Challenge 1: Pink Zebra

```
UEsDBBQAAAAIAF0BfVJIgDy0oAEAANkDAAAMABwAY2hhbGxlbmdlLnB5VVQJAAPy/mBgfFRlYHV4CwABBPUBAA
AEFAAAAHVTS2/CMAy+51dYcBhIFTRtWpLDTozDTpM2btMOXRfWTqVFfYzBr5+dpB2PUcmOY8efHefrGNaZhqRs
9rqGtoIWd7vueCw05A1sqq78gLw07u+kzpN3DLysls+r9Ywya33XQFUWB1Qa0qRNsxkbw+MGDlUHbbLdIew+bz
NEILwc0y1W0WmoNg4LjxQFpFlSfuoZzOeM5dtdVbdQNYwtnx5WcA+vDPDjvgcy9GCBotCOlQdhgCu3K+ecVOSB
iMlAl1qQEXsuX7iQH5LiNmZTuE8qCFxAIL6KrHA/6AHMTrozfVwQkD9Eoh6RsF1/Jp1s1fdKx5SwhcSNi4nQrd
ICKGnsc1EXdr93hUJpVpNPfh5ggYUTMXTzNz5x0uBgcLzf2R0ISqLEkUMSJz68dySufLYF9c/Z4MInzzFNy1jI
AEThZSC2XVw75Q2I60BsZ67cvFXoDWQjf4wskoGxDYAU58QLlRXLrbgfZT84877cV4aOlkjETUMx5VlamrmLYf
bXRKKnskRjb4zpH51ORqPZV5WXk22ym6RZ7QH9LdPplP0CUEsBAh4DFAAAAAgAXQF9UkiAPLSgAQAA2QMAAAwA
GAAAAAAAAQAAAKSBAAAAAGNoYWxsZW5nZS5weVVUBQAD8v5gYHV4CwABBPUBAAAEFAAAAFBLBQYAAAAAAQABAF
IAAADmAQAAAAA=
```

Hint: We are looking for a color and an animal.

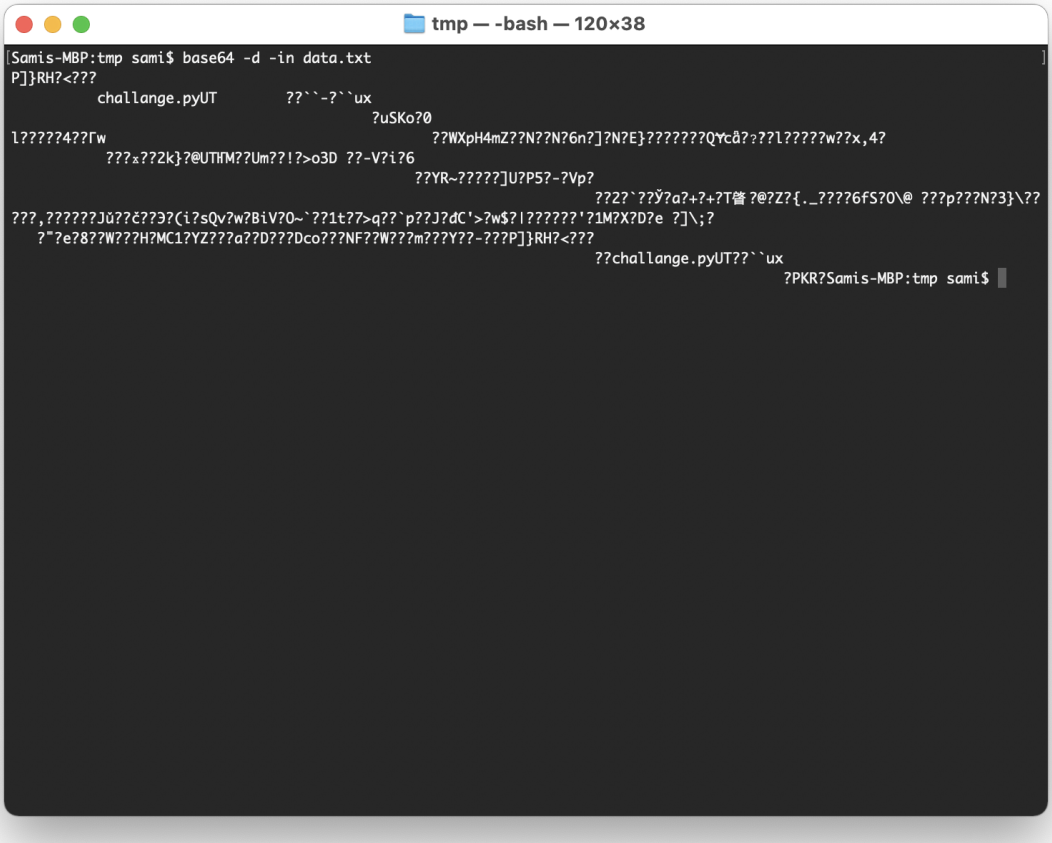
The Solution

A trained eye will easily recognize the string as base64 encoding. Base64 was designed as a way to send binary data over text based protocols like SMTP and HTTP. By grouping the binary data in 6-bit groups instead of bytes (8-bit) we can represent each such group with 64 standard ASCII characters (A-Za-z0-9+/). Because the binary data, normally grouped in octets, cannot always be evenly divided in groups of six the data is padded with zeros which is represented by the equals sign (=).

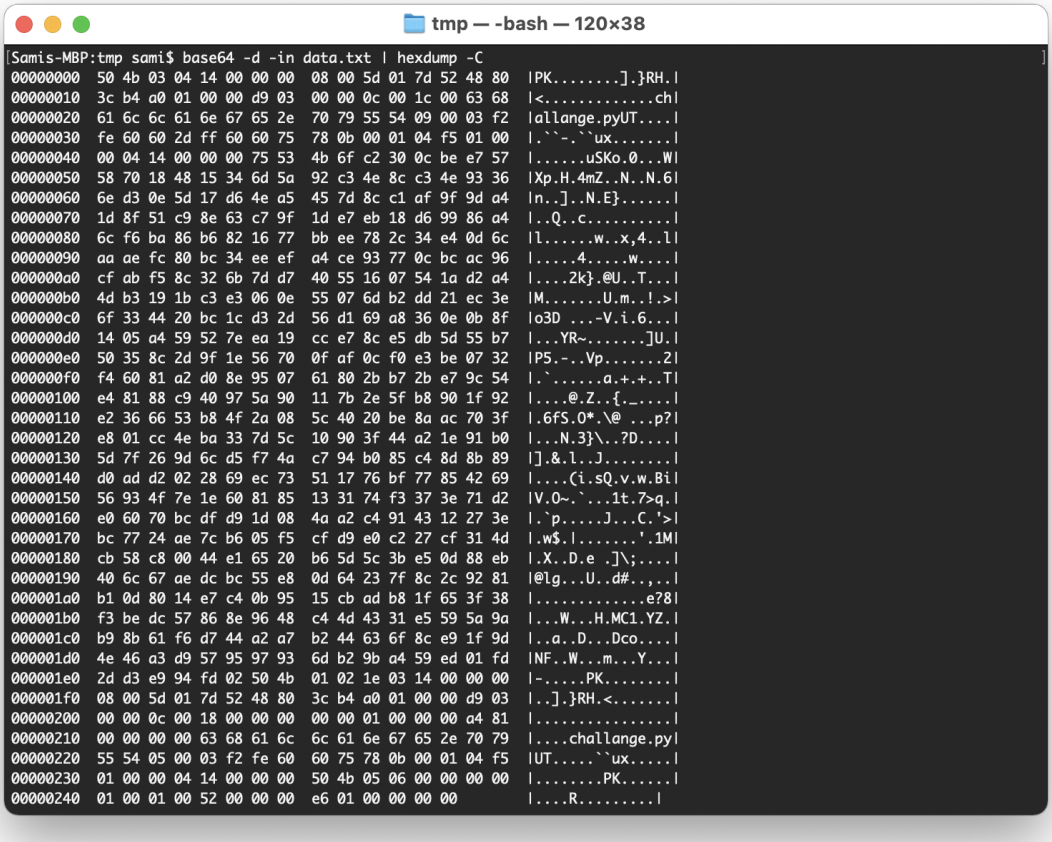
There is a common misconception that Base64 is a form of encryption which it's not. Just as 2021-03-31 and 03/31/2021 is two different ways of representing the same date, binary and base64 is just two different ways of representing the same data. **Never use Base64 with the purpose of hiding or obfuscating information.**

Sometimes developers encode images as Base64 in data URIs for embedding them in CSS or SVG-files. Note that data encoded in Base64 is about 33%-36% larger than it's binary equivalent and can actually have a significant impact on your bandwidth bill. **If you want to consolidate images to save requests on the server, put them in a jpeg och png sprite instead.**

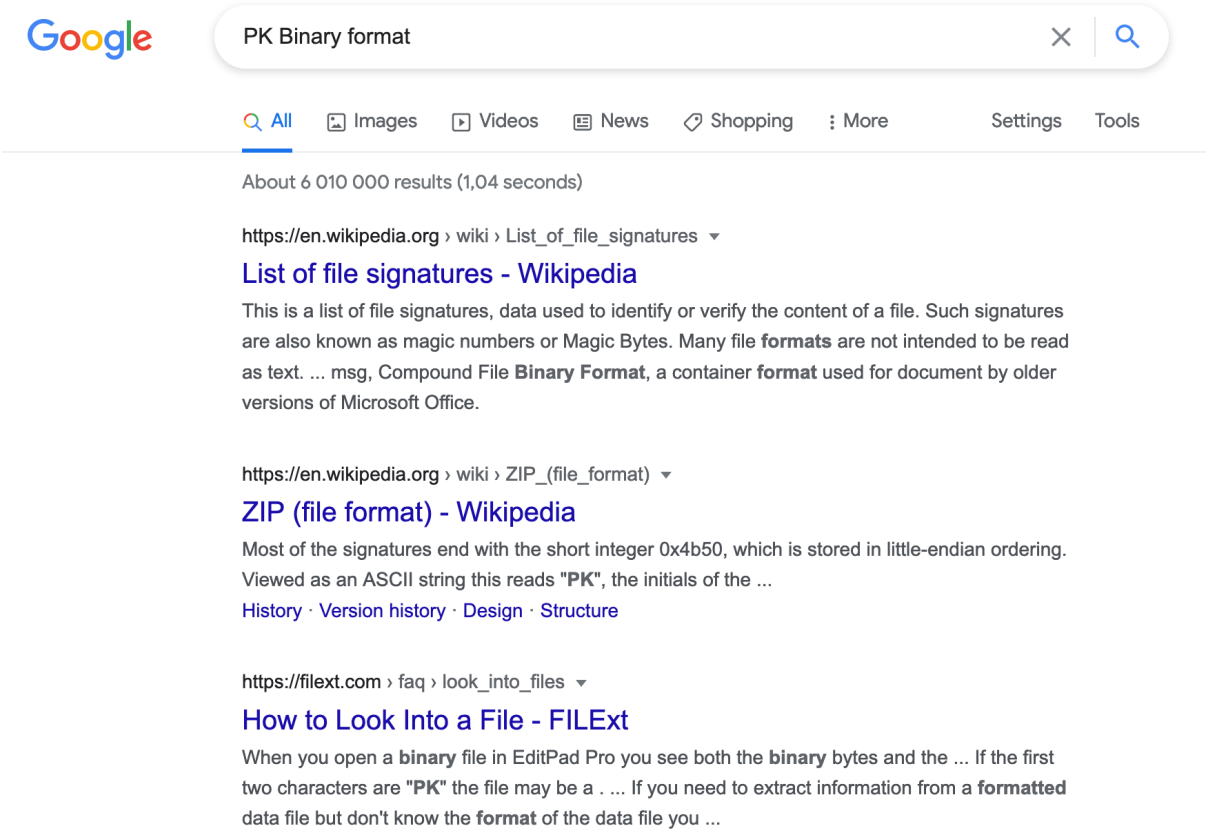
If we put the base64 encoded string into a text-file (in this case called data.txt) we can decode it with the command line tool base64:



Unfortunately the output appears as total gibberish as the terminal is trying to output binary data as text. A better way to inspect binary data is with the command **hexdump**:



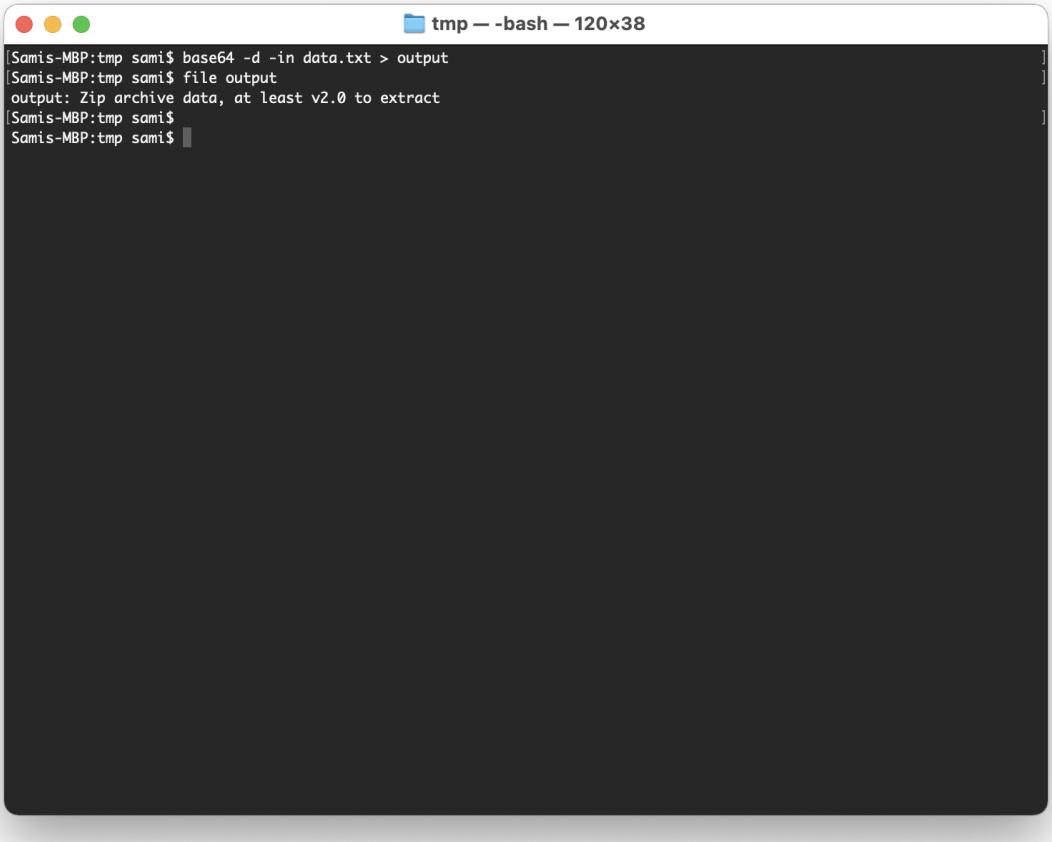
Most binary formats begins with a couple of bytes identifying what format it is. This file begins with the two bytes PK. If we don't recognize the format we can try Google it:



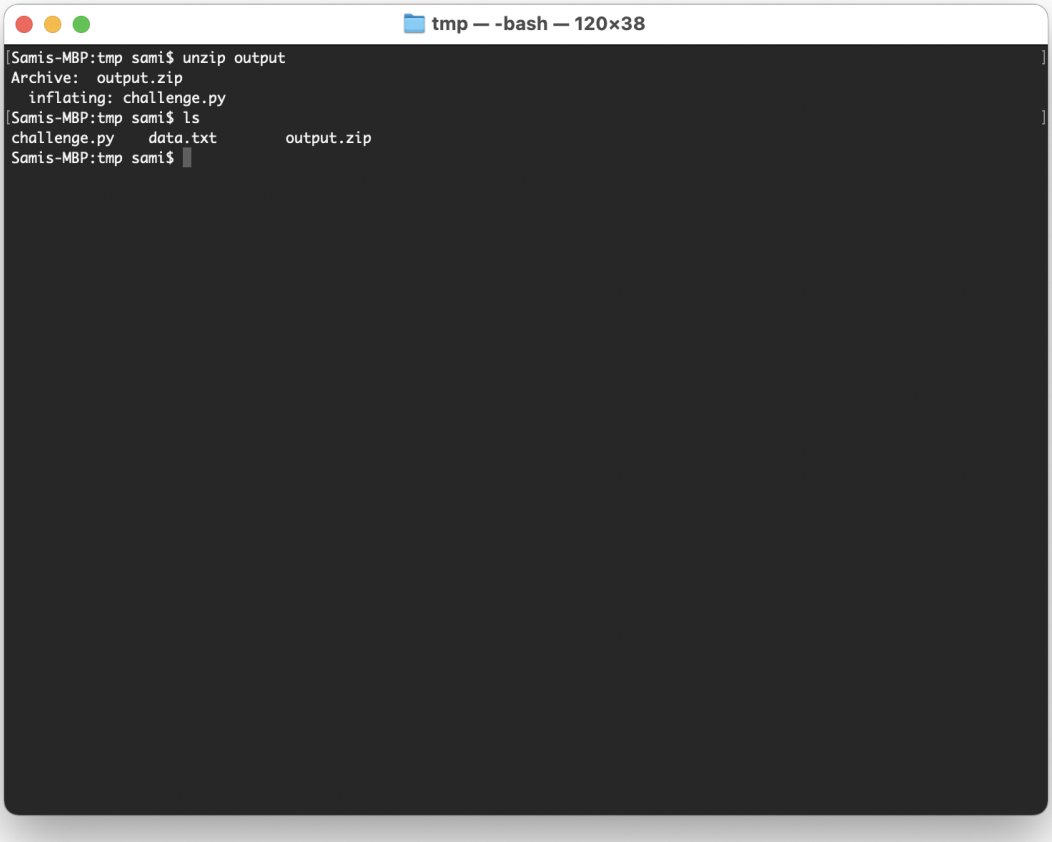
The first result shows a list on Wikipedia with common binary formats and its corresponding file signatures and the first two bytes PK indicates it's a ZIP-file.

4D 5A	MZ	0	exe dll	DOS MZ executable file format and its descendants (including NE and PE)
50 4B 03 04 50 4B 05 06 (empty archive) 50 4B 07 08 (spanned archive)	PK..	0	zip aar apk docx epub ipa jar kmz maff odp ods odt pk3 pk4 pptx usdz vsdx xlsx xpi	zip file format and formats based on it, such as EPUB, JAR, ODF, OOXML
52 61 72 21 1A 07 00	Rar!...	0	rar	RAR archive version 1.50 onwards[11]

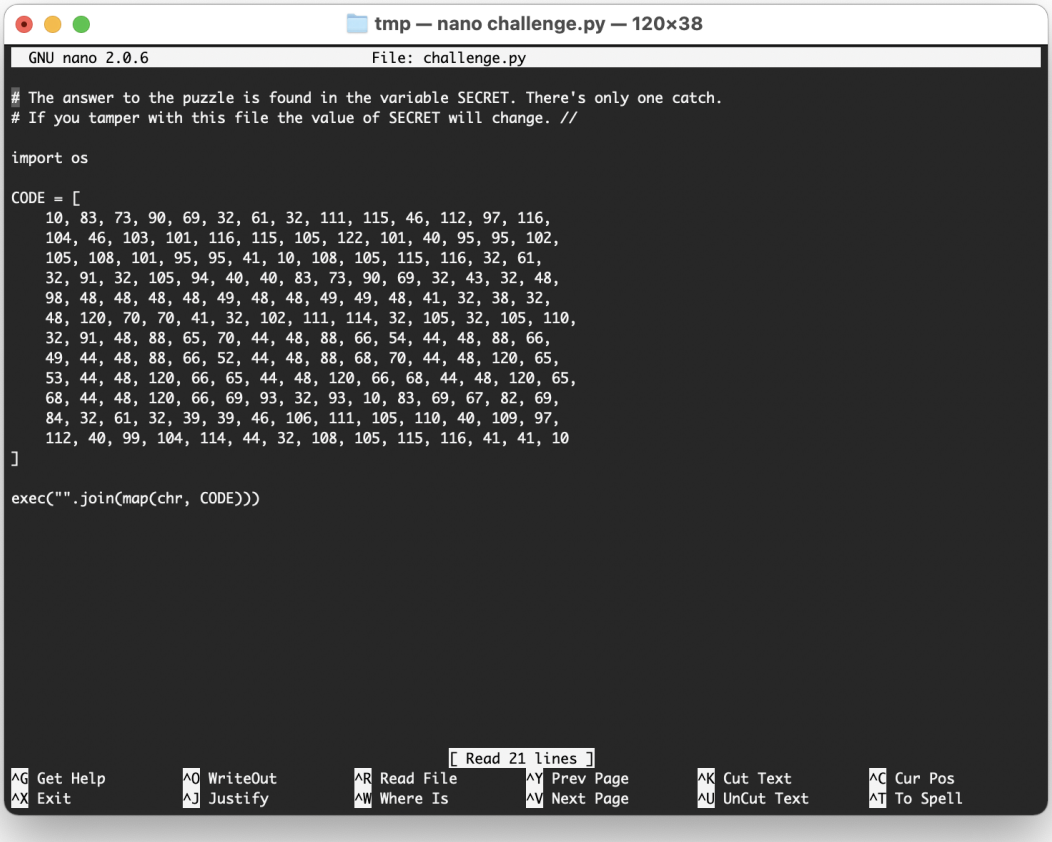
The **file** command confirms our suspicion:



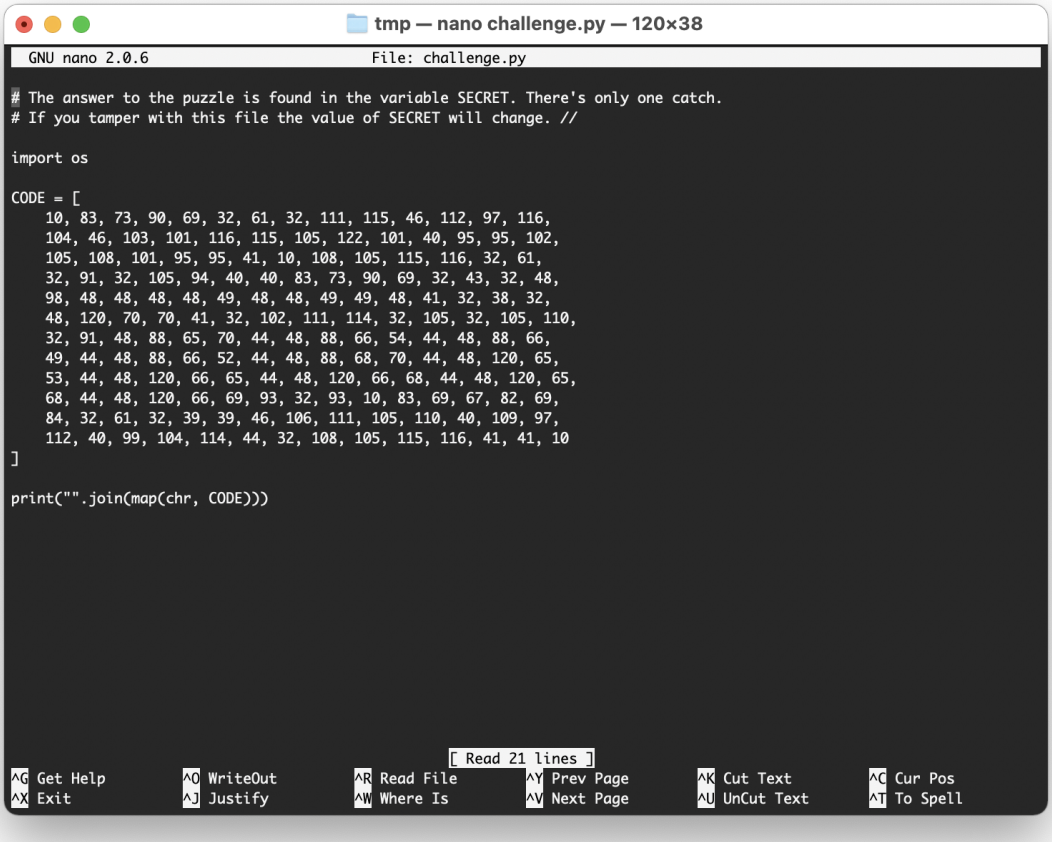
Now that we know we are dealing with a zip file. Lets **unzip** it!



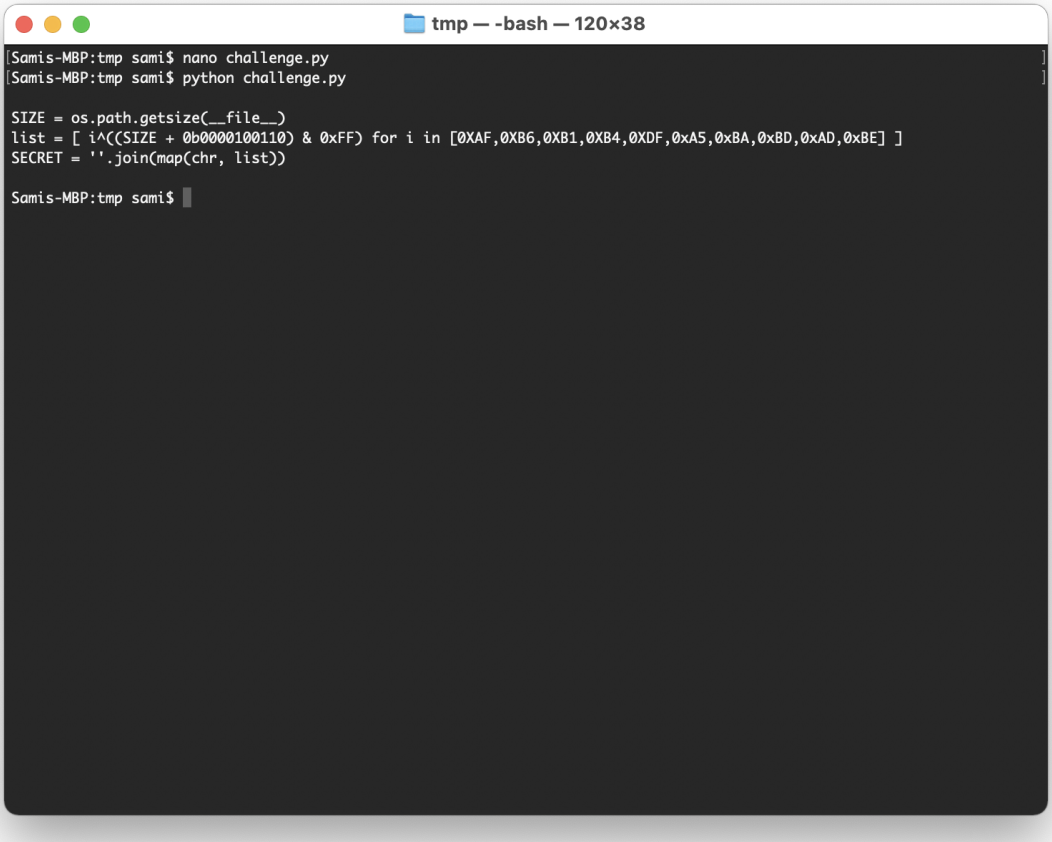
Unzipping the file results in a new file called **challenge.py**. Let's look at the content of the file:



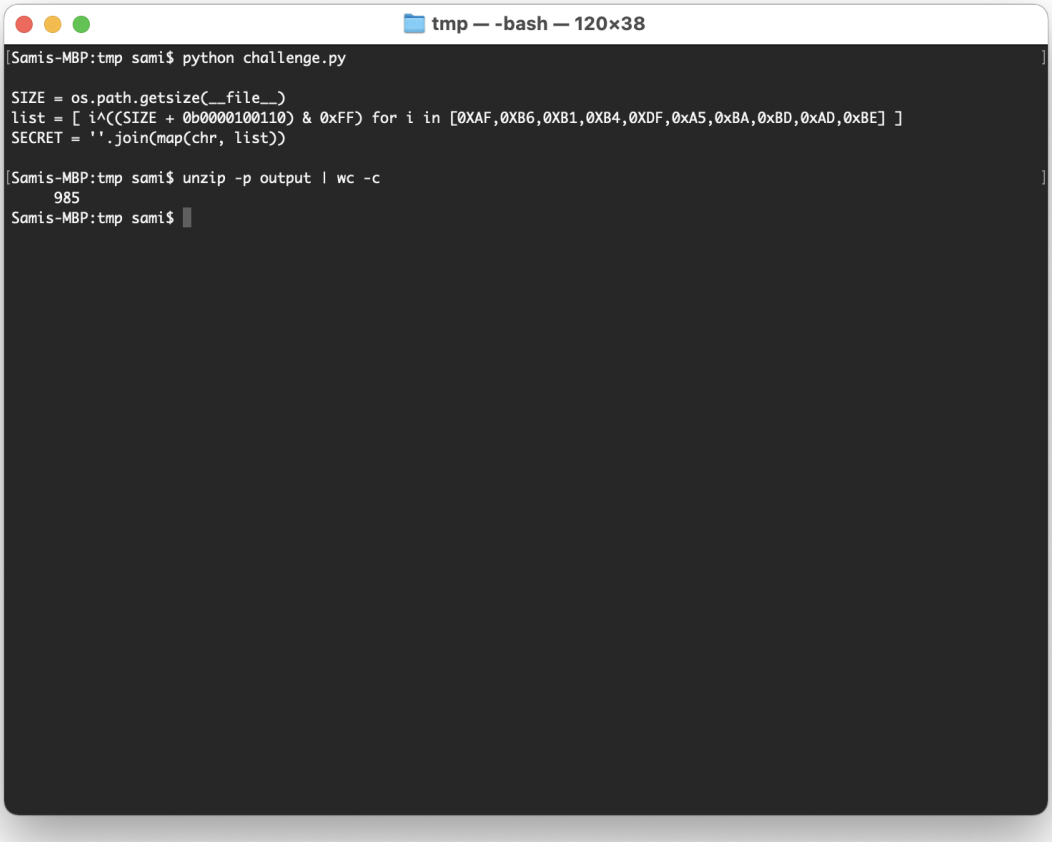
We can see this is a **Python** script and by the comment we know that the answer to the challenge is set in a variable called SECRET. However, apparently we cannot change the content of the file without changing the content of the variable. To understand what the script actually does we will replace the function exec with a print like this:



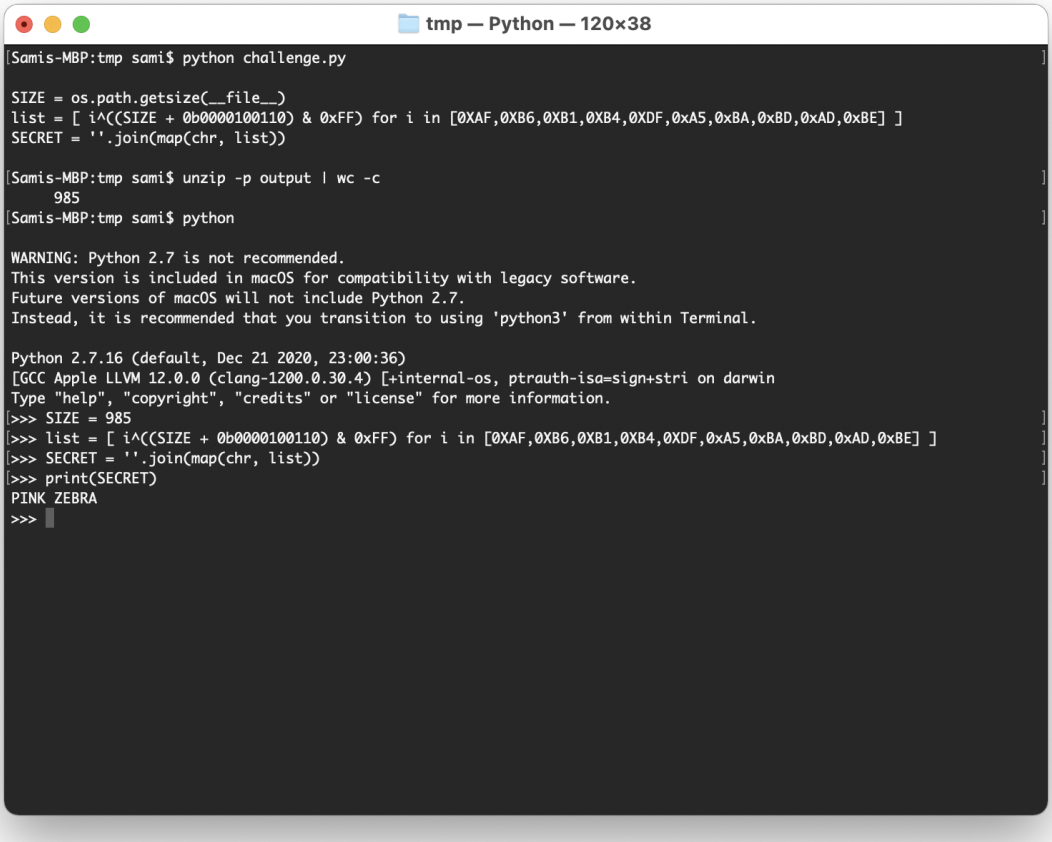
Running the script will result in the following output:



Apparently the variable SECRET is calculated using the size of the python script itself. This is why changing the file will change the value of SECRET. Now we have several options on how to proceed. But first we need to get the original file size of challenge.py either by undoing our change, or just unzip the original base64 decoded file straight into the command **wc**.



Apparently the original file was 985 bytes in size. If we open up a Python shell and just run the code we got from challenge.py but with SIZE=985 we can then read the variable SECRET to reveal the right answer to this puzzle:



And that's it. The right answer to this challenge was **PINK ZEBRA.**