For developing project will use Python in Pycharm.

First using anaconda shell will create an virtual env named “tele-venv”.

And will install the necessary libraries.

**Step 2: Install Libraries**

1. Install core libraries:

bash

Copy code

pip install python-telegram-bot beautifulsoup4 requests openai pandas python-dotenv

1. If using MongoDB:

bash

Copy code

pip install pymongo

1. If using Firebase:

bash

Copy code

pip install firebase-admin

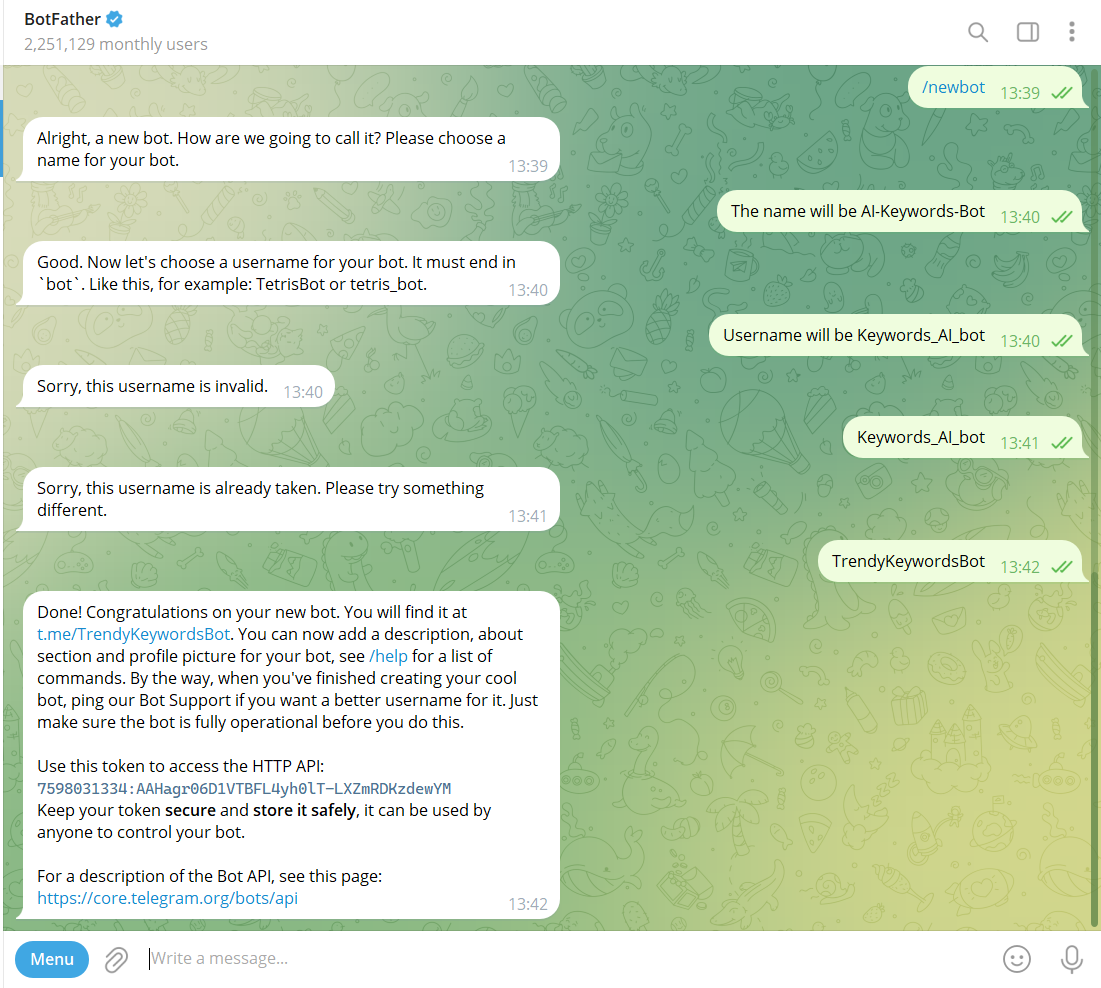
1. Install additional dependencies for development:

bash

Copy code

pip install pylint black

After setting the libraries for python now, we need to chat to @BotFather in telegram and set the following things-



Now –

**Create a Basic Bot**

**Step 1: Start Script**

1. Create a new Python file (bot.py) in PyCharm.
2. Add the following code to test bot functionality:

python

Copy code

from telegram import Update

from telegram.ext import Updater, CommandHandler, CallbackContext

def start(update: Update, context: CallbackContext) -> None:

update.message.reply\_text('Hello! I am your AI-powered business assistant bot.')

def main():

updater = Updater("YOUR\_BOT\_API\_KEY")

dispatcher = updater.dispatcher

dispatcher.add\_handler(CommandHandler("start", start))

updater.start\_polling()

updater.idle()

if \_\_name\_\_ == "\_\_main\_\_":

main()

1. Replace YOUR\_BOT\_API\_KEY with the API key from BotFather.

**Step 2: Run the Bot**

1. Activate the environment in the terminal:

bash

Copy code

conda activate tele-venv

1. Run the bot:

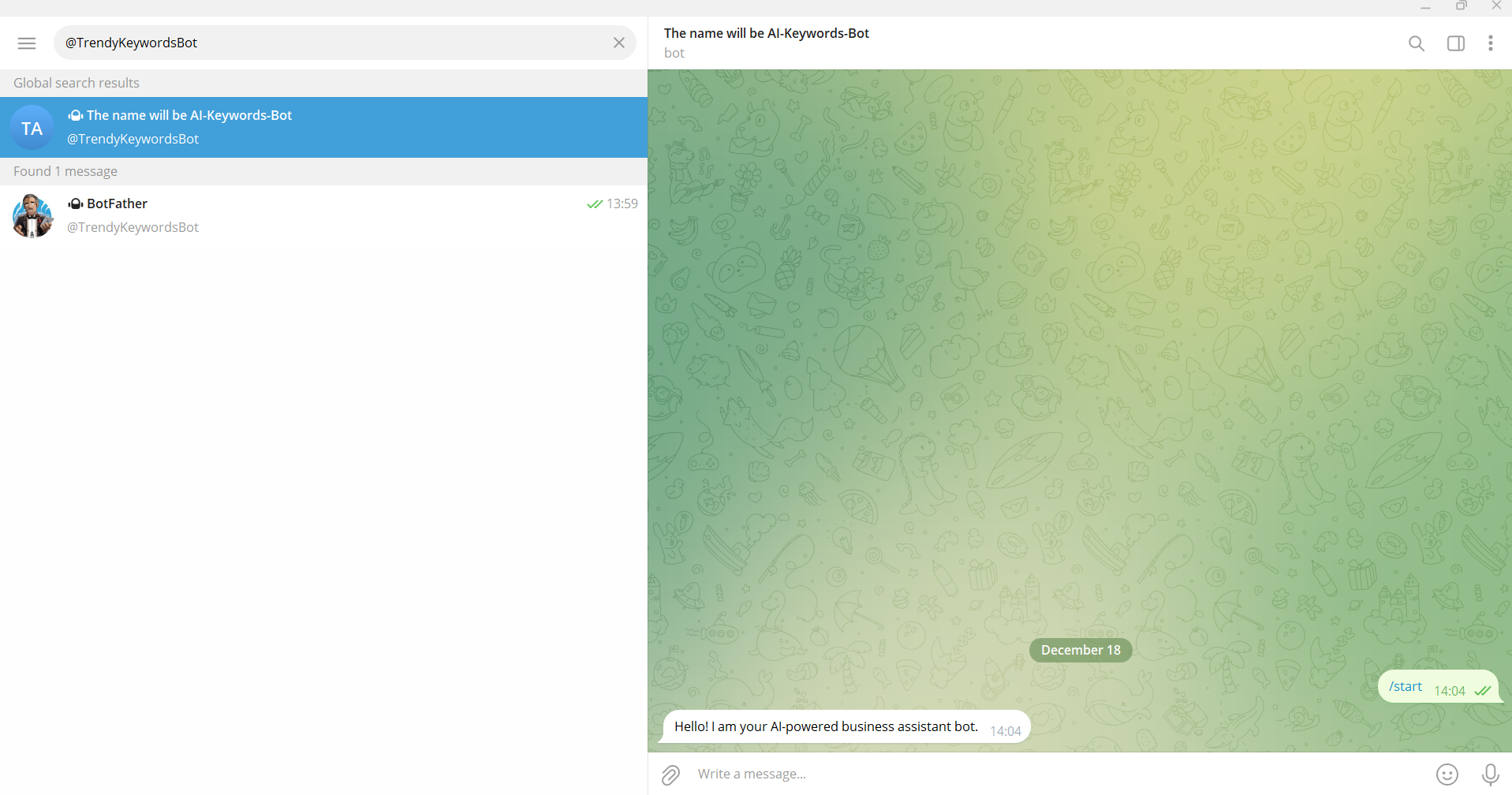
bash

Copy code

python bot.py

1. Test it by typing /start in the bot chat on Telegram.

Now then we have pasted our initial token and in the activated venv in the terminal – type – python bot.py



After searching the name in the telegram , we have successfully developed our telegram bot.

from telegram import Update  
from telegram.ext import (  
 Application,  
 CommandHandler,  
 MessageHandler,  
 ConversationHandler,  
 filters,  
 CallbackContext,  
)  
import os  
from dotenv import load\_dotenv  
import openai  
import requests  
from bs4 import BeautifulSoup  
  
# Load API keys from .env file  
load\_dotenv()  
TELEGRAM\_BOT\_API\_KEY = os.getenv("TELEGRAM\_BOT\_API\_KEY")  
OPENAI\_API\_KEY = os.getenv("OPENAI\_API\_KEY")  
  
# Configure OpenAI  
openai.api\_key = OPENAI\_API\_KEY  
  
# Conversation states  
INDUSTRY, OBJECTIVE, WEBSITE, SOCIAL\_MEDIA, PPC, AUDIENCE, LOCATION = range(7)  
  
# Send welcome message directly when the bot starts  
async def send\_welcome\_message(application):  
 updates = await application.bot.get\_updates() # Await the updates coroutine  
 for update in updates:  
 await application.bot.send\_message(  
 chat\_id=update.message.chat.id,  
 text=(  
 "Welcome to the Business Assistant Bot! I can help you with finding trendy keywords based on the answers you give me. "  
 "Be ready, I will ask you questions below one by one.\n\n"  
 "Here are the commands you can use:\n"  
 "- To start, use: /start\n"  
 "- To know the trends in PPC and more: /trends\n"  
 "- To ask business-related questions: /faq\n\n"  
 "Let’s get started!"  
 )  
 )  
  
# Start command  
async def start(update: Update, context) -> int:  
 await update.message.reply\_text("What industry is your business in?")  
 return INDUSTRY  
  
# Collect industry input  
async def industry(update: Update, context) -> int:  
 context.user\_data['industry'] = update.message.text  
 await update.message.reply\_text("What is your business objective (e.g., lead generation, sales)?")  
 return OBJECTIVE  
  
# Collect objective input  
async def objective(update: Update, context) -> int:  
 context.user\_data['objective'] = update.message.text  
 await update.message.reply\_text("Do you have a website? If yes, please share the URL.")  
 return WEBSITE  
  
# Collect website input  
async def website(update: Update, context) -> int:  
 context.user\_data['website'] = update.message.text  
 await update.message.reply\_text("Do you have any social media platforms? If yes, please share the URL(s).")  
 return SOCIAL\_MEDIA  
  
# Collect social media input  
async def social\_media(update: Update, context) -> int:  
 context.user\_data['social\_media'] = update.message.text  
 await update.message.reply\_text("Do you use PPC campaigns? (yes/no)")  
 return PPC  
  
# Collect PPC input  
async def ppc(update: Update, context) -> int:  
 context.user\_data['ppc'] = update.message.text  
 await update.message.reply\_text("Who are you trying to reach? (e.g., young adults, professionals, etc.)")  
 return AUDIENCE  
  
# Collect audience input  
async def audience(update: Update, context) -> int:  
 context.user\_data['audience'] = update.message.text  
 await update.message.reply\_text("What location(s) would you like to target?")  
 return LOCATION  
  
# Collect location input and generate keywords  
async def location(update: Update, context) -> int:  
 context.user\_data['location'] = update.message.text  
  
 # Collect all user inputs  
 industry = context.user\_data['industry']  
 objective = context.user\_data['objective']  
 website = context.user\_data.get('website', "No website provided")  
 social\_media = context.user\_data.get('social\_media', "No social media provided")  
 ppc = context.user\_data.get('ppc', "No PPC campaigns")  
 audience = context.user\_data['audience']  
 location = context.user\_data['location']  
  
 # Generate keywords using GPT  
 prompt = (  
 f"Generate a list of trending and relevant keywords for a {industry} business with the following details:\n"  
 f"- Objective: {objective}\n"  
 f"- Website: {website}\n"  
 f"- Social Media: {social\_media}\n"  
 f"- PPC Campaigns: {ppc}\n"  
 f"- Target Audience: {audience}\n"  
 f"- Target Location(s): {location}"  
 )  
  
 try:  
 response = openai.ChatCompletion.create(  
 model="gpt-3.5-turbo",  
 messages=[  
 {"role": "system", "content": "You are an expert in digital marketing."},  
 {"role": "user", "content": prompt}  
 ],  
 max\_tokens=200  
 )  
 keywords = response.choices[0].message['content'].strip()  
 await update.message.reply\_text(f"Here are the trending keywords for your business:\n{keywords}")  
 except Exception as e:  
 await update.message.reply\_text("Sorry, I couldn't generate keywords. Please try again later.")  
 print(f"Error: {e}")  
  
 return ConversationHandler.END  
  
# Fetch PPC data  
async def fetch\_ppc\_data():  
 url = "https://databox.com/ppc-industry-benchmarks"  
 response = requests.get(url)  
 soup = BeautifulSoup(response.text, 'html.parser')  
 # Example: Simplify this based on actual HTML structure  
 data = soup.find\_all("table")  
 return "Parsed PPC data (example)" # Replace with actual parsed data  
  
# Send PPC trends  
async def trends(update: Update, context: CallbackContext):  
 try:  
 data = await fetch\_ppc\_data()  
 await update.message.reply\_text(f"Here are the latest PPC trends:\n{data}")  
 except Exception as e:  
 await update.message.reply\_text("Sorry, I couldn't fetch the trends at the moment.")  
 print(f"Error: {e}")  
  
# Handle FAQ command  
async def faq(update: Update, context: CallbackContext):  
 user\_question = update.message.text.replace("/faq", "").strip()  
 if not user\_question:  
 await update.message.reply\_text("Please ask a question after the /faq command, like this:\n/faq How do I improve my ad performance?")  
 return  
  
 try:  
 response = openai.ChatCompletion.create(  
 model="gpt-3.5-turbo",  
 messages=[  
 {"role": "system", "content": "You are a digital marketing assistant."},  
 {"role": "user", "content": user\_question}  
 ],  
 max\_tokens=150,  
 temperature=0.7  
 )  
 ai\_response = response.choices[0].message['content'].strip()  
 await update.message.reply\_text(ai\_response)  
 except Exception as e:  
 await update.message.reply\_text("Sorry, I couldn't process your question. Please try again later.")  
 print(f"Error: {e}")  
  
# Cancel conversation  
async def cancel(update: Update, context) -> int:  
 await update.message.reply\_text("Operation cancelled.")  
 return ConversationHandler.END  
  
# Main function  
def main():  
 # Create application  
 application = Application.builder().token(TELEGRAM\_BOT\_API\_KEY).build()  
  
 # Send welcome message as the bot starts  
 application.bot.send\_message(  
 chat\_id="@AI-Keywords-Bot", # Set your default chat ID here  
 text="Welcome to the Business Assistant Bot! Let's get started!"  
 )  
  
 # Conversation handler for generating keywords  
 conv\_handler = ConversationHandler(  
 entry\_points=[CommandHandler("start", start)],  
 states={  
 INDUSTRY: [MessageHandler(filters.TEXT & ~filters.COMMAND, industry)],  
 OBJECTIVE: [MessageHandler(filters.TEXT & ~filters.COMMAND, objective)],  
 WEBSITE: [MessageHandler(filters.TEXT & ~filters.COMMAND, website)],  
 SOCIAL\_MEDIA: [MessageHandler(filters.TEXT & ~filters.COMMAND, social\_media)],  
 PPC: [MessageHandler(filters.TEXT & ~filters.COMMAND, ppc)],  
 AUDIENCE: [MessageHandler(filters.TEXT & ~filters.COMMAND, audience)],  
 LOCATION: [MessageHandler(filters.TEXT & ~filters.COMMAND, location)],  
 },  
 fallbacks=[CommandHandler("cancel", cancel)],  
 )  
  
 # Add handlers  
 application.add\_handler(CommandHandler("trends", trends)) # Add trends handler  
 application.add\_handler(CommandHandler("faq", faq)) # Add FAQ handler  
 application.add\_handler(conv\_handler)  
  
 # Run the bot  
 application.run\_polling()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

now my code is updated- use /start, /trends, and /faq to interact with the bot.

now for proper deployment will use the Heroku cli –

[**Get Started with the Heroku CLI**](https://devcenter.heroku.com/articles/heroku-cli#get-started-with-the-heroku-cli)

After you install the CLI, run the heroku login command.

$ heroku login

heroku: Press any key to open up the browser to login or q to exit

› Warning: If browser does not open, visit

› https://cli-auth.heroku.com/auth/browser/\*\*\*

heroku: Waiting for login...

Logging in... done

Logged in as me@example.com

If you prefer to stay in the CLI to enter your credentials, run heroku login -i.

​​You can’t use the -i option if you have [**multi-factor authentication**](https://devcenter.heroku.com/articles/multi-factor-authentication#log-in-with-mfa) enabled due to a technical dependency on web browsers for verification.

$ heroku login -i

heroku: Enter your login credentials

Email: me@example.com

Password: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Two-factor code: \*\*\*\*\*\*\*\*

Logged in as me@heroku.com

The CLI saves your email address and an API token to ~/.netrc for future use. For more information, see [Heroku CLI Authentication](https://devcenter.heroku.com/articles/authentication).

Now you’re ready to create your first Heroku app.

$ cd ~/myapp

$ heroku create

Creating app... done, ⬢ sleepy-meadow-81798

https://sleepy-meadow-81798.herokuapp.com/ | https://git.heroku.com/sleepy-meadow-81798.git

Check out your preferred language’s [getting started guide](https://devcenter.heroku.com/start) for a comprehensive introduction to deploying your first app. For a full list of commands, see [Heroku CLI Commands](https://devcenter.heroku.com/articles/heroku-cli-commands).