Learning to make a Discord bot in Python.

By Sihanern Thitisan (1TE19250T), Electrical Engineering and Computer Science student.
Kyushu University, 24th May 2021.
Subject: Programming Methodology I

Nowadays, we cannot deny that the world has become more interconnected than in the past. Most of us have access to some means of online communication or social platforms. For people around me, one of the popular social platforms is called “Discord”.

Discord offers a means to connect to friends in a group known as “server” and connect to influential people through a larger server called “community server”. Even though the structure of Discord looks like some other social platform, the creators of Discord encourage the customization of the platform more than other platforms. One of the additions to experience in Discord created by many communities is Bot. People have created various bots for various uses in Discord servers. I also want to try to create a unique bot from my idea in Discord too. This assignment has given me a chance to try to do as I intended. Aside from creating a usable bot, I have learned a lot about programming through the process as well.

After I set the goal to create a usable bot, the first thing I must learn is how to interact with the Discord application without an interface for humans. I have learned that I have to interact with any related application via Application Programming Interface (API). APIs are made for specific people (usually developers) to have access to specific data efficiently. In short, my bot is going to need to access all the data via the API of the app too. Later, I have learned that interacting with API directly has a lot of complexity on its own. However, there are API wrappers to make it more friendly to use. API wrappers are the language-specific packages that make complex API functions easier to use. And thanks to the Discord community, there is an API wrapper for Discord API called discord.py, which is written in Python language. Hence, this is the main Python library that I will use throughout the process.

During the process of making this bot. I have encountered many problems and learned a lot of new aspects of programming. Most of them can be found in my presentation and submitted code. I cannot explain everything using text without the context of the code itself. The final result is a bot that will search for people who own the specified game on Steam platform and in the server that this bot was used.

The result itself was simple to understand, yet the underlying process of making it took a long time to learn and implement. By doing this project, it made me appreciate the people who made feature-rich bots that seem useful and user-friendly upfront. Moreover, I am thankful for the simplicity that the developers have tried to make the life of end-users easier. Although it might seem hard to understand when you tried to do it at first, you will realize that widely available things have gone through many revisions and improvements.

Technical explanation of what I have learned.

During the time I have spent learning about programming to bring this project to its fruition. I have learned many aspects of programming. The first big one is the idea of asynchronous programming. Simply speaking, Asynchronous is one of many solutions for speeding up the program, and let each task running in an overlapping manner while other tasks are waiting for further instruction or computation. In python, the library made for controlling this is “asyncio” library. It has ‘async’ and ‘await’ as keywords for marking the asynchronous on the manner of functions. However, there are many lower-level implementations I have skipped over in the explanation. But thanks to the API wrapper I am focusing on, it was already implemented in the module itself. So, I can focus on learning the data structure of it instead.

Interestingly, the data structure of discord.py uses classes with a huge number of attributes and methods that intertwine with each other. After a little bit of practice, and trial and error, I have a better idea of where things belong in the data structure of each section. When I started to understand it, I moved to interact with Steam Web API directly because there are not many functions I want to use in the available API wrapper.

The next thing I have learned about is making HTTP(s) requests from python. There are many different types of requests, but I have used only ‘GET’ method inside this code. Each method has its strength and weaknesses on its own.

After successfully implementing the interaction with steam. For searching for a game. I have researched how to search for results from a given keyword. Even though I have tried a little bit about fuzzy string matching algorithm (I have mentioned it in the presentation), but it did not make it to my final design since there are too many varieties of abbreviation for game names. In the end, I use a google search module to find the result instead, which is also where I have learned about HTTP error 429 a hard way for the first time.

Overall, the bot does come out working as intended eventually. However, there is still much room for improvement. Like better help commands, or more features. In the far future, I can deploy a bot in a cloud server to make it runs all the time, not only when I execute the program.

For more technical information regarding implementation please see comments in my code, and these materials I have studied from.

discord.py documentation

Steam Web API documentation

aiohttp documentation

Googlesearch library

SteamID library