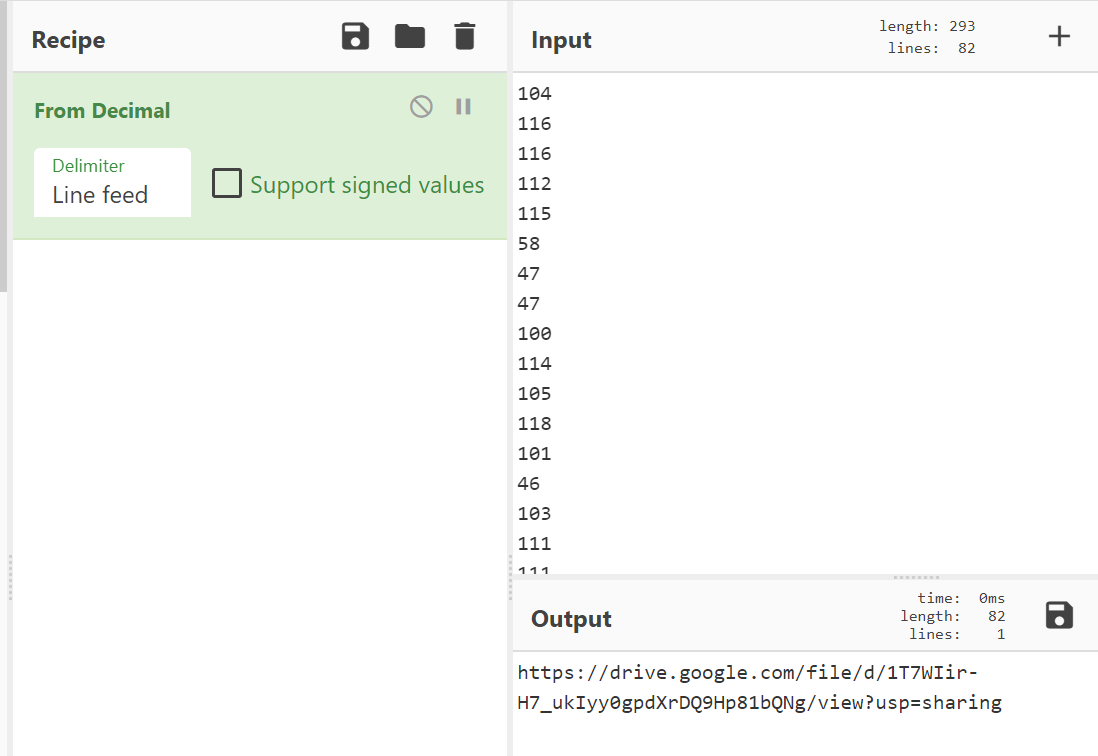
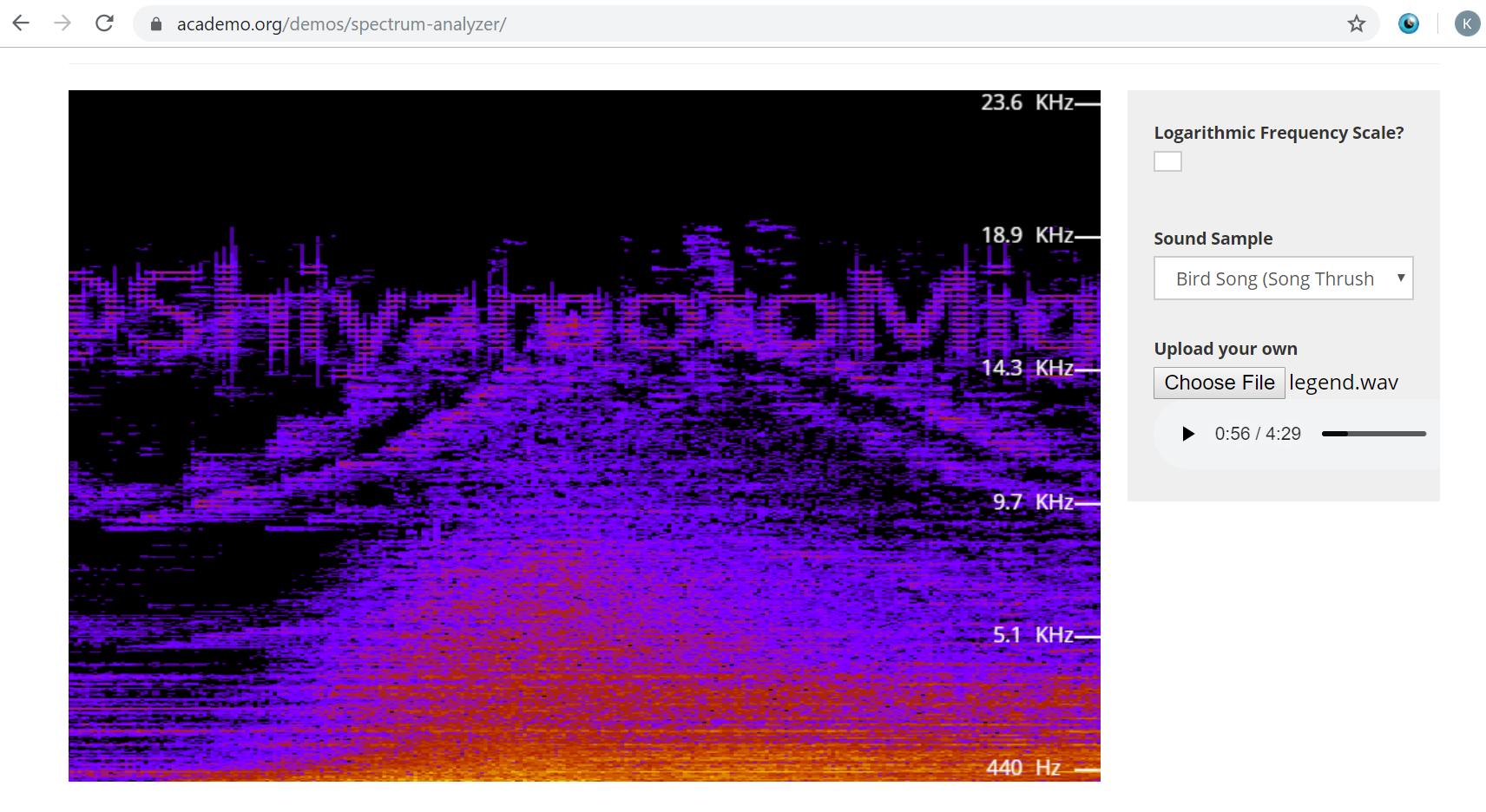
LEGEND Flag - Solving the Challenge

The quest begins with a .txt file of a decimal-encoded Google Drive Link.

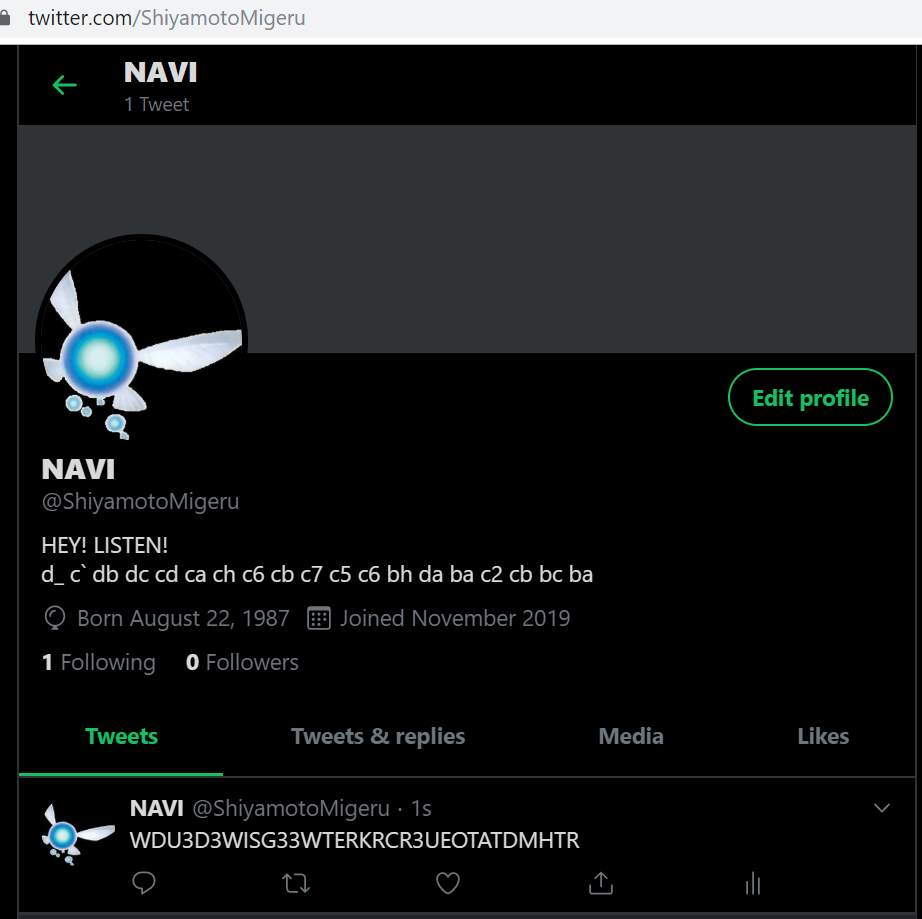


Going to that link gives you the orchestral version of a *Legend of Zelda* theme.

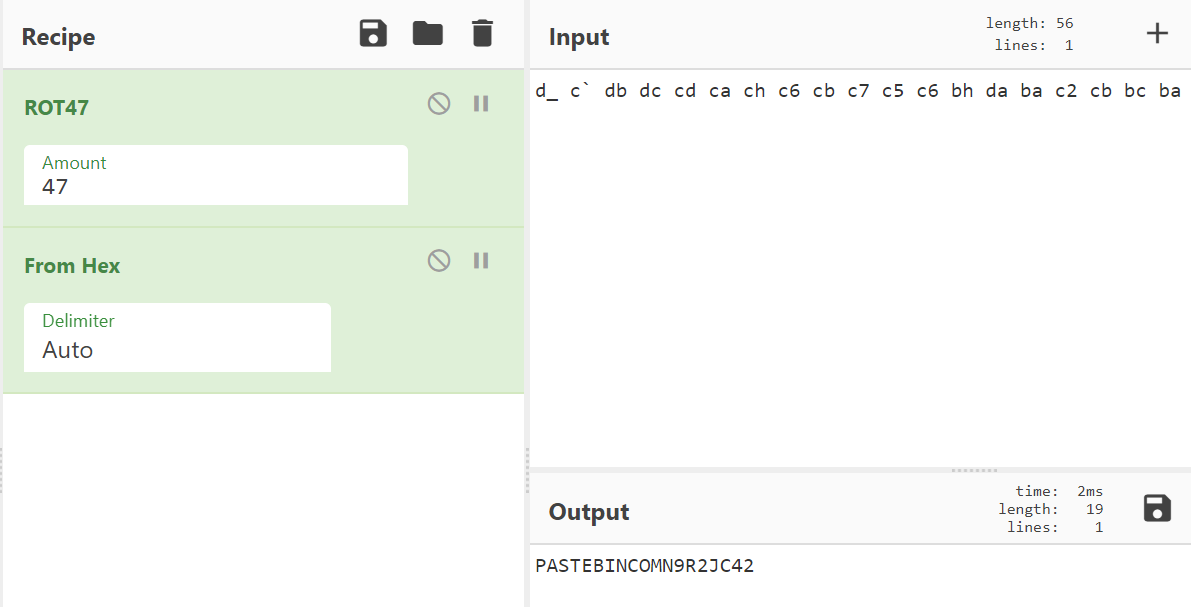
Examining the song with a spectrogram analyzer reveals a twitter handle embedded around the “chorus” of the song (around 0:51). The handle is @ShiyamotoMigeru.



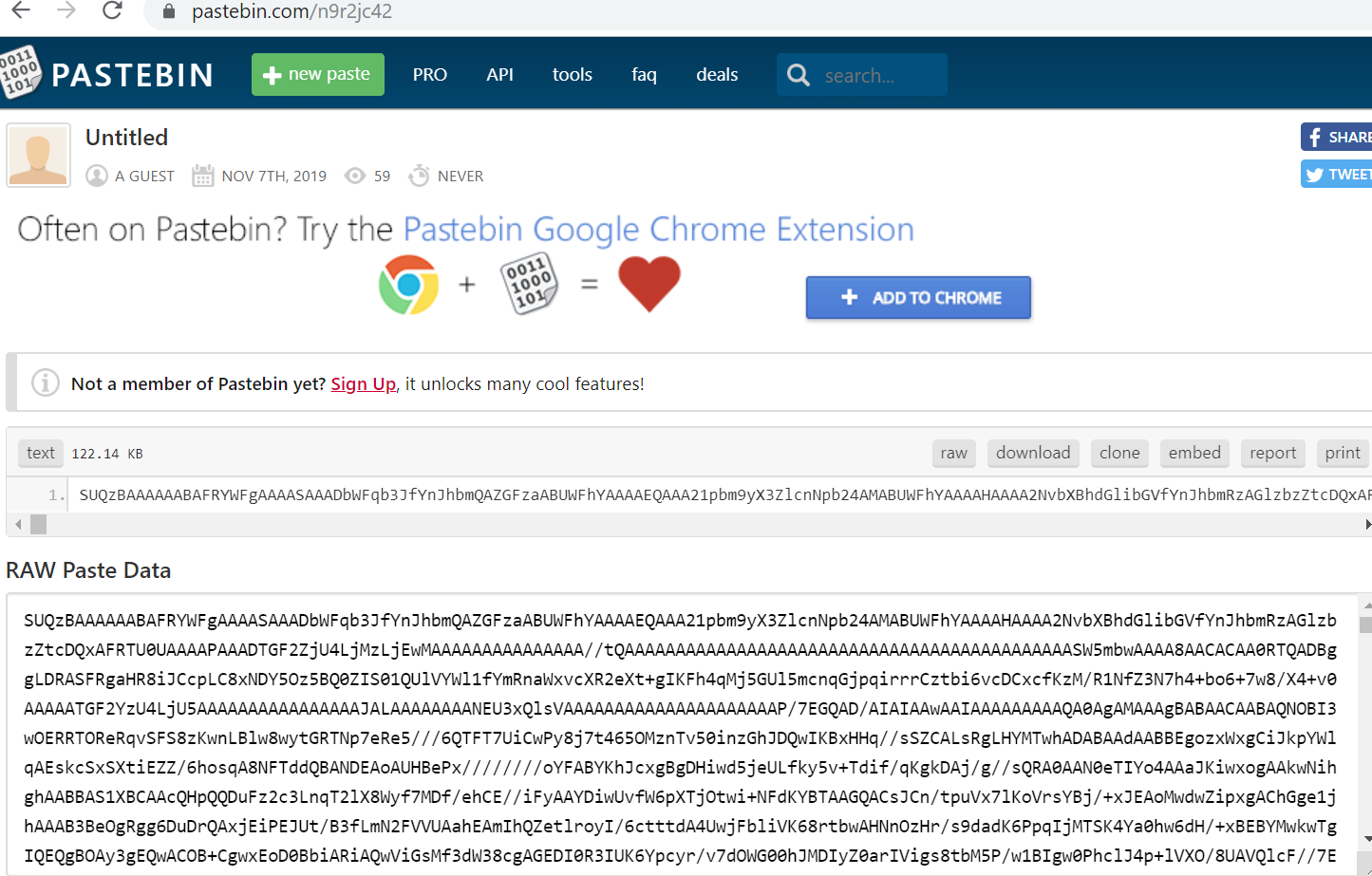
ShiyamotoMigeru is a transposition of the name Shigeru Miyamoto, the creator of *Legend of Zelda*. Twitter.com/shiyamotomigeru is a twitter account of Navi with both an encoded bio and an encrypted tweet.



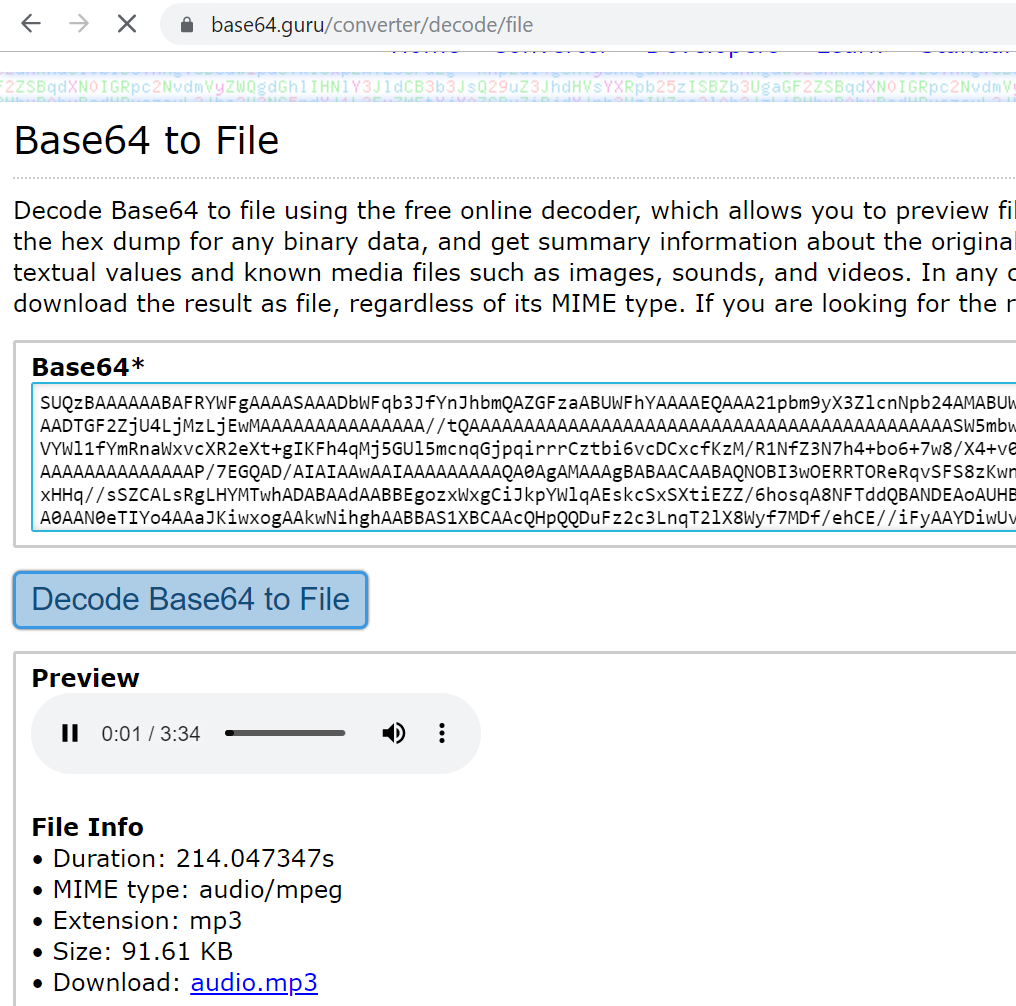
If we decode the bio with ROT47 and then HEX, we receive a pastebin address.



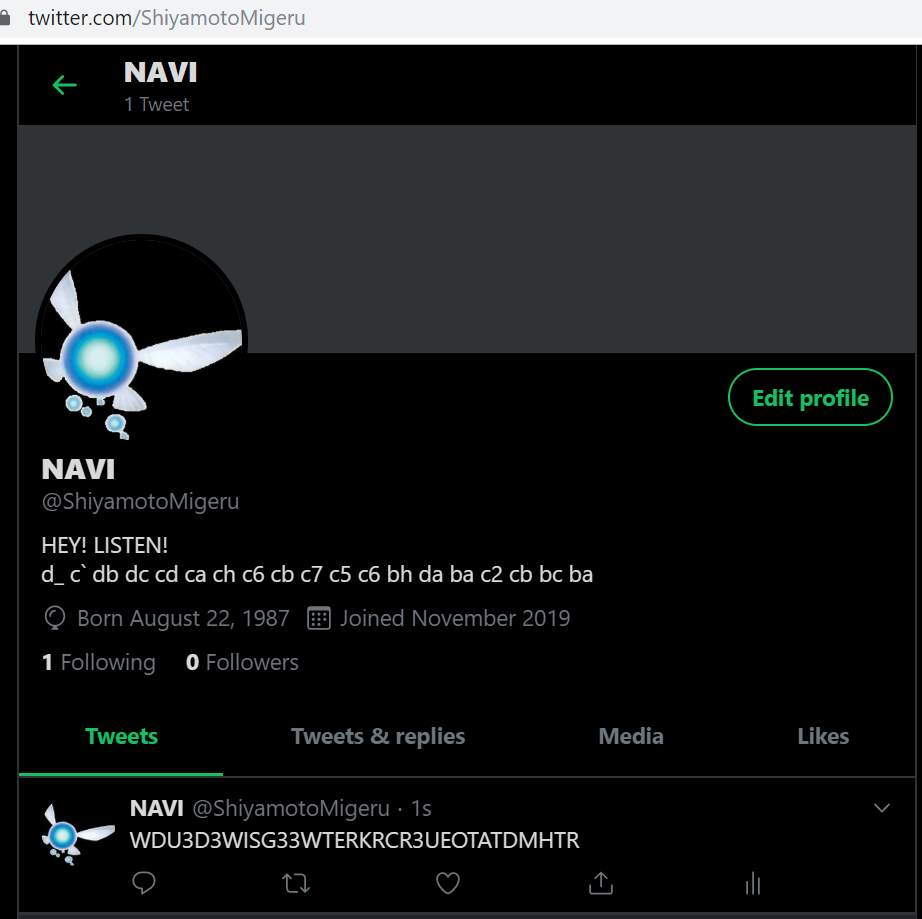
Pastebin.com/n9r2jc42 provides us with a slew of base64-encoded data. If you decode the base64 data, it provides what looks to be information that could be compiled into a file.



Taking that base64 data and decoding it into a file provides a potato-quality version of Rick Astley’s “Never Gonna Give You Up”. This is a dead end and NOT the way to complete the challenge.



Going back to Navi, we find an encrypted ciphertext as a tweet.



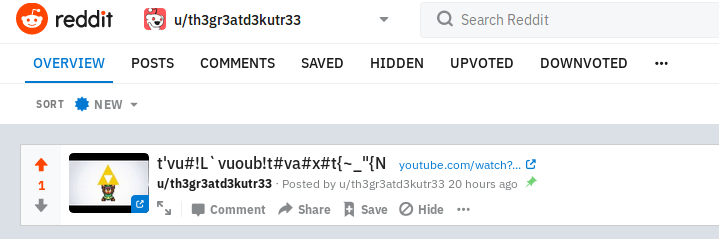
The ciphertext was encrypted using the transposition cipher.

**WDU3D3WISG33WTERKRCR3UEOTATDMHTR**

The key for the file is a length of 6. Decrypting provides an unformatted and padded reddit address (see the bottom of this document for a python method to break the transposition cipher)

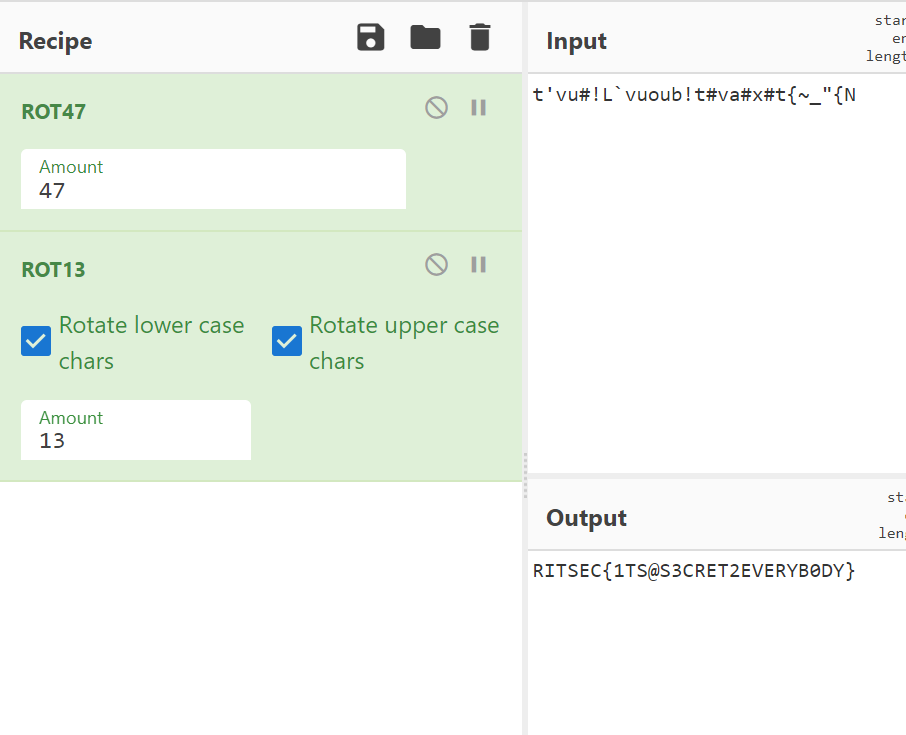
**WWWREDDITCOMUSERTH3GR3ATD3KUTR33**

[www.reddit.com/user/th3gr3atd3kutr33](http://www.reddit.com/user/th3gr3atd3kutr33)



The user is The Great Deku Tree in l33tsp3ak. It has posted a link to the “opening chest” sound in the Legend of Zelda games with a strange name for the post.

**t'vu#!L`vuoub!t#va#x#t{~\_"{N**



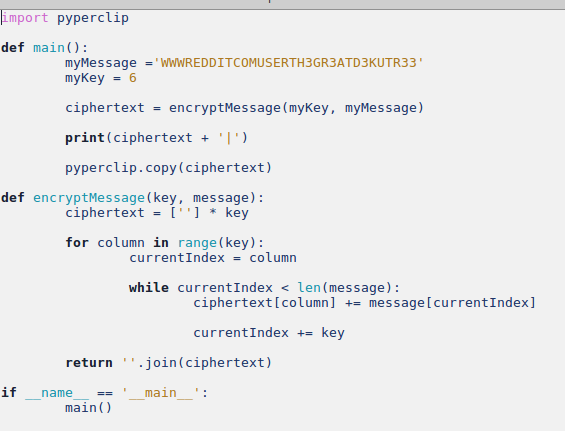
Decoding this text with ROT47 and then ROT43, we are provided with the flag for this challenge.

**RITSEC{1TS@S3CRET2EVERYB0DY}**

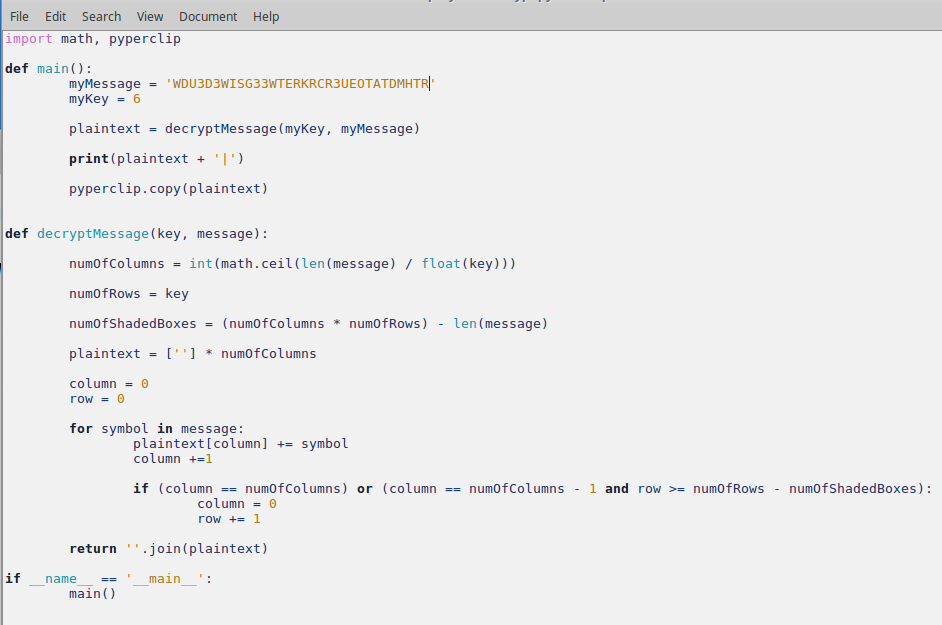
Solving the Transposition Cipher

Encryption and decryption both use pyperclip.py from Al Sweigart.

<https://inventwithpython.com/pyperclip.py>

How the plaintext was encrypted:  


How the ciphertext was decrypted:



What the command line prompt looks like.

