Google Coding Challenge

Bright Network Internship, July 2021

Overview

Thank you for choosing to take part in Google's Coding Challenge for Bright Network, we're very happy to have you here. We hope you have as much fun implementing this challenge as we had creating it!

In this challenge you are going to implement various components of a command-line application simulating YouTube. The result of your code will be a fully working command-line application, simulating how a user would interact with YouTube in real life but without any of the web elements. This is a big part of what a software engineer would be doing at Google, just on a much greater scale.

Your interactions with the tool will look something like below. Note that the $ is to emphasize the fact that this is done in a shell and is not part of the actual command nor the output.

```
$ ./youtube // Start the YouTube application. This will vary depending on your
            // chosen language.
Hello and welcome to Youtube. Please type a command. // [...] truncated for brevity
YT> HELP
Here is a list of commands:
  NUMBER_OF_VIDEOS
     Shows how many videos are in the library.
  SHOW_ALL_VIDEOS
     Lists all videos from the library.
  PLAY <video_id>
     Plays specified video.
  PLAY_RANDOM
     Plays a random video from the library.
[...] // output truncated for brevity

YT> SHOW_ALL_VIDEOS
Here's a list of all available videos:
  Amazing Cats (amazing_cats_video_id) [#cat #animal]
  Another Cat Video (another_cat_video_id) [#cat #animal]
  Funny Dogs (funny_dogs_video_id) [#dog #animal]
[...] // output truncated for brevity

YT> PLAY amazing_cats_video_id
Playing video: Amazing Cats
```

How to get started

The starting code will be provided as well as pre-written unit tests you can use to verify your progress as you work through this challenge. Keep in mind that in the beginning, all tests will fail, as you have not yet implemented any of the required commands. This approach is also called TDD ("Test Driven Development"). Don't worry about your code-quality while you first attempt this challenge. The idea of this project is to get something working and you can always come back to it and improve it later on.

To get started, choose your preferred language and follow the instructions below. You can choose to implement this challenge in Java, Python or C++.

You can find the starting code here:
https://github.com/internship-experience-uk/google-code-sample

Java

Project link: https://github.com/internship-experience-uk/google-code-sample/tree/main/java

How to get set up:

```
$ git clone https://github.com/internship-experience-uk/google-code-sample.git
$ cd google-code-sample/java
```

The java/ directory is considered to be the root directory if you are choosing to attempt this challenge in Java. You will be required to have Java 11 installed on your machine as well Maven. You will also require JUnit 5.4, which will automatically be installed by Maven as soon as you run the tests.

You can find more information on how to get setup and run the application as well as run the tests in the README that is part of the repository.

Python

Project link: https://github.com/internship-experience-uk/google-code-sample/tree/main/python

How to get set up:

```
$ git clone https://github.com/internship-experience-uk/google-code-sample.git
$ cd google-code-sample/python
```

The python/ directory is considered to be the root directory if you are choosing to attempt this challenge in Python. You will be required to have Python 3.7 or higher installed on your machine.

Furthermore, the tests are using pytest, which you will need to install using pip. If you installed Python from the official source, pip will already be installed together with your Python installation.

You can find more information on how to get setup and run the application as well as run the tests in the README that is part of the repository.

C++

Project link: https://github.com/internship-experience-uk/google-code-sample/tree/main/cpp

How to get set up:

```
$ git clone https://github.com/internship-experience-uk/google-code-sample.git
$ cd google-code-sample/cpp
```

The cpp/ directory is considered to be the root directory if you are choosing to attempt this challenge in C++. You will be required to have C++11 or higher installed on your machine. You can find more information on how to get setup and run the application as well as run the tests in the README that is part of the repository.

The starting code

The starting code consists of the following classes:

- Video: A representation of a video.
    - Some basic methods are already implemented but you can make changes here if you need to.
- VideoLibrary: A representation of a video library. This class parses the videos.txt file as Videos and keeps track of them. It provides ways to get all available videos as well as access video objects by their video id.
    - This class is already fully implemented and you are not expected to make any changes to it (you can, if you want to, though). You will be using this class in the video player.
- VideoPlayer: A representation of a video player. This class will contain most of the logic this challenge focuses on.
    - You will write most of the code here.
- VideoPlaylist: A representation of a playlist containing videos.
    - This class will need implementation as part of this challenge.
- CommandParser: The class that parses the commands.
    - There will be no changes required to this class, so you can feel free to ignore it.

As part of this challenge you will mostly edit the VideoPlayer class. However, some changes to the Video, VideoPlaylist and VideoLibrary classes will possibly be needed to achieve your

desired results. **Keep in mind that you can make changes wherever you want and you are not limited to implementing the methods layed out for you**. **You are free to also add new classes, methods and properties where required or even modify existing code as you please. While the challenge can be solved using the standard libraries for each language alone, you are free to use external libraries throughout your code - as long as the tests still pass.**

The tests only verify that the expected output is written to **the console (standard output)** by the video player. What you do with all the other classes is entirely up to you.

Commands

Below is a list of each of the commands to be implemented. Note that the commands are listed in increasing difficulty and grouped into parts that can be completed independently.

You are not required to implement all of the commands. Running the tests will give you feedback over your progress to 100% completion of the challenge (i.e. all commands implemented). We recommend that you run the relevant tests after implementing each command in order to verify your progress.

Please refer to the code examples in this document on what the printed output to each command should look like. They will also showcase what the warning messages should look like, should the command require any.

Part 1

The below commands are all related to video playing. You may want to get started by having a look at the CommandParser class.

NUMBER_OF_VIDEOS

This command shows how many videos are in the library. It is **already implemented** and is here to help you get started. Have a look at the VideoPlayer to see how this method works.

```
YT> NUMBER_OF_VIDEOS
5 videos in the library
```

SHOW_ALL_VIDEOS

This command will list all available videos in the format: “title (video_id) [tags]”. The videos should be shown in lexicographical order by title. If there are no tags available, display empty brackets.

```
YT> SHOW_ALL_VIDEOS
Here's a list of all available videos:
  Amazing cats (amazing_cats_video_id) [#cat #animal]
  Another Cat Video (another_cat_video_id) [#cat #animal]
  Funny Dogs (funny_dogs_video_id) [#dog #animal]
  Life at Google (life_at_google_video_id) [#google #career]
  Video about nothing (nothing_video_id) []
```

PLAY <video_id>

Play the specified video. If a video is currently playing, display a note that this video will be stopped, even if the same video is already playing. If the video doesn’t exist, display a warning message (and don’t stop the currently playing video).

```
YT> PLAY amazing_cats_video_id
Playing video: Amazing Cats

YT> PLAY funny_dogs_video_id
Stopping video: Amazing Cats
Playing video: Funny Dogs

YT> PLAY funny_dogs_video_id
Stopping video: Funny Dogs
Playing video: Funny Dogs

YT> PLAY some_other_video_id
Cannot play video: Video does not exist
```

STOP

Stop the current playing video. If no video is currently playing, display a warning message "Cannot stop video: No video is currently playing" and do nothing.

```
YT> PLAY amazing_cats_video_id
Playing video: Amazing Cats

YT> STOP
Stopping video: Amazing Cats

YT> STOP
Cannot stop video: No video is currently playing
```

PLAY_RANDOM

Play a random video. If a video is currently playing, display a note that this video will be stopped, even if the same video is already playing.

If there are no videos available, print out "No videos available". You can assume that the case where there are no videos available never happens, unless you get to Part 4 and implement flagging of videos.

```
YT> PLAY_RANDOM
Playing video: Life at Google

YT> PLAY_RANDOM
Stopping video: Life at Google
Playing video: Funny Dogs
```

PAUSE

Pause the current playing video. If a video is already paused, display a warning message and do nothing. Equally, If no video is currently playing, display a warning message and do nothing.

```
YT> PLAY amazing_cats_video_id
Playing video: Amazing Cats

YT> PAUSE
Pausing video: Amazing Cats

YT> PAUSE
Video already paused: Amazing Cats

YT> STOP
Stopping video: Amazing Cats
```

```
YT> PAUSE
Cannot pause video: No video is currently playing
```

If a new video is played while another one is paused, the new video should be playing and ignore the fact that the previous video was in paused status. This also means that the new video is pauseable again.

```
YT> PLAY amazing_cats_video_id
Playing video: Amazing Cats

YT> PAUSE
Pausing video: Amazing Cats

YT> PLAY another_cat_video_id
Stopping video: Amazing Cats
Playing video: Another Cat Video

YT> PAUSE
Pausing video: Another Cat Video
```

CONTINUE

Continues a currently paused video. If the currently playing video is not paused, display a warning message and do nothing. If no video is playing at all, also display a warning message and do nothing.

```
YT> PLAY amazing_cats_video_id
Playing video: Amazing Cats

YT> CONTINUE
Cannot continue video: Video is not paused

YT> PAUSE
Pausing video: Amazing Cats

YT> CONTINUE
Continuing video: Amazing Cats

YT> CONTINUE
Cannot continue video: Video is not paused

YT> STOP
Stopping video: Amazing Cats

YT> CONTINUE
Cannot continue video: No video is currently playing
```

SHOW_PLAYING

Displays the title, video_id, video tags and paused status of the video that is currently playing. If no video is currently playing, display a message.

```
YT> PLAY amazing_cats_video_id
Playing video: Amazing Cats

YT> SHOW_PLAYING
Currently playing: Amazing Cats (amazing_cats_video_id) [#cat #animal]

YT> PAUSE
Pausing video: Amazing Cats

YT> SHOW_PLAYING
Currently playing: Amazing Cats (amazing_cats_video_id) [#cat #animal] - PAUSED

YT> STOP
Stopping video: Amazing Cats

YT> SHOW_PLAYING
No video is currently playing
```

Time to run the tests

At this point you might want to run the tests for Part 1. If they all pass, you have completed this part.

Part 2

The below commands are all related to playlist management. You don't need to have completed Part 1 to complete this part.

CREATE_PLAYLIST <playlist_name>

Create a new (empty) playlist with a unique name. If a playlist with the same name already exists, display a warning to the user and do nothing.

Note: The playlist name should be a string with no whitespace. Furthermore, the playlist name should not be case sensitive, e.g. "My_playlist" and "my_PLAYLIST" should be considered the same playlist. You should always respond to the user with the same case they entered in the command, even if it differs from the case the playlist was created with. **However, internally you should still keep the case the playlist was originally created with, as you will need this for commands that are introduced later on.**

```
YT> CREATE_PLAYLIST my_PLAYlist
Successfully created new playlist: my_PLAYlist

YT> CREATE_PLAYLIST my_PLAYLIST
Cannot create playlist: A playlist with the same name already exists
```

ADD_TO_PLAYLIST <playlist_name> <video_id>

Adds the specified video to a playlist. If either the video or the playlist don't exist, show a warning message. If both don't exist, display the warning message for the playlist first. The playlist should not allow duplicate videos and display a warning message if a video is already present in the playlist. As the name is not case sensitive, always use the same case in the response as the user entered in their command.

```
YT> CREATE_PLAYLIST my_playLIST
Successfully created new playlist: my_playLIST

YT> ADD_TO_PLAYLIST my_playlist amazing_cats_video_id
Added video to my_playlist: Amazing Cats

YT> ADD_TO_PLAYLIST my_PLAYlist amazing_cats_video_id
Cannot add video to my_PLAYlist: Video already added

YT> ADD_TO_PLAYLIST my_playlist some_other_video_id
Cannot add video to my_playlist: Video does not exist

YT> ADD_TO_PLAYLIST another_playlist some_other_video_id
Cannot add video to another_playlist: Playlist does not exist
```

SHOW_ALL_PLAYLISTS

Show all the available playlists (name only). if the list is empty, display the user friendly message "No playlists exist yet". The playlists should be shown in lexicographical order by playlist name. You can ignore the case of the playlist when sorting, so it doesn't matter if you display "A_playlist" or "a_playlist" first. **The playlist names should be displayed in the same case that they were originally created in the CREATE_PLAYLIST command.**

```
YT> SHOW_ALL_PLAYLISTS
No playlists exist yet

YT> CREATE_PLAYLIST MY_playlist
Successfully created new playlist: MY_playlist

YT> SHOW_ALL_PLAYLISTS
Showing all playlists:
  MY_playlist
```

SHOW_PLAYLIST <playlist_name>

Show all the videos in the specified playlist in the following format: "title (video_id) [tags]". If the playlist doesn't exist, display a warning message. If the playlist is empty, display "No videos here yet" instead of a video list. The videos should be listed in the same order they were added.

As the name is not case sensitive, use the playlist name in the same case that the user entered when printing the response message, even if the case is different from the original.

```
YT> CREATE_PLAYLIST my_playlist
Successfully created new playlist: my_playlist

YT> SHOW_PLAYLIST my_PLAYLIST
Showing playlist: my_PLAYLIST
  No videos here yet

YT> ADD_TO_PLAYLIST my_playlist amazing_cats_video_id
Added video to my_playlist: Amazing Cats

YT> SHOW_PLAYLIST my_playlist
Showing playlist: my_playlist
  Amazing Cats (amazing_cats_video_id) [#cat #animal]

YT> SHOW_PLAYLIST another_playlist
Cannot show playlist another_playlist: Playlist does not exist
```

REMOVE_FROM_PLAYLIST <playlist_name> <video_id>

Remove the specified video from the specified playlist. If either does not exist, display a relevant warning message. In all cases, check for the playlist existence first: This means if both don't exist, you'll be printing the message that the playlist doesn't exist. If the video is not in the playlist to begin with, display a warning message and do nothing.

When printing the response message, keep the case of the playlist name as the user entered it. Remember that it's still the same playlist, as the name should be handled the same, irrespective of case.

```
YT> CREATE_PLAYLIST my_playlist
Successfully created new playlist: my_playlist

YT> ADD_TO_PLAYLIST my_PLAYlist amazing_cats_video_id
Added video to my_PLAYlist: Amazing Cats

YT> REMOVE_FROM_PLAYLIST my_playLIST amazing_cats_video_id
Removed video from my_playLIST: Amazing Cats

YT> REMOVE_FROM_PLAYLIST my_playlist amazing_cats_video_id
Cannot remove video from my_playlist: Video is not in playlist

YT> REMOVE_FROM_PLAYLIST my_playlist some_other_video_id
Cannot remove video from my_playlist: Video does not exist

YT> REMOVE_FROM_PLAYLIST another_playlist amazing_cats_video_id
Cannot remove video from another_playlist: Playlist does not exist

YT> REMOVE_FROM_PLAYLIST another_playlist some_other_video_id
Cannot remove video from another_playlist: Playlist does not exist
```

CLEAR_PLAYLIST <playlist_name>

Removes all the videos from the playlist, but doesn't delete the playlist itself. If the playlist doesn't exist, display a warning message.

```
YT> CREATE_PLAYLIST my_playlist
Successfully created new playlist: my_playlist

YT> ADD_TO_PLAYLIST my_playlist amazing_cats_video_id
Added video to my_playlist: Amazing Cats

YT> CLEAR_PLAYLIST my_playlist
Successfully removed all videos from my_playlist

YT> SHOW_PLAYLIST my_playlist
Showing playlist: my_playlist
```

```
  No videos here yet.

YT> CLEAR_PLAYLIST another_playlist
Cannot clear playlist another_playlist: Playlist does not exist
```

DELETE_PLAYLIST <playlist_name>

Delete the specified playlist. Display a warning if the playlist doesn't exist.

```
YT> CREATE_PLAYLIST my_playlist
Successfully created new playlist: my_playlist

YT> DELETE_PLAYLIST my_playlist
Deleted playlist: my_playlist

YT> DELETE_PLAYLIST my_playlist
Cannot delete playlist my_playlist: Playlist does not exist
```

Time to run the tests

At this point you might want to run the tests for Part 2. If they all pass, you have completed this part.

Part 3

The below commands are related to searching videos. You do not need Part 1 and Part 2 to complete this part. However, it will be helpful if you have completed the PLAY command outlined in Part 1.

SEARCH_VIDEOS <search_term>

Display all videos in the library whose titles contain the specified search term. For example, if the user searches for “cat”, all videos whose titles contain “cat” would be returned: [Amazing Cats]. The search should not be case sensitive, so even if the user searches for “CaT”, the video “Amazing Cats” should be displayed. For simplicity, the search term cannot contain any whitespace or special characters.

Display a list in lexicographical order (by title) and ask the user if they’d like to play one of the videos. Read in the answer from the standard input (check the main function of your language for an example on how we do this to read in the commands) - If the answer is a valid number (one of the listed videos) then play that video, otherwise assume the answer is no and do nothing.

If there are no search results, display a nice message.

```
YT> SEARCH_VIDEOS cat
Here are the results for cat:
  1) Amazing Cats (amazing_cats_video_id) [#cat #animal]
  2) Another Cat Video (another_cat_video_id) [#cat #animal]
Would you like to play any of the above? If yes, specify the number of the video.
If your answer is not a valid number, we will assume it's a no.
Nope!

YT> SEARCH_VIDEOS cat
Here are the results for cat:
  1) Amazing Cats (amazing_cats_video_id) [#cat #animal]
  2) Another Cat Video (another_cat_video_id) [#cat #animal]
Would you like to play any of the above? If yes, specify the number of the video.
If your answer is not a valid number, we will assume it's a no.
2
Playing video: Another Cat Video

YT> SEARCH_VIDEOS blah
No search results for blah
```

Note that the sentences “Would you like to play....” and “If your answer is not a valid number...” are on separate lines.

SEARCH_VIDEOS_WITH_TAG <tag_name>

Show all videos whose list of tags contains the specified hashtag. The search should not be case sensitive, so even if the user searches for "#CaT", the video "Amazing Cats" should be displayed.

Just like above, display a list in lexicographical order by title and ask the user if they'd like to play one of the videos. Read in the answer - If the answer is a valid number (one of the listed videos) then play that video, otherwise assume the answer is no and do nothing.

If there are no search results, display a nice message. This should also be the case if the user forgets the hashtag.

```
YT> SEARCH_VIDEOS_WITH_TAG #cat
Here are the results for #cat:
  1) Amazing Cats (amazing_cats_video_id) [#cat #animal]
  2) Another Cat Video (another_cat_video_id) [#cat #animal]
Would you like to play any of the above? If yes, specify the number of the video.
If your answer is not a valid number, we will assume it's a no.
No

YT> SEARCH_VIDEOS_WITH_TAG #cat
Here are the results for #cat:
  1) Amazing Cats (amazing_cats_video_id) [#cat #animal]
  2) Another Cat Video (another_cat_video_id) [#cat #animal]
Would you like to play any of the above? If yes, specify the number of the video.
If your answer is not a valid number, we will assume it's a no.
2
Playing video: Another Cat Video

YT> SEARCH_VIDEOS_WITH_TAG #blah
No search results for #blah

YT> SEARCH_VIDEOS_WITH_TAG cat
No search results for cat
```

Time to run the tests

At this point you might want to run the tests for Part 3. If they all pass, you have completed this part.

Part 4

The following commands are extras for those of you who like the extra challenge. You need to have Parts 1, 2 and 3 completed before you can implement the below.

FLAG_VIDEO <video_id> <flag_reason>

Mark a video as flagged with a supplied reason. The flag-reason is optional - if no reason is supplied by the user, it should default to “Not supplied”. If a video is already flagged or does not exist, display a warning message.

As the video is now considered flagged, a warning message should be displayed when the user tries to play a flagged video or add this video to a playlist. For simplicity, the flag reason should be a string without whitespaces.

The error checking and corresponding messages should stay the same if a playlist or a video don’t exist.

```
YT> FLAG_VIDEO amazing_cats_video_id dont_like_cats
Successfully flagged video: Amazing Cats (reason: dont_like_cats)

YT> FLAG_VIDEO another_cat_video_id
Successfully flagged video: Another Cat Video (reason: Not supplied)

YT> FLAG_VIDEO amazing_cats_video_id
Cannot flag video: Video is already flagged

YT> FLAG_VIDEO video_does_not_exist flag_video_reason
Cannot flag video: Video does not exist

YT> PLAY amazing_cats_video_id
Cannot play video: Video is currently flagged (reason: dont_like_cats)

YT> PLAY another_cat_video_id
Cannot play video: Video is currently flagged (reason: Not supplied)

YT> ADD_TO_PLAYLIST my_playlist amazing_cats_video_id
Cannot add video to my_playlist: Video is currently flagged (reason:
 dont_like_cats)

YT> ADD_TO_PLAYLIST my_playlist another_cat_video_id
Cannot add video to my_playlist: Video is currently flagged (reason:
 Not supplied)
```

If a video is already part of a playlist and the user tries to add it after flagging, print the error message regarding the flagging, irrespective of the fact that the video was already part of the playlist. This is because even if the user removed that video from the playlist, they still couldn't add the video because it is currently flagged. We want to show error messages in the order the user has to fix them to be able to add the video to the playlist.

```
YT> CREATE_PLAYLIST my_playlist
Successfully created new playlist: my_playlist

YT> ADD_TO_PLAYLIST my_playlist amazing_cats_video_id
Added video to my_playlist: Amazing Cats

YT> FLAG_VIDEO amazing_cats_video_id dont_like_cats
Successfully flagged video: Amazing Cats (reason: dont_like_cats)

YT> ADD_TO_PLAYLIST my_playlist amazing_cats_video_id
Cannot add video to my_playlist: Video is currently flagged (reason:
 dont_like_cats)
```

Flagged videos should no longer be selected by PLAY_RANDOM. Furthermore, if all available videos are flagged, PLAY_RANDOM should behave just as it does when there are no videos available and print "No videos available".

```
YT> FLAG_VIDEO amazing_cats_video_id
Successfully flagged video: Amazing Cats (reason: Not supplied)

YT> FLAG_VIDEO another_cat_video_id
Successfully flagged video: Another Cat Video (reason: Not supplied)

YT> FLAG_VIDEO life_at_google_video_id
Successfully flagged video: Life at Google (reason: Not supplied)

YT> FLAG_VIDEO funny_dogs_video_id
Successfully flagged video: Funny Dogs (reason: Not supplied)

YT> FLAG_VIDEO nothing_video_id
Successfully flagged video: Video about nothing (reason: Not supplied)

YT> PLAY_RANDOM
No videos available
```

When the user requests all videos (SHOW_ALL_VIDEOS) or when the user asks for all the videos in a specified playlist (SHOW_PLAYLIST <playlist_name>) the video should be shown as flagged.

```
YT> FLAG_VIDEO amazing_cats_video_id dont_like_cats
Successfully flagged video: Amazing Cats (reason: dont_like_cats)

YT> SHOW_ALL_VIDEOS
Here's a list of all available videos:
  Amazing cats (amazing_cats_video_id) [#cat #animal] - FLAGGED (reason:
    dont_like_cats)
  Another Cat Video (another_cat_video_id) [#cat #animal]
  Funny Dogs (funny_dogs_video_id) [#dog #animal]
  Life at Google (google_video_id) [#google #career]
  Video about nothing (nothing_video_id) []
```

```
YT> CREATE_PLAYLIST my_playlist
Successfully created new playlist: my_playlist

YT> ADD_TO_PLAYLIST my_playlist amazing_cats_video_id
Added video to my_playlist: Amazing Cats

YT> FLAG_VIDEO amazing_cats_video_id dont_like_cats
Successfully flagged video: Amazing Cats (reason: dont_like_cats)

YT> SHOW_PLAYLIST my_playlist
Showing playlist: my_playlist
  Amazing Cats (amazing_cats_video_id) [#cat #animal] - FLAGGED (reason:
    dont_like_cats)
```

Flagged videos should not show up in search results (neither when searching by text nor by tag).

```
YT> SEARCH_VIDEO cat
Here are the results for cat:
  1) Amazing Cats (amazing_cats_video_id) [#cat #animal]
  2) Another Cat Video (another_cat_video_id) [#cat #animal]
Would you like to play any of the above? [...]

YT> FLAG_VIDEO amazing_cats_video_id dont_like_cats
Successfully flagged video: Amazing Cats (reason: dont_like_cats)

YT> SEARCH_VIDEO cat
Here are the results for cat:
  1) Another Cat Video (another_cat_video_id) [#cat #animal]
Would you like to play any of the above? [...]

YT> SEARCH_VIDEOS_WITH_TAG #cat
Here are the results for #cat:
  1) Another Cat Video (another_cat_video_id) [#cat #animal]
```

```
Would you like to play any of the above? [...]
```

If a video is flagged while it's playing (or paused), the flagging also serves as a STOP command and should stop the video immediately. This also applies to videos that are currently paused.

```
YT> PLAY_VIDEO amazing_cats_video_id
Playing video: Amazing Cats

YT> SHOW_PLAYING
Currently playing: Amazing Cats (amazing_cats_video_id) [#cat #animal]

YT> FLAG_VIDEO amazing_cats_video_id dont_like_cats
Stopping video: Amazing Cats
Successfully flagged video: Amazing Cats (reason: dont_like_cats)

YT> SHOW_PLAYING
No video is currently playing
```

Note that you don't have to stop a currently playing video if it's not the same as the one being flagged.

ALLOW_VIDEO <video_id>

Attempts to allow (un-flag) a video. If a video doesn't exist or is not currently flagged, display a warning message.

```
YT> FLAG_VIDEO amazing_cats_video_id
Successfully flagged video: Amazing Cats (reason: Not supplied)

YT> ALLOW_VIDEO amazing_cats_video_id
Successfully removed flag from video: Amazing Cats

YT> ALLOW_VIDEO amazing_cats_video_id
Cannot remove flag from video: Video is not flagged

YT> ALLOW_VIDEO non_existing_video_id
Cannot remove flag from video: Video does not exist
```

This will update a video to allow it to be playable again and it will no longer be marked with the additional flagged information when showing all videos (SHOW_ALL_VIDEOS), showing a playlist (SHOW_PLAYLIST), or searching videos (SEARCH_VIDEOS, SEARCH_VIDEOS_WITH_TAG).

```
YT> CREATE_PLAYLIST my_playlist
Successfully created new playlist: my_playlist

YT> ADD_TO_PLAYLIST my_playlist amazing_cats_video_id
Added video to my_playlist: Amazing Cats
```

```
YT> FLAG_VIDEO amazing_cats_video_id dont_like_cats
Successfully flagged video: Amazing Cats (reason: dont_like_cats)

YT> SHOW_PLAYLIST my_playlist
Showing playlist: my_playlist
  Amazing Cats (amazing_cats_video_id) [#cat #animal] - FLAGGED (reason:
    dont_like_cats)

YT> ALLOW_VIDEO amazing_cats_video_id
Successfully removed flag from video: Amazing Cats

YT> SHOW_PLAYLIST my_playlist
Showing playlist: my_playlist
  Amazing Cats (amazing_cats_video_id) [#cat #animal]
```

Time to run the tests

At this point you might want to run the tests for Part 4. If they all pass, you have completed this part.

Want more?

There are a lot of ways to extend this application. If you want to challenge yourself further, consider implementing the following:

- If a playlist doesn't exist yet, ask the user if they'd like to create one with the specified name (Similarly to how you ask the user if they'd like to play one of the videos after a search).
- When showing the playlists, consider adding the number of videos as well (e.g. "my_playlist (12 videos)".
    - Keep in mind that when there is only 1 video you might want to print "1 video" instead of "1 videos".
- Implement a command to play a whole playlist and command to advance to the next video in the playlist.
    - Consider a PLAY_PLAYLIST <playlist_name> command
    - Consider a NEXT command - Think about the cases when this command should not be doing anything and print a warning message instead.
    - This would have to work with PLAY <id>, STOP, PAUSE, CONTINUE. You can also consider adding a SHOW_CURRENT_PLAYLIST or similar to show which playlist is currently playing.
- Implement a rating system for the videos and extend the listing commands to also show the average rating and order by the rating.
    - Consider adding a RATE_VIDEO <video_id> <rating> command.
    - Consider adding validation such that the rating can only be in a certain range (e.g. 1-5).
- Implement an UNDO command
- Extend the video library to add real URLs. You can extend the commands that list videos to also show the URLs. Consider adding validation for those URLs.
- Think about how you might search the video title and tags if the list was very large. How could you optimize the search?

Final words

Thank you for taking part in this challenge, we hope you had fun and enjoyed yourselves trying to come up with a solution to all the problems posed here.

We wish you continued success and fun in your journey ahead.