Custom LLM

This notebook goes over how to create a custom LLM wrapper, in case you want to use your own LLM or a different wrapper than one that is supported in LangChain.

There is only one required thing that a custom LLM needs to implement:

1. A `_call` method that takes in a string, some optional stop words, and returns a string

There is a second optional thing it can implement:

1. An `_identifying_params` property that is used to help with printing of this class. Should return a dictionary.

Let's implement a very simple custom LLM that just returns the first N characters of the input.

```
from typing import Any, List, Mapping, Optional

from langchain.callbacks.manager import CallbackManagerForLLMRun
from langchain.llms.base import LLM
```

API Reference:

- CallbackManagerForLLMRun from `langchain.callbacks.manager`
- LLM from `langchain.llms.base`

```
class CustomLLM(LLM):
    n: int

    @property
    def _llm_type(self) -> str:
        return "custom"

    def _call(
        self,
        prompt: str,
        stop: Optional[List[str]] = None,
```

```
        run_manager: Optional[CallbackManagerForLLMRun] = None,
    ) -> str:
        if stop is not None:
            raise ValueError("stop kwargs are not permitted.")
        return prompt[: self.n]

    @property
    def _identifying_params(self) -> Mapping[str, Any]:
        """Get the identifying parameters."""
        return {"n": self.n}
```

We can now use this as an any other LLM.

```
llm = CustomLLM(n=10)
```

```
llm("This is a foobar thing")
```

```
    'This is a '
```

We can also print the LLM and see its custom print.

```
print(llm)
```

```
    CustomLLM
    Params: {'n': 10}
```