Logging to file

This example shows how to print logs to file. It shows how to use the `FileCallbackHandler`, which does the same thing as `StdOutCallbackHandler`, but instead writes the output to file. It also uses the `loguru` library to log other outputs that are not captured by the handler.

```
from langchain.callbacks import
FileCallbackHandler
from langchain.chains import LLMChain
from langchain.prompts import PromptTemplate
from langchain_openai import OpenAI
from loguru import logger

logfile = "output.log"

logger.add(logfile, colorize=True,
enqueue=True)
handler = FileCallbackHandler(logfile)

llm = OpenAI()
prompt = PromptTemplate.from_template("1 +
{number} = ")

# this chain will both print to stdout
```

```
(because verbose=True) and write to
'output.log'
# if verbose=False, the FileCallbackHandler
will still write to 'output.log'
chain = LLMChain(llm=llm, prompt=prompt,
callbacks=[handler], verbose=True)
answer = chain.run(number=2)
logger.info(answer)
```

```
> Entering new LLMChain chain...
Prompt after formatting:
1 + 2 =

> Finished chain.
```

```
2023-06-01 18:36:38.929 | INFO     |
__main__:<module>:20 -

3
```

Now we can open the file `output.log` to see that the output has been captured.

```
%pip install --upgrade --quiet  ansi2html >
```

```
/dev/null
```

```
from ansi2html import Ansi2HTMLConverter
from IPython.display import HTML, display

with open("output.log", "r") as f:
    content = f.read()

conv = Ansi2HTMLConverter()
html = conv.convert(content, full=True)

display(HTML(html))
```

```
> Entering new LLMChain chain...
Prompt after formatting:
1 + 2 =
> Finished chain.
2023-06-01 18:36:38.929 | INFO     | __main__:<module>:20 -
3
```