



# Challenge 1: Pink Zebra

```
UEsDBBQAAAIAF0BFVJlgDy0oAEAANKDAAAMABwAY2hhbGx1bmd1LnB5VVQJAAPy/mBgfFR1YHV4CwABBPUBAA  
AEFAAAAHTS2/CMAY+51dYcBhIFTRtWpLDTozDTpM2btM0XRFWTqVFFYzBr5+dpB2PUcm0Y8efHefrGNaZhqRs  
9rqGtoIwd7vueCw05A1sqq78gLw07u+kzpN3DLysls+r9Ywyaa33XQFUWB1Qa0qRNsxbw+MGDlUHbbLdIew+bz  
NEILwc0y1W0WmoNg4LjxQFpFlSfuozZoeM5tdvbdQNYwttx5WCA+vDPDjvgcy9GCBotC0lQdhgCu3K+ecV0SB  
iMLAL1qQExsuX7iQH5LiNmZTuE8qCFxAIl6KrHA/6AHMTrozfVwQkD9Eoh6RsF1/Jp1s1fdKx5SwhcSNi4nQrd  
ICKGnsc1Exdr93hUjpVpNPfh5ggYUTMXTzNz5x0uBgcLzf2R0ISqLEKUMSJz68dySufLYF9c/Z4MINzzFNy1jI  
AEThZSC2XVw75Q2I60BsZ67cvFXoDWQjf4wskoGxDYAU58QLlRXLrbgfZT84877cV4a0lkjETUMx5VlamrmLYf  
bXRKKnskRjb4zpH510RqPZV5WXk22ym6RZ7QH9LdPp1P0CUEsBAh4DFAAAAAGAXQF9UkiAPLSgAQAA2QMAAAwA  
GAAAAAAAQAAKSBBBBBAAAGNoYWxsZW5nZS5weVVUBQAD8v5gYHV4CwABBPUBAAAEEFAAAAFBLBQYAAAAAQABAF  
IAAADmAQAAAA=
```

**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-Z-a-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 are two different ways of representing the same date, binary and base64 are 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 URLs 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:

The screenshot shows a terminal window titled "tmp — bash — 120x38". The command entered is "base64 -d < data.txt". The output consists of a long string of characters, mostly question marks and other non-printable symbols, which is the decoded base64 representation of the image file.

```
[Samis-MBP:tmp sami$ base64 -d < data.txt
P] }RH?<??
challange.pyUT      ??`-?`ux
                      ?uSKo?0
1?????4??rw          ??Wxph4mZ??N??N?6n?]N?E}??????Qyc??:?1?????w??x,4?
??x??2k}?@UTHM??Um??!>o3D ??-V?i?6
                      ??YR-?????]U?PS?-?Vp?
????,??????J0??E???(i?sQv?w?BiV?0~`??1t??>q??`p??J?dC'>?w$?I?????'?1M?X?D?e ?]`\;?
?"?e?8??W??H?MC1?YZ??a??D???Dco??NF??W??m??Y??-???P]RH?<??
                      ??challange.pyUT??`ux
?PKR?Samis-MBP:tmp sami$ ]
```

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**:

```

[Samis-MBP:tmp sami$ base64 -d -in data.txt | hexdump -C
00000000  50 4b 03 04 14 00 00 00  08 00 5d 01 7d 52 48 80  IPK.....].}RH.I
00000010  3c b4 a0 01 00 00 d9 03  00 00 00 01 c0 00 63 68  <.....chI
00000020  61 6c 6c 61 6e 67 65 2e  70 79 55 54 09 00 03 f2  lallange.pyUT....|
00000030  fe 60 60 2d ff 60 60 75  78 0b 00 01 04 f5 01 00  .`-.`ux.....|
00000040  00 04 14 00 00 00 75 53  4b 6f c2 30 0c be e7 57  I.....uSKo.0...W|
00000050  58 70 18 48 15 34 6d 5a  92 c3 4e 8c c3 4e 93 36  IXp.H.4mZ..N..N.6|
00000060  6e d3 0e 5d 17 d6 4a 5  45 7d 8c c1 af 9f 9d a4  In.].N.E}.....|
00000070  1d 8f 51 c9 8e 63 c7 9f  1d e7 eb 18 d6 99 86 a4  I..Q..c.....|
00000080  6c f6 ba 86 b6 82 16 77  bb ee 78 2c 34 e4 0d 6c  [l.....w..x,4..l|
00000090  aa ae fc 80 bc 34 ee ef  a4 ce 93 77 0c bc ac 96  I.....4.....w....|
000000a0  cf ab f5 8c 32 6b 7d d7  40 55 16 07 54 1a d2 a4  I....2k}.@U..T...|
000000b0  4d b3 19 1b c3 e3 06 0e  55 07 6d b2 dd 21 ec 3e  IM.....U.m.!>|
000000c0  6f 33 44 20 bc 1c d3 2d  56 d1 69 a8 36 0b 8f  l03D ...-V.i.6...|
000000d0  14 05 a4 59 52 7e ea 19  cc e7 8c e5 db 5d 55 b7  I...YR~.....|JU.|
000000e0  50 35 8c 2d 9f 1e 56 70  0f af 0c f0 e3 be 07 32  IP5.-..Vp.....|2|
000000f0  f4 60 81 a2 d0 8e 95 07  61 80 2b b7 2b e7 9c 54  I.`.....a.+..T|
00000100  e4 81 88 c9 40 97 5a 90  11 7b 2e 5f b8 90 1f 92  I....@.Z..{.....|
00000110  e2 36 66 53 b8 4f 2a 08  5c 40 20 be 8a ac 70 3f  I.6fS.0*.@...p?|
00000120  e8 01 cc 4e ba 33 7d 5c  10 90 3f 44 a2 1e 91 b0  I...N.3}\..\..D....|
00000130  5d 7f 26 9d 6c d5 f7 4a  c7 94 b0 85 c4 8d 8b 89  I].&l..J.....|
00000140  d0 ad d2 02 28 69 ec 73  51 17 76 bf 77 85 42 69  I....(i.sQ.v.w.Bil|
00000150  56 93 4f 7e 1e 60 81 85  13 31 74 f3 37 3e 71 d2  IV.0~.`...1t.7>q.|
00000160  e0 60 70 bc df d9 1d 08  4a a2 c4 91 43 12 27 3e  I.`p....J...C.'>I|
00000170  bc 77 24 ae 7c b6 05 f5  cf d9 e0 c2 27 cf 31 4d  I..w$.|.....'.1M|
00000180  cb 58 c8 00 44 e1 65 20  b6 5d 5c 3b e5 0d 88 eb  I.X..D.e .]`\;....|
00000190  40 6c 67 ae dc bc 55 e8  0d 64 23 7f 8c 2c 92 81  @!g...U..df#....|
000001a0  b1 0d 80 14 e7 c4 0b 95  15 cb ad b8 1f 65 3f 38  I.....e?8|
000001b0  f3 be dc 57 86 8e 96 48  c4 4d 43 31 e5 59 5a 9a  I..W...H.MC1.YZ.|
000001c0  b9 8b 61 f6 d7 44 a2 a7  b2 44 63 6f 8c e9 1f 9d  I..a..D...Dco....|
000001d0  4e 46 a3 d9 57 95 97 93  6d b2 9e a4 59 ed 01 fd  INF..W..m...Y...|
000001e0  2d d3 e9 94 fd 02 50 4b  01 02 1e 03 14 00 00 00  I.....PK.....|
000001f0  08 00 5d 01 7d 52 48 80  3c b4 a0 01 00 00 d9 03  I..].}RH.<.....|
00000200  00 00 0c 00 18 00 00 00  00 00 01 00 00 00 a4 81  I.....|
00000210  00 00 00 00 63 68 61 6c  6c 61 6e 67 65 2e 70 79  I....challange.py|
00000220  55 54 05 00 03 f2 fe 60  60 75 78 0b 00 01 04 f5  IUT....`ux....|
00000230  01 00 00 04 14 00 00 00  50 4b 05 06 00 00 00 00 00  I.....PK.....|
00000240  01 00 01 00 52 00 00 00  e6 01 00 00 00 00 00 00 00  I....R.....|

```

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 screenshot shows a Google search results page. The search bar at the top contains the query "PK Binary format". Below the search bar, there are several navigation links: "All" (highlighted in blue), "Images", "Videos", "News", "Shopping", "More", "Settings", and "Tools". The main content area displays search results. The first result is a link to "List of file signatures - Wikipedia" with the URL "https://en.wikipedia.org/wiki/List\_of\_file\_signatures". The second result is a link to "ZIP (file format) - Wikipedia" with the URL "https://en.wikipedia.org/wiki/ZIP\_(file\_format)". The third result is a link to "How to Look Into a File - FILExt" with the URL "https://filext.com/faq/look\_into\_files".

About 6 010 000 results (1,04 seconds)

[https://en.wikipedia.org/wiki/List\\_of\\_file\\_signatures](https://en.wikipedia.org/wiki/List_of_file_signatures) ▾

### List of file signatures - Wikipedia

This is a list of file signatures, data used to identify or verify the content of a file. Such signatures are also known as magic numbers or Magic Bytes. Many file formats are not intended to be read as text. ... msg, Compound File Binary Format, a container format used for document by older versions of Microsoft Office.

[https://en.wikipedia.org/wiki/ZIP\\_\(file\\_format\)](https://en.wikipedia.org/wiki/ZIP_(file_format)) ▾

### ZIP (file format) - Wikipedia

Most of the signatures end with the short integer 0x4b50, which is stored in little-endian ordering. Viewed as an ASCII string this reads "PK", the initials of the ...

[History](#) · [Version history](#) · [Design](#) · [Structure](#)

[https://filext.com/faq/look\\_into\\_files](https://filext.com/faq/look_into_files) ▾

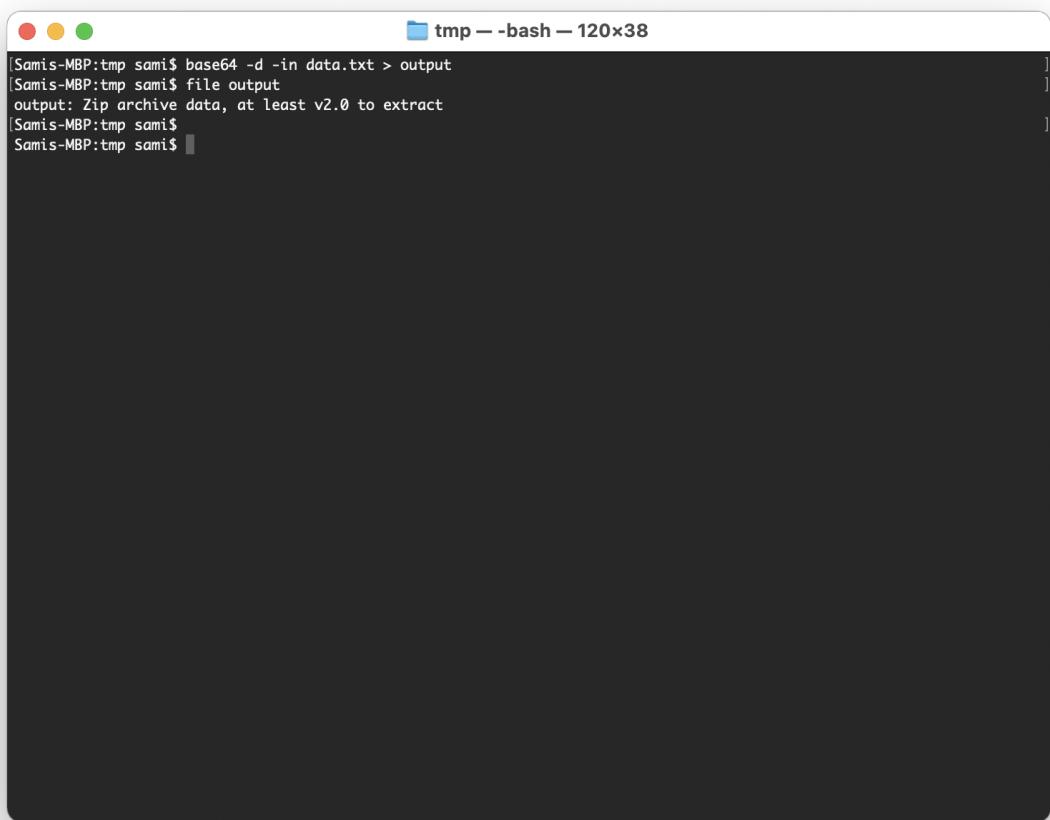
### How to Look Into a File - FILExt

When you open a binary file in EditPad Pro you see both the binary bytes and the ... If the first two characters are "PK" the file may be a ... If you need to extract information from a formatted data file but don't know the format of the data file ...

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			zip aar apk docx epub <a href="#">ipa</a> jar kmz <a href="#">maff</a> odp ods odt pk3 pk4 pptx usdz vsdx xlsx xpi	
50 4B 05 06	PK..	0		<a href="#">zip file format</a> and formats based on it, such as EPUB, JAR, ODF, OOXML
(empty archive)				
50 4B 07 08				
(spanned archive)				
52 61 72 21 1A 07 00	Rar!...	0	rar	RAR archive version 1.50 onwards <sup>[11]</sup>

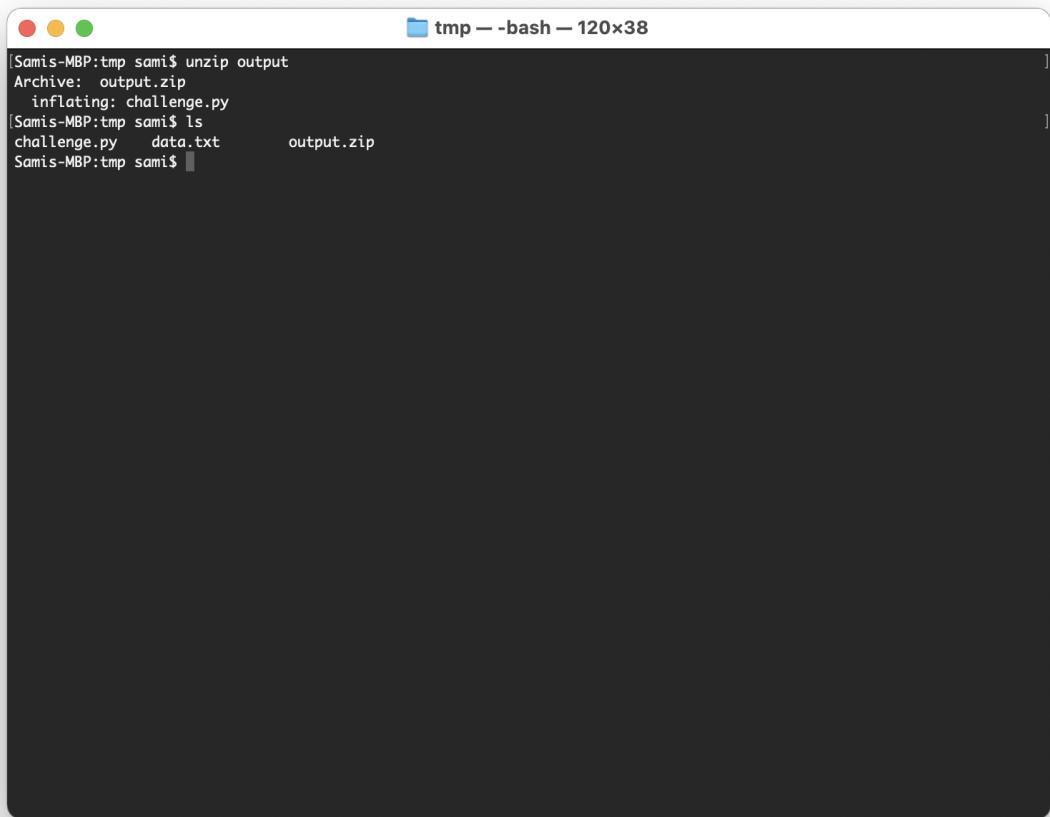
The **file** command confirms our suspicion:



A screenshot of a macOS terminal window titled "tmp -- bash - 120x38". The window has three colored window control buttons (red, yellow, green) at the top left. The terminal content is as follows:

```
[Samis-MBP:tmp sami$ base64 -d -in data.txt > output
[Samis-MBP:tmp sami$ file output
output: Zip archive data, at least v2.0 to extract
[Samis-MBP:tmp sami$ Samis-MBP:tmp sami$ ]
```

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



A screenshot of a macOS terminal window titled "tmp -- bash - 120x38". The window shows the following command-line session:

```
[Samis-MBP:tmp sami$ unzip output
Archive: output.zip
  inflating: challenge.py
[Samis-MBP:tmp sami$ ls
challenge.py    data.txt      output.zip
Samis-MBP:tmp sami$ ]
```

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

```
tmp — nano challenge.py — 120x38
GNU nano 2.0.6          File: challenge.py

# The answer to the puzzle is found in the variable SECRET. There's only one catch.
# If you tamper with this file the value of SECRET will change. //

import os

CODE = [
    10, 83, 73, 90, 69, 32, 61, 32, 111, 115, 46, 112, 97, 116,
    104, 46, 103, 101, 116, 115, 105, 122, 101, 40, 95, 95, 102,
    105, 108, 101, 95, 95, 41, 10, 108, 105, 115, 116, 32, 61,
    32, 91, 32, 105, 94, 40, 40, 83, 73, 90, 69, 32, 43, 32, 48,
    98, 48, 48, 48, 49, 48, 48, 49, 49, 48, 41, 32, 38, 32,
    48, 120, 70, 70, 41, 32, 102, 111, 114, 32, 105, 32, 105, 110,
    32, 91, 48, 88, 65, 70, 44, 48, 88, 66, 54, 44, 48, 88, 66,
    49, 44, 48, 88, 66, 52, 44, 48, 88, 68, 70, 44, 48, 120, 65,
    53, 44, 48, 120, 66, 65, 44, 48, 120, 66, 68, 44, 48, 120, 65,
    68, 44, 48, 120, 66, 69, 93, 32, 93, 10, 83, 69, 67, 82, 69,
    84, 32, 61, 32, 39, 39, 46, 106, 111, 105, 110, 40, 109, 97,
    112, 40, 99, 104, 114, 44, 32, 108, 105, 115, 116, 41, 41, 10
]

exec("".join(map(chr, CODE)))
```

[ Read 21 lines ]

^G Get Help   ^O WriteOut   ^R Read File   ^Y Prev Page   ^K Cut Text   ^C Cur Pos  
^X Exit   ^J Justify   ^W Where Is   ^V Next Page   ^U UnCut Text   ^T To Spell

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:

```
tmp — nano challenge.py — 120x38
GNU nano 2.0.6          File: challenge.py

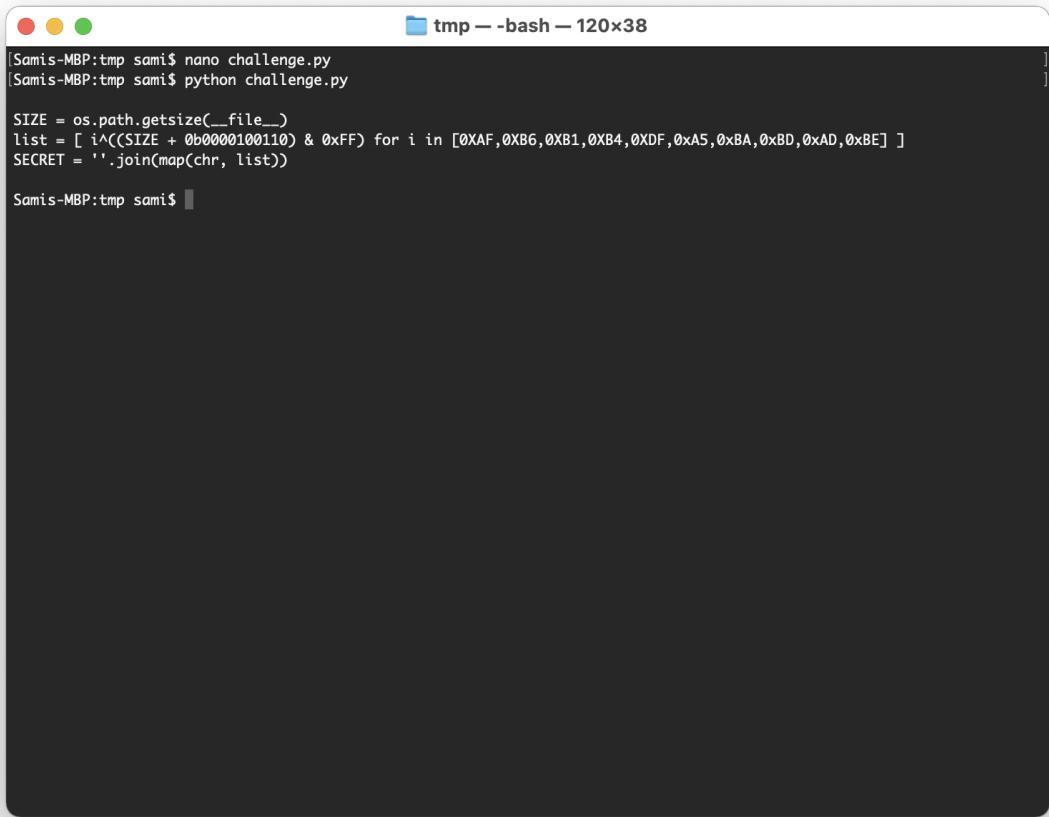
# The answer to the puzzle is found in the variable SECRET. There's only one catch.
# If you tamper with this file the value of SECRET will change. //

import os

CODE = [
    10, 83, 73, 90, 69, 32, 61, 32, 111, 115, 46, 112, 97, 116,
    104, 46, 103, 101, 116, 115, 105, 122, 101, 40, 95, 95, 102,
    105, 108, 101, 95, 95, 41, 10, 108, 105, 115, 116, 32, 61,
    32, 91, 32, 105, 94, 40, 40, 83, 73, 90, 69, 32, 43, 32, 48,
    98, 48, 48, 48, 49, 48, 48, 49, 49, 48, 41, 32, 38, 32,
    48, 120, 70, 70, 41, 32, 102, 111, 114, 32, 105, 32, 105, 110,
    32, 91, 48, 88, 65, 70, 44, 48, 88, 66, 54, 44, 48, 88, 66,
    49, 44, 48, 88, 66, 52, 44, 48, 88, 68, 70, 44, 48, 120, 65,
    53, 44, 48, 120, 66, 65, 44, 48, 120, 66, 68, 44, 48, 120, 65,
    68, 44, 48, 120, 66, 69, 93, 32, 93, 10, 83, 69, 67, 82, 69,
    84, 32, 61, 32, 39, 39, 46, 106, 111, 105, 110, 40, 109, 97,
    112, 40, 99, 104, 114, 44, 32, 108, 105, 115, 116, 41, 41, 10
]

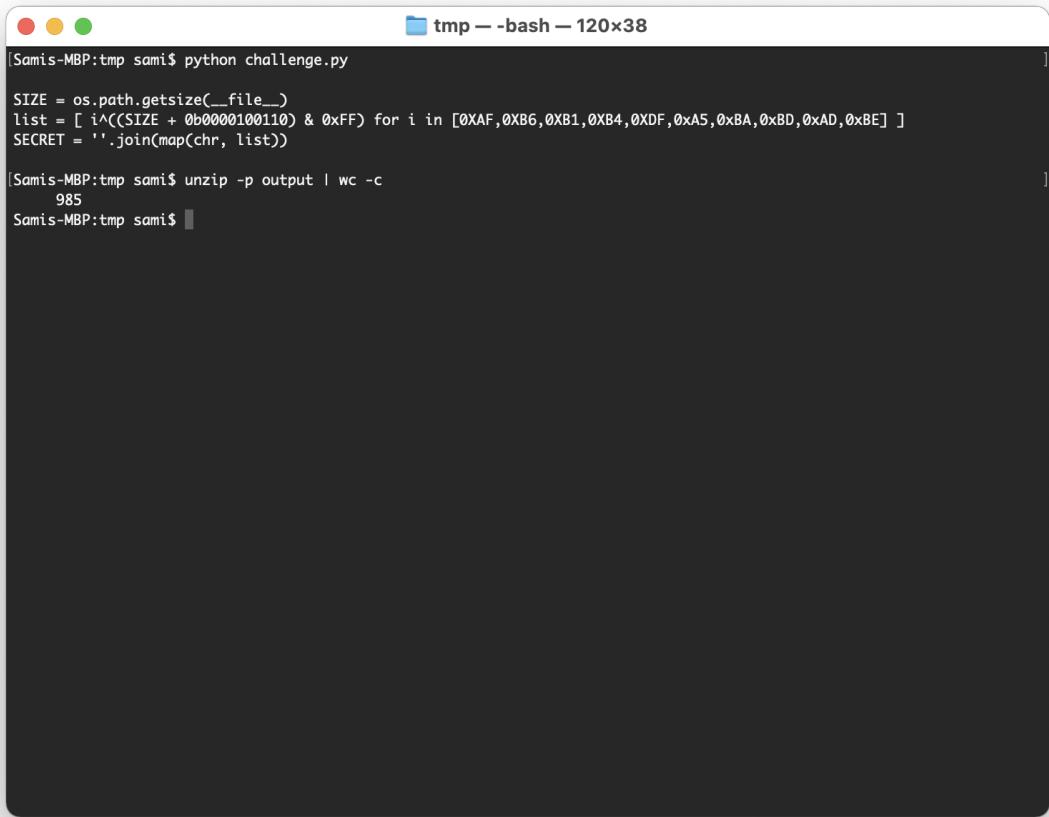
print"".join(map(chr, CODE))
```

Running the script will result in the following output:



```
[Samis-MBP:tmp sami$ nano challenge.py
[Samis-MBP:tmp sami$ python challenge.py
SIZE = os.path.getsize(__file__)
list = [ i^((CSIZE + 0b0000100110) & 0xFF) for i in [0xAF,0XB6,0XB1,0XB4,0XDF,0xA5,0xBA,0xBD,0xAD,0xBE] ]
SECRET = ''.join(map(chr, list))
Samis-MBP:tmp sami$ ]
```

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**.

A screenshot of a macOS terminal window titled "tmp -- bash - 120x38". The window contains the following text:

```
[Samis-MBP:tmp sami$ python challenge.py
SIZE = os.path.getsize(__file__)
list = [ i^((SIZE + 0b0000100110) & 0xFF) for i in [0xAF,0XB6,0XB1,0XB4,0XDF,0xA5,0xBA,0xBD,0xAD,0xBE] ]
SECRET = ''.join(map(chr, list))
[Samis-MBP:tmp sami$ unzip -p output | wc -c
985
Samis-MBP:tmp sami$ ]
```

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:

```
[Samis-MBP:tmp sami$ python challenge.py
SIZE = os.path.getsize(__file__)
list = [ i^((SIZE + 0b0000100110) & 0xFF) for i in [0xAF,0XB6,0XB1,0XB4,0XDF,0xA5,0xBA,0xBD,0xAD,0xBE] ]
SECRET = ''.join(map(chr, list))

[Samis-MBP:tmp sami$ unzip -p output | wc -c
985
[Samis-MBP:tmp sami$ python

WARNING: Python 2.7 is not recommended.
This version is included in macOS for compatibility with legacy software.
Future versions of macOS will not include Python 2.7.
Instead, it is recommended that you transition to using 'python3' from within Terminal.

Python 2.7.16 (default, Dec 21 2020, 23:00:36)
[GCC Apple LLVM 12.0.0 (clang-1200.0.30.4) [+internal-os, ptrauth-is=sign+stri on darwin
Type "help", "copyright", "credits" or "license" for more information.
[>>> SIZE = 985
[>>> list = [ i^((SIZE + 0b0000100110) & 0xFF) for i in [0xAF,0XB6,0XB1,0XB4,0XDF,0xA5,0xBA,0xBD,0xAD,0xBE] ]
[>>> SECRET = ''.join(map(chr, list))
[>>> print(SECRET)
PINK ZEBRA
>>>
```

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