Ciphers with Python

•••

By: Ryan Gordon

Overview

- What is a cipher?
- Common ciphers
  - Caesar cipher
  - Transposition cipher
  - Vigenère cipher
- Modified Vigenère cipher
- Significance for encryption
- Comparison
- Github repository
- Works cited

What is a cipher?

- Encoded message
- Secret way to communicate
- Any algorithm to obfuscate a message
- Primitive
    - Caesar cipher
    - Transposition
- Complex
    - Modern encryption
    - One-time pad
    - AES256

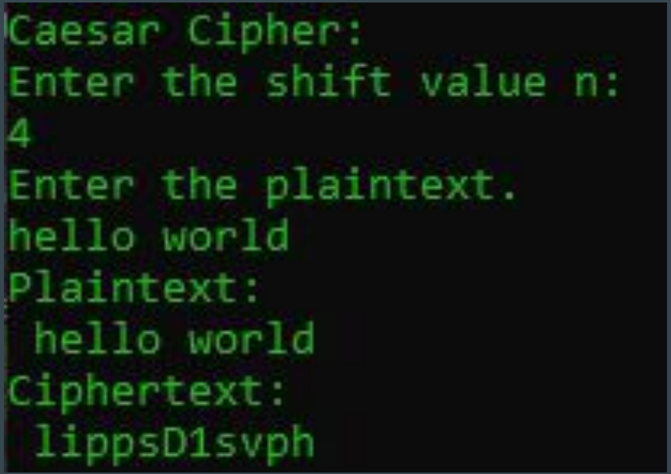
Caesar Cipher

- Most commonly taught cipher
- Named after Julius Caesar
- Also known as a ‘shift cipher’
- Accomplished by shifting across an established alphabet by a set number of letters
    - Before: ABCDEFGHIJKLMNOPQRSTUVWXYZ
    - Shift by n=3
    - After: DEFGHIJKLMNOPQRSTUVWXYZABC

Caesar Cipher in Python

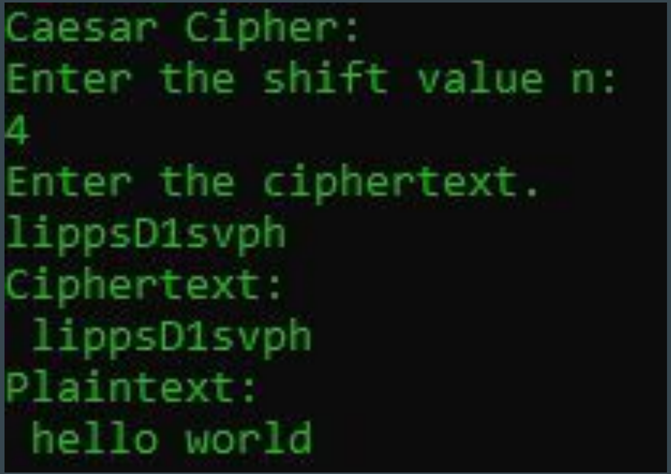
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890~!@#$%^&*()_+`-=[]\{}|;':",./<>?

Encrypt



```
Caesar Cipher:
Enter the shift value n:
4
Enter the plaintext.
hello world
Plaintext:
 hello world
Ciphertext:
 lippsD1svph
```

Decrypt



```
Caesar Cipher:
Enter the shift value n:
4
Enter the ciphertext.
lippsD1svph
Ciphertext:
 lippsD1svph
Plaintext:
 hello world
```

Transposition Cipher

- Plaintext characters are shifted in a regular pattern
- Ciphertext is ultimately one of `n!` possible permutations of the plaintext characters, where `n` is the length of the message
    - This means that the characters are generally the same in the ciphertext as the plaintext (except for null characters in some ciphers if `(n % columns != 0)`)
- Because the frequency of each character in the plaintext is (generally) the same as the frequency in the ciphertext, transposition ciphers can be broken through brute-force rearrangement of characters until a meaningful message is produced

Column Transposition Cipher

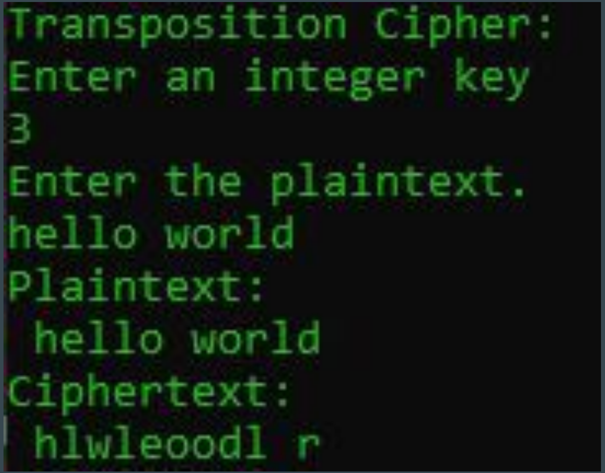
- key = `3`
- plaintext = `hello world`
  - The plaintext is written on the rows of the table of length key
- ciphertext = `hlwleoodl r`
  - The ciphertext is read from the columns of the table
- Message length is constant
- Character frequency remains constant:

h	e	l	o	w	r	s	
1	1	3	2	1	1	1	1

1	2	3
h	e	l
l	o	
w	o	r
l	d	

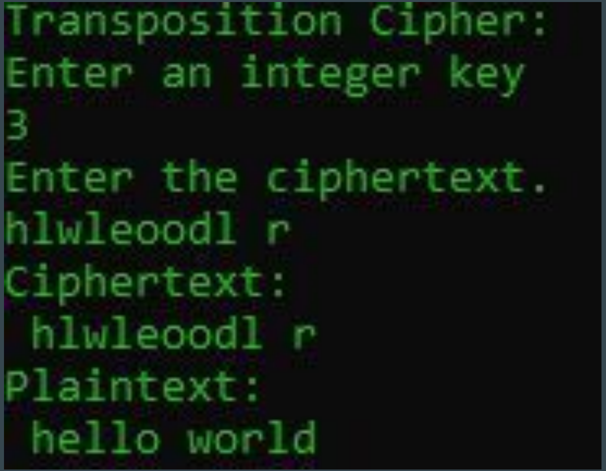
Transposition Cipher in Python

Encrypt



```
Transposition Cipher:
Enter an integer key
3
Enter the plaintext.
hello world
Plaintext:
 hello world
Ciphertext:
 hlwleoodl r
```

Decrypt



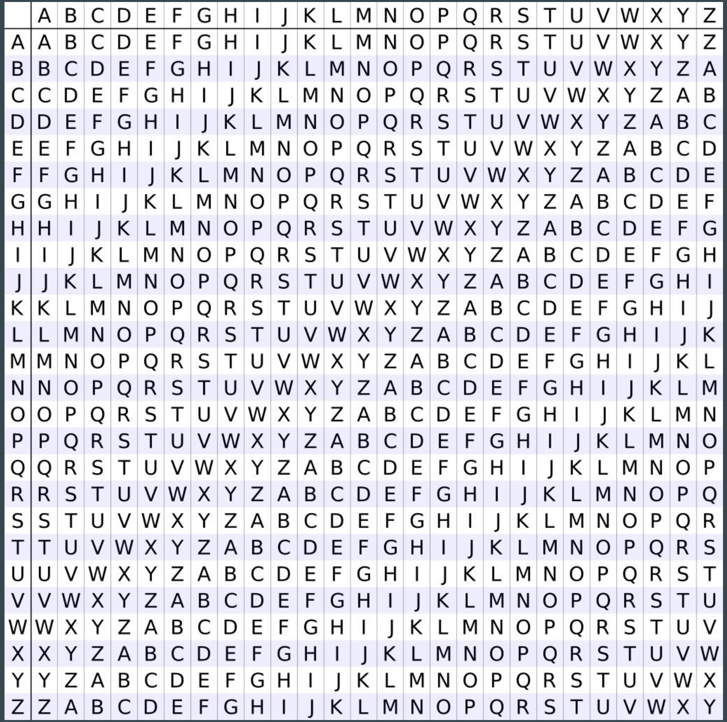
```
Transposition Cipher:
Enter an integer key
3
Enter the ciphertext.
hlwleoodl r
Ciphertext:
 hlwleoodl r
Plaintext:
 hello world
```

Vigenère Cipher

- Le chiffre indéchiffrable
  - ‘The indecipherable cipher’
- In use since the 1500s
- Mainly unbroken until the 1900s
  - A few exceptions exist in the 1800s
- Works through polyalphabetic substitution

Vigenère Cipher

- Key on the top row
- Message on the left side
- plaintext = `cipher`
- key = `key` = `keykey`
- Go to `c` on the left side and `k` on the top
  - = `m`
- Go to `i` on the left side and `e` on the top
  - = `m`
- Go to `p` on the left side and `y` on the top
  - = `n`
- etc.
- ciphertext = `mmnrip`

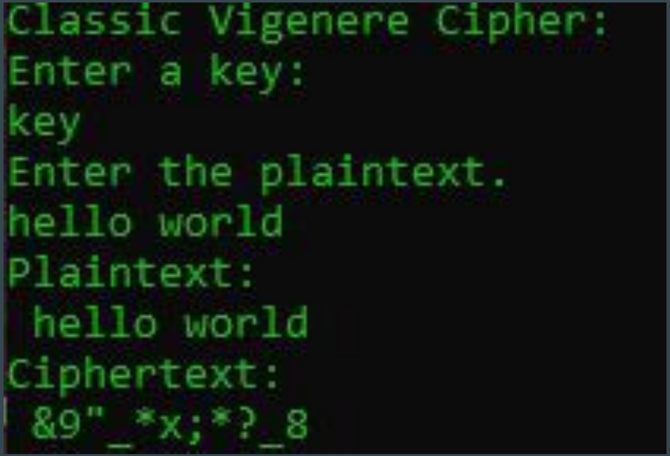


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T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

Vigenère Cipher in Python

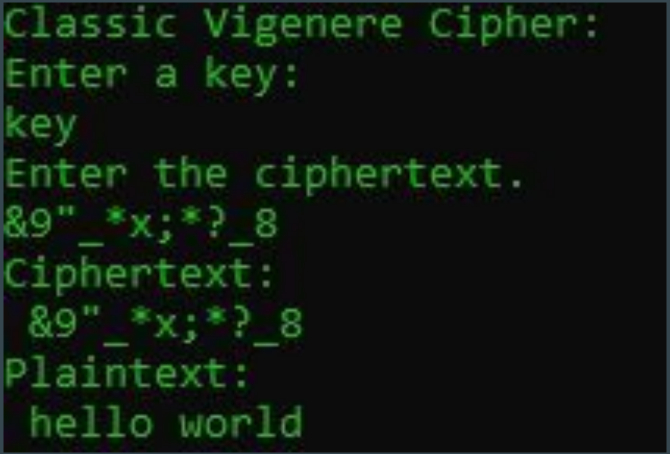
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890~!@#$%^&*()_+`-=[]\{}|;':",./<>?

Encrypt



```
Classic Vigenere Cipher:
Enter a key:
key
Enter the plaintext.
hello world
Plaintext:
 hello world
Ciphertext:
 &9"_*x;*?_8
```

Decrypt



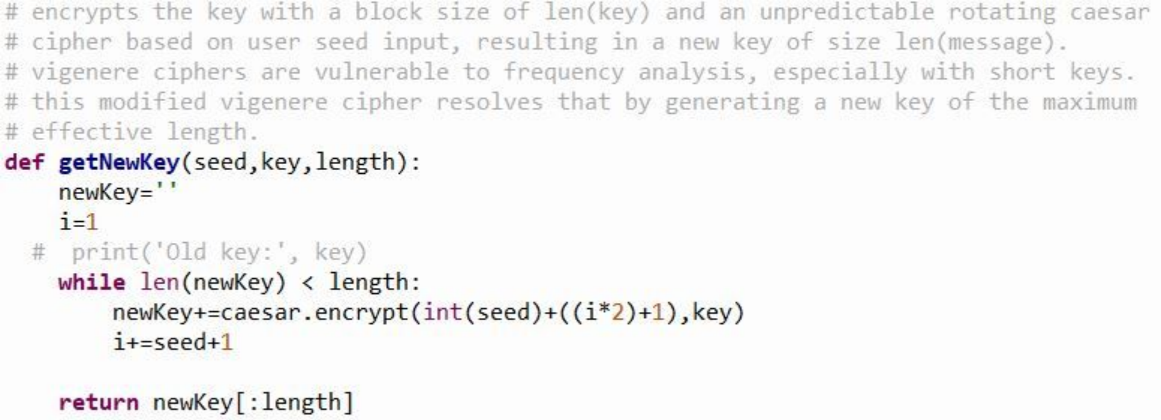
```
Classic Vigenere Cipher:
Enter a key:
key
Enter the ciphertext.
&9"_*x;*?_8
Ciphertext:
 &9"_*x;*?_8
Plaintext:
 hello world
```

Vigenère Keys

- Classic Vigenère is weak with short keys
    - Key repeats over length of the entire message
    - Message: `hello world`
    - Entered key: `key`
    - Used key: `keykeykeyke`

Modified Vigenère Keys

- Run an algorithm to ‘randomize’ the key
    - Dependent on seed value and repeated Caesar-shift of the key
    - Length of key = length of message
        - Increases key entropy (unpredictability)
        - Reduces risk of frequency analysis



```
# encrypts the key with a block size of len(key) and an unpredictable rotating caesar
# cipher based on user seed input, resulting in a new key of size len(message).
# vigenere ciphers are vulnerable to frequency analysis, especially with short keys.
# this modified vigenere cipher resolves that by generating a new key of the maximum
# effective length.
def getNewKey(seed,key,length):
    newKey=''
    i=1
  #  print('Old key:', key)
    while len(newKey) < length:
        newKey+=caesar.encrypt(int(seed)+((i*2)+1),key)
        i+=seed+1

    return newKey[:length]
```

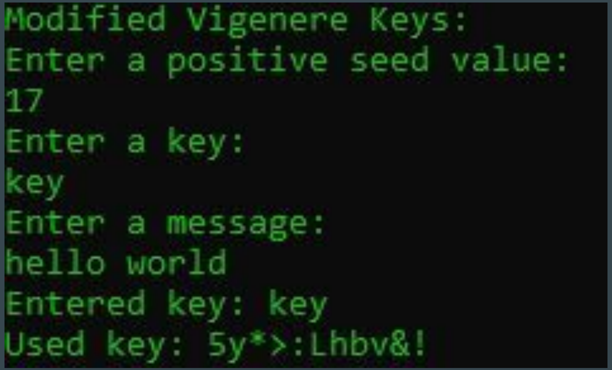
Modified Vigenère Keys

- Modified Vigenère is strong even with short keys
    - Message: `hello world`
    - Entered key: `key`
    - Seed: `17`
    - Unmodified key: `keykeykeyke`
    - Modified key: `5y*>:Lhbv&!`
- Makes frequency analysis based on short keys with repeating characters more difficult/impossible

Modified Vigenère Key Values

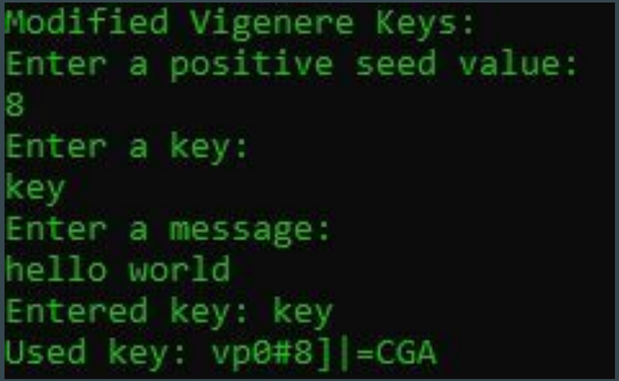
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890~!@#$%^&*()_+`-=[]\{}|;':",./<>?

Seed = 17



```
Modified Vigenere Keys:
Enter a positive seed value:
17
Enter a key:
key
Enter a message:
hello world
Entered key: key
Used key: 5y*>:Lhbv&!
```

Seed = 8



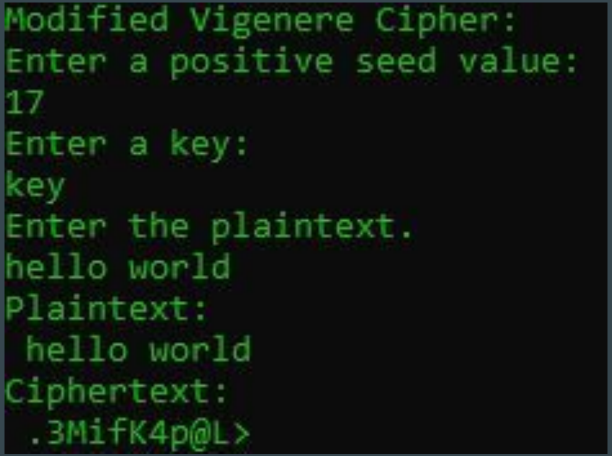
```
Modified Vigenere Keys:
Enter a positive seed value:
8
Enter a key:
key
Enter a message:
hello world
Entered key: key
Used key: vp0#8]|=CGA
```

Modified Vigenère Cipher in Python

ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890~!@#$%^&*()_+`-=[]\{}|;':",./<>?

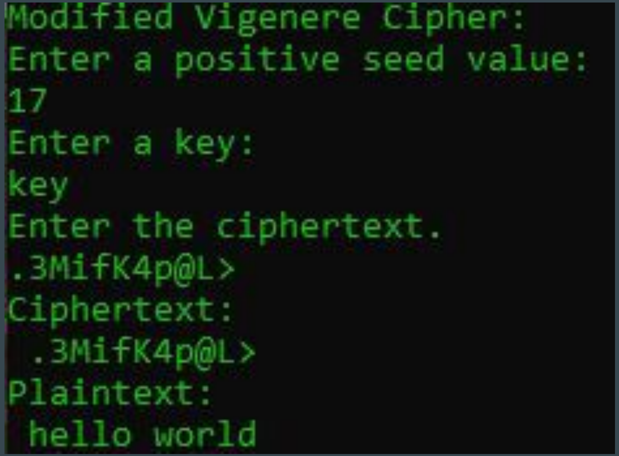
Seed = 17

Encrypt



```
Modified Vigenere Cipher:
Enter a positive seed value:
17
Enter a key:
key
Enter the plaintext.
hello world
Plaintext:
 hello world
Ciphertext:
 .3MifK4p@L>
```

Decrypt



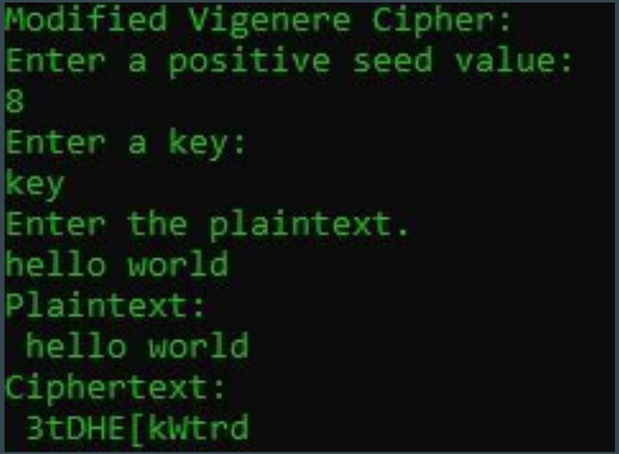
```
Modified Vigenere Cipher:
Enter a positive seed value:
17
Enter a key:
key
Enter the ciphertext.
.3MifK4p@L>
Ciphertext:
 .3MifK4p@L>
Plaintext:
 hello world
```

Modified Vigenère Cipher in Python

ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890~!@#$%^&*()_+`-=[]\{}|;':",./<>?

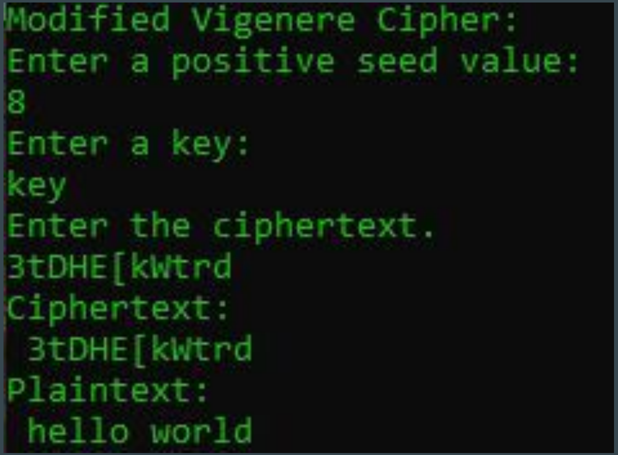
Seed = 8

Encrypt



```
Modified Vigenere Cipher:
Enter a positive seed value:
8
Enter a key:
key
Enter the plaintext.
hello world
Plaintext:
 hello world
Ciphertext:
 3tDHE[kWtrd
```

Decrypt



```
Modified Vigenere Cipher:
Enter a positive seed value:
8
Enter a key:
key
Enter the ciphertext.
3tDHE[kWtrd
Ciphertext:
 3tDHE[kWtrd
Plaintext:
 hello world
```

Comparison

Cipher	Shift/Key	Seed	Key Used	Plaintext	Ciphertext
Caesar	4	N/A	4	hello world	lippsD1svph
Transposition	3	N/A	3	hello world	hlwleoodl r
Classic Vigenère	key	N/A	keykeykeyke	hello world	&9”_*x;*?_8
Modified Vigenère	key	17	5y*>:Lhbv&!	hello world	.3MifK4p@L>

Github Repository

- Available at https://github.com/flashrgordon/encryption
- Covered under GNU General Public License 3.0

Works Cited

"Caesar Cipher." Wikipedia. Wikimedia Foundation, October 17, 2020. https://en.wikipedia.org/wiki/Caesar_cipher.

Sweigart, Al. *Cracking Codes with Python: an Introduction to Building and Breaking Ciphers*. San Francisco, California: No Starch Press,Inc., 2018.

"Transposition Cipher," October 13, 2020. https://en.wikipedia.org/wiki/Transposition_cipher.

"Vigenère Cipher." Wikipedia. Wikimedia Foundation, October 25, 2020. https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher.