Input Style:

[Command] [Input] [Option(s)] [Additional Input]

--

Command Requirements:

Q (Quit)

* Quits the program immediately

L (-f, -s, -e)

- Lists all items **THEN** directories. This will not list the contents of directories inside of the specified path.

- f

-f excludes directories from results

-s

-s takes in a file name and only outputs files/directories with that given name

-e

-e takes in a suffix and only outputs files with that given suffix

L -r (-f, -s, -e)

* Lists all items **THEN** directories. This is done recursively.
* F
* S
* E

Part 2:

C

* Creates a new file in a specified directory with a .dsu suffix
* If the files already exists, print (‘ERROR’)

D

* Deletes a user given .dsu file. Errors otherwise
* Should print the location of the file that was deleted followed by ‘ DELETED’ afterwards.

R

* Reads the contents of a file. Prints ‘EMPTY’ if the file is empty

Problematic Inputs:

L D:\Video Projects

L D:\Video Projects -r

Things to consider:

.extend

.contains

* A Note: A few of the command extensions require secondary input. Thankfully, these command extensions also **always appear before** the secondary input. To avoid white space splitting issues, we can index these command extensions (‘ -s ‘and ’ -e ‘, add 4 to the index value to account for the length of the extensions, and take the substring from there to the end in order to avoid spacing problems

All command extensions (-r, -s, -f, -e, etc…) all start with a dash. To capture the user entered path (regardless if there’s spaces in the path), we can index the user’s input from either:

* Input[2:]
  + This works if there’s NO extensions.
* Input[2:(first instance of ‘-‘)]
  + This works if there’s ANY extensions.

Valid Input:

* Commands[‘Q’, ‘L’, ‘C’, ‘D’, ‘R’]
* Valid Extensions[‘ -r’, ‘ -f’, ‘ -s’, ‘ -e’]

Valid commands (and their extensions) are the following:

* Q
* L (-r, -f) -s/-e (specifications)
* C (path[given this exists]) -n (name)
* D (path)
* R (path)

**Program Psuedocode:**

Main:

Input\_terminal()

Input\_terminal(): #Takes in inputs and tells the program what to do with them

User\_Input = input()

While input() not ‘Q’:

Is\_valid = cmd\_checker(User\_Input)

If Is\_valid:

Cmd\_processor(Valid\_command)

Else:

Error()

User\_Input = input()

Cmd\_checker(user\_input): -> bool:

Commands = [‘L ’, ‘C ’, ‘D ’, ‘R ’]

User\_cmd = user\_input[0:2]

Cmd\_exists = User\_cmd in Commands

If User\_cmd = ‘L’:

L\_valid\_extensions = [‘-r’, ‘-f’]

L\_additional\_input = [‘-s’, ‘-e’]

L\_user\_extensions = []

For character in user\_input:

If character == ‘-‘

L\_user\_extensions.append(user\_input[character.index]: user\_input[character.index]: )

Return Cmd\_exists and valid\_extension

Error():

Print(‘ERROR’)