Final Project
Algorithm & Programming

Project Name: **Alfininator**

Name: **Muhammad Alfin Rizqullah**

Student ID: **2502036842**

Class: **L1CC**

Project Specification

With Discord being my main source of communication with my friends I decided to create a bot that would suit my need why also satisfying me and my friend's group sense of humor. The bot itself would be able to receive an input regarding an invitation to a game that me and my friend would regularly play called Bloons TD6, the bot then will respond with a random GIF that has the "bloon" in its link, it might be small but me and my friend does find it quite amusing. While my bot would be useful for me and my friends need, there is one glaring issue that will impact the longevity of my bot, and that is the module I mainly use for this project, discord.py. The basic rundown on what happen is that the person that is responsible for creating the module is fed up of Discord itself in citing increased control of it's code and restriction that forces him to change his code, and with no support from discord themselves he has decided that he will stop updating his module, which means that for now it is usable but if Discord ever brings an update that is significantly different in terms of how lines of code interact with Discord itself, then this bot will be unusable.

Input

1. Invitation link to the specified game
2. /Use
3. /bloons
4. /gog
5. /hello

Output

1. A random gif that has bloons in it's name
2. The code for the game invitation link
3. What command that is available for the bot
4. Functions that is available

Solution Design

1. /use
2. Given the command for the user to use
3. Accepting input
4. Output

Uses of the bot

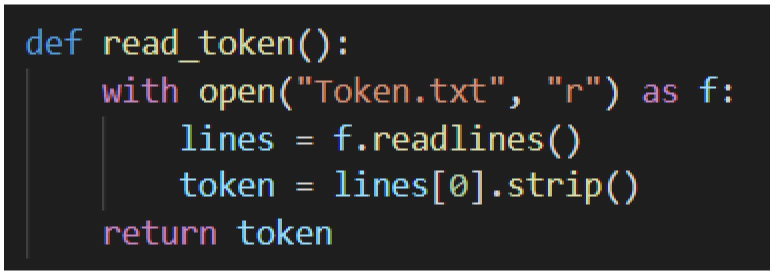
Randomize gif

The bot will read that there is a link from the game using keywords I've setup that ensured that it is a proper link and will give out a random gif while also giving out the code for the game lobby itself.

Implementation of code

Main_final.py:

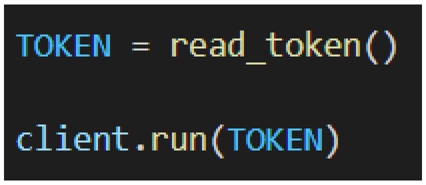
Reading Token



```python
def read_token():
    with open("Token.txt", "r") as f:
        lines = f.readlines()
        token = lines[0].strip()
    return token
```

In order to create a discord bot that can be run on a server, you need to first log in to discord dev in where you would make the name and description of your bot while also allowing you to give your bot as much permission as you want. When everything has been set up, discord would then give a token for your bot, think of it as the key to your bot, if someone else took it they will be given access to do whatever they want with your bot. This is why I have the function read_token() this is because in order to not give anyone else the token I decided to put it in a different text file and use the function created above to access it so later on it can be utilize so that the bot itself can be launched.

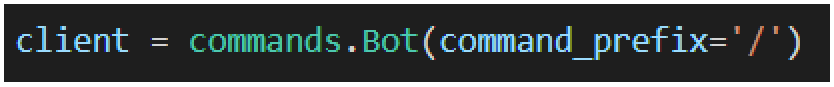
Launching the bot



```python
TOKEN = read_token()

client.run(TOKEN)
```

When launching the bot, it requires an internet connection due to the code having to run the token through the discord database in order to ensure that this is a legimate bot.

Command Prefix



```
client = commands.Bot(command_prefix='/')
```

For every Discord Bot that have commands, every single one of them will always have some sort of prefix before the actual command, this is so that the user will not accidently start a command while typing normally, the command used is **commands.bot** which is a module from discord.py

The difference between .event and . command

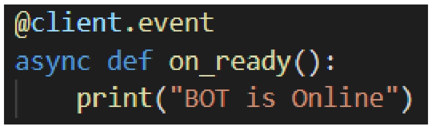
“.event” work almost constantly without the need of a prefix before the given function which makes it perfect if you want to have the bot respond to a message, since the user would not have to change anything specific when writing the message since the bot will simply scan the message and if the conditions are met, the code inside “.event” will proceed.

“.command” requires a prefix for it to work and it will not scan entire message, only if the message starts with the prefix, the bot will then proceed to fullfill its function.

Async over Sync

With the amount of users that are potentially going to be using the bot at a single time, it is best practice to use async because it can run multiple at the same time, while also being able to pause the run time of a code if it still requires information from a different module/ database meaning that it can run other code while it waits, sync meanwhile would stop the entire queue until the first call is finish, delaying the others.

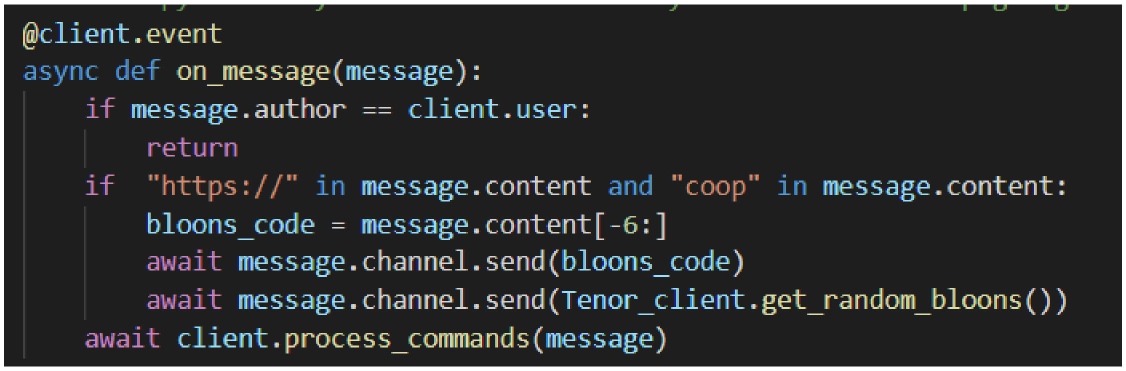
BOT Activation Message



```
@client.event
async def on_ready():
    print("BOT is Online")
```

When we do launch the bot, whether through an online service or simply from our machine, we need an indication that the bot is actually on, **on_ready()** is what is printed in discord.py when the bot has successfully been turned on, normally we can’t see since it is run on the module, but we can grab it and simply give a new output if the bot receive the required output.

Sending the invite and the randomize gif



```
@client.event
async def on_message(message):
    if message.author == client.user:
        return
    if  "https://" in message.content and "coop" in message.content:
        bloons_code = message.content[-6:]
        await message.channel.send(bloons_code)
        await message.channel.send(Tenor_client.get_random_bloons())
    await client.process_commands(message)
```

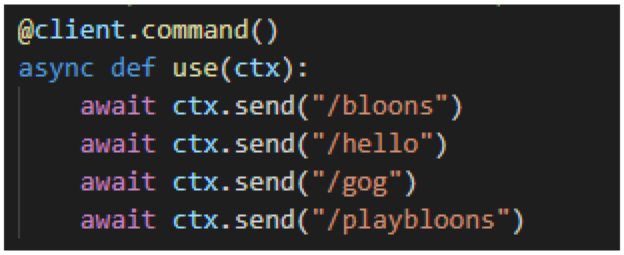
The **on_message()** function of discord.py would tell the bot to scan every single message that has been posted in the server that it has access to, and the first **if** variable is there to ensure that it is a message sent by a user or a bot in the server, and simply not a line of code from discord.py.

In order for this code to run, there is 2 requirement. One of them is that it requires the message to include an "**https://**" to ensure that the message contains a link while if this is the only requirement, we found that other discord bot that paste links like music bots, could accidently trigger it which I fixed but adding another requirement, which is that the message must also contain the word **"coop"** which is rarely used especially in conjunction with **https://**. There is still the off chance of a link with the word **"coop"** could trigger the bot, but after evaluation the likeliness of it is very low and if that does happen too often I could simply add in more condition before the bot would trigger.

The purpose of **"bloons_code"** is to print out the invite code from the link and text I set the range so that is flexible in case the invite link/ sentence beforehand change in a future update, but what will always be the same is the location of the invite code, which will be at the end of the message, so if we create a range but count it from the right we can use **"-6"** for the start of the invite code until the end of the message which would ensure that the code will be extracted from no matter how long the sentence before hand is.

The code **"Tenor_client.get_random_bloons**()**"** is extracted from a class file I created which I would be explaining later on, but this line would simply send the randomize gif link into the chat server.

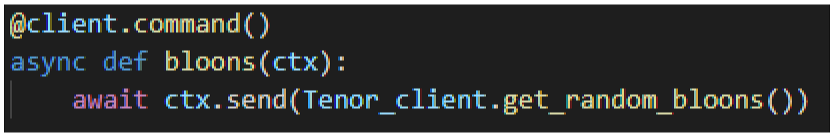
Knowing what commands are available



```
@client.command()
async def use(ctx):
    await ctx.send("/bloons")
    await ctx.send("/hello")
    await ctx.send("/gog")
    await ctx.send("/playbloons")
```

This command will send all of the commands to the user in their respective text channel, this is done due to hindsight of previous coding exercise in where I only see it from the perspective of the developer which means that I have access to all the commands and have the knowledge of the names of said commands. When I saw someone try using my code they kept asking me on what are the commands which made me realize that I need someway to show all the commands that my bot have. While it is not perfect I atleast need to only say to type /use and after that the command names are pretty explanatory.

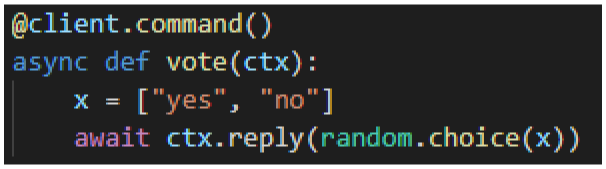
If people just want random bloons gif



```
@client.command()
async def bloons(ctx):
    await ctx.send(Tenor_client.get_random_bloons())
```

This will allow them to get random gif to fullfill their amusement because sometimes in our friend group the easiest way to make someone laugh is by showing them a random monkey gif, though I did not put a maturity filter on it which means that more 18+ type content could appear.

Vote Code

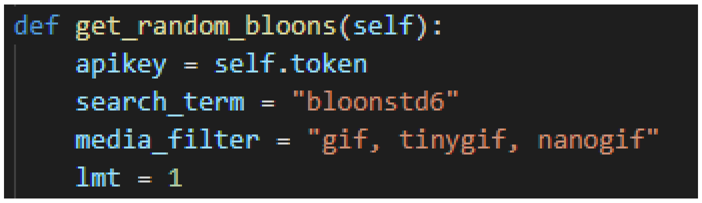


```
@client.command()
async def vote(ctx):
    x = ["yes", "no"]
    await ctx.reply(random.choice(x))
```

Sometimes me and my friend have issue deciding something, this will simply give a third party an option to vote for us to what to do next.

GIF.py

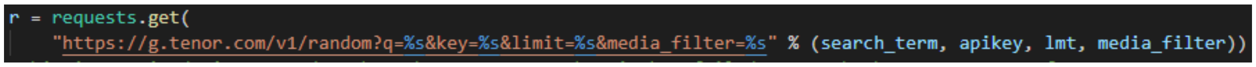
Variable for API interaction



```
def get_random_bloons(self):
    apikey = self.token
    search_term = "bloonstd6"
    media_filter = "gif, tinygif, nanogif"
    lmt = 1
```

This section of the code is required by “Tenor” in order to access their api correctly, these are the parameters in which it will use the api to search the tenor gif website for the gif that conform to these parameters.

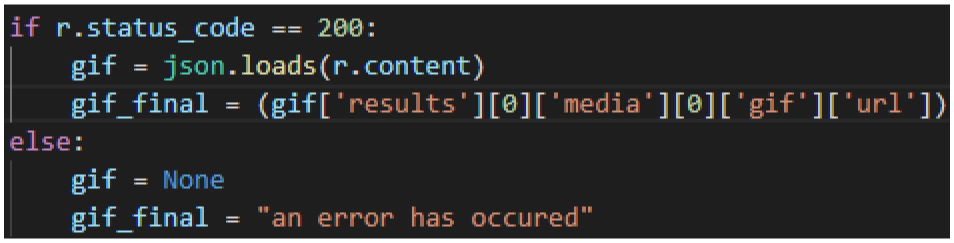
Requests



```
r = requests.get(
    "https://g.tenor.com/v1/random?q=%s&key=%s&limit=%s&media_filter=%s" % (search_term, apikey, lmt, media_filter))
```

The use of the module **“requests”** is to be able to access the api/link of websites which would extract the code for the websites in a relatively abstract route.

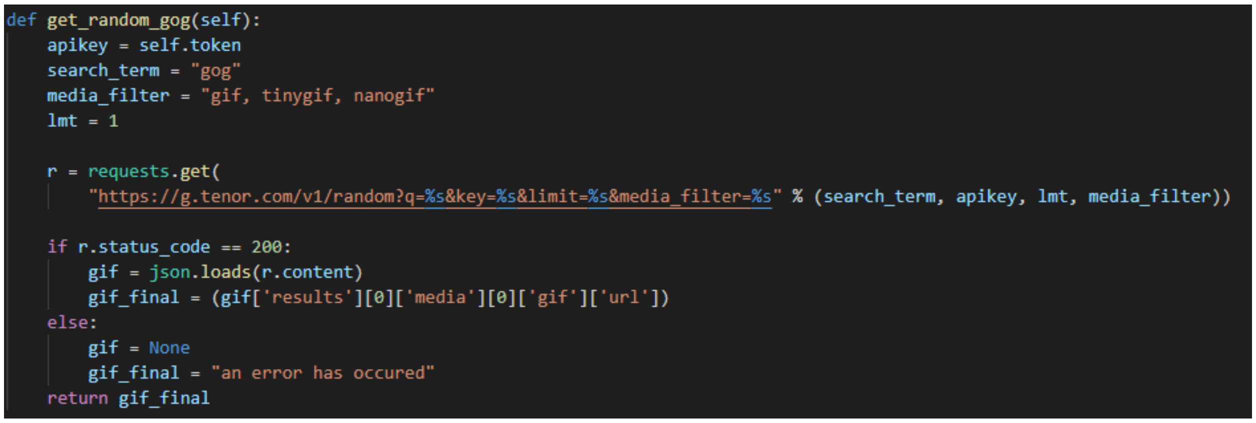
The status of connection



```
if r.status_code == 200:
    gif = json.loads(r.content)
    gif_final = (gif['results'][0]['media'][0]['gif']['url'])
else:
    gif = None
    gif_final = "an error has occured"
```

Status code 200 means that the connection between the bot and the tenor server are running smoothly, if the code is not 200 then it is an error and usually it would simply crash the bot, but in this case it would print the string in the message.

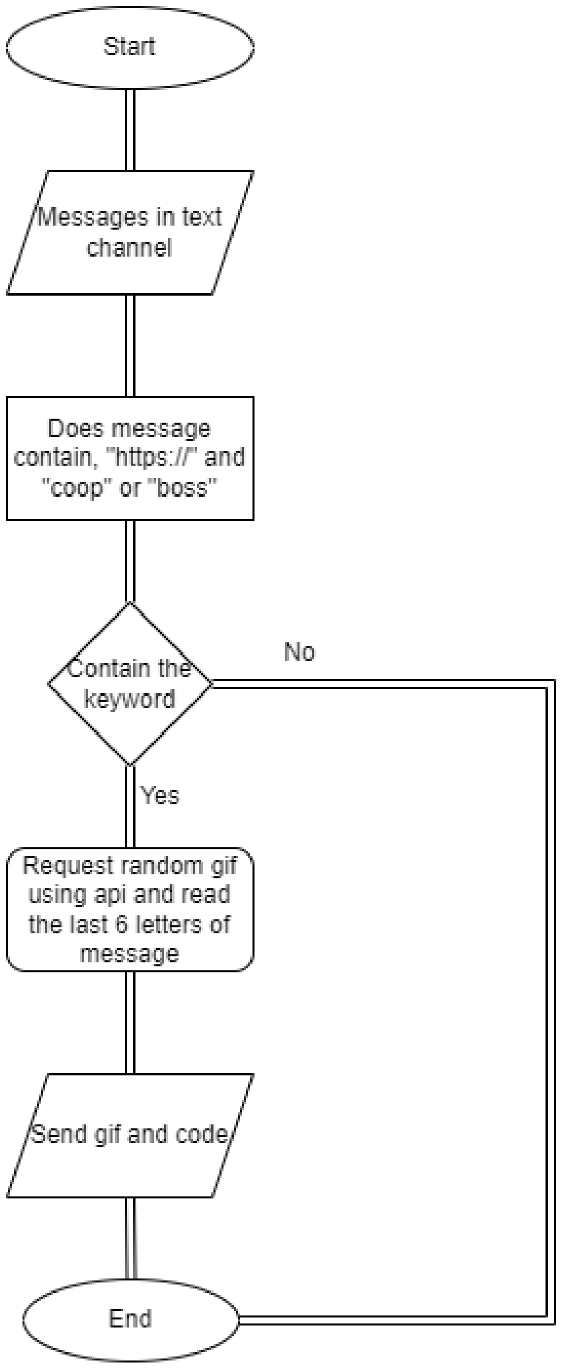
Modularity


```python
def get_random_gog(self):
    apikey = self.token
    search_term = "gog"
    media_filter = "gif, tinygif, nanogif"
    lmt = 1

    r = requests.get(
        "https://g.tenor.com/v1/random?q=%s&key=%s&limit=%s&media_filter=%s" % (search_term, apikey, lmt, media_filter))

    if r.status_code == 200:
        gif = json.loads(r.content)
        gif_final = (gif['results'][0]['media'][0]['gif']['url'])
    else:
        gif = None
        gif_final = "an error has occured"
    return gif_final
```

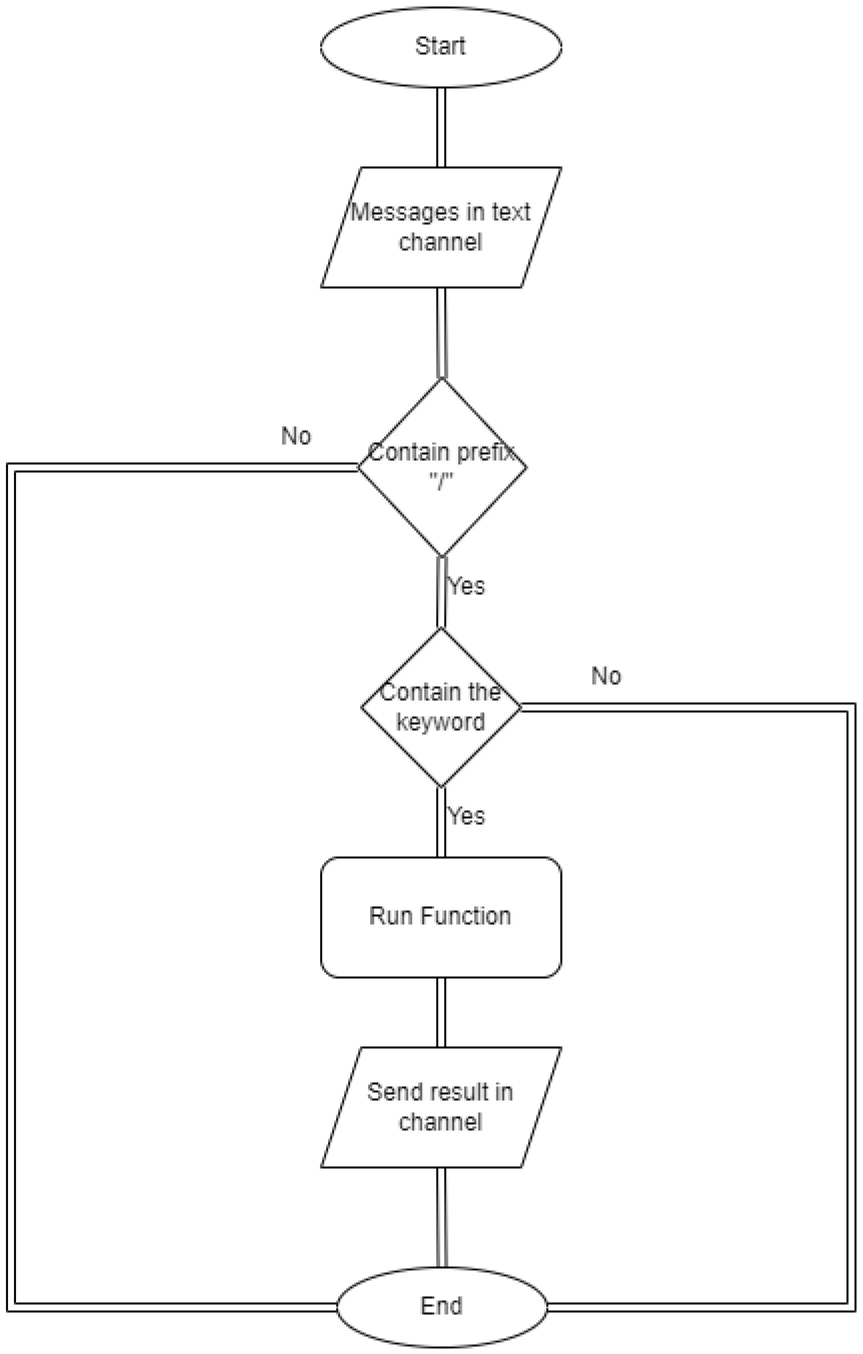
I added this code because my also find the topic “gog” quite funny and this shows how modular my code is because I could simply add more of these type of code and after changing a certain parameter it could randomize another topic, plus if I want to make an app with the same randomizer option I could still use the same file with a different main.py.

Flow Chart

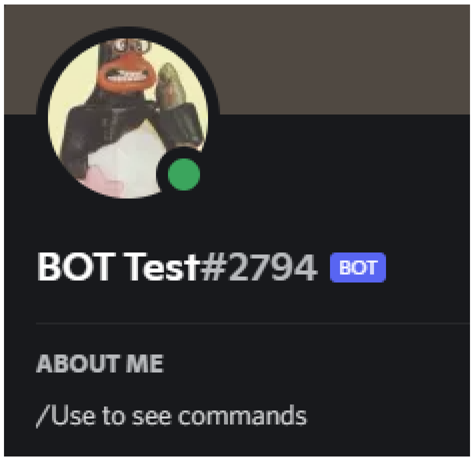
Random GIF


Start
Messages in text channel
Does message contain, "https://" and "coop" or "boss"
Contain the keyword
No
Yes
Request random gif using api and read the last 6 letters of message
Send gif and code
End

Commands



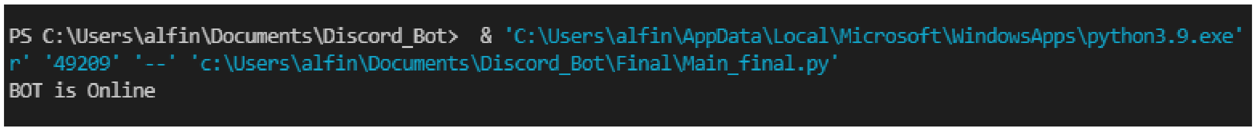
Proof of program



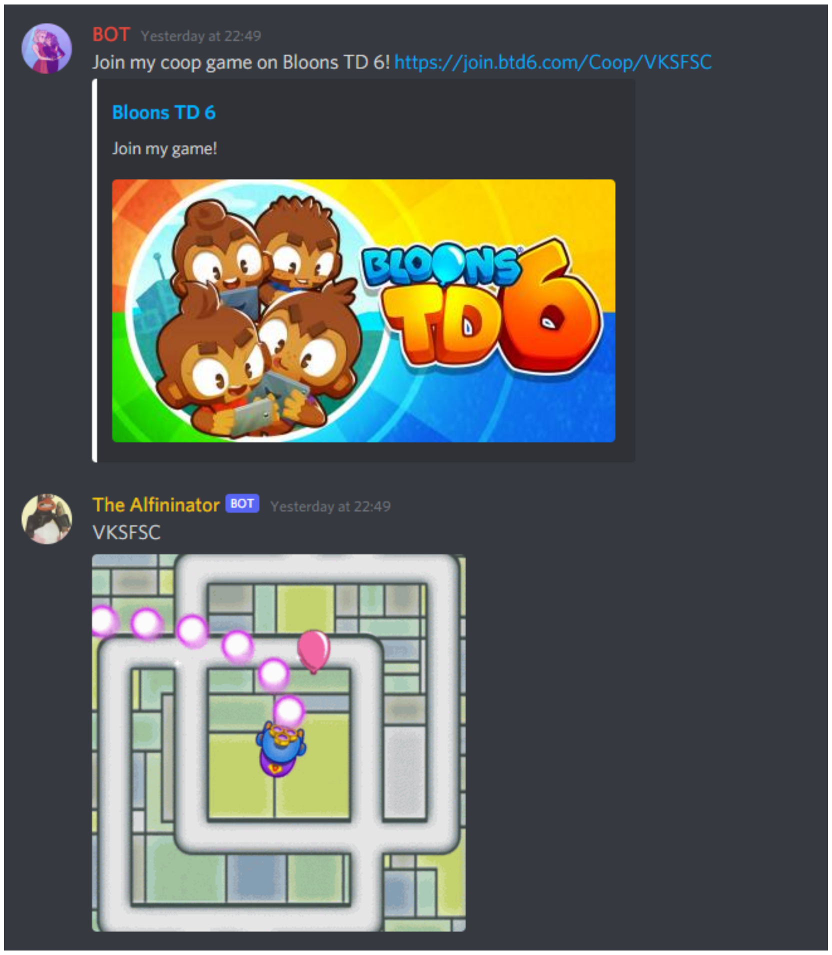
BOT Test#2794 BOT

ABOUT ME

/Use to see commands



```
PS C:\Users\alfin\Documents\Discord_Bot>  & 'C:\Users\alfin\AppData\Local\Microsoft\WindowsApps\python3.9.exe'
r' '49209' '--' 'c:\Users\alfin\Documents\Discord_Bot\Final\Main_final.py'
BOT is Online
```



BOT Yesterday at 22:49

Join my coop game on Bloons TD 6! https://join.btd6.com/Coop/VKSFSC

Bloons TD 6

Join my game!

The Alfininator BOT Yesterday at 22:49

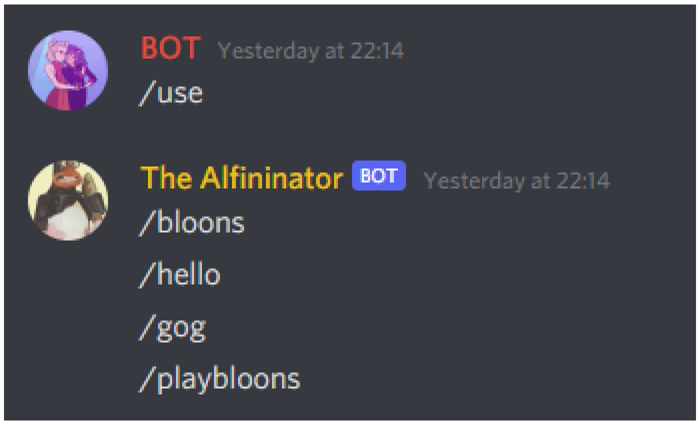
VKSFSC

Sunk by 13 Torpedoes Yesterday at 22:18
/bloons

The Alfininator BOT Yesterday at 22:18

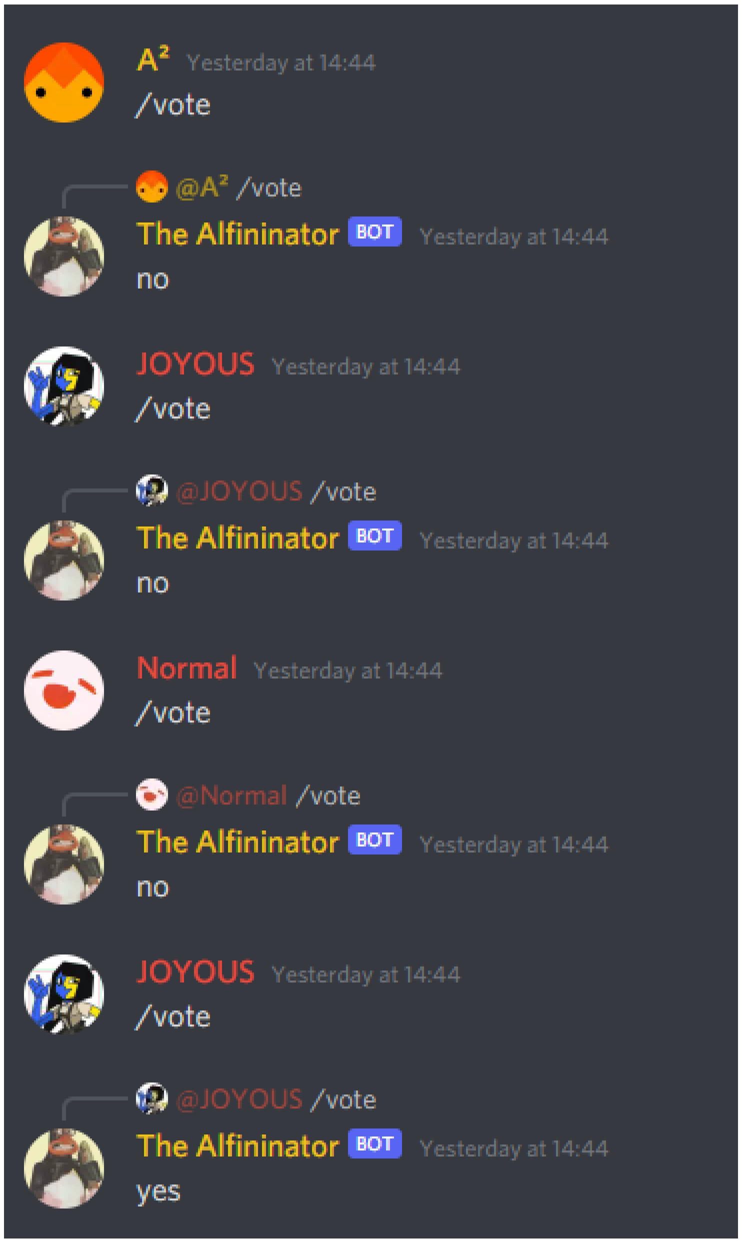
A² Yesterday at 22:14
/gog

The Alfininator BOT Yesterday at 22:14



BOT Yesterday at 22:14
/use

The Alfininator BOT Yesterday at 22:14
/bloons
/hello
/gog
/playbloons



A² Yesterday at 14:44
/vote

@A² /vote
The Alfininator BOT Yesterday at 14:44
no

JOYOUS Yesterday at 14:44
/vote

@JOYOUS /vote
The Alfininator BOT Yesterday at 14:44
no

Normal Yesterday at 14:44
/vote

@Normal /vote
The Alfininator BOT Yesterday at 14:44
no

JOYOUS Yesterday at 14:44
/vote

@JOYOUS /vote
The Alfininator BOT Yesterday at 14:44
yes

Reflection and experience

Overall I’m satisfied with the code that I’ve done because for the most part I actually enjoyed, even coding and beta testing this bot for over 6 hours with only an hour break because I was genuinely enjoying python coding that much (HTML, CSS and JS is a different story) I truly did lost track of time doing this which is rare for me because that kind of lost of awareness only appears when I play games and hangout with friends, activities that I truly enjoy. The first idea for this project was to make a bot that can play music while also be able to remind me of homework which in the grand scheme of things seems pretty advance since I didn’t even have any experience in interacting with web service and json files. After doing this project though I do want to add those feature in the future since it truly become a passion project, and something that is very useful for me and my friends and really that’s the main reason why I went with a discord bot instead of making a game for this project. My thinking was If I make a game, why would anyone play it? Because games are a wide market and simply I can’t compete with what developer releases and I didn’t want to make something just to play it once or twice for the presentation and never use it again, I wanted something that be can be updated to suit my needs in the future and a discord bot is the perfect candidate for this, though with the discontinuation of discord.py I will have to migrate this code to either JavaScript or the module “Hikari”.

When I finished the bot and started to review the code, I felt like it was to simple because the code itself doesn’t look complex, but what I realize is that it looks simple because of my additional research in understanding how to read json file and understanding how to use api’s/ scrapping data from website (well it could still be quite simple, comparatively). Going back to when I first tried to understand json, I was completely dumbfounded because of how complex it was at the start, but the reason it was hard to comprehend in the beginning was because I saw the website as being this giant code where everything is randomize, but when I realize it was just a giant dictionary/ list I started to understand that it was much simpler than what I thought. What makes this experience so smooth was the fact that the gif hosting site “Tenor” provided its own api which was such a convenience, though if they didn’t I would simply just scrap the link from the website itself.

The biggest thing that I used to downplay is debugging/ testing, as I stated earlier I spent 6 hours in doing the code itself, but I also spend another 2 hours with a friend of mine debugging everything. At first I thought like “everything should work properly” but after an hour we found a couple of bugs that we thought I fixed, but then I would be getting random error message in the terminal and for a while we just couldn’t find the issue, but after looking at every single text channel in the server, what we find is that my bot was interacting with the music bot, because the condition that was set earlier was written in away that it only requires a link by the form of “https://” to be present, which means that when the music bot added a music link into a text channel the bot will try to respond to it, but for the extraction of the invite code, it was hard coded that the message needs to be a certain length which makes it so that it creates an error with the bot, thankfully after the next hour it was relatively trouble free and after that I realized on how useful debugging is.

Overall I enjoyed the project and it did bring back my motivation for coding and after showing this code to my friend who have more experience in coding than me, he was impress in how neat the code was, because for someone that has just started coding, how I format it is very neat compared to when he started. My other motivation in finishing this project is not really the score, but to make Sir Jude happy, since he is one of the best teacher in my class and the way you teach sir does make me want to do good in class.

Code a Discord Bot with Python - Host for Free in the Cloud - YouTube
python - How can I randomly select an item from a list? - Stack Overflow
How to Communicate with APIs in Python with the Requests Package - YouTube