Different call methods

All classes inherited from `Chain` offer a few ways of running chain logic. The most direct one is by using `__call__`:

```
chat = ChatOpenAI(temperature=0)
prompt_template = "Tell me a {adjective} joke"
llm_chain = LLMChain(llm=chat, prompt=PromptTemplate.from_template(prompt_template))

llm_chain(inputs={"adjective": "corny"})
```

```
{'adjective': 'corny',
 'text': 'Why did the tomato turn red? Because it saw the salad dressing!'}
```

By default, `__call__` returns both the input and output key values. You can configure it to only return output key values by setting `return_only_outputs` to `True`.

```
llm_chain("corny", return_only_outputs=True)
```

```
{'text': 'Why did the tomato turn red? Because it saw the salad dressing!'}
```

If the `Chain` only outputs one output key (i.e. only has one element in its `output_keys`), you can use `run` method. Note that `run` outputs a string instead of a dictionary.

```
# llm_chain only has one output key, so we can use run
llm_chain.output_keys
```

```
['text']
```

```
llm_chain.run({"adjective": "corny"})
```

```
'Why did the tomato turn red? Because it saw the salad dressing!'
```

In the case of one input key, you can input the string directly without specifying the input mapping.

```
# These two are equivalent
llm_chain.run({"adjective": "corny"})
llm_chain.run("corny")

# These two are also equivalent
llm_chain("corny")
llm_chain({"adjective": "corny"})
```

```
{'adjective': 'corny',
 'text': 'Why did the tomato turn red? Because it saw the salad dressing!'}
```

Tips: You can easily integrate a `Chain` object as a `Tool` in your `Agent` via its `run` method. See an example here.