

# Final Project Exercises

**Note:** These exercises are designed to give you more hands-on about langchain. In these exercises you will be exploring langchain agents, tools and how we can implement conversational memory for LLMs so that it is able to respond on our previous queries. Make changes to the code that has been provided to you by following this

exercise book.

# **Table of Contents**

Exercise # 01:	
Exploring different langchain agent tools	
Exercise # 02:	10
Adding Conversation Memory in Basic Chatbot	10

### Exercise # 01:

## **Exploring different langchain agent tools**

Agents are one of the most powerful and fascinating approaches to using Large Language Models (LLMs). The explosion of interest in LLMs has made agents incredibly prevalent in Al-powered use cases. Using agents allows us to give LLMs access to tools. These tools present an infinite number of possibilities. With tools, LLMs can search the web, do math, run code, and more.

In the code that has been provided to you we have used the **duckduckgosearch** tool through which you can query about recent events. In this exercise, you will be exploring the following tools:

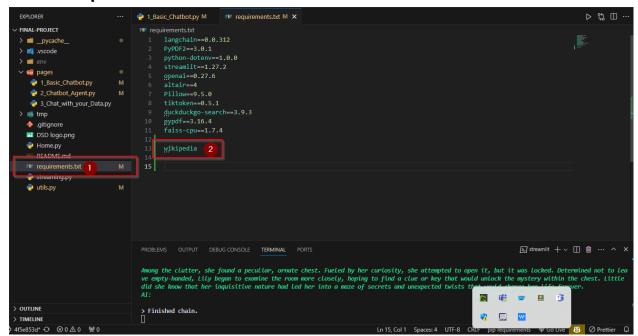
- Wikipedia: Ask about a specific topic, person, or event.
- Python Repl: Ask python codes, functions etc.

You can explore more tools from:

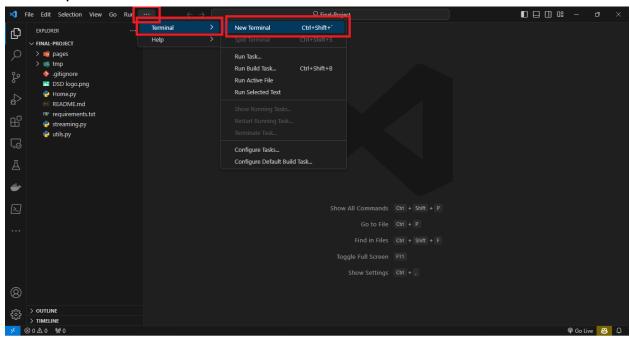
https://python.langchain.com/docs/integrations/tools

Follow these steps to add tools to the **Chatbot Agent**:

 Go to requirements.txt in the Final-Project folder and add wikipedia to it.

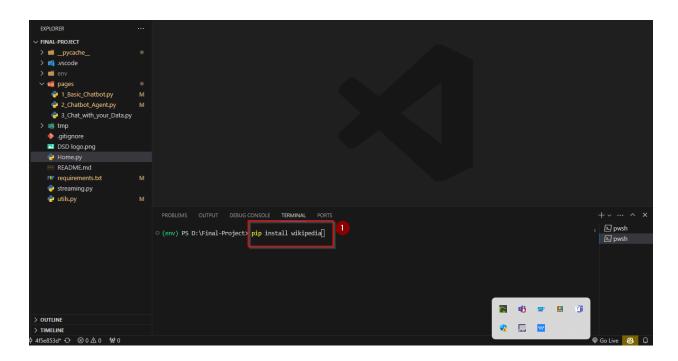


#### 2. Open terminal in Visual Studio Code



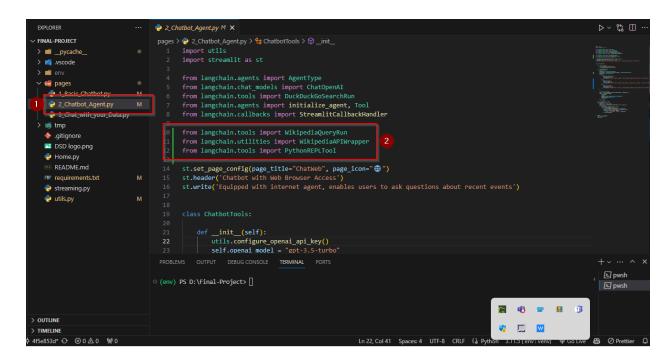
3. Run the following command in the terminal. You can **copy** the **command** from below.

#### pip install wikipedia



4. Now, goto **pages** folder, open **2\_Chatbot\_Agent.py** and add the following imports in it. You can **copy** the **code** from below.

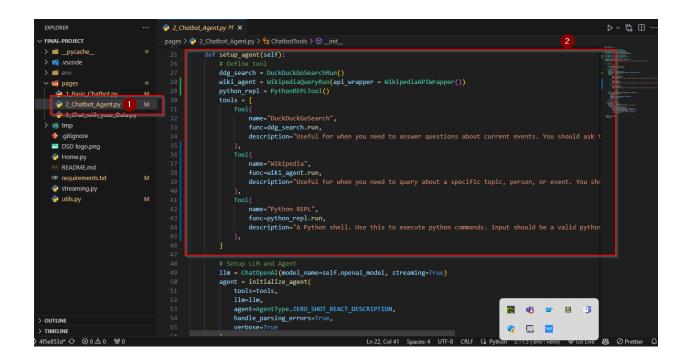
```
from langchain.tools import WikipediaQueryRun
from langchain.utilities import WikipediaAPIWrapper
from langchain.tools import PythonREPLTool
```



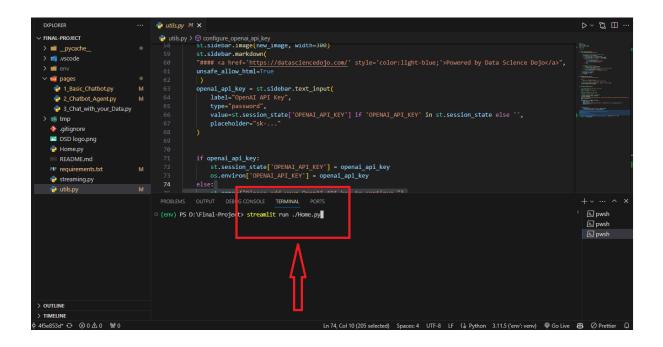
5. Now, change the code of **setup\_agent** function in the same file with the following code. You can **copy** the **code** from below.

```
def setup agent(self):
       ddg search = DuckDuckGoSearchRun()
        wiki agent = WikipediaQueryRun(api wrapper =
WikipediaAPIWrapper())
       python repl = PythonREPLTool()
                func=ddg search.run,
                description="Useful for when you need to answer
           ),
                func=wiki agent.run,
                description="Useful for when you need to query about
           Tool(
                name="Python REPL",
                func=python repl.run,
                description="A Python shell. Use this to execute
python commands. Input should be a valid python command. If you
        llm = ChatOpenAI(model name=self.openai model,
streaming=True)
        agent = initialize agent(
            tools=tools,
            llm=llm,
            agent=AgentType.ZERO SHOT REACT DESCRIPTION,
```

```
handle_parsing_errors=True,
    verbose=True
)
return agent
```



# 6. Now, run the code. Open **Terminal** and write **streamlit run** ./**Home.py**



### Exercise # 02:

## **Adding Conversation Memory in Basic Chatbot**

Conversation memory in the context of chatbots refers to the ability of the system to retain and utilize information from past interactions. It enables the chatbot to have a more context-aware and coherent conversation with the user. After completing this exercise you would witness that the **Basic Chatbot** would be able to respond on your previous responses

Follow these steps to add the memory in the **Basic Chatbot**:

7. Open the **Final-Project** folder from the **desktop** with **Visual Studio Code** and go to utils.py and add the following code to the end of it.

You can **copy** the **code** from below.

```
def join_messages(role_list, role):
    messages = [message["content"] for message in role_list if
message["role"] == role]
    return " ".join(messages)
```

```
⊳ ∨ ቈ Ш
                                     🗬 utils.py M 🗙
                                                 st.sidebar.markdown(
                                                  "#### <a href='https://datasciencedojo.com/' style='color:light-blue;'>Powered by Data Science Dojo</a>",
     1_Basic_Chatbot.py
                                                 openai_api_key = st.sidebar.text_input(
label="OpenAI API Key",
type="password",
     2 Chatbot_Agent.py
     🍦 3_Chat_with_your_Data.py
                                                     value=st.session state['OPENAI API KEY'] if 'OPENAI API KEY' in st.session state else '',
   M DSD logo.png
    Home.py
                                                 if openai_api_key:
      README.md
                                                     os.environ['OPENAI_API_KEY'] = openai_api_key
                                                 st.stop()
return openai_api_key
                                             def join_messages(role_list, role):
                                                 messages = [message["content"] for message in role_list if message["role"] == role]
return " ".join(messages)
                                                                                                                                             🎅 🔥 👱 📙 🧃
> OUTLINE
                                                                                                                                                  .... W
 f5e853d* ↔ ⊗ 0 🛦 0 🐪 0
                                                                                                    Ln 81, Col 30 Spaces: 4 UTF-8 LF ( } P
```

8. Now, goto **pages** folder and open **1\_Basic\_Chatbot.py** and add the following imports in it. You can **copy** the **code** from below.

from langchain.memory import ConversationBufferMemory

```
🕏 utils.py M 💝 2_Chatbot_Agent.py M
                                                                               2 1_Basic_Chatbot.py M X 3_Chat_with_your_Data.py
FINAL-PROJECT
                                         pages > 🜓 1 Basic Chatbot.py > ...
                                                  import utils
import streamlit as st
from streaming import StreamHandler
> = _pycache_
                                                  from langchain.llms import OpenAI from langchain.chains import ConversationChain
                                               from langchain.memory import ConversationBufferMemory
    3_Chat_with_your_Data.py
                                                st.set_page_config(page_title="Chatbot", page_icon="♥")
st.header('Basic Chatbot')
st.write(' Allow users to interact with the OpenAI LLMs ')
   .gitignore

    DSD logo.png

     ■ README.md
   requirements.txt
   streaming.py
                                                     def __init__(self):
    utils.configure_openai_api_key()
    self.openai_model = 'gpt-3.5-turbo
                                                          # Setup memory for contextual conversation
llm = OpenAI(model_name=self.openai_model,
                                                          return chain
OUTLINE
```

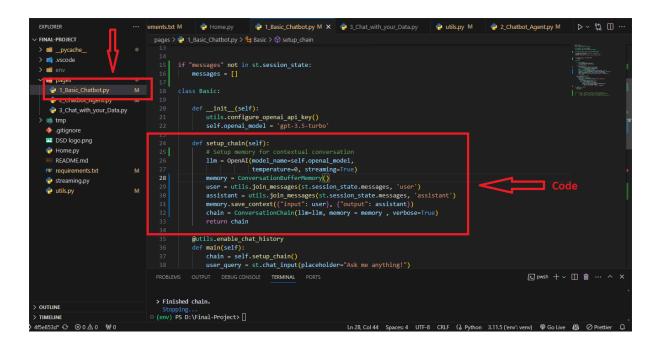
9. Now, add the following code in the same file before the **Basic Class** constructor begins. You can **copy** the **code** from below.

```
if "messages" not in st.session_state:
    st.session_state["messages"] = []
```

```
pages > 🌏 1_Basic_Chatbot.py > ..
> ii _pycache
                                                                   from langchain.chains import ConversationChain
                                                                  from langchain.memory import ConversationBufferMemory
    1_Basic_Chatbot.py
                                                                   st.set page config(page title="Chatbot", page icon=" ")
    3_Chat_with_your_Data.py
                                                                  st.header('Basic Chatbot')
st.write(' Allow users to interact with the OpenAI LLMs ')
   DSD logo.png
                                                         if "messages" not in st.session_state:

st.session_state["messages"] = []
   e Home.py
      README.md
    requirements.txt
   streaming.py
                                                                        def __init__(self):
    utils.configure_openai_api_key()
    self.openai_model = 'gpt-3.5-turbo'
                                                                            llm = OpenAI(model name=self.openai model.
                                                                                                                                                                                                                                         ∑ streamlit + ∨ □ 🛍 ··· ^ ×
                                                          File "D:\Final-Project\env\Lib\site-packages\openai\api requestor.py", line 687, in interpret response line
                                                        File "D:\Final-Project\env\Lib\site-package\text{openal\api}_requestor.py", line 687, in _interpret_response_line naise self-handle_error_response_Companal_response_Line (openal_error_APIError: Bad gateway. "grams": None, 'type': 'cf_bad_gateway')} (*Obate': Fri, 20 Oct 2023 02:07:21 GMT, 'Content-Type': 'application/jso n, 'Content-Length': '84', 'Connection': 'keep-allive', 'K-Frame-Options': 'SAMEGRIGIN', 'Referrer-Policy': 'same-origin', 'Cache-Control': 'priva te, max-age-0, no-store, no-cache, must-revallidate, post-check-0, 'Expires': 'Thu, 01 Jan 1970 00:00:01 GMT', 'Server': 'cloudflare', 'CF-RAY': '818054ebla2e28f4-RMI', 'alt-svc': 'h3=":443"; ma=66400')
                                                                                                                                                    Ln 50, Col 14 Spaces: 4 UTF-8 CRLF () Python 3.11.5 ('env': venv) @ Go Live 🔠 🕢 Pro
```

 Now, change the code of setup\_chain function in the same file with the following code. You can copy the code from below.



# 11. Now, run the code. Open **Terminal** and write **streamlit run** ./**Home.py**

