

Assignment 3

Assignment 3

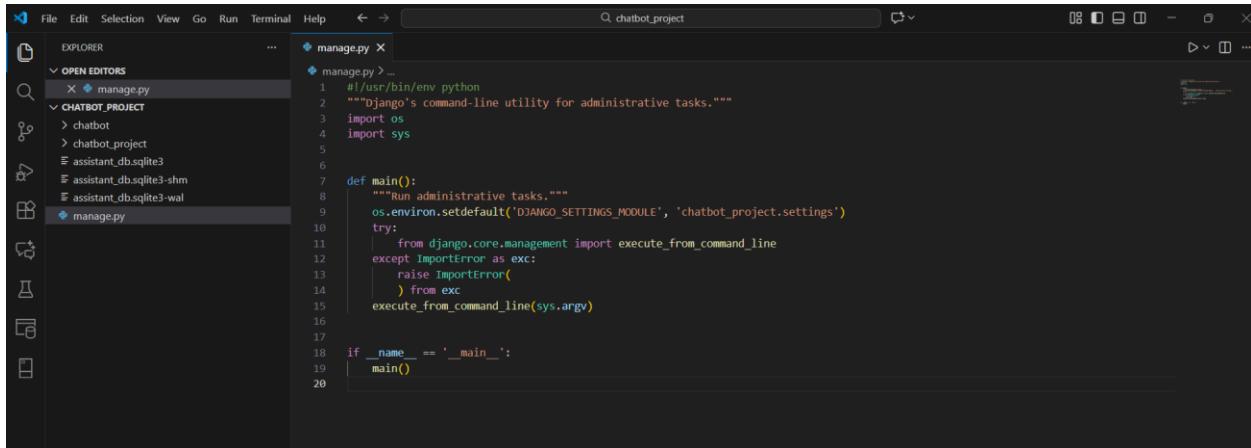
Prashanna Acharya

Advance Artificial Intelligence (MSCS-633-B01)

University of the Cumberlands

Dr. Ulrich Vouama

manage.py

A screenshot of a code editor window titled "manage.py". The window shows the Python code for the manage.py script. The code defines a main() function that sets the DJANGO_SETTINGS_MODULE environment variable to 'chatbot_project.settings', imports execute_from_command_line from django.core.management, and then calls it with sys.argv. It also includes a check for __name__ == '__main__' to handle direct execution. The code editor has a dark theme with syntax highlighting for Python keywords and comments.

The manage.py file is a script that Django automatically generates when a new project is created. Its role is to provide a command-line tool for controlling and working with the Django application. When executed, it sets the appropriate environment variable, so Django knows which settings configuration to load. After that, it brings in Django's command-handling machinery and passes control to it. This setup enables to run various project-related commands, such as launching the development server, managing database migrations, creating apps, or executing custom commands. Essentially, manage.py acts as the main gateway for performing administrative operations in a Django project and ensures everything is properly configured before those tasks run.

chatclient.py

The screenshot shows a code editor interface with the following details:

- File Explorer:** On the left, it lists the project structure under "CHATBOT_PROJECT". The "management" directory contains "commands", "chatclient.py", and "manage.py".
- Editor Area:** The main area displays the content of "chatclient.py".
- Search Bar:** At the top right, there is a search bar with the text "chatbot_project".
- Code Content:**

```
chatbot > management > commands > chatclient.py > Command > handle
1 import sys
2 from django.core.management.base import BaseCommand
3 from chatterbot import ChatBot
4 from chatterbot.trainers import ChatterBotCorpusTrainer
5
6
7 class Command(BaseCommand):
8     help = 'Launching an interactive terminal conversation with the AI assistant'
9
10    def handle(self, *args, **options):
11        # Creating an AI assistant instance with database and logic adapters
12        ai_assistant = ChatBot(
13            'TerminalAssistant',
14            storage_adapter='chatterbot.storage.SQLStorageAdapter',
15            logic_adapters=[
16                'chatterbot.logic.BestMatch',           # Handling general conversation
17                'chatterbot.logic.TimeLogicAdapter',    # Responding with current time
18                'chatterbot.logic.MathematicalEvaluation' # Solving mathematical expressions
19            ],
20            database_uri='sqlite:///assistant_db.sqlite3'
21        )
22
23
24        # Training the AI assistant on English language corpus
25        trainer = ChatterBotCorpusTrainer(ai_assistant)
26        trainer.train('chatterbot.corpus.english')
27
28        # Displaying startup message and informing how to exit
29        self.stdout.write(self.style.SUCCESS(
30            'Starting conversation! (Typing "exit" or using Ctrl-C to quit)\n'
31        ))
32
33
34        # Running conversation loop continuously
35        while True:
36            try:
37                # Asking user for input continuously
38                user_message = input('You: ').strip()
39
39                # Skipping if the user is entering nothing
40                if not user_message:
41                    continue
42
43
44                # Checking if the user is typing exit command
45                if user_message.lower() == 'exit':
```

The `chatclient.py` file is implemented as a Django management command that is intended to operate a ChatterBot-driven chatbot within the terminal. A Command class is defined by extending Django's `BaseCommand`, allowing the script to be detected and executed through the command `python manage.py chatclient`. Inside the command, a `ChatBot` instance is configured with a storage adapter, multiple logic adapters, and an SQLite database in which conversational data is stored. Training is then carried out using ChatterBot's English corpus so that a basic level of conversational ability is provided. Once training has been completed, an interactive loop is initiated, through which user input is continuously received and responses from the chatbot are generated in real time. The session is maintained until the user types "exit" or interrupts the program. Through this design, the Django project is enabled to function as a terminal-based chatbot system controlled by a custom management command.

Output on Terminal

File Edit Selection View Go Run Terminal Help ↺ ↻ chatbot_project

EXPLORER
OPEN EDITORS
CHATBOT_PROJECT
CHAT

```
~~~~~  
desc='ChatterBot Corpus Trainer',  
~~~~~  
disable=self.disable_progress  
~~~~~  
):  
File "C:\Users\prash\AppData\Local\Programs\Python\Python313\Lib\site-packages\tqdm\std.py", line 1181, in __iter__  
for obj in iterable:  
~~~~~  
File "C:\Users\prash\AppData\Local\Programs\Python\Python313\Lib\site-packages\chatterbot\corpus.py", line 84, in load_corpus  
corpus_data = read_corpus(file_path)  
File "C:\Users\prash\AppData\Local\Programs\Python\Python313\Lib\site-packages\chatterbot\corpus.py", line 56, in read_corpus  
raise OptionalDependencyImportError(message)  
chatterbot.exceptions.OptionalDependencyImportError: Unable to import "yaml".  
Please install "yaml" to enable chatterbot corpus functionality:  
pip install yaml  
PS C:\Users\prash\Desktop\Masters\Fall 2025\Advanced AI\chatbot_project> pip install PyYAML  
>>>  
Collecting PyYAML  
  Downloading pyyaml-6.0.3-cp313-cp313-win_amd64.whl.metadata (2.4 kB)  
  Downloading pyyaml-6.0.3-cp313-cp313-win_amd64.whl (154 kB)  
Installing collected packages: PyYAML  
Successfully installed pyyaml-6.0.3  
PS C:\Users\prash\Desktop\Masters\Fall 2025\Advanced AI\chatbot_project> python manage.py chatclient  
>>>  
ChatterBot Corpus Trainer: 19it [00:01, 10.21it/s]  
Starting conversation! (Typing "exit" or using Ctrl-C to quit)  
  
You: Hello  
Assistant: Hi  
You: How are you?  
Assistant: I am doing well.  
You: What is 2+@  
Assistant: h is to o as o is to v.  
You: Sorry I meant 2+2  
Assistant: 2 + 2 = 4  
You: Okay, can you tell me the time right now?  
Assistant: Yes, that is correct.  
You: Ok, you clearly cannot  
Assistant: What's up?  
You: What is your name?  
Assistant: I am still young by your standards.  
You: Nevermind, what is 3+3  
Assistant: 3 + 3 = 6  
You: Ok, that is all I need for this project  
Assistant: What are you working on?  
You: |
```

Build with agent mode
Let's get started
Add context (#), extensions (@), commands

Build Workspace Show Config

AI responses may be inaccurate.

Ln 57, Col 78 Spaces: 4 UTF-8 CRLF ⌂ Python ⌂ Finish Setup 3.13.1 3:57 PM 11/23/2025

66°F Sunny