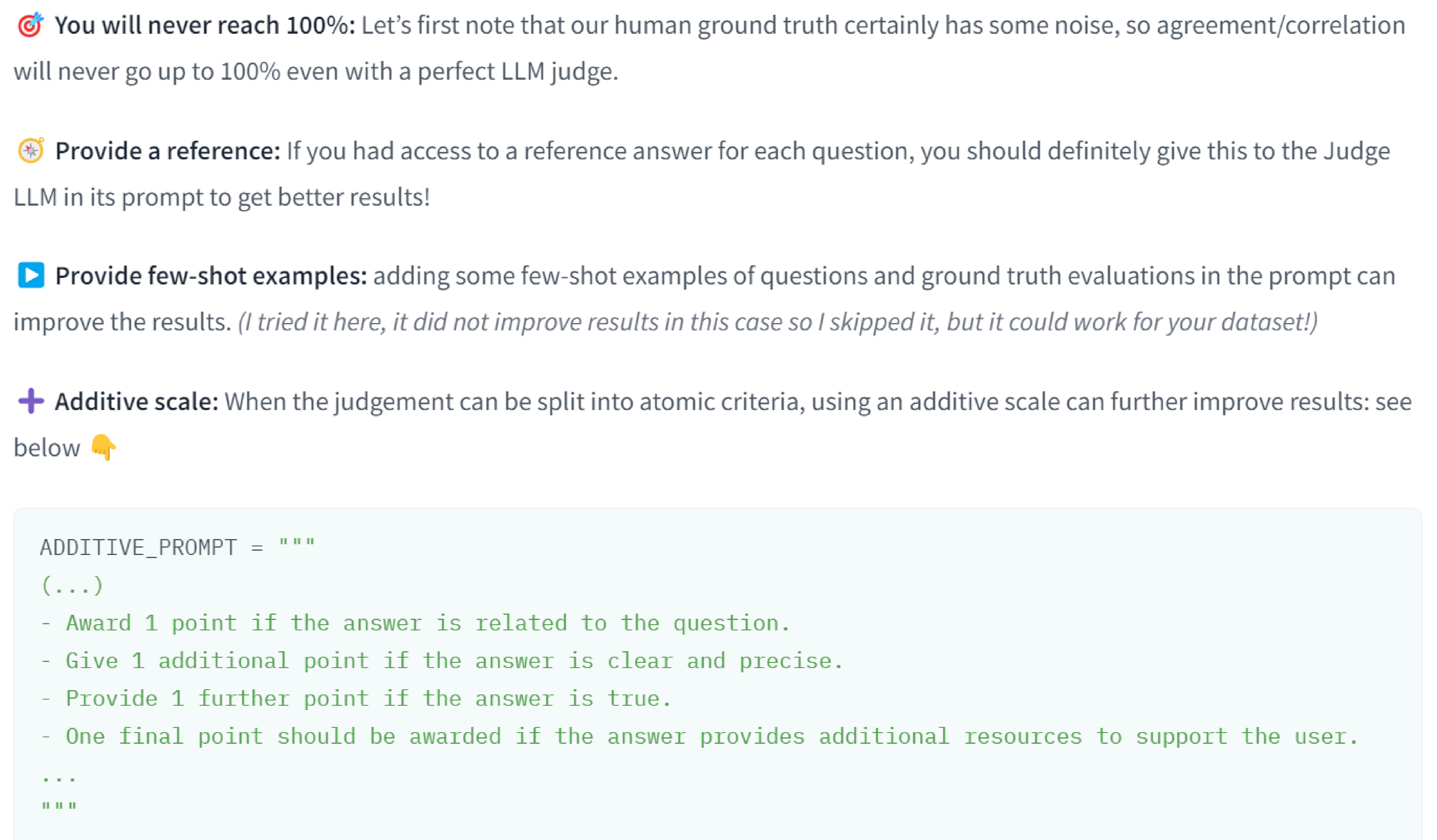
- **There is no perfect judge prompt, it totally depends on the problem and scenario but using the previous mentioned points are always good**

- **Overall try to make the scoring criteria clear and quantitative**

- **Few-shot examples dramatically help in improving performance also**

Source: https://huggingface.co/learn/cookbook/en/llm_judge

Other Considerations



🎯 **You will never reach 100%:** Let’s first note that our human ground truth certainly has some noise, so agreement/correlation will never go up to 100% even with a perfect LLM judge.

🧭 **Provide a reference:** If you had access to a reference answer for each question, you should definitely give this to the Judge LLM in its prompt to get better results!

▶️ **Provide few-shot examples:** adding some few-shot examples of questions and ground truth evaluations in the prompt can improve the results. *(I tried it here, it did not improve results in this case so I skipped it, but it could work for your dataset!)*

➕ **Additive scale:** When the judgement can be split into atomic criteria, using an additive scale can further improve results: see below 👇

```
ADDITIVE_PROMPT = """
(...)
- Award 1 point if the answer is related to the question.
- Give 1 additional point if the answer is clear and precise.
- Provide 1 further point if the answer is true.
- One final point should be awarded if the answer provides additional resources to support the user.
...
"""
```

Source: https://huggingface.co/learn/cookbook/en/llm_judge