

**Create a Simple Chatbot using a Traditional Approach**

Alan L. Dennis

University of the Cumberland

## **Create a Simple Chatbot using Traditional Approaches**

Chatbots are a pervasive way that humans interact with computers. In this document, I present the requirements for the fifth topic project assignment and walk through the starting point. I discuss the general instructions, Microsoft Bot Framework, setting up the Anaconda environment, and then changing the sample. Lastly, I present the conclusion.

### **General Instructions**

After reviewing the lecture, use this document to create a simple Chatbot. Ensure the source code is stored or exported to a GitHub repository and that your instructor has access to that repository (and knows its name). Additionally, submit your source code in an archive file (ZIP) in the course. Document the process in a report. The report should be 5-7 pages of content and should include a title and references page using APA format. Document any challenges you met and mitigation strategies you utilized.

The chatbot should respond to multiple prompts, provide a list of its capabilities, and handle malformed questions or input. You are not required to use the framework presented here. However, your bot must support simple interactions and be sufficient for extension (connecting to an AI as a service offering, such as Azure's Cognitive Services). Ensure that you document your progress with screenshots and a narrative.

When you submit your paper, ensure you include an archive (ZIP file) of your solution, along with the URL of your GitHub repository.

### **Microsoft Bot Framework**

The Microsoft Bot Framework (MBF) is an open-source platform for building bots. MBF includes a set of software development kits (SDKs) and tools to help build and test bots (Microsoft, n.d.). The framework supports development in Python, JavaScript, and C#. For this

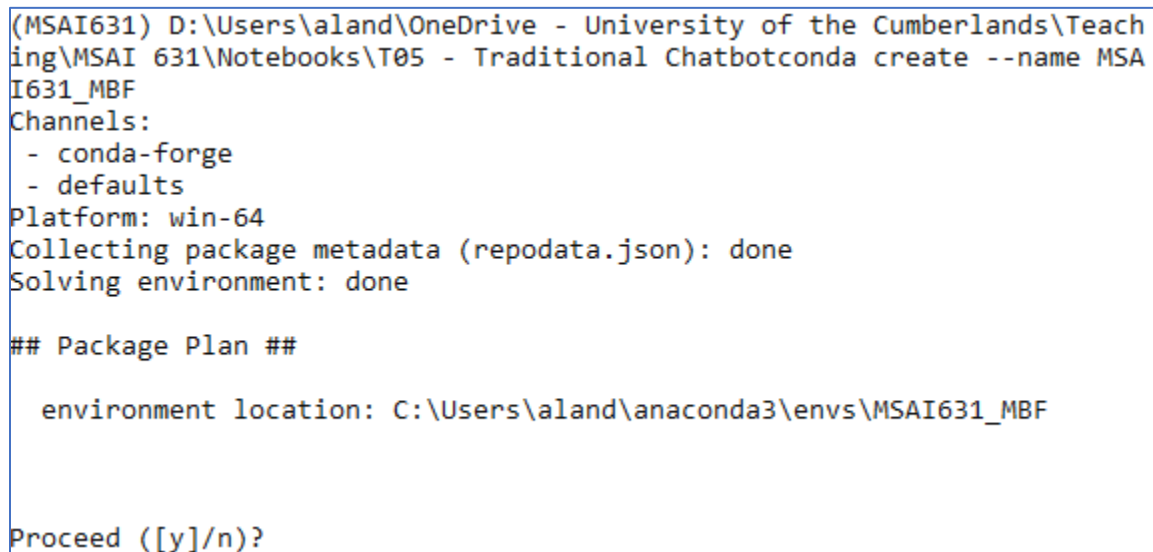
example, we will use Python (in keeping with the other elements of the course).

### Anaconda Environment

It is often helpful to have different environments for different workloads. The first step is to create an Anaconda environment. Anaconda's documentation on environments includes creation and activation (Anaconda, 2017). Ensure you specify Python 3.8.2 during the environment creation. Otherwise, you will encounter numerous errors. To do this, we use the command:

```
conda create --name MSAI631_MBF python==3.8.2
```

This creates an environment named MSAI631\_MBF. You can see the results of executing the command in *Figure 1*.



```
(MSAI631) D:\Users\aland\OneDrive - University of the Cumberland\Teaching\MSAI 631\Notebooks\T05 - Traditional Chatbotconda create --name MSAI631_MBF
Channels:
 - conda-forge
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\aland\anaconda3\envs\MSAI631_MBF

Proceed ([y]/n)?
```

*Figure 1: Create environment*

Entering y at the prompt causes the environment to be created, as shown in Figure 2.

```
Proceed ([y]/n)? y
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate MSAI631_MBF
#
# To deactivate an active environment, use
#
#     $ conda deactivate
```

Figure 2: Created environment

Once you follow the instruction printed on the screen to activate the environment, we can move forward with using it. For this example, we will use one of the samples located at in the GitHub repo at <https://github.com/microsoft/BotBuilder-Samples/tree/main/samples/python>. We will look at the EchoBot. The EchoBot accepts input from the user and returns it. We will need to install the Bot Framework Emulator to run it locally. However, our first step is to insure all the requirements for the application are met. We do this by using the command:

```
pip install -r requirements.txt
```

Using the command window, we used earlier to create the MSAI631\_MBF environment, we navigate to that folder and issue the command, as shown in Figure 3.

```
(MSAI631_MBF) D:\Repos\botbuilder-samples\samples\python\02.echo-bot>dir
Volume in drive D is Extreme SSD
Volume Serial Number is 9C89-09D9

Directory of D:\Repos\botbuilder-samples\samples\python\02.echo-bot

01/03/2024  08:41 AM    <DIR>          .
01/03/2024  08:41 AM    <DIR>          ..
01/03/2024  08:41 AM                3,097 app.py
01/03/2024  08:41 AM    <DIR>          bots
01/03/2024  08:41 AM                352 config.py
01/03/2024  08:41 AM    <DIR>          deploymentTemplates
01/03/2024  08:41 AM                2,268 README.md
01/03/2024  08:41 AM                40 requirements.txt
               4 File(s)                5,757 bytes
               4 Dir(s)  716,126,703,616 bytes free

(MSAI631_MBF) D:\Repos\botbuilder-samples\samples\python\02.echo-bot>pip install -r requirements.txt
Collecting botbuilder-integration-aiohttp==4.14.0 (from -r requirements.txt (line 1))
  Downloading botbuilder_integration-aiohttp-4.14.7-py3-none-any.whl.metadata (4.0 kB)
Collecting botbuilder-schema==4.14.7 (from botbuilder-integration-aiohttp==4.14.0->-r requirements.txt (line 1))
  Using cached botbuilder_schema-4.14.7-py2.py3-none-any.whl.metadata (3.7 kB)
Collecting botframework-connector==4.14.7 (from botbuilder-integration-aiohttp==4.14.0->-r requirements.txt (line 1))
  Using cached botframework_connector-4.14.7-py2.py3-none-any.whl.metadata (5.9 kB)
Collecting botbuilder-core==4.14.7 (from botbuilder-integration-aiohttp==4.14.0->-r requirements.txt (line 1))
  Using cached botbuilder_core-4.14.7-py3-none-any.whl.metadata (3.9 kB)
Collecting yarl==1.8.1 (from botbuilder-integration-aiohttp==4.14.0->-r requirements.txt (line 1))
  Downloading yarl-1.9.4-cp39-cp39-win_amd64.whl.metadata (32 kB)
Collecting aiohttp==3.8.5 (from botbuilder-integration-aiohttp==4.14.0->-r requirements.txt (line 1))
  Downloading aiohttp-3.8.5-cp39-cp39-win_amd64.whl.metadata (8.0 kB)
Requirement already satisfied: attrs==17.3.0 in c:\python39\lib\site-packages (from aiohttp==3.8.5->botbuilder-integration-aioh
Requirement already satisfied: charset-normalizer<4.0.0, >2.0 in c:\python39\lib\site-packages (from aiohttp==3.8.5->botbuilder-int
```

Figure 3: Installing required libraries

We are now ready to try and start the application using the command:

```
python app.py
```

Depending on your environment, you may need to use `python3` instead of `python` in your command. If you see an exception during startup, you may need to run the following command:

```
pip -vvv install --upgrade --force-reinstall cffi
```

Now that the application is running, you need to download and install the Bot Framework Emulator (<https://github.com/microsoft/botframework-emulator>). This program makes it easy to test your bots, without having to deploy them to an Azure environment. Clicking on the GitHub releases link will bring up the most recent builds for each supported platform, as shown in

**Error! Reference source not found..**

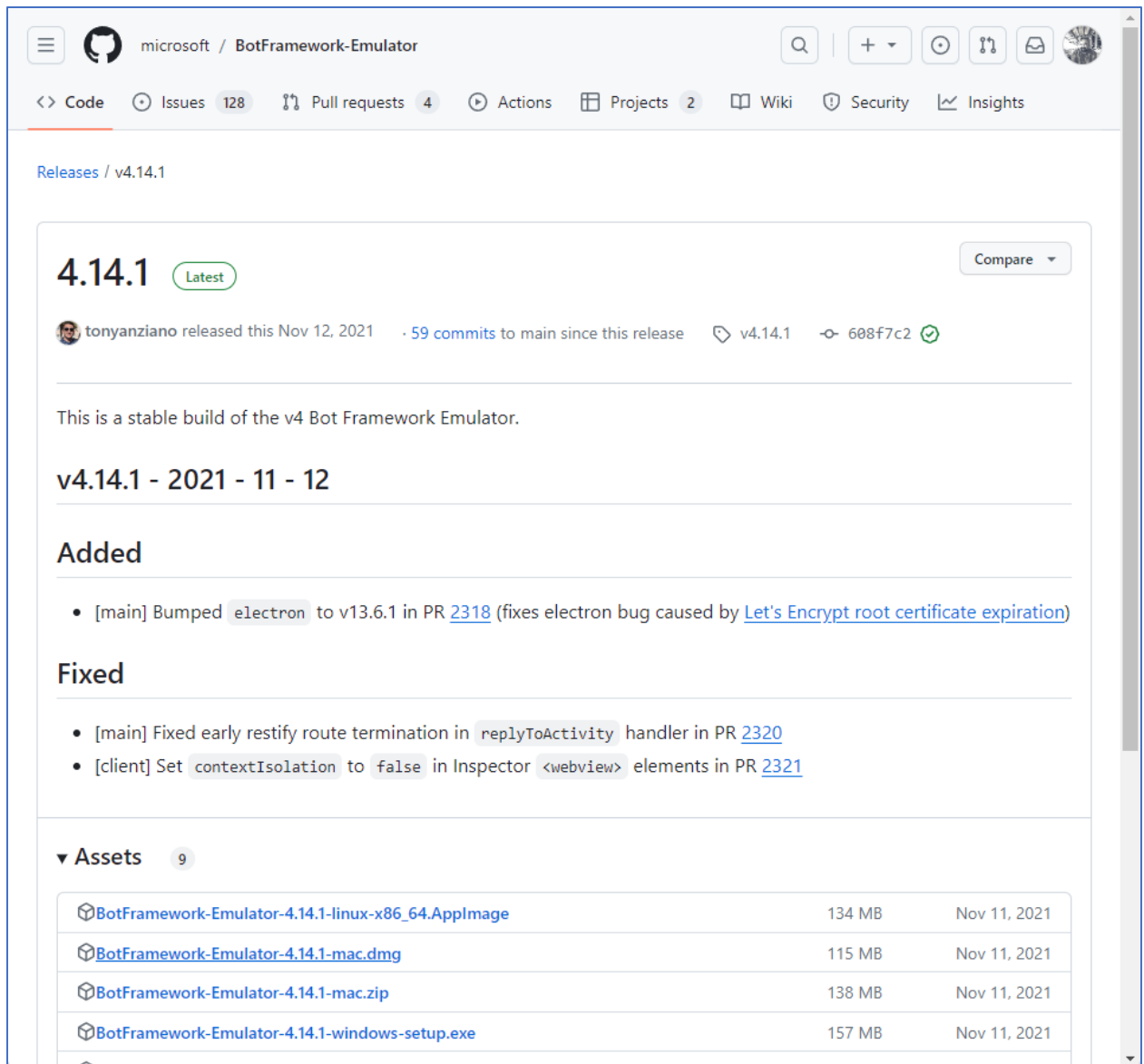


Figure 4: Bot Framework Emulator

One potential error occurs if you connect to the wrong end point in the emulator. Ensure that you include `/api/messages` in the url when connecting to the bot, as shown Figure 5.

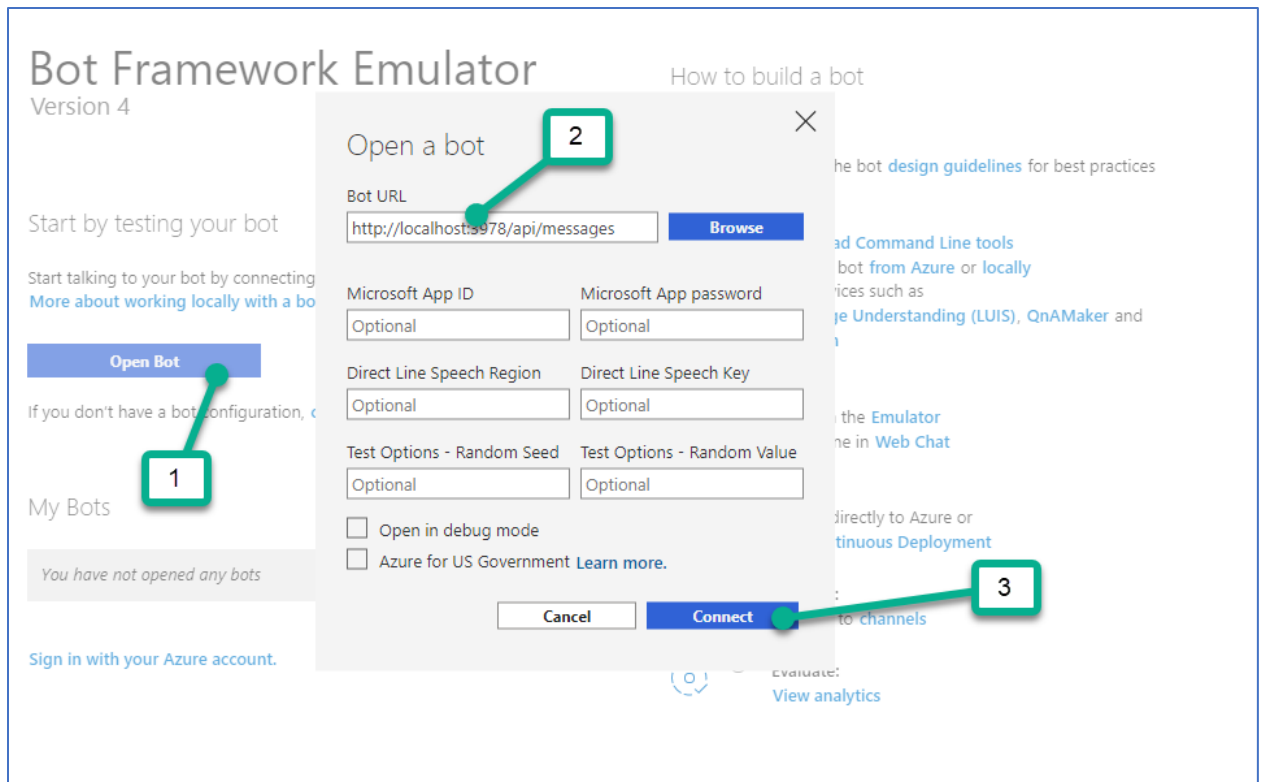


Figure 5: Connecting to the running bot

When you open the Bot Framework Emulator, you will see a page with text “Start by testing your bot” and a button labeled Open Bot. Click that button, and enter

```
http://localhost:3978/api/messages
```

in the text box, then click Connect. If you do not include api/messages you will get 500 errors. Once the bot is running and the emulator is connected you can type a message in the text box at the bottom and get a response, as shown in Figure 6.

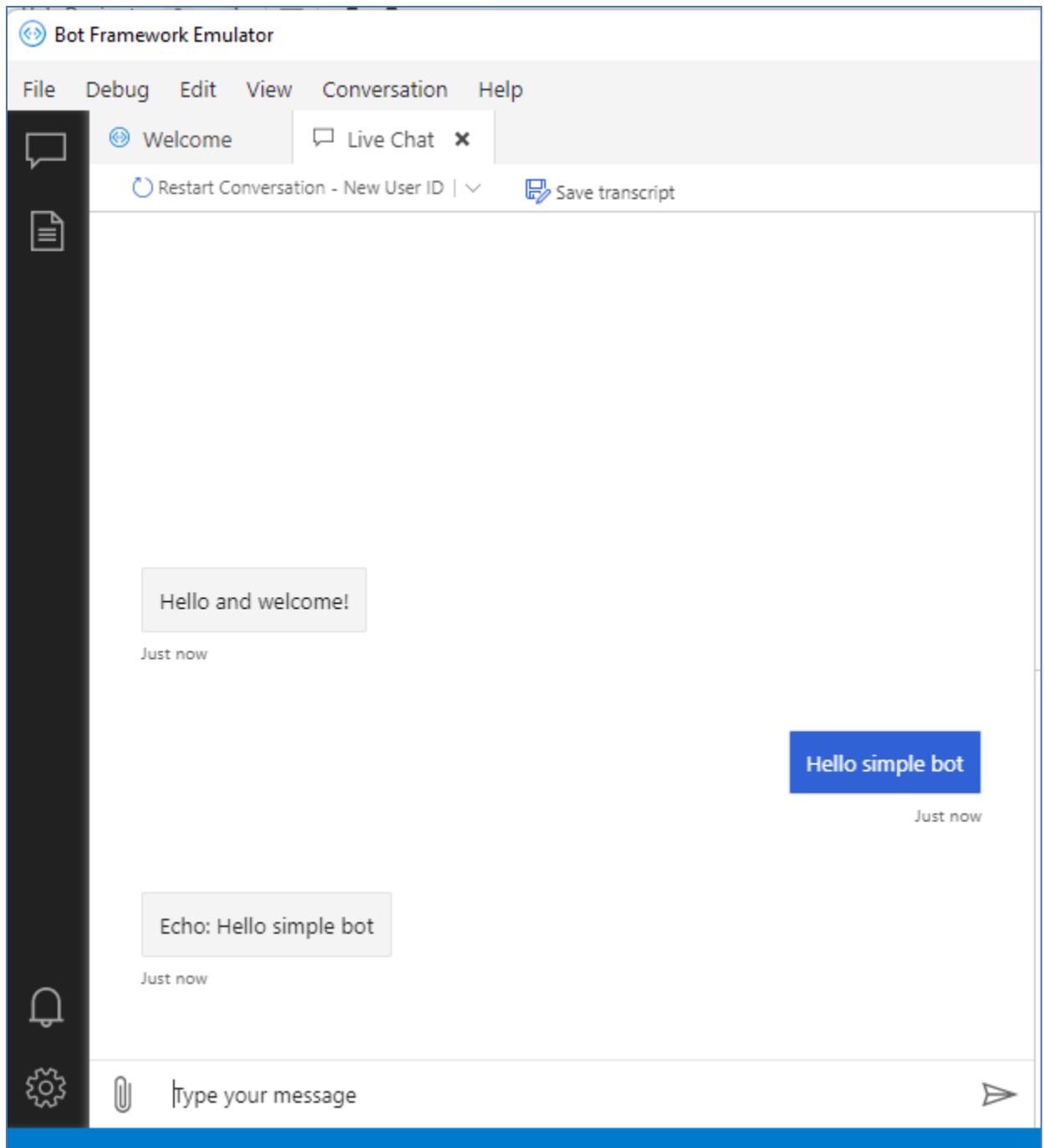


Figure 6: Running Chatbot Emulator

### Change the Sample

Now that we have it running, let us explore how it works. In this case I use Visual Studio Code (<https://code.visualstudio.com/>) a free open-source editor that runs on most platforms. The method of interest is named messages, as shown in Figure 7. To make this code a little more



interesting we modify the text contained within the body dictionary, reversing the string. Another key piece of this application is the route statement:

```
APP = web.Application(middlewares=[aiohttp_error_middleware])
```

```
APP.router.add_post("/api/messages", messages)
```

Which binds the method to the api/messages URI, which was previously discussed.

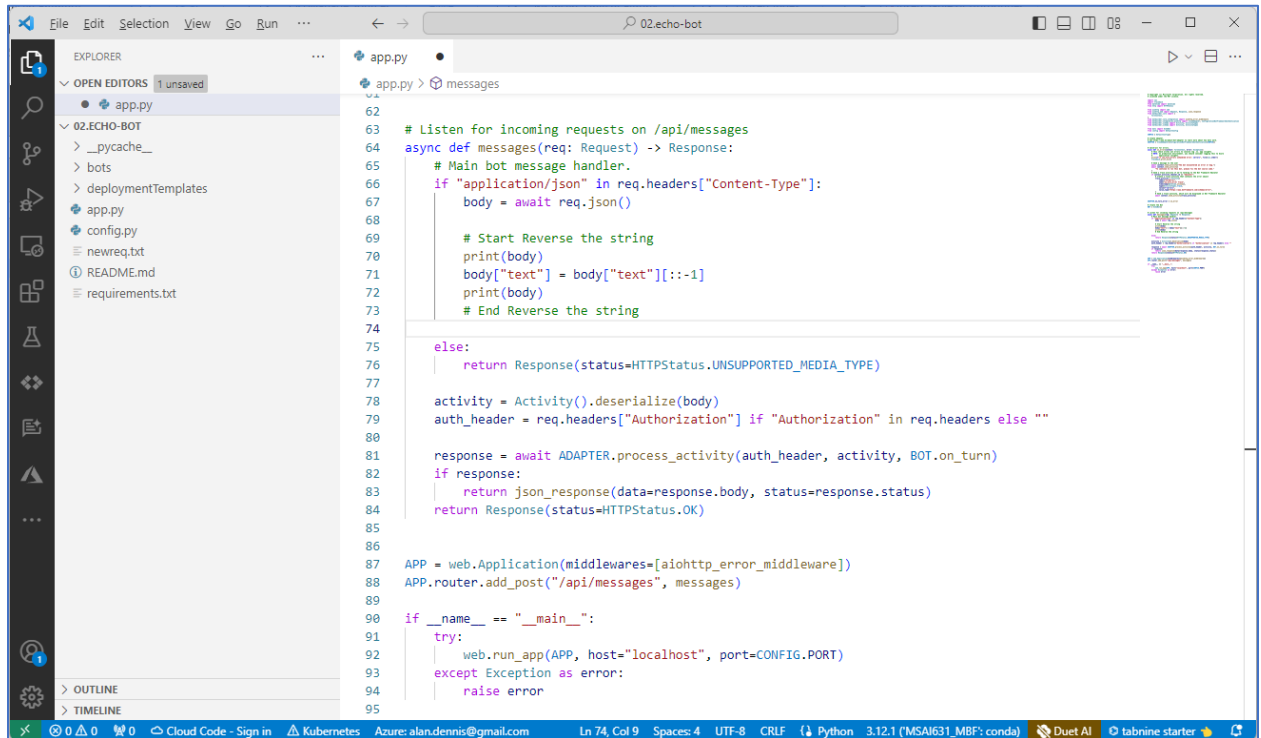


Figure 7: Changing the bot

We can then save the file and restart the bot. If all went well, you should see that the results of the echo are reversed, as shown in Figure 8.

```
[11:27:43] POST 200 directline/conversations/<conversationId>/activities
[11:28:25] -> message Echo: This is a test to see if it reverses the
str...
[11:28:25] <- message Echo: .gnirts eht sesrever ti fi ees ot tset a si
...
[11:28:25] POST 200 directline/conversations/<conversationId>/activities
[11:29:14] -> message .gnirts eht sesrever ti fi ees ot tset a si sihT
:...
[11:29:14] <- message Echo: Echo: This is a test to see if it reverses
t...
```

Figure 8: Testing reversing the string

## Conclusion

I have shared the steps to get a Microsoft Bot Framework working using Python and the Bot Emulator. This will get you most of the way through your assignment. Feel free to expand the functionality of the bot. You should now be equipped with the information necessary to complete this assignment.

## References

Anaconda, I. (2017). *Managing environments*. <https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html>

Microsoft. (n.d.). *Microsoft Bot Framework*. <https://dev.botframework.com/>