Caesar / Rot13

Shifting cipher, which was used by Julius Caesar

- Cipher
- Description

Input (plaintext) length: 50

```
Hello this is a test.
Please enter your text
here.
```

Encrypt

Decrypt

Key: - 1 +

Output (ciphertext) length: 50

```
Ifmmp uijt jt b uftu.
Qmfbtf foufs zpvs ufyu
ifsf.
```

- Options
- Alphabet

Blocks of five

Keep non-alphabet characters

Plaintext alphabet ABCDEFGHIJKLMNOPQR Ciphertext alphabet BCDEFGHIJKLMNOPQRS

Upper case

Lower case

Digits

Punctuation marks

Umlauts

Blanks

Use constructed alphabet

Define own alphabet

This application is also available implemented in Python. Running it as Python downloads a local runtime environment (8MB compressed) into your browser. You can edit the Python code and it will use the values from the upper GUI components. Feel free to try it:

Hide local Python implementation

If the Python code below is visible, it is used to calculate the upper output. To use the default (JavaScript) implementation again, just click the hide button. The code below can be edited, so you can experiment with it. You can also download the code and run it in a local terminal on your computer.

Terminal command: `python caesarpy --encrypt --key 1 --message 'Hello this is a test. Please enter your text here.' --alphabet 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz' --keep-non-alp`

This website would like to use cookies for Google Analytics.

Accept Reject

Run again | Reset code | Download code

me:

ipse

ose shadowfox yeti solarized

t size: 12

```python
import argparse
import sys


def caesar(text, key, alphabet, b_encrypt, b_keep_chars, b_block_of_five):
    ciphertext = ""

    # iterate through text
    for old_character in text:
        new_character = ""

        # if character is in alphabet append to ciphertext
        if(old_character in alphabet):
            index = alphabet.index(old_character)

            if(b_encrypt):  # if text is to be encrypted
                new_index = (index + key) % len(alphabet)

            else:  # if text is to be decrypted
                new_index = (index - key) % len(alphabet)

            new_character = alphabet[new_index]

        else:

            # if the symbol is not in alphabet then regard block_of_five and b_encrypt
            if(not b_keep_chars):
                continue
            else:
                if(b_block_of_five and b_encrypt):
                    if(old_character != " "):
                        new_character = old_character
                    else:
                        continue
                else:
                    new_character = old_character

        ciphertext = ciphertext + new_character

        # if blocks_of_five is true, append a space after every 5 characters
        if(b_block_of_five and b_encrypt):
            if(len(ciphertext.replace(" ", "")) % 5 == 0):
                ciphertext = ciphertext + " "

    # Output
    print(ciphertext)


if __name__ == "__main__":

    # add command line arguments
    parser = argparse.ArgumentParser()
    parser.add_argument("-k", "--key", help="key for encrypt / decrypt", type=int)
    parser.add_argument("-e", "--encrypt", dest="crypt_modus", action="store_true")
    parser.add_argument("-d", "--decrypt", dest="crypt_modus", action="store_false")
    parser.add_argument("-f", "--blocks-of-five", dest="blocks_of_five", action="store_true")
    parser.add_argument("-m", "--message", help="message for encrypt / decrypt", type=str, required=True)
    parser.add_argument("-ka","--keep-non-alp", help="keep non-alphabet characters", dest="keep_chars", action=
    parser.add_argument("-a", "--alphabet", help="defined alphabet", type=str)

    if len(sys.argv) == 1:
        sys.exit(1)

    args = parser.parse_args()
```

Yes ✖No

ckground

The Caesar cipher is named after the Roman military and political leader Gaius Julius Caesar (100 BC – 44 BC). form of ciphering to encipher military messages.

Description

The classic version uses the capital letters A-Z, but, in principle, an arbitrary alphabet can be used. The first ste times.

```
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  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
```

Now, the bottom alphabet is shifted by an arbitrary number of positions. The number of positions is the value alphabet 3 positions to the right yields the following result:

```
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
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
```

The letter `A` becomes the letter `D`, `B` is replaced by `E`, and `C` replaced by `F`, etc. The word `example` would be en alphabet is called 'plaintext alphabet' and the lower alphabet is called 'ciphertext alphabet'.

Security

The number of possible keys is identical to the size of the given alphabet. Using the capital letters A-Z as alph the 26th key rendered meaningless because it would map each letter to itself. With only 25 meaningful keys, i possible keys until the correct one is found (brute-force analysis). The Caesar cipher can also easily be cracked

Internal working of the local Python version

About the code

In this plugin you can control the encryption process in two ways: