Partial prompt templates

Like other methods, it can make sense to "partial" a prompt template - eg pass in a subset of the required values, as to create a new prompt template which expects only the remaining subset of values.

LangChain supports this in two ways:

1. Partial formatting with string values.
2. Partial formatting with functions that return string values.

These two different ways support different use cases. In the examples below, we go over the motivations for both use cases as well as how to do it in LangChain.

Partial With Strings

One common use case for wanting to partial a prompt template is if you get some of the variables before others. For example, suppose you have a prompt template that requires two variables, `foo` and `baz`. If you get the `foo` value early on in the chain, but the `baz` value later, it can be annoying to wait until you have both variables in the same place to pass them to the prompt template. Instead, you can partial the prompt template with the `foo` value, and then pass the partialed prompt template along and just use that. Below is an example of doing this:

```
from langchain.prompts import PromptTemplate
```

```
prompt = PromptTemplate(template="{foo}{bar}", input_variables=["foo", "bar"])
partial_prompt = prompt.partial(foo="foo");
print(partial_prompt.format(bar="baz"))
```

```
    foobaz
```

You can also just initialize the prompt with the partialed variables.

```
prompt = PromptTemplate(template="{foo}{bar}", input_variables=["bar"], partial_variables={"foo": "foo"})
print(prompt.format(bar="baz"))
```

```
    foobaz
```

Partial With Functions

The other common use is to partial with a function. The use case for this is when you have a variable you know that you always want to fetch in a common way. A prime example of this is with date or time. Imagine you have a prompt which you always want to have the current date. You can't hard code it in the prompt, and passing it along with the other input variables is a bit annoying. In this case, it's very handy to be able to partial the prompt with a function that always returns the current date.

```
from datetime import datetime

def _get_datetime():
```

```
    now = datetime.now()
    return now.strftime("%m/%d/%Y, %H:%M:%S")
```

```
prompt = PromptTemplate(
    template="Tell me a {adjective} joke about the day {date}",
    input_variables=["adjective", "date"]
);
partial_prompt = prompt.partial(date=_get_datetime)
print(partial_prompt.format(adjective="funny"))
```

```
    Tell me a funny joke about the day 02/27/2023, 22:15:16
```

You can also just initialize the prompt with the partialed variables, which often makes more sense in this workflow.

```
prompt = PromptTemplate(
    template="Tell me a {adjective} joke about the day {date}",
    input_variables=["adjective"],
    partial_variables={"date": _get_datetime}
);
print(prompt.format(adjective="funny"))
```

```
    Tell me a funny joke about the day 02/27/2023, 22:15:16
```