Oracle Discord Bot

Zachary Campbell, Story County

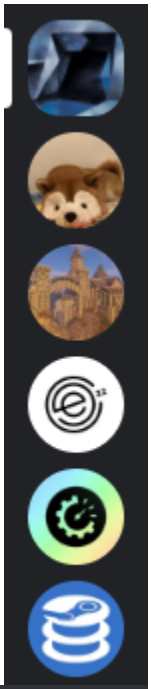
Iowa State Fair 2021

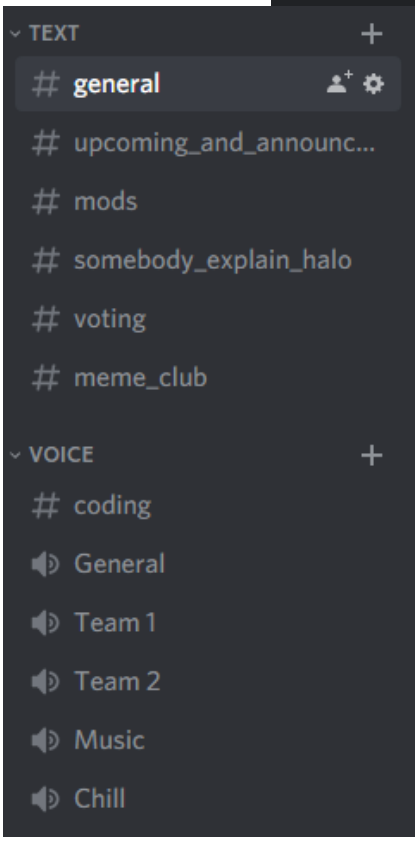
My 4-H Project is a computer program I wrote to be a bot user of Discord; it is named Oracle. This bot can do helpful things such as tell you the weather; retrieve images, comics and information; and modify and change images it is given. This bot functions through Discord, a group messaging application primarily for people who play video games, but also for other groups.

Discord

Discord is a messaging application originally intended for people who play video games. It has grown into an all-purpose messaging and communication application for study groups, hangouts, communities, and more. Discord works on a basis of *servers*, which are essentially chat rooms or group chats. On the right you can see various servers, identified by their icons. The top one is the main server my friends and I use. A few other examples shown are one for my Youth Group, one for an ISU programming club, and one for a database hosting site.

Discord also has various *channels*, which are essentially different places to have conversations. Shown in the picture below are various channels on Oracle's main server. These are so I can have a text conversation with someone in one channel, and simultaneously have one with someone else in a different channel. This is also helpful for organizing conversations. For example, we confine all our discussion of the game Halo to the chat called "somebody_explain_halo" because that's the intended place to talk about Halo. The chats in the example under "Voice" are voice channels, where you can use a microphone to talk to people like a Zoom call or Google Meet.





Discord provides support for bot users. A bot user is similar to a regular user of Discord, except it is run by a program instead of a person with an account. Bot users can send messages, read messages, send images, save images, and even join and play audio through voice chats. Playing music is actually a primary use of bots across Discord; this allows everyone in a call to listen together.

Oracle

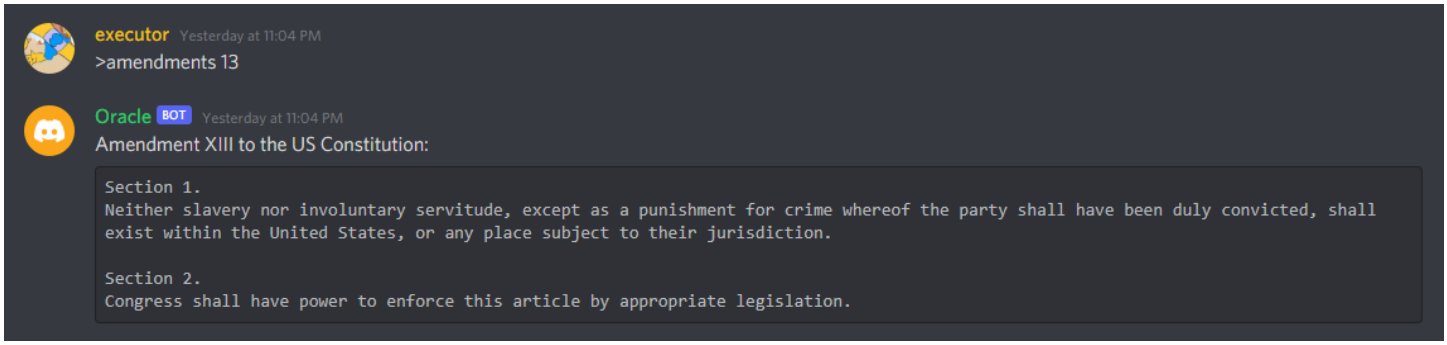
Oracle is the name of my bot, an application running off my computer at my house written in the programming language Python. To make this bot, I started out by watching a series of Youtube videos from a channel called Carberra Tutorials. The first 12 of these videos were very helpful; after that, I branched off, believing I could figure everything else out from there. There is a library for Python called Discord.py which allows you to interface with Discord and do everything you need to - send messages, read messages, respond to commands, etc. Discord.py has very good documentation on their website https://discordpy.readthedocs.io/. This is essentially a form of Wikipedia, meaning if I want to know about how to send an image, I can look that up on this website and it will tell me how to do so. All programming is knowing how to tie various pieces of a language together in a way that will do what you want it to do.

Oracle has, as of this writing, 37 main commands and 5 additional behind-the-scenes commands (for use by me, the creator of the bot). I've

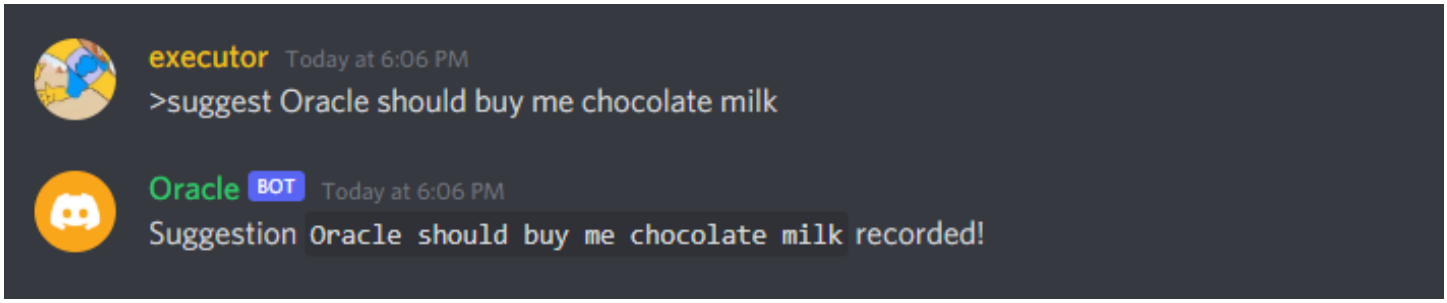
chosen to go through many of the commands that I find most interesting and fun. To use any of Oracle's commands, one must type a right arrow (>) and then the name of the command. For example, ">greet" would call the "greet" command, which simply causes Oracle to greet you with a salutation. I've categorized the commands I want to discuss into the following categories: Informational, Images, Community, and Miscellaneous.

Informational Commands

1. **Amendments:** The first interesting command is "**amendments**." This command takes a number between 0 and 27 and returns the text of any of the amendments to the US Constitution. So, to get the text of the 13th amendment, you would type the following and receive the following response:



2. **Suggest:** The "suggest" command simply saves whatever you type to my computer in a text file for me to read later:



3. **Poll:** Discord has a *reaction* feature, which allows you to "react" with an emoji to a message. For example, someone might say it is his/her birthday and you could react with a birthday cake emoji. Multiple people can react and it can be used to make a poll, as in the example below with thumbs-up and thumbs-down:

4. **NOAA:** This command retrieves the weather forecast from the National Oceanographic & Atmospheric Administration for a given latitude and longitude or a default location. In this example I have it set such that anytime the NOAA command is called on Oracle's home server it searches for 50010. However, you can also give it a latitude and longitude and it will search anywhere you like. This command works by using both the Discord Interface and NOAA's own interface. Oracle retrieves the information off the website before converting it to a form that can be sent through Discord, all automatically and behind the scenes:

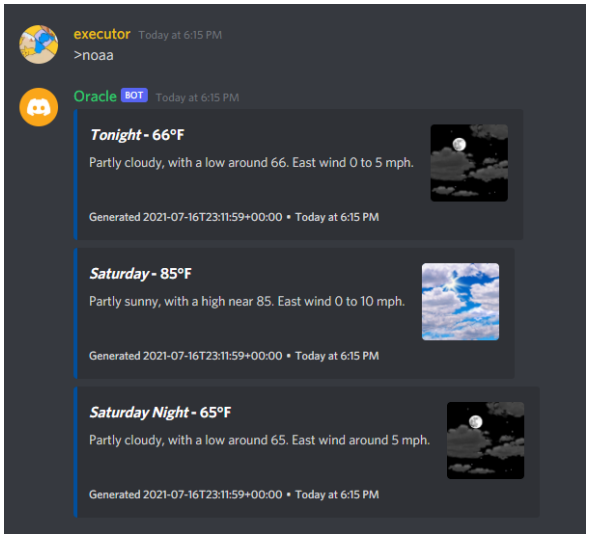
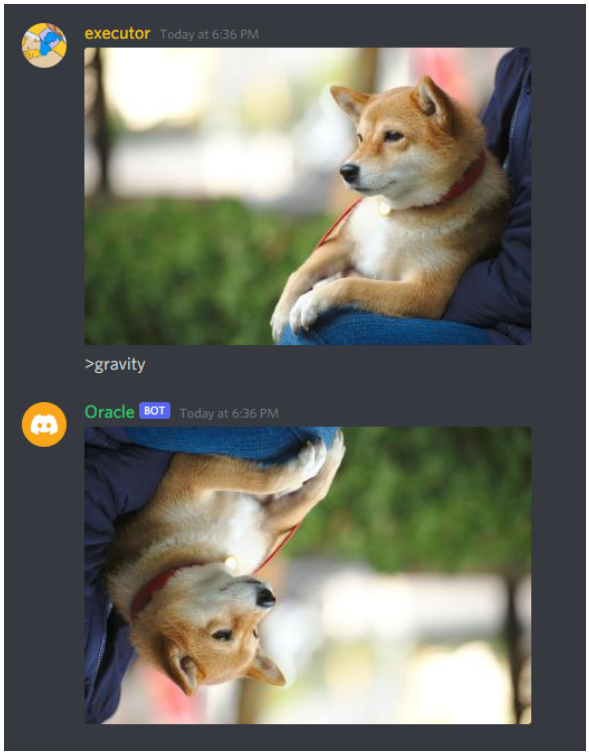


Image Commands

I learned how to use Python to modify images in a class last year using a library called PIL. PIL lets you do more or less anything you like with an image, providing you know how to use it. In addition to making some really cool commands, working on Oracle has garnered me a lot of experience working with PIL and with images in general. To show you the effect of these commands, I have Oracle start with the image on the right and then tell it to make each modification one at a time in sequence. To be clear, this is all done through Discord and can be run from a laptop, phone, or anything with access to Discord. This is a creative commons image that can be found here: commons.wikimedia.org/wiki/File:Momo_(15563002723).jpg



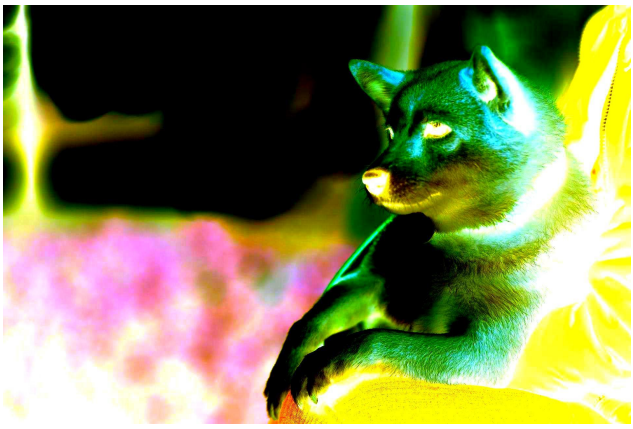
5. **Gravity:** This command rotates the image 180 degrees, or flips it across both horizontal and vertical axes. This example image also shows you that you can send an image in the chat, and then start using commands instead of having to attach the image to the command every time.
6. **Rcha/Gcha/Bcha:** These commands are all very similar - they modify a color channel of the image. Much of PIL works with images in RGB format, meaning every pixel is stored as a combination of red, green, and blue values. These values range from 0-255 and are ordered as red, green, blue. 0 is none of a color and 255 is as much of that color as possible. Thus, a pixel with the RGB value "0, 255, 0" is going to appear only green and as green as possible. Each of these commands takes a different channel and multiplies every pixel's value by the given number, as a percentage. So, running ">bcha 200" goes through every pixel in the image and multiplies the blue value by 200%, as has been done on the right. Running ">bcha 100" would multiply each blue value by 100%, or 1, and nothing would change.

7. **Pixelate:** This command makes an image pixelated, as you would expect. The way it does this is by using the RGB system I discussed above. Pixelate takes a number as part of using the command - the image to the right was created using “>pixelate 60.” To make the explanation simpler, let’s use the number 3. Oracle takes the image and first makes sure it is divisible by 3, cutting off excess as needed to make sure both the length and width of the image are divisible by 3. With that finished, it divides it up into 3x3 squares. Each square has all the individual red, green and blue values for all 9 pixels added together and then divided - a simple mathematical average. Then, the average pixel consisting of the average RGB values is placed back in that 3x3 square for all 9 pixels, causing it to lose resolution/detail and appear more pixelated. This works for any number you use as long as the image is longer and wider than the number.

8. **Revert:** Because a user of the bot’s image capabilities won’t know how a modification looks until after running the command, I created another command that “undoes” the last modification performed. Revert deletes the last image Oracle sent and allows you to continue working from where you were before, in this case undoing the pixelation effect (example not shown).

9. **Invert:** This command causes the colors and pixels to invert - 0 becomes 255, 100 becomes 155, 255 becomes 0, and so on, for every pixel in the image. This is something PIL can do quite easily using a built-in channel operation function. Implementing it into the bot was not very difficult; it shows the power of PIL on its own.

10. **Contrast:** The contrast command does exactly what it sounds like - increasing or decreasing the contrast depending on the decimal value it is given. Values greater than 1 increase the contrast, while less than 1 decrease the contrast. The default value is 2 or 200% increased contrast. I also ran the gravity command again because I was tired of looking at the beautiful dog upside-down.



11. **Enhance_Color:** This command “enhances” the color using another built-in PIL function. As you can see it amplifies the pink color of the background on the left of the image.

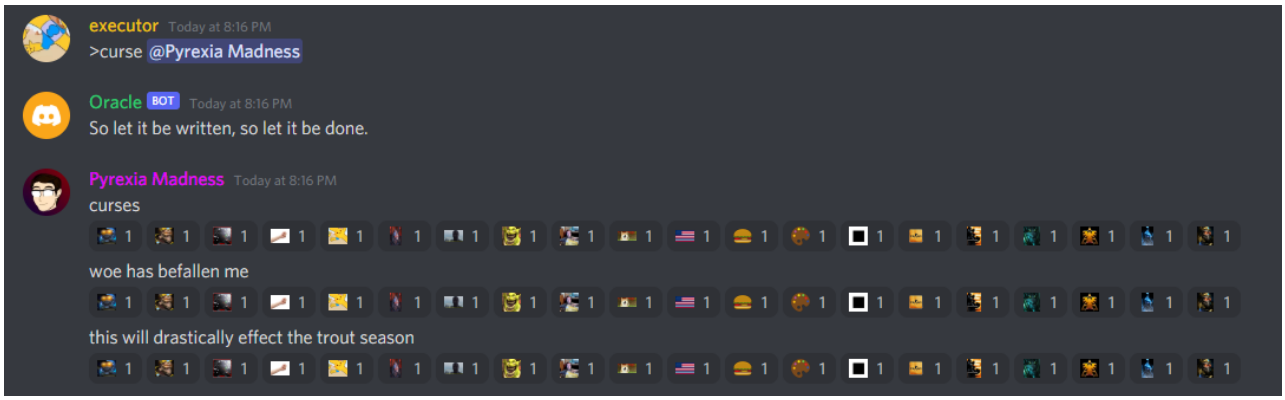
12. **Grayscale/BNW:** To showcase these last couple of Image commands, I'm going to use a different image, so the effects are clear. On the left is a picture of my dog, Rolo, taking a nap. In the middle is a standard grayscale of that image - a very simple effect. On the right is something more interesting. Grayscale, or luminosity, assigns each pixel a value between 0 and 255, with 0 being black and 255 being white. The image on the right is from the "**bnw**" command, short for black-n-white, because it takes the grayscale image and takes it a step further - every pixel is either 0=black, or 255=white. The cutoff for where to go from making pixels all black to making them all white is determined by the number used with it. For this image, that number was 100. Everything greater than 100 is white and everything else is black.



Community Commands

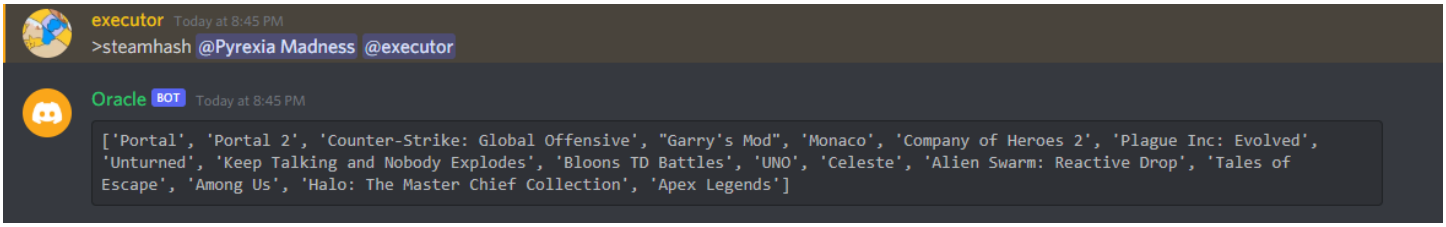
These are the commands for doing things as a group and actually have some of the most interesting coding behind them. Without further ado:

13. **Curse:** This first command is something I created when I realized I could have the bot add reactions to messages. As I mentioned above, reactions are the little emojis that get put underneath a message when someone clicks the "react" button with an emoji. These things can also be annoying when misused. So, the curse command makes it so the bot adds all the reaction emojis it can to a message, and then does that to every message sent by the cursed person. See example below:



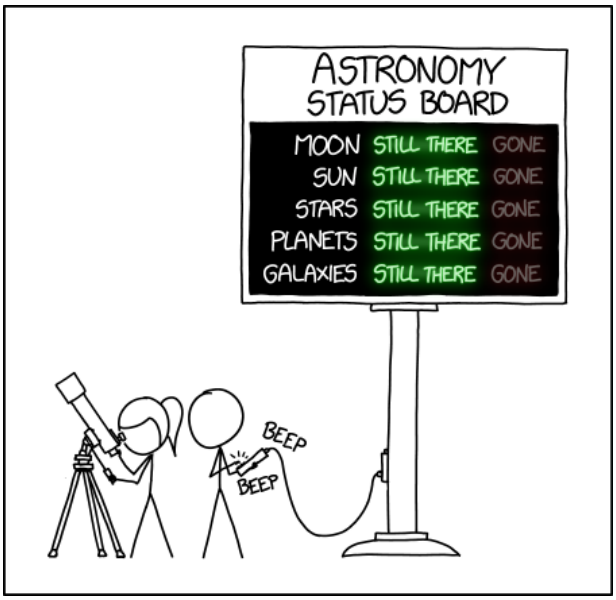
Something I didn't mention earlier is that Discord lets you upload custom emojis to a server. Oracle naturally uses these emojis first whenever it goes to enact the curse. The great part about the curse, of course, is that someone who is cursed cannot un-curse themselves - someone else has to do it.

14. **Steamhash:** To explain the Steamhash command, I first must give a brief overview of Steam to anyone who is not familiar. Steam is a platform used to play video games - a launcher of sorts that allows you to buy games from developers and play them with your friends. Steam is a massive part of playing games on a PC and is used by nearly everyone I know. Steamhash is a method to determine who can play what games together. To use Steamhash, you first have to call the "**steam_register**" command and give the bot your Steam ID - this is so it can link your Discord name and your Steam name and be able to find the games you own on Steam. Then, as in the example below, you can call the steamhash command with a series of names and it will return just the games that everyone named can play. The best part about this is that once you are registered, any new games you get are automatically added to the list.



A Few More...

These are a few more commands I didn't have a great spot for in the above categories. I still think they're pretty neat, and want to share them with you.



15. **xkcd:** If you're not familiar, xkcd is a webcomic published by Randall Munroe about math, science, and the internet at large. His comics are pretty good, so I decided to make a command that gets them from his website and sends them in the chat. *xkcd.com is licensed under a Creative Commons Attribution-NonCommercial 2.5 License.*
16. **Dilbert/Garfield**: Because I had nothing else to do that day, I made similar commands for the Dilbert and Garfield comics as well, from their respective websites. I'm not going to put any of those here, as I am confident you are familiar with them and the copyright rules are less clearly stated as with xkcd.
17. **Flickr:** Flickr is an image-sharing website intended for artists and photographers, and has an interface that I can easily use Python to interact with. Thus, Oracle can search Flickr for images, tags, and even geotagged locations with latitude and longitude. Largely because of copyright reasons I do not have a great example to demonstrate with, but it is a fun command to play around with nonetheless.
18. **Dailies:** Comics like Dilbert and Garfield are posted on their websites every day. xkcd is posted three days a week. What if you wanted these comics delivered to your Discord Server every day, automatically? Thus, the dailies command. I won't explain it in detail because it's boring, but the basic gist is this: You, the end user of the bot, without assistance from me, the creator, can configure Oracle

to post whichever of those three comics you want sent to a specific chat in your Discord server every day (this functionality could also be expanded to include more comics like Foxtrot or Calvin and Hobbes). The dailies command expands the bot from something simple that my friends and I mess around with, to something that could be used by anyone. Discord easily allows you to add bots you have no connection with to your servers. This means that anyone could invite Oracle to their server and have it perform everything I have told you about. I think that's pretty cool.

Thanks for reading! If you're interested in learning more about Oracle, you can reach me at zacharyhcampbell@gmail.com and I will be happy to email you back.

Write-Up Questions:

1. **What was your exhibit goal(s)? (What did you plan to learn or do?)**

My goal for this project was to create a Discord Bot in the programming language Python using the Discord.py library. I wanted the bot to be able to modify and edit images. My goal then expanded to include using Internet APIs (Application Programming Interfaces) to do things like retrieve the weather forecast and images. I did not have much of a time limit for this project, but I was very dedicated especially early on and that allowed me to get a lot done. Being homeschooled, this was my main Python project for the last month or two of school.

2. **What steps did you take to learn or do this?**

The first source I used for this was a series of Youtube videos by the channel Carberra Tutorials. The first 12 of these were very helpful. After those, I decided to try to make it work on my own. The Discord.py documentation (https://discordpy.readthedocs.io/) was also extremely helpful. The last source I used a few times was the Discord.py Discord Server, which had some helpful people who were willing to answer questions I had when I got stuck. For a more detailed explanation of what exactly I did, see the included paper.

3. **What were the most important things you learned?**

The most important thing I learned was definitely about Internet APIs and programming using Python to get information off of the web. I didn't know much about that at all before starting this project and my goals were very helpful in forcing me to figure out how to achieve them. I also learned a lot about image manipulation and writing functions to perform modifications on images from scratch. That was a good challenge and a lot of fun to figure out on my own.