Prompting Principles

Principle 1: Write clear and specific instructions

Tactic 1: Use delimiters:

```
Summarize the text delimited by triple <backticks/quotes/tags> into a single sentence.
```{text}```
```

Tactic 2: Ask for a structured output:

```
Provide results in JSON format with the following keys: <book_id>, <title>, <author>, <genre>.
```

Tactic 3: Ask the model to check whether conditions are satisfied:

```
If the text does not contain <something> then simply write "<something>"
```

Tactic 4: "Few-shot" prompting:

```
Your task is to answer in a consistent style.
<student>: text/question
<teacher>: text/answer
<student>: text/question
```

Principle 2: Give the model time to "think"

Tactic 1: Specify the steps required to complete a task:

```
Perform the following actions:
1 - Summarize the following text delimited by triple backticks with <1> sentence
2 - Translate the summary into <French>
3 - List each name in the summary
4 - Output a json object that contains the following keys: summary, num_names

Separate your answers with line breaks.

Text:
```{text}```
```

Tactic 2: Ask for output in a specified format:

```
Your task is to perform the following actions:
1 - Step 1.
2 - Step 2.
3 - Step 3.
4 - Step N.

Use the following format:
Text: <text to summarize>
Summary: <summary>
Translation: <summary translation>
Names: <list of names in French summary>
Output JSON: <json with summary and num_names>

Text: <{text}>
```

Tactic 2: Instruct the model to work out its own solution before rushing to a conclusion:

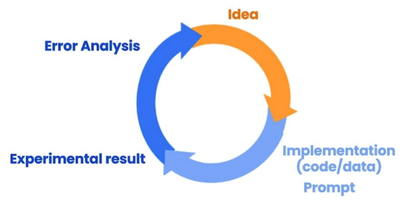
```
Your task is to determine if the student's solution is correct or not.
To solve the problem do the following:
- First, work out your own solution to the problem.
- Then compare your solution to the student's solution and evaluate if the student's solution is correct or not.

Don't decide if the student's solution is correct until you have done the problem yourself.

Use the following format:
Question:
```
question here
```
Student's solution:
```
student's solution here
```
Actual solution:
```
steps to work out the solution and your solution here
```
Is the <student's solution> the same as actual solution just calculated:
```
yes or no
```
Student grade:
```
correct or incorrect
```

Question:
```
I'm building a solar power installation and I need help working out the financials.
- Land costs $100 / square foot
- I can buy solar panels for $250 / square foot
- I negotiated a contract for maintenance that will cost me a flat $100k per year, and an additional $10 / square foot
What is the total cost for the first year of operations as a function of the number of square feet.
```
Student's solution:
```
Let x be the size of the installation in square feet..
Costs:
1. Land cost: 100x
2. Solar panel cost: 250x
3. Maintenance cost: 100,000 + 100x
Total cost: 100x + 250x + 100,000 + 100x = 450x + 100,000
```
Actual solution:
```

Iterative Prompt Development:



There is no perfect prompt for everything. Develop your prompt in an iterative way:
- Try something and be clear and specific.
- Analyze why result does not give desired output.
- Refine the idea and the prompt
- Repeat

Deal with Issues:

Output text is too long:

Limit the number of words/sentences/characters

```
Use at most 50 words.
```

Output text focuses on the wrong details:

```
The output/answer/description is intended for <specific_group>, so should be <technical> in nature and focus on <some_aspects>.
```

Output text should be organized in a table:

```
After the description, include a table that gives the <information>. The table should have <two> columns.
In the first column include <first_name>.
In the second column include <grades>.

Give the table the title 'Student_Grades'.

Format everything as <HTML/Markdown> that can be used in a website.
Place the description in a <div> element.
```

Summarize vs. Extract:

- Summaries include topics that my not be related to the topic of focus.

```
Your task is to generate a short summary of a <product_review> from an <ecommerce_site> to give feedback to the <pricing_department>, responsible for determining the price of the product.

Summarize the review below, delimited by triple backticks, in at most 30 words, and focusing on any aspects that are relevant to the price and perceived value.

Review: ```{prod_review}```
```

```
Your task is to extract relevant information from a <product_review> from an <ecommerce_site> to give feedback to the Shipping department.

From the review below, delimited by triple quotes extract the information relevant to <shipping and delivery>. Limit to 30 words.

Review: ```{prod_review}```
```

Inferring:

- Inferring tasks mean that the model takes a text as input and performs some kind of analysis, e.g., extracting labels, extracting names, analyzing sentiment of a text.

```
Identify the following items from the <review text>:
- Sentiment (positive or negative)
- Is the reviewer expressing anger? (true or false)
- Item purchased by reviewer
- Company that made the item

The review is delimited with triple backticks.
Format your response as a <JSON_object> with "Sentiment", "Anger", "Item" and "Brand" as the keys.
If the information isn't present, use "unknown" as the value.
Make your response as short as possible.
Format the Anger value as a boolean.

Review text: ```{review}```
```

```
Determine five topics that are being discussed in the following text, which is delimited by triple backticks.

Make each item one or two words long.

Format your response as a list of items separated by commas.

Text sample: ```{story}```
```

```
Determine whether each item in the following list of topics is a topic in the text below, which is delimited with triple backticks.

Give your answer as list with 0 or 1 for each topic.

List of topics: {", ".join(topic_list)}

Text sample: ```{story}```
```

Temperature:

- Governs the randomness/creativity of the responses.
- Set `temperature` to `0` for building reliable and predictable systems.
- Set `temperature` to `0 - 0.3` for extraction, standardization, format conversion, and grammar fixes tasks.
- Set `temperature` to `0.5` for writing tasks.

OpenAI API call

```
def get_completion(prompt,
                   model="gpt-3.5-turbo"):
messages = [{"role": "user",
             "content": prompt}]

response = openai.ChatCompletion.create(
             model=model,
             messages=messages,
             temperature=0)
```


Role

```
messages =
[
 {"role": "system",
  "content": "You are an assistant… "},
 {"role": "user",
  "content": "tell me a joke "},
 {"role": "assistant",
  "content": "Why did the chicken… "},
 …
]
```

Prompting Principles

Principle 1: Write clear and specific instructions

Tactic 1: Use delimiters:

```
Summarize the text delimited by triple <backticks/quotes/tags> into a single sentence.
```{text}```
```

Tactic 2: Ask for a structured output:

```
Provide results in JSON format with the following keys: <book_id>, <title>, <author>, <genre>.
```

Tactic 3: Ask the model to check whether conditions are satisfied:

```
If the text does not contain <something> then simply write "<something>"
```

Tactic 4: "Few-shot" prompting:

```
Your task is to answer in a consistent style.
<student>: text/question
<teacher>: text/answer
<student>: text/question
```

Principle 2: Give the model time to "think"

Tactic 1: Specify the steps required to complete a task:

```
Perform the following actions:
1 - Summarize the following text delimited by triple backticks with <1> sentence
2 - Translate the summary into <French>
3 - List each name in the summary
4 - Output a json object that contains the following keys: summary, num_names

Separate your answers with line breaks.

Text:
```{text}```
```

Tactic 2: Ask for output in a specified format:

```
Your task is to perform the following actions:
1 - Step 1.
2 - Step 2.
3 - Step 3.
4 - Step N.

Use the following format:
Text: <text to summarize>
Summary: <summary>
Translation: <summary translation>
Names: <list of names in French summary>
Output JSON: <json with summary and num_names>

Text: <{text}>
```

Tactic 2: Instruct the model to work out its own solution before rushing to a conclusion:

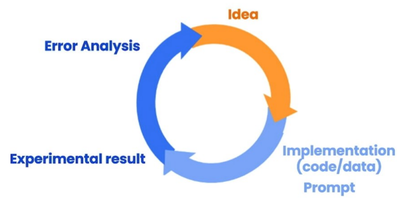
```
Your task is to determine if the student's solution is correct or not.
To solve the problem do the following:
- First, work out your own solution to the problem.
- Then compare your solution to the student's solution and evaluate if the student's solution is correct or not.

Don't decide if the student's solution is correct until you have done the problem yourself.

Use the following format:
Question:
```
question here
```
Student's solution:
```
student's solution here
```
Actual solution:
```
steps to work out the solution and your solution here
```
Is the <student's solution> the same as actual solution just calculated:
```
yes or no
```
Student grade:
```
correct or incorrect
```

Question:
```
I'm building a solar power installation and I need help working out the financials.
- Land costs $100 / square foot
- I can buy solar panels for $250 / square foot
- I negotiated a contract for maintenance that will cost me a flat $100k per year, and an additional $10 / square foot
What is the total cost for the first year of operations as a function of the number of square feet.
```
Student's solution:
```
Let x be the size of the installation in square feet..
Costs:
1. Land cost: 100x
2. Solar panel cost: 250x
3. Maintenance cost: 100,000 + 100x
Total cost: 100x + 250x + 100,000 + 100x = 450x + 100,000
```
Actual solution:
```

Iterative Prompt Development:



There is no perfect prompt for everything. Develop your prompt in an iterative way:
- Try something and be clear and specific.
- Analyze why result does not give desired output.
- Refine the idea and the prompt
- Repeat

Deal with Issues:

Output text is too long:

Limit the number of words/sentences/characters

```
Use at most 50 words.
```

Output text focuses on the wrong details:

```
The output/answer/description is intended for <specific_group>, so should be <technical> in nature and focus on <some_aspects>.
```

Output text should be organized in a table:

```
After the description, include a table that gives the <information>. The table should have <two> columns.
In the first column include <first_name>.
In the second column include <grades>.

Give the table the title 'Student_Grades'.

Format everything as <HTML/Markdown> that can be used in a website.
Place the description in a <div> element.
```

Summarize vs. Extract:

- Summaries include topics that my not be related to the topic of focus.

```
Your task is to generate a short summary of a <product_review> from an <ecommerce_site> to give feedback to the <pricing_department>, responsible for determining the price of the product.

Summarize the review below, delimited by triple backticks, in at most 30 words, and focusing on any aspects that are relevant to the price and perceived value.

Review: ```{prod_review}```
```

```
Your task is to extract relevant information from a <product_review> from an <ecommerce_site> to give feedback to the Shipping department.

From the review below, delimited by triple quotes extract the information relevant to <shipping and delivery>. Limit to 30 words.

Review: ```{prod_review}```
```

Inferring:

- Inferring tasks mean that the model takes a text as input and performs some kind of analysis, e.g., extracting labels, extracting names, analyzing sentiment of a text.

```
Identify the following items from the <review text>:
- Sentiment (positive or negative)
- Is the reviewer expressing anger? (true or false)
- Item purchased by reviewer
- Company that made the item

The review is delimited with triple backticks.
Format your response as a <JSON_object> with "Sentiment", "Anger", "Item" and "Brand" as the keys.
If the information isn't present, use "unknown" as the value.
Make your response as short as possible.
Format the Anger value as a boolean.

Review text: ```{review}```
```

```
Determine five topics that are being discussed in the following text, which is delimited by triple backticks.

Make each item one or two words long.

Format your response as a list of items separated by commas.

Text sample: ```{story}```
```

```
Determine whether each item in the following list of topics is a topic in the text below, which is delimited with triple backticks.

Give your answer as list with 0 or 1 for each topic.

List of topics: {", ".join(topic_list)}

Text sample: ```{story}```
```

Temperature:

- Governs the randomness/creativity of the responses.
- Set `temperature` to `0` for building reliable and predictable systems.
- Set `temperature` to `0 - 0.3` for extraction, standardization, format conversion, and grammar fixes tasks.
- Set `temperature` to `0.5` for writing tasks.

OpenAI API call

```
def get_completion(prompt,
                   model="gpt-3.5-turbo"):
messages = [{"role": "user",
             "content": prompt}]

response = openai.ChatCompletion.create(
             model=model,
             messages=messages,
             temperature=0)
```


Role

```
messages =
[
 {"role": "system",
  "content": "You are an assistant… "},
 {"role": "user",
  "content": "tell me a joke "},
 {"role": "assistant",
  "content": "Why did the chicken… "},
 …
]
```