

VR Code-Off!

May 2019

About the Code-Off	1
Event format	1
<i>What we will build</i>	1
Rules/guidelines	2
Preparation for the Code-Off	3
Programming languages and libraries	3
<i>Choice of language</i>	3
<i>Discord API libraries</i>	3
Creating a Discord bot	4
<i>Creating an application and token</i>	4
<i>Inviting your bot to the VR Code-Off server</i>	4
<i>Permissions</i>	5
Requirements	6
Command format	6
<i>Commands that accept arguments</i>	6
Errors	7
<i>Error handling</i>	7
Functions	8
Calculation function	9
<i>Extension</i>	9
<i>Super extension</i>	9
Help function	10
<i>Extension</i>	10
Message tools	11
<i>Swear filter</i>	11
<i>What's hot?</i>	11
Roles function	12
<i>Extension</i>	13
System information function	14
<i>Extension</i>	14

Unit converter

15

Extension

15

About the Code-Off

The Code-Off is an informal event to practice programming, to explain to and learn from each other in order to develop understandings of algorithms, best programming practices, and to encourage examining different ways of approaching similar tasks.

This is not a competition: we can (and it is heavily encouraged that participants should) work together to derive better understandings of what is required, building each other's understanding of the nature of the problems and solutions.

Participation is voluntary and there is no prize.

Event format

The event will take place on the weekend of the 25th to 26th of May, 2019, starting at 10am and finishing at 2pm Australian Eastern Standard Time (8pm to 12am the night before, Eastern Daylight Time). Some participants may stream their progress online through video sharing sites such as Twitch.

AEST Start	AEST Finish	EDT Start	EDT Finish
25/09 10:00	25/09 14:00	24/09 20:00	25/09 00:00
26/09 10:00	26/09 14:00	25/09 20:00	26/09 00:00

What we will build

All participants will independently (or, with prior arrangement, as a pair) build a Discord bot that is capable of joining a given Discord server, respond to user input, and provide correct output.

There are three mandatory requirements that the bot of each participant/pair should implement. The bot should also handle errors gracefully. Further details are given in the next section, 'Requirements'.

Rules/guidelines

As the Code-Off is a friendly event, there aren't really any formal rules. However, please keep the following guidelines in mind:

- Participants may have more or less programming experience than others. Please don't denigrate participants who may be struggling with a problem that you find easy; rather, try to be helpful and offer suggestions.
 - Please do not be offended if participants choose not to heed your suggestions: they may wish to try solving the problem in their own way.
 - On the other hand, please do not be offended if someone offers you a suggestion. Assume that people are trying to be helpful.
- Not everybody is a natural-born programmer. Some participants may work professionally with code but not as a hobby; don't be discouraging to those who are unfamiliar with certain workflows that don't come up in the course of their professional work.
- You may have a preferred way of tackling a problem; however, that doesn't make your solution automatically better than someone else's. It is preferable to have a discussion about how your approaches might differ than to denigrate other peoples' code.
- Not every programming language or Discord library may be conducive to your personal or preferred way of coding. Please be aware of this when providing advice to other participants; what you may perceive as an ugly hack might be the only way to accomplish something in a given language!

Preparation for the Code-Off

Programming languages and libraries

Choice of language

It is encouraged that each participant/pair should use a different programming language from the other users. However, this is not mandatory; please feel free to use the programming language with which you are most comfortable. That said, you may also feel free to challenge yourself with an unfamiliar language if you so desire.

Discord API libraries

Please be advised that creating a Discord bot is easiest when done in conjunction with an existing library that exposes the Discord API in an idiomatic manner. Though Discord has a list of officially-vetted libraries, please be aware that they all vary in terms of API coverage and functionality; some libraries in the same language may have significantly different approaches.

You may find the list of libraries at the following web address:

<https://discordapp.com/developers/docs/topics/community-resources#libraries>

There are also other libraries not in that list that are available via the Discord API server. Note that this is an unofficial server but is where the bulk of discussion around the API and libraries occurs. You may access it at the following web address:

<https://discordapp.com/invite/discord-api>

You may find other libraries for other languages through a Google search.

Creating a Discord bot

Creating an application and token

In order to create a bot, you need to create a Discord application on the Discord Developer Portal web site. The following instructions detail how to do this:

1. Access the Discord Developer Portal website at the following web address:
<https://discordapp.com/developers/applications/>
2. At the top-right, click "New Application"
3. When asked for the application name, give it the name you wish for your bot.
4. On the left, click Bot.
5. On the right, click "Add Bot". Then, click "Yes, do it!"
6. Click on "Click to Reveal Token"
7. Copy and paste the revealed token into a file. You will need to use this in your bot's code when connecting to Discord.
 - This token is private. Don't reveal it to anybody. If you use versioning software that sends your code to a public repository, such as to Github or Sourceforge, be sure to not commit any files with your token in them.
 - If you are streaming, do not show this on stream! If you do reveal it by accident, come back to the Discord Developer Portal and regenerate the token.

Inviting your bot to the VR Code-Off server

You should invite your bot to the VR Code-Off server. That way, all participants can see the fruits of your coding, help out, and any Twitch viewers can also come help.

All participants have been provided a sandbox channel. Please test your bot commands in your respective channels.

Only TDNZap can invite bots to the server. You will need to provide TDNZap with the invitation link for your bot. To get the link:

1. Go to the Discord Developer Portal.
2. Click on your bot application.
3. On the left, make sure General Information is selected.
4. Copy and paste the Client ID to TDNZap

Permissions

Your bot will be added to the server with the following permissions:

- Read text channels
- Send messages
- Send TTS messages
- Manage messages
- Embed links
- Attach files
- Read previous messages
- Use external emoji
- Add reactions
- See voice channels
- Connect
- Speak

Be aware that the bot may not have these permissions in every channel.

Requirements

Your Discord bot must implement at least three functions, detailed in the subsection 'Required functions'.

Command format

All bots in the Code-Off should use the same trigger for commands, a single exclamation mark **!** followed by the name of the command. Command names should be strings containing alphanumeric characters only. For example, if the user wishes to use a command called 'help', the command would be triggered as follows:

```
!help
```

Commands should **not** be case-sensitive. 'Help' and 'help' should resolve to the same command.

Commands that accept arguments

If the command takes arguments, these should be separated by spaces. For example, if a user wishes to use a command called 'help' that takes the name of another command 'stats' as an argument, the command would be triggered as follows:

```
!help stats
```

Positive numbers should be formatted following normal programming conventions; integers are a string of Arabic numerical characters, fixed or floating point numbers containing a period character **.** with an arbitrary number of digits afterwards. For example, if the user wishes to use a command called 'double' to multiply a given number by 2, the command would be triggered as follows:

```
!double 14
```

```
!double 14.576
```

Negative numbers are prefixed with a minus symbol **-** but are otherwise formatted like positive numbers. For example, if the user wishes to use a command called 'multiply' to multiply a given number by another, and at least one of the inputted numbers is negative, the command would be triggered as follows:

```
!multiply 2 -24
```

```
!multiply -53.0 -7
```

Errors

Your bot should gracefully handle errors. That is, your finished product should show evidence that the bot does not crash.

In order to help your code be more robust and resilient against errors and potential crashes, consider the following scenarios:

- a user triggers a command with improper arguments, including:
 - numbers out of range
 - malformed strings
 - incorrect types
- a user or the bot does not have sufficient permissions to perform a certain action
- sudden loss of connection to Discord's servers

Error handling

In the event that an error does occur, your bot should:

- inform users that an error occurred, if appropriate (such as in the case of a bad response to a user-triggered command, incorrect input, or a network error)
- provide a diagnosis and possible means of correction, especially if the error comes from bad user input
 - for example, if a bot command expects three arguments but only two are supplied, the bot should inform the user of the expected number of arguments

Functions

These functions have been suggested by participants in the Code-Off. It is expected that all participants attempt to implement these, but there is no penalty if anybody is unable to do so for any of these functions.

Each function offers a particular set of challenges. Even if the function has no particular pertinence to you, give it a shot: the aim is to demonstrate or learn about ways to tackle the task!

Other participants may suggest functions that all our bots should implement. Please check the VR Code-Off server #chat channel for details.

Calculation function	9
Help function	10
Message tools	11
Roles function	12
System information function	14
Unit converter	15

The following table outlines the functions that all participants/pairs should implement in their bots. Other functions to implement may be agreed between all participants/pairs during or prior to the competition but are *in addition* to the following functions.

These functions have been chosen to allow participants to demonstrate their approach to the following areas:

- String manipulation
- Personal choice of data structures
- Working with unfamiliar or arbitrary data structures
- Persisting and retrieving data in a database/file

Calculation function

Hint: this is a simple starter command — some basic maths, string manipulation, and error handling!

Your bot must implement a <code>calc</code> function that allows users to perform a maths operation on exactly two numbers.	<code>!calc 1 + 2</code>	3
The only operators allowed are: + addition - subtraction * multiplication / division	<code>!calc 24 * 3.3</code>	79.2
	<code>!calc 5 - 9</code>	-4
	<code>!calc 10 / 2.0</code>	5.0
	<code>!calc 1 / 0</code>	Error: invalid calculation
Do not allow standard form (scientific notation) input, only plain integers or decimal numbers. However, you may <i>output</i> to standard form.	<code>!calc 6.022e23 - 128</code>	Error: invalid format. For more information, type <code>!help calc</code>
The numbers and operator must be separated by spaces.	<code>!calc 1+1</code>	Error: invalid format. For more information, type <code>!help calc</code>
Do not allow text of any kind to be passed as arguments.	<code>!calc banana</code>	Error: invalid format. For more information, type <code>!help calc</code>
Always require two numerical arguments.	<code>!calc</code>	Error: invalid format. For more information, type <code>!help calc</code>

Extension

Allow for an arbitrary number of numbers and operators, but only allow left associativity — BEDMAS does not apply in this command.

Super extension

Make BEDMAS apply!

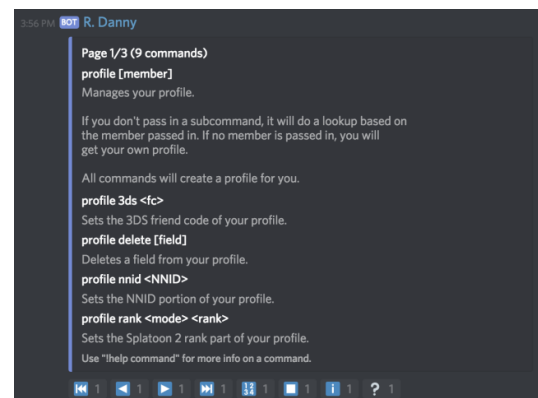
Help function

Hint: think about the design of your command setup — how can you flexibly retrieve command information?

Your bot must implement a help function that outputs a list of all available command names in alphabetical order.	<code>!help</code>	<pre>Commands:\n - aCommand\n - bot\n - calc\n - help\n - zoot\n</pre>
This command takes one optional argument, a string containing the name of another command.	<code>!help help</code>	<pre>**help**:\n Shows information on available commands.\n \n ``!help [command]``\n \n ``[command]`` *(optional)*:\n the name of a command</pre>
The command then outputs a description of the command and required/optional inputs in expected order.	<code>!help calc</code>	<pre>**calc**:\n Performs a math calculation on two numbers.\n \n ``!calc <numberA> <operator> <numberB>``\n \n ``<numberA>``: a valid integer or decimal number\n ``<operator>``: a maths operator (``+``, ``-``, ``*``, ``/``)\n ``<numberB>``: a valid integer or decimal number</pre>
Return an error message if an invalid command is specified	<code>!help banana</code>	Error: invalid command

Extension



Format the output of your command as a Discord Embed with multiple pages, similar to R Danny (see example, right). Each page must contain only up to five fields each. The user should be able to navigate through the pages either by clicking on reactions (as per R Danny) or entering a command — the spelling and arguments for this command are at your discretion.



Message tools

Hint: for these bots, you will need to handle the CREATE MESSAGE event.

Swear filter

Your bot must delete messages containing swear words, then repost the message with the swear words censored using • symbols for all but the first letter.	someuser21: holy fuck, that hurt!	**someuser21**: holy f•••, that hurt!
You must censor the following words: - fuck - shit - cunt - arse	Champ: Where did you hear that shit?	**Champ**: Where did you hear that s•••?
You must also censor variants such as 'fucking', 'fucked', leaving prefixes/suffixes in place and only censoring the main part of the word	 TennisFan98: that's fucking bullshit	**  TennisFan98**: that's f•••ing bulls•••

Extension: instead of replacing the swear word with • symbols, generate !@#\$-style swear words.

What's hot?

Your bot must implement a whatshot command that posts the most frequently-used word in the past 24 hours.	!whatshot	The hot topic is: ``banana``
The command must persist data across bot instances. That is, if you restart the bot, it should continue to know what words have been used previously.	!whatshot (after the bot restarted)	The hot topic is: ``banana``
Words are defined as substrings of a Discord Message content, separated by a space. Words must contain alphabetical characters from the Latin alphabet.	!whatshot	NOT The hot topic is: ``12`` NOT The hot topic is: ``?`` NOT The hot topic is: ``👏👏👏``

Extension: instead of revealing just one word, reveal three separate words.

The hot topics are: ``banana``, ``apple``, and ``fruit``

Roles function

Hint: this function is going to require a lot of error checking. Think about what checks you will need to do, what will happen when an error occurs, and how Discord's role-based permission system works.

Your bot must implement a roles command that gives a Discord Role to a user, and allows an admin user to specify what roles are permitted.	<code>!roles</code>	<code>**Available roles**:\n- twitch_subscriber\n- twitch_follower\n- viewer</code>
The arguments for the command are actually subcommands: - giveme <role>: gives a role/group to the caller - giveto <user> <role>: gives a role/group to a specified user	<code>Someguy23: !roles giveme viewer</code>	<code>*Someguy23 received the viewer role*</code>
Roles with spaces in their actual names should be referred to with underscores when used in a command. Users should be able to refer to the role names in lower case.	<code>Role name: Twitch Subscriber</code>	<code>In commands, users can type: twitch_subscriber</code>
giveto's <user> refers to a highlight (when you type @ and select someone's name), not a username or nickname		<code>Highlight without nickname: <@87118368078835712></code> <code>The numbers are a person's snowflake (identifier)</code>
Don't allow users to receive a role that isn't in the list of accessible roles.	<code>Someguy23: !roles giveme not_in_list</code>	<code>Error: inaccessible role. Check the list of accessible roles with !roles</code>
If your bot lacks the Manage Members permission, it should alert the user that it cannot assign a role.	<code>Someguy23: !roles giveme viewer</code>	<code>Error: insufficient permission to give role to Someguy23</code>
	<code>Orange: !roles giveto <@87118368078835712> viewer</code>	<code>*👉TDNZap received the viewer role from Orange*</code>
If the bot's role is lower than the receiver's role in the role hierarchy, the bot won't have permission to add a role.	<code>Orange: !roles giveto <@87118368078835712> viewer</code>	<code>Error: insufficient permission to give role to 👉TDNZap</code>
Alert the caller when the specified recipient doesn't exist.	<code>Orange: !roles giveto <@87118368078835712> viewer</code>	<code>Error: specified recipient does not exist in this server</code>

There are also admin-only commands: <ul style="list-style-type: none"> - addrole <role>: adds a role to the list of roles for which a user can ask - remrole <role>: removes a role from the list of roles for which a user can ask 	Admin: !roles addrole viewer	*Admin added viewer to accessible roles*
Only allow adding a role if it already exists in the server.	!roles addrole truffle_spud	Error: no such role 'truffle_spud' in server roles
	Admin: !roles remrole twitch_subscriber	*Admin removed twitch_subscriber from accessible roles*
Make sure to alert the user when they mistype a role name.	!roles addrole twitch_vader	Error: no such role 'twitch_vader' in accessible roles

Extension

Add a new set of subcommands to the roles command.		
addgroup <group> ([!]<role>...) This subcommand will create a role group called <group>. This subcommand is followed by an indefinite number of roles.	Admin: !roles addgroup sub twitch_subscriber viewer	*Admin created a role group called 'sub': - twitch_subscriber\s - viewer*
! before a role indicates that if a user asks for a role group and already has the role prepended by the !, then that role must be removed . These roles and ones without ! are mutually exclusive.	Admin: !roles addgroup sub twitch_subscriber !twitch_follower viewer	*Admin created a role group called 'sub': - twitch_subscriber\s - viewer\ This group will remove: - twitch_follower*
remgroup <group> This subcommand deletes the group altogether.	Admin: !roles remgroup sub	*Admin removed the group 'sub'*
Extend the giveme and giveto subcommands to allow giving/receiving groups — however, the group name must be prepended by g: This changes the syntax to: <ul style="list-style-type: none"> - giveme [g:]<role_or_group> - giveto <user> [g:]<role_or_group> 	Someguy23: ! roles giveme g:sub	*Someguy23 received the 'twitch_subscriber' and 'viewer' roles — and the 'twitch_follower' role was removed*
Of course, do the necessary error checking: don't let users ask for groups that don't exist, make sure the bot has permission to add the roles, etc.		

System information function

Hint: you will need to look into the creation of Discord Embeds. There are websites that help you to visualise Embeds; these websites often provide a code snippet as well.

Your bot must implement a sysinfo function that outputs information about the system on which the bot is running.														
<p>The output should contain the following information:</p> <ul style="list-style-type: none">- operating system name- kernel version/Windows build no.- system uptime in days- available RAM/total RAM- boot disk name & storage capacity <p>The formatting (titles, bold, italics, etc.) is up to you, however the output must be in the form of a Discord Embed.</p>	<code>!sysinfo</code>	<div><p>Embed Preview</p><div><div>System information</div><table><tr><td>Operating system</td><td>Kernel version</td></tr><tr><td>macOS 10.14.5</td><td>18.5.0</td></tr><tr><td>Uptime</td><td>RAM</td></tr><tr><td>Less than 1 day</td><td>19/32 GB</td></tr><tr><td>Storage (Macintosh HD)</td><td></td></tr><tr><td>1.04 TB</td><td></td></tr></table></div></div>	Operating system	Kernel version	macOS 10.14.5	18.5.0	Uptime	RAM	Less than 1 day	19/32 GB	Storage (Macintosh HD)		1.04 TB	
Operating system	Kernel version													
macOS 10.14.5	18.5.0													
Uptime	RAM													
Less than 1 day	19/32 GB													
Storage (Macintosh HD)														
1.04 TB														

Extension

Add information relating to graphics — the GPU name and VRAM amount, desktop resolution, and scaling factor (on HiDPI displays) expressed as a percentage.

Unit converter

Implement a from function that converts a number from one unit of measurement to another.	+from 10 cm to m	0.1 m
<p>The function should be able to make the following conversions:</p> <ul style="list-style-type: none"> - distance <ul style="list-style-type: none"> - Metric: mm, cm, m, km - US: in, ft, yd - volume <ul style="list-style-type: none"> - Metric: mL (aka cc), L - US: oz (fluid), qt, gal - temperature <ul style="list-style-type: none"> - c - f - k 	+from 5 m to yd	5.47 yd
<p>The command should take four arguments separated by spaces.</p> <ol style="list-style-type: none"> 1. a number (integer, decimal) 2. a unit symbol 3. the word 'to' 4. another unit symbol 	+from 0 c to f	0 f
It is an error to convert between types of measurement. For example, one cannot convert from centimetres to fluid ounces!	+from 29.1 cm to oz	Error: cannot convert distance to volume. For more information, type !help from
As with the calc function, check that the numbers are always in the correct format. You may wish to share code between these two functions.	+from banana L to gal	Error: invalid format. For more information, type !help from
Make sure both unit symbols are specified.	+from 63 k to	Error: invalid format. For more information, type !help from
Unit symbols must be specified all lowercase. One exception: mL and L must contain a capital L.	+from 1 l to qt	Error: invalid format. For more information, type !help from

Extension

Add currency conversion. You can decide how you fetch the conversion rates. Only allow the following currencies: **aud**, **cad**, **gbp**, **nzd**, **usd**, and **yen**.