Universidad San Francisco de Quito

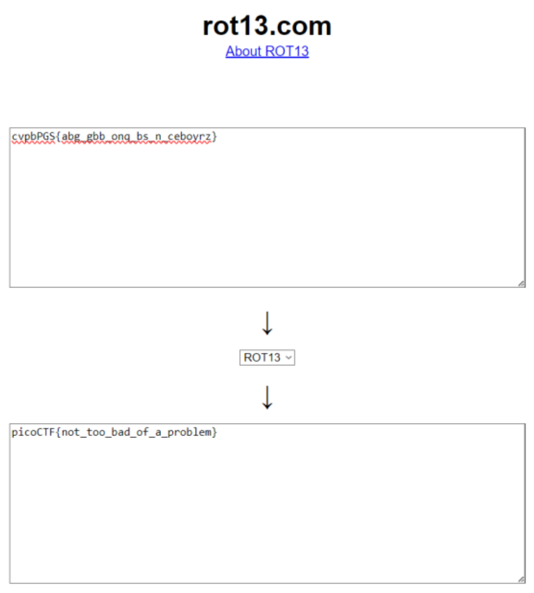
Computer Security

Name: Mateo Ruiz

Banner: 00212195

Exercises:

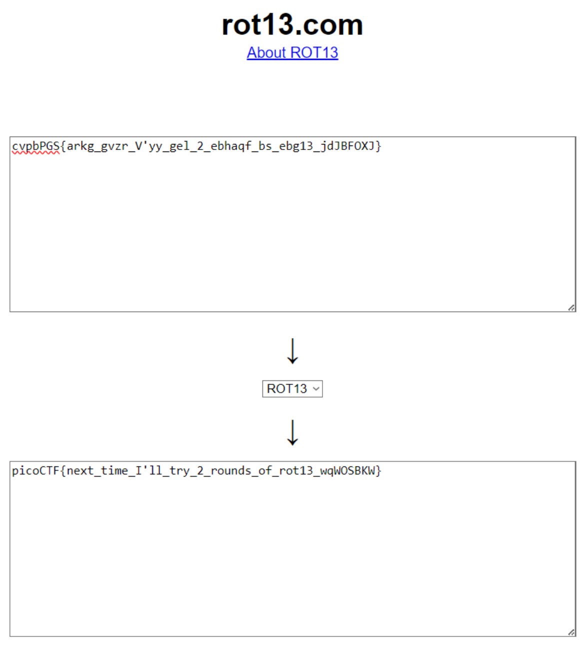
13:



```
cvpbPGS{abg_gbb_onq_bs_n_ceboyrz}
```

```
picoCTF{not_too_bad_of_a_problem}
```

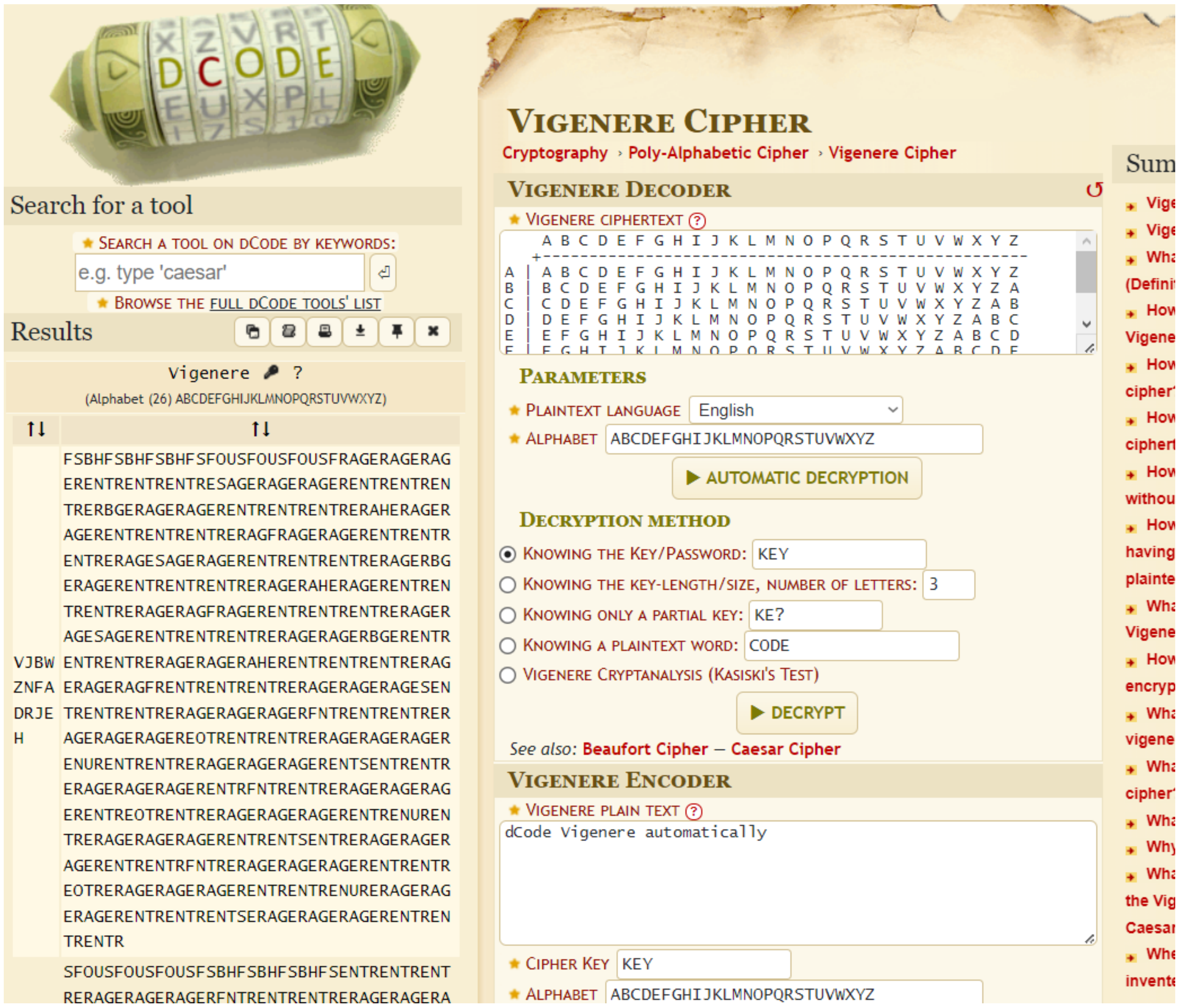
Mod 26:



```
cvpbPGS{arkg_gvzr_V'yy_gel_2_ebhaqf_bs_ebg13_jdJBFOXJ}
```

```
picoCTF{next_time_I'll_try_2_rounds_of_rot13_wqWOSBKW}
```

Easy 1:



Search for a tool

SEARCH A TOOL ON DCODE BY KEYWORDS:

e.g. type 'caesar'

BROWSE THE FULL DCODE TOOLS' LIST

Results

Vigenere ?

(Alphabet (26) ABCDEFGHIJKLMNOPQRSTUVWXYZ)

↑↓	↑↓
VJBW ZNFA DRJE H	FSBHFSBHFSBHFSFOUSFOUSFOUSFRAGERAGERAG ERENTRENTRENTRESAGERAGERAGERENTRENTREN TRERBGERAGERAGERENTRENTRENTRERAHERAGER AGERENTRENTRENTRERAGFRAGERAGERENTRENTR ENTRERAGESAGERAGERENTRENTRENTRERAGERBG ERAGERENTRENTRENTRERAGERAHERAGERENTREN TRENTRERAGERAGFRAGERENTRENTRENTRERAGER AGESAGERENTRENTRENTRERAGERAGERBGERENTR ENTRENTRERAGERAGERAHERENTRENTRENTRERAG ERAGERAGFRENTRENTRENTRERAGERAGERAGESEN TRENTRENTRERAGERAGERAGERFNTRENTRENTRER AGERAGERAGEREOTRENTRENTRERAGERAGERAGER ENURENTRENTRERAGERAGERAGERENTSENTRENTR ERAGERAGERAGERENTRFNTRENTRERAGERAGERAG ERENTREOTRENTRERAGERAGERAGERENTRENUREN TRERAGERAGERAGERENTRENTSENTRERAGERAGER AGERENTRENTRFNTRERAGERAGERAGERENTRENTR EOTRERAGERAGERAGERENTRENTRENURERAGERAG ERAGERENTRENTRENTSERAGERAGERAGERENTREN TRENTR
	SFOUSFOUSFOUSFSBHFSBHFSBHFSENTRENTRENT RERAGERAGERAGERFNTRENTRENTRERAGERAGERA

VIGENERE CIPHER

Cryptography › Poly-Alphabetic Cipher › Vigenere Cipher

VIGENERE DECODER

VIGENERE CIPHERTEXT ?

```
    A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
   +---------------------------------------------------
A  | A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
B  | B C D E F G H I J K L M N O P Q R S T U V W X Y Z A
C  | C D E F G H I J K L M N O P Q R S T U V W X Y Z A B
D  | D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
E  | E F G H I J K L M N O P Q R S T U V W X Y Z A B C D
F  | F G H I J K L M N O P Q R S T U V W X Y Z A B C D E
```

PARAMETERS

PLAINTEXT LANGUAGE: English

ALPHABET: ABCDEFGHIJKLMNOPQRSTUVWXYZ

AUTOMATIC DECRYPTION

DECRYPTION METHOD

[✓ KNOWING THE KEY/PASSWORD: KEY]

KNOWING THE KEY-LENGTH/SIZE, NUMBER OF LETTERS: 3

KNOWING ONLY A PARTIAL KEY: KE?

KNOWING A PLAINTEXT WORD: CODE

VIGENERE CRYPTANALYSIS (KASISKI'S TEST)

DECRYPT

See also: Beaufort Cipher – Caesar Cipher

VIGENERE ENCODER

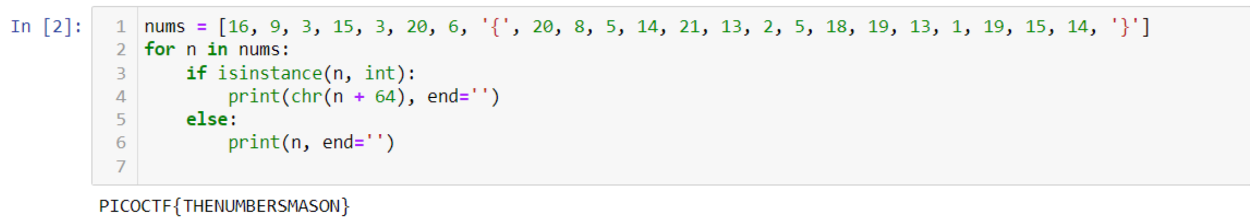
VIGENERE PLAIN TEXT ?

dCode Vigenere automatically

CIPHER KEY: KEY

ALPHABET: ABCDEFGHIJKLMNOPQRSTUVWXYZ

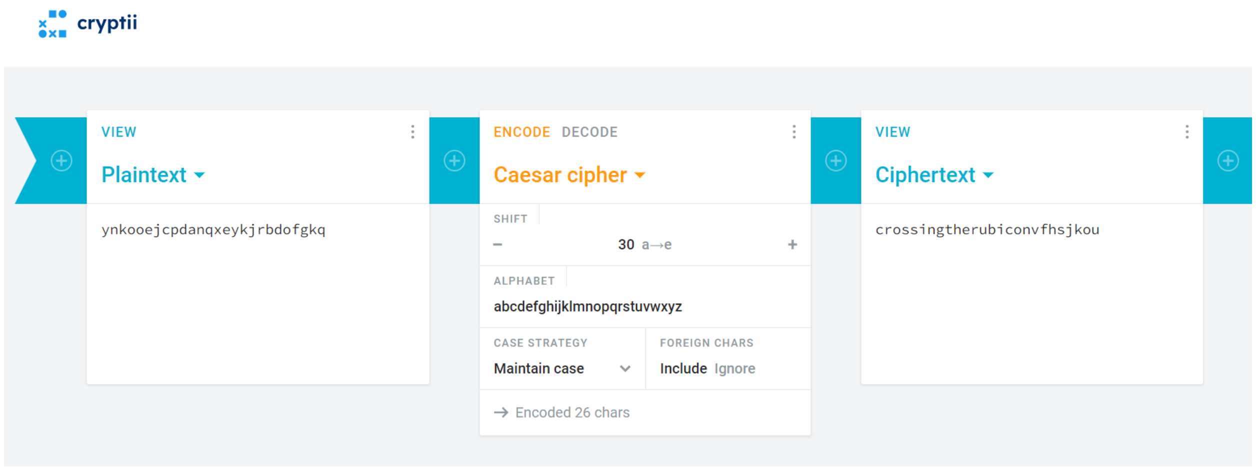
The Numbers:



```
In [2]:
nums = [16, 9, 3, 15, 3, 20, 6, '{', 20, 8, 5, 14, 21, 13, 2, 5, 18, 19, 13, 1, 19, 15, 14, '}']
for n in nums:
    if isinstance(n, int):
        print(chr(n + 64), end='')
    else:
        print(n, end='')

```

```
PICOCTF{THENUMBERSMASON}
```

Caesar:



ynkooejcpdanqxeykjrbdofgkq

SHIFT
30 a→e

ALPHABET
abcdefghijklmnopqrstuvwxyz

CASE STRATEGY
Maintain case

FOREIGN CHARS
[✓ Include] Ignore

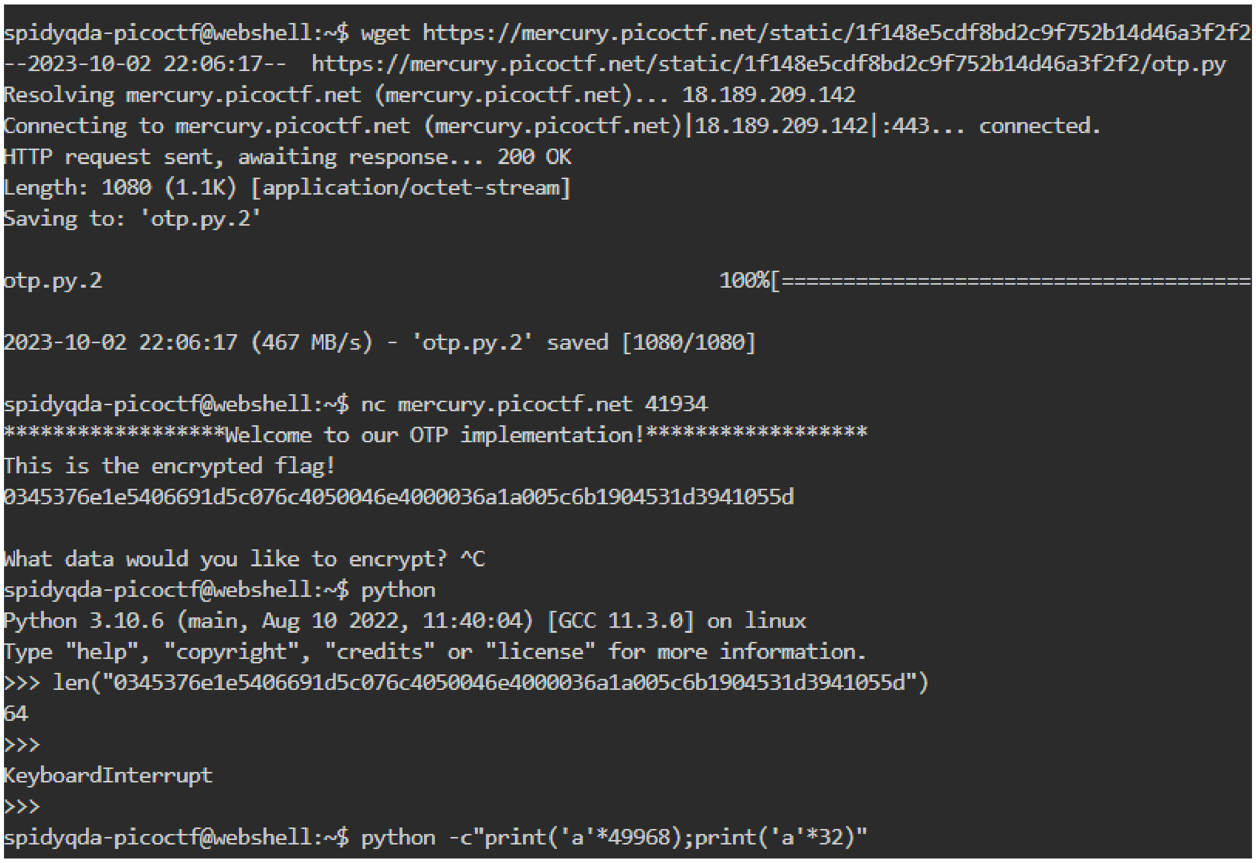
→ Encoded 26 chars

crossingtherubiconvfhsjkou

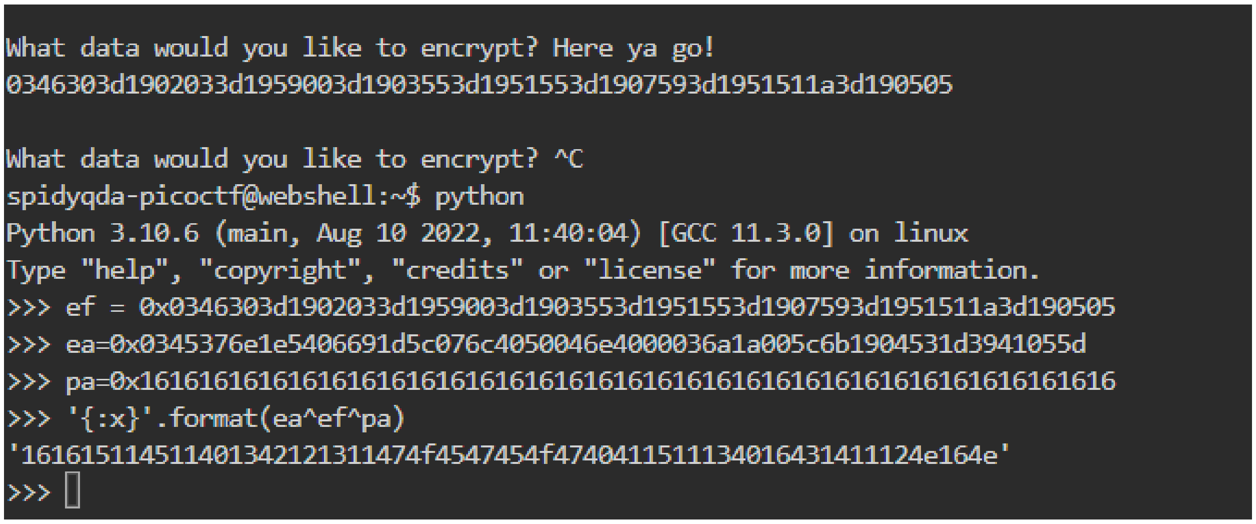
New Caesar:

```python
import string
import random

ALPHABET = string.ascii_lowercase[:16]
LOWERCASE = ord("a")

def encode_b16(text):
    encoded_text = ""
    for char in text:
        binary = "{0:08b}".format(ord(char))
        encoded_text += ALPHABET[int(binary[:4], 2)]
        encoded_text += ALPHABET[int(binary[4:], 2)]
    return encoded_text

def shift_characters(char, key_char):
    char_value = ord(char) - LOWERCASE
    key_value = ord(key_char) - LOWERCASE
    return ALPHABET[(char_value + key_value) % len(ALPHABET)]

flag_text = "".join(random.choice(string.ascii_letters + string.digits) for _ in range(8))
key = "".join(random.choice(ALPHABET) for _ in range(8))
assert all([k in ALPHABET for k in key])
assert len(key) == 8

encoded_flag = encode_b16(flag_text)
encrypted_text = ""
for i, char in enumerate(encoded_flag):
    encrypted_text += shift_characters(char, key[i % len(key)])
print(encrypted_text)

```

Easy Peasy



```
spidyqda-picoctf@webshell:~$ wget https://mercury.picoctf.net/static/1f148e5cdf8bd2c9f752b14d46a3f2f2
--2023-10-02 22:06:17--  https://mercury.picoctf.net/static/1f148e5cdf8bd2c9f752b14d46a3f2f2/otp.py
Resolving mercury.picoctf.net (mercury.picoctf.net)... 18.189.209.142
Connecting to mercury.picoctf.net (mercury.picoctf.net)|18.189.209.142|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1080 (1.1K) [application/octet-stream]
Saving to: 'otp.py.2'

otp.py.2                                       100%[=====================================

2023-10-02 22:06:17 (467 MB/s) - 'otp.py.2' saved [1080/1080]

spidyqda-picoctf@webshell:~$ nc mercury.picoctf.net 41934
******************Welcome to our OTP implementation!******************
This is the encrypted flag!
0345376e1e5406691d5c076c4050046e4000036a1a005c6b1904531d3941055d

What data would you like to encrypt? ^C
spidyqda-picoctf@webshell:~$ python
Python 3.10.6 (main, Aug 10 2022, 11:40:04) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> len("0345376e1e5406691d5c076c4050046e4000036a1a005c6b1904531d3941055d")
64
>>>
KeyboardInterrupt
>>>
spidyqda-picoctf@webshell:~$ python -c"print('a'*49968);print('a'*32)"
```



```
What data would you like to encrypt? Here ya go!
0346303d1902033d1959003d1903553d1951553d1907593d1951511a3d190505

What data would you like to encrypt? ^C
spidyqda-picoctf@webshell:~$ python
Python 3.10.6 (main, Aug 10 2022, 11:40:04) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> ef = 0x0346303d1902033d1959003d1903553d1951553d1907593d1951511a3d190505
>>> ea=0x0345376e1e5406691d5c076c4050046e4000036a1a005c6b1904531d3941055d
>>> pa=0x1616161616161616161616161616161616161616161616161616161616161616
>>> '{:x}'.format(ea^ef^pa)
'16161511451140134212131147 4f4547454f474041151113401643141 1124e164e'
>>> 
```

Spelling quiz

```python
import random
import os

file_list = [os.path.join(path, file) for path, _, files in os.walk('.')
             for file in files if file.split('.')[-1] == 'txt']

original_alphabet = list('abcdefghijklmnopqrstuvwxyz')
shuffled_alphabet = list(original_alphabet)
random.shuffle(shuffled_alphabet)
char_mapping = dict(zip(original_alphabet, shuffled_alphabet))

for filename in file_list:
    with open(filename, 'r') as file:
        text = file.read()

    encrypted_text = ''.join(char_mapping[c] if c in char_mapping else c for c in text)

    encrypted_filename = filename + '.encrypted'
    with open(encrypted_filename, 'w') as encrypted_file:
        encrypted_file.write(encrypted_text)
```