Some Background:

For non Family Guy viewers, the names Brian and Jillian certainly are new names. However, here's some background:

Brian -> Refers to Brian Griffin. The family's dog. Pretentious, Stewie's best friend. And most importantly one of my most relatable characters. Tries to keep a social image even though internally this guy feels something else altogether.

Jillian -> Refers to Jillian Wilcox ne`e Russell. Brian's longest girlfriend. I guess the next most was the crush on Lois Griffin. However, yes Jillian Russell was his longest girlfriend and I guess the most important individual in Brian's life.

Introduction:

Brian on Jillian Kernel better known as **BOJK** is a directory workup program designed by Anna and works under the Perry Architecture BOJK allows you to change directories in a good fashion. It was inspired from the classic `roscd` command of Robot Operating System(`ros`)

History:

BOJK started as a batch file in early August 2020 under the Meg Architecture `carl.bat` and `brian.bat` being two batch files on the system. In fact this was what `carl.bat` had in its code:

```
@echo off
set "arg=%1"

if arg=="ds" (
    cd /d "D:\P2\Python Projects\Data Science Projects"
    rem This is just to change directory and the drive
)

rem some more commands come in here
```

`brian.bat` is also similar. However, it must be pretty obvious that this program cannot be used directly in scaling. Since batch files cannot be given proper configuration using backend files(like XML or YAML) files.

After this, I decided to change the architecture to the Perry Architecture using python files. The first Brian program was called `brian.py` , using this form:

`brian.bat` : -> Sends Arguments to `brian.py` *Batch File was set on PATH*
`brian.py` : -> Writes output to `output_brian.bat` *Contains all the necessary output*
`output_brian.bat` : -> Run on the Windows CMD to execute the action

Back then, Brian was hardcoded, similarly carl had to be hardcoded too. Commands given were parsed by `sys.argv` and not using `argparse.ArgumentParser`

Introduction of `argparse` :

`argparse` was initiated in about late September and used to parse the arguments. For the first time `brian` changed from a botched up python script to a professional looking program.

By this time, Brian had the following arguments usable

```
usage: brian.py [-h] [-c COMMAND] [-o]

optional arguments:
  -h, --help            show this help message and exit
  -o, --open            Opens the directory and not cd into it
  -c COMMAND, --command COMMAND
                        Command to be read by Brian Dictionary
```

And it was working with the same old `brian.bat -> brian.py -> output_brian.bat` format. A classic of the Perry Architecture

However, since the command was hardcoded, Brian had to have four copies of itself, just having different commands hardcoded. since there were different commands to be used, `brian.py` was handling `D:\Toolkit` , `carl.py` was handling `D:\P2` , `cherry.py` was handling `D:\Personals` and `isabella.py` was handling `D:\Third Sem` .

Introduction of the YAML File:
Since hardcoding all the files were especially difficult, Brian was given a back of a `yaml` file in mid October. This was especially useful since by now there were different commands to be used and hardcoding was tough.

Just imagine the hardcoding which was used before the YAML file was initiated. Here's a sample that was in the most optimized version just before YAML file was initiated.

```
def command_parser(command):

    data_dict = {
                "brian" : "D:\\Toolkit\\Extra\\Brian" ,
                "custom" : "D:\\Toolkit\\Custom" ,
                "extra" : "D:\\Toolkit\\Extra" ,
                "sienna" : "D:\\Toolkit\\Extra\\Sienna" ,
                "system" : "D:\\Toolkit\\System"
    }

    folder_ = data_dict.get(command , None)

    if folder_ == None:
        #Command does not exist
        return "D:\\Toolkit\\Extra"
        #Default Folder Kinda thing

    return folder_
```

There was so much of hardcoding which got simplified after the YAML file was introduced. Sure, the YAML file had the same dictionary stored in it, however, it could be much easily modified now. And instead of four different Python files needed, now I could work with just a single python file and four different YAML files needed.

By now, these were the arguments:

```
usage: brian.py [-h] [-c COMMAND] [-o] [-y YAML]

optional arguments:
  -h, --help            show this help message and exit
  -o, --open            Opens the directory and not cd into it
  -c COMMAND, --command COMMAND
                        Command to be read by Brian Dictionary
  -y YAML, --yaml YAML  YAML File to be used
```

Search Engine Functionality:

Ever created a folder `Riley Reid Sanskaari Photos` somewhere to be hidden and now you have no fucking clue where it is. Probably it is somewhere `D:\Study Material\Second Sem Courses\Minor Course\Reference Books\Teacher Notes\Riley Reid Sanskaari Photos` . To obtain such folders, I created the functionality called Search Engine. Which basically listed all the folders in an overall_path and searched for the directories in it against a given search query and returned some of the top results. This was useful to search `Riley Reid Sanskaari Photos` among all directories of `D:\Study Material`

Brian changed completely when this feature was initialized in late October. This is where the term `Jillian Kernel` comes from. This is where `Brian` becomes `Brian on Jillian Kernel` . *[Sex Comedy Completely Intended]*

Basically though Jillian Kernel is just another python file named `Jillian_Kernel.py` . It was named Kernel because it was a sort of central system which performed the entire search and arranging and return the directory useful. Was about a 130 line code which did something like this:

`brian.bat` -> Arguments Accepted. And send to `brian.py`
`brian.py` ->: Check the user input, if command used , give corresponding output from the YAML file. If search engine to be used -> give parameters to *`Jillian_Kernel.Jillian`*
`Jillian_Kernel.py` -> Returns output. Brian checks it, writes all the data to the batch file
`output_brian.py` -> Windows Command Prompt Order

I didn't have any idea about the cool python libraries present like `FuzzyWuzzy` .

I had an idea about the Levenshtein distance idea, so I created a useable Levenshtein distance algorithm and this was the structure

```
def Jillian(return_index , overall_path , search_query):
    #Returns a list of all the folders matching with the search .
```

Introduction of RapidFuzz:
This changed the entire program utility. RapidFuzz was introduced in about early November. Brian on Jillian Kernel changed in its entire structure post introduction of RapidFuzz. `RapidFuzz` is a custom library under development(so not safe for professional use. Yeah, as if BOJK is a professional program and I care). It is a C-Based equivalent(lower in space and time complexities) to the much popular Python Library `FuzzyWuzzy`. This was used, and the search engine functionality was faster than ever. By now very complex searches also took max 3 seconds or so. These were the commands here:

```
usage: brian.py [-h] [-c COMMAND] [-o] [-y YAML] [-s SEARCH_QUERY]

optional arguments:
  -h, --help            show this help message and exit
  -o, --open            Opens the directory and not cd into it
  -c COMMAND, --command COMMAND
                        Command to be read by Brian Dictionary
  -y YAML, --yaml YAML  YAML File to be used
  -s SEARCH_QUERY, --search SEARCH_QUERY
                        Search Engine Query
```

Okay by now, think about this, now `Jillian Kernel` is dead right? `RapidFuzz` took its place? And yup that's right. Jillian Kernel was dead. But well, let's think of it this way....
Jillian Russell left Brian Griffin and went for Derek Wilcox. Derek was murdered by Diane Simmons but Brian never got Jillian back. Now I ain't that cold hearted lol. So though Jillian Kernel wasn't useful anymore, I let the name be. It was still `Brian on Jillian Kernel`

Functionality Improvement:

After early December, BOJK was never modified at all. It remained the exact same. However, on `January 10` it was completely revamped and the final pieces were added to the code. `Brian on Jillian Kernel` was revamped with the addition of the following new features:

1. *Introduction of a cache feature*: Since walking the directories in real-time for the search engine functionality might not be optimized for the program; a binary pickle file cache was generated for each directory being controlled by one of Brian's sister configurations.

2. *File Extensions*: The extensions `.bojk` and `.jillian` were assigned to the cache file and the YAML configuration file respectively.

3. *More options*: The option -so was added which changes the directory and the directory is opened in the explorer as well. Also the directory path could be copied to the clipboard(`--copy`) and the directory path could be printed(`--print`) onto the terminal. Just the program's additional features.

4. *More search utilities*: Two more features full path search(`-fp` , `--full-path-search`) and search by file(`-sbf` , `--search-by-file`) were added. Basically let's take the Riley Reid example once

again. There is directory `D:\Study Material\Second Sem Courses\Minor Course\Reference Books\Teacher Notes\Riley Reid Sanskaari Photos` . Now if you use the normal search engine feature, you will have to search only using `Riley Reid Sanskaari Photos` and nothing else works. However, if you have two directories, say `D:\Study Material\Second Sem Courses\Minor Course\Reference Books\Teacher Notes\Riley Reid Sanskaari Photos` and `D:\Personals\My Photos\Canada Tour 2017\Blurry Photos\Riley Reid Sanskaari Photos` , you could use `Teacher Notes\Riley Reid Sanskaari Photos` to get the output from `D:\Study Material\Second Sem Courses\Minor Course\Reference Books\Teacher Notes\Riley Reid Sanskaari Photos` . However, suppose you have a file say `Riley_XXX2` in `D:\Study Material\Second Sem Courses\Minor Course\Reference Books\Teacher Notes\Riley Reid Sanskaari Photos` , you could use the search by file feature an d obtain the directory `D:\Study Material\Second Sem Courses\Minor Course\Reference Books\Teacher Notes\Riley Reid Sanskaari Photos`

5. *Version and Usage Text* : The two commands `-V` and `-u` were introduced to check the present version of BOJK and to check the usage of the program respectively.

6. *Use of colored print*: You can use the flag `-tc` to use termcolor on the terminal. This helps in viewing contents more clearly. And screening important lines from a lot of output(if any)

Presently these are the arguments BOJK can use:

```
usage: bojk.exe [-h] [-o] [-so] [-c COMMAND] [-s SEARCH_] [-fp] [-f] [-y YAML] [-V] [-u] [-g]
                [--copy] [--print]

optional arguments:
  -h, --help            show this help message and exit
  -o, --open            Opens the directory in the explorer
  -so, --shell-and-open
                        Opens the final obtained directory both in the shell and in explorer
  -c COMMAND, --command COMMAND
                        Use a command from the YAML like Configuration file
  -s SEARCH_, --search-engine SEARCH_
                        Search for this directory
  -fp, --full-path-search
                        Forces the search engine to search for the entire path. Defaults to on
  -f, --config          Shows the present configuration using the YAML like Configuration File
  -y YAML, --yaml YAML  Change the YAML like configuration file. Default is `config.jillian` i
  -V, --version         Show details about version of BOJK
  -u, --usage           Show how to use Brian on Jillian Kernel
  -g, --generate        Generate a blank config YAML like file in the current working director
  -i, --init-cache      Initialize the BOJK Cache feature which can be used later
  --cache-file CACHE_FILE
                        Cache File to be used for BOJK. Default to `Directories.bojk`in the do
  -uc, --use-cache      Use Cache Instead of Walking the Folders in Realtime for the Search En
  -sbf, --search-by-file
                        Searches the directory based on the files inside it. NOTE CACHE FEATUR
  -tc, --term-color     If flagged, the terminal prints a very colorful text. Sets to OFF by d
  --copy                Copy the final output folder instead of changing directory into it
  --print               Print the final output on the screen
```

APIable functions:
The source files are present in the `src` directory within the BOJK program. These can be used as APIs if needed. The functions are well documented. However note that the entire script cannot be used on non Windows machines. There is enough data implementation within the script using functions. Also the program shuts if used as a python file on non-`nt` systems.

There are two DTTEs(Dummy Transmission Text Editors) used for the two extensions. `.jillian` is opened in Atom Nightly and `.bojk` is opened in Pickle Viewer.

This is the present final configuration of the Brian on Jillian Kernel

Sister Programs:
Presently BOJK has four programs:

1. `Brian` Configuration : Uses `D:\Toolkit` as the default search path and some corresponding commands
2. `Carl` Configuration: Uses `D:\P2` instead of Brian configuration's `D:\Toolkit`
3. `Nora` Configuration: Was old named Cherry. Uses `D:\Personals` as the default search path and some corresponding commands
4. `Isabella` Configuration: Uses Academic Directory. Presently works on `D:\Fourth Sem`