PromptTemplates

Prompt template classes.

pydantic model langchain.prompts.**BaseChatPromptTemplate** [source]

format(***kwargs: Any*) → str [source]

Format the prompt with the inputs.

Parameters:

kwargs – Any arguments to be passed to the prompt template.

Returns:

A formatted string.

Example:

```
prompt.format(variable1="foo")
```

abstract **format_messages**(***kwargs: Any*) → List[langchain.schema.BaseMessage] [source]

Format kwargs into a list of messages.

format_prompt(***kwargs: Any*) → langchain.schema.PromptValue [source]

Create Chat Messages.

pydantic model langchain.prompts.**BasePromptTemplate** [source]

Base class for all prompt templates, returning a prompt.

field **input_variables**: *List[str] [Required]*

A list of the names of the variables the prompt template expects.

field **output_parser**: *Optional[langchain.schema.BaseOutputParser] = None*

How to parse the output of calling an LLM on this formatted prompt.

Skip to main content

Return dictionary representation of prompt.

abstract **format**(***kwargs: Any*) → str [source]

Format the prompt with the inputs.

Parameters:

kwargs – Any arguments to be passed to the prompt template.

Returns:

A formatted string.

Example:

```
prompt.format(variable1="foo")
```

abstract **format_prompt**(***kwargs: Any*) → langchain.schema.PromptValue [source]

Create Chat Messages.

partial(***kwargs: Union[str, Callable[[], str]]*) → langchain.prompts.base.BasePromptTemplate [source]

Return a partial of the prompt template.

save(*file_path: Union[pathlib.Path, str]*) → None [source]

Save the prompt.

Parameters:

file_path – Path to directory to save prompt to.

Example: .. code-block:: python

> prompt.save(file_path=”path/prompt.yaml”)

pydantic model langchain.prompts.**ChatPromptTemplate** [source]

format(***kwargs: Any*) → str [source]

Format the prompt with the inputs.

Skip to main content

kwargs – Any arguments to be passed to the prompt template.

Returns:

A formatted string.

Example:

```
prompt.format(variable1="foo")
```

format_messages(***kwargs: Any*) → List[langchain.schema.BaseMessage] [source]

Format kwargs into a list of messages.

partial(***kwargs: Union[str, Callable[[], str]]*) → langchain.prompts.base.BasePromptTemplate [source]

Return a partial of the prompt template.

save(*file_path: Union[pathlib.Path, str]*) → None [source]

Save the prompt.

Parameters:

file_path – Path to directory to save prompt to.

Example: .. code-block:: python

> prompt.save(file_path="path/prompt.yaml")

pydantic model langchain.prompts.**FewShotPromptTemplate** [source]

Prompt template that contains few shot examples.

field **example_prompt**: *langchain.prompts.prompt.PromptTemplate [Required]*

PromptTemplate used to format an individual example.

field **example_selector**: *Optional[langchain.prompts.example_selector.base.BaseExampleSelector] = None*

Skip to main content

ExampleSelector to choose the examples to format into the prompt. Either this or examples should be provided.

field **example_separator**: *str = '\n\n'*

String separator used to join the prefix, the examples, and suffix.

field **examples**: *Optional[List[dict]] = None*

Examples to format into the prompt. Either this or example_selector should be provided.

field **input_variables**: *List[str] [Required]*

A list of the names of the variables the prompt template expects.

field **prefix**: *str = ''*

A prompt template string to put before the examples.

field **suffix**: *str [Required]*

A prompt template string to put after the examples.

field **template_format**: *str = 'f-string'*

The format of the prompt template. Options are: 'f-string', 'jinja2'.

field **validate_template**: *bool = True*

Whether or not to try validating the template.

dict(***kwargs: Any*) → Dict [source]

Return a dictionary of the prompt.

format(***kwargs: Any*) → str [source]

Format the prompt with the inputs.

Parameters:

kwargs – Any arguments to be passed to the prompt template.

Returns:

A formatted string.

Example:

Skip to main content

```
prompt.format(variable1="foo")
```

pydantic model **langchain.prompts.FewShotPromptWithTemplates** [source]

Prompt template that contains few shot examples.

field **example_prompt**: *langchain.prompts.prompt.PromptTemplate [Required]*

PromptTemplate used to format an individual example.

field **example_selector**: *Optional[langchain.prompts.example_selector.base.BaseExampleSelector] = None*

ExampleSelector to choose the examples to format into the prompt. Either this or examples should be provided.

field **example_separator**: *str = '\n\n'*

String separator used to join the prefix, the examples, and suffix.

field **examples**: *Optional[List[dict]] = None*

Examples to format into the prompt. Either this or example_selector should be provided.

field **input_variables**: *List[str] [Required]*

A list of the names of the variables the prompt template expects.

field **prefix**: *Optional[langchain.prompts.base.StringPromptTemplate] = None*

A PromptTemplate to put before the examples.

field **suffix**: *langchain.prompts.base.StringPromptTemplate [Required]*

A PromptTemplate to put after the examples.

field **template_format**: *str = 'f-string'*

The format of the prompt template. Options are: ‘f-string’, ‘jinja2’.

field **validate_template**: *bool = True*

Skip to main content

dict(***kwargs: Any*) → Dict [source]

Return a dictionary of the prompt.

format(***kwargs: Any*) → str [source]

Format the prompt with the inputs.

Parameters:

kwargs – Any arguments to be passed to the prompt template.

Returns:

A formatted string.

Example:

```
prompt.format(variable1="foo")
```

pydantic model langchain.prompts.**MessagesPlaceholder** [source]

Prompt template that assumes variable is already list of messages.

format_messages(***kwargs: Any*) → List[langchain.schema.BaseMessage] [source]

To a BaseMessage.

property **input_variables**: *List[str]*

Input variables for this prompt template.

langchain.prompts.**Prompt**

alias of `langchain.prompts.prompt.PromptTemplate`

pydantic model langchain.prompts.**PromptTemplate** [source]

Schema to represent a prompt for an LLM.

Example

```
from langchain import PromptTemplate
prompt = PromptTemplate(input_variables=["foo"], template="Say {foo}")
```

field **input_variables**: *List[str] [Required]*

Skip to main content

A list of the names of the variables the prompt template expects.

field **template**: *str [Required]*

The prompt template.

field **template_format**: *str = 'f-string'*

The format of the prompt template. Options are: 'f-string', 'jinja2'.

field **validate_template**: *bool = True*

Whether or not to try validating the template.

format(***kwargs: Any*) → str [source]

Format the prompt with the inputs.

Parameters:

kwargs – Any arguments to be passed to the prompt template.

Returns:

A formatted string.

Example:

```
prompt.format(variable1="foo")
```

classmethod **from_examples**(*examples: List[str], suffix: str, input_variables: List[str], example_separator: str = '\n\n', prefix: str = '', **kwargs: Any*) → langchain.prompts.prompt.PromptTemplate [source]

Take examples in list format with prefix and suffix to create a prompt.

Intended be used as a way to dynamically create a prompt from examples.

Parameters:

- **examples** – List of examples to use in the prompt.
- **suffix** – String to go after the list of examples. Should generally set up the user's input.
- **input_variables** – A list of variable names the final prompt template will expect.

Skip to main content

- **example_separator** – The separator to use in between examples. Defaults to two new line characters.
- **prefix** – String that should go before any examples. Generally includes examples. Default to an empty string.

Returns:

The final prompt generated.

classmethod `from_file`(*template_file: Union[str, pathlib.Path]*, *input_variables: List[str]*, ***kwargs: Any*) → `langchain.prompts.prompt.PromptTemplate` [source]

Load a prompt from a file.

Parameters:

- **template_file** – The path to the file containing the prompt template.
- **input_variables** – A list of variable names the final prompt template will expect.

Returns:

The prompt loaded from the file.

classmethod `from_template`(*template: str*, ***kwargs: Any*) → `langchain.prompts.prompt.PromptTemplate` [source]

Load a prompt template from a template.

pydantic model `langchain.prompts.StringPromptTemplate` [source]

String prompt should expose the format method, returning a prompt.

`format_prompt`(***kwargs: Any*) → `langchain.schema.PromptValue` [source]

Create Chat Messages.

`langchain.prompts.load_prompt`(*path: Union[str, pathlib.Path]*) → `langchain.prompts.base.BasePromptTemplate` [source]

Unified method for loading a prompt from LangChainHub or local fs.