

Forum:[Hack Forums](#)

Thread: [Custom Base64 Alphabet Encoder x Decode \(Alphabet Translator\)](#)

Replace - November 9, 2025, 2:02 pm

Was going through some of my previous notes and found this and thought to share this up, I thought was valuable, a simple Custom base64 Alphabet Translator. This can be extended to support CLI/WebApp integration. Also after some GIT searching, added in interactive mode below and also found this great tool which is already doing what this skeleton aimed for and much more. You can use any one of the works posted bellow, the one I provided or the one from GIT, just don't forget to like this post, will encourage me to share more with the community. ?

GIT *(Encode or decode base64 or base32 using a custom alphabet):

<https://github.com/darienhuss/custombase>

So, here it is raw:

Code import string
import base64

```
# custom encoding function
def EncodeCustomBase64(stringToBeEncoded,custom_b64):
    standard_b64 =
'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
    encoded = base64.b64encode(stringToBeEncoded)
    final_encoded = ""
    for ch in encoded:
        if (ch in custom_b64):
            final_encoded += custom_b64[string.find(standard_b64,str(ch))]
        elif (ch == '='):
            final_encoded += '='
    print(final_encoded)
```

```
# custom decoding function
def DecodeCustomBase64(stringToBeDecoded,custom_b64):
    standard_b64 =
'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
    temp_decoded = ""
    for ch in stringToBeDecoded:
        if (ch in custom_b64):
            temp_decoded += standard_b64[string.find(custom_b64,str(ch))]
        elif (ch == '='):
            temp_decoded += '='
    decoded = base64.b64decode(temp_decoded)
    print(decoded)
```

```
print "#####"
print "#### CAB-64 - Custom Alphabet Base64 [en|de]coder v0.1 ####"
print "#####"
```

```

while True:
# enter custom base64 alphabet
i.e"QpaZlivj4ndG=H021y+NO5RSTxPgUz67FMhYq8b3wemKfkJLBocCDrs9VtWXIEuA"
custom_base_64 = raw_input("> Enter your custom base64 alphabet (must be 64 char long):\n")
if len(custom_base_64) != 65:
    print "\n\t - [!] incorrect alphabet length - alphabet must be 64 char long\n"
else:
    while True:
        action = raw_input("> Are you Encoding (E) or Decoding (D)\n")
        if action != "D" and action != "E" and action != "e" and action != "d":
            print "\n\t[!] incorrect selection"
        elif action == "E" or action == "e":
            cleartext = raw_input("> Enter your cleartext that needs to be encoded:\n")
            EncodeCustomBase64(cleartext,custom_base_64)
        elif action == "D" or action == "d":
            ciphertext = raw_input("> Enter your ciphertext that needs to be deencoded\n")
            DecodeCustomBase64(ciphertext,custom_base_64)

```