

---

# SecretPy Documentation

*Release 1.0*

**[Read the Docs](#)**

**Jul 27, 2021**



---

## Contents:

---

|          |                                  |          |
|----------|----------------------------------|----------|
| <b>1</b> | <b>SecretPy</b>                  | <b>1</b> |
| 1.1      | Description . . . . .            | 1        |
| 1.2      | Installation . . . . .           | 2        |
| 1.3      | Usage . . . . .                  | 2        |
| 1.3.1    | Direct way . . . . .             | 2        |
| 1.3.2    | CryptMachine . . . . .           | 3        |
| 1.3.3    | CompositeMachine . . . . .       | 4        |
| 1.4      | Maintainers . . . . .            | 5        |
| <b>2</b> | <b>API</b>                       | <b>7</b> |
| 2.1      | ADFGX . . . . .                  | 7        |
| 2.1.1    | Examples . . . . .               | 8        |
| 2.2      | ADFGVX . . . . .                 | 10       |
| 2.2.1    | Examples . . . . .               | 10       |
| 2.3      | Affine . . . . .                 | 12       |
| 2.3.1    | Examples . . . . .               | 12       |
| 2.4      | Atbash . . . . .                 | 14       |
| 2.4.1    | Examples . . . . .               | 15       |
| 2.5      | Autokey . . . . .                | 16       |
| 2.5.1    | Examples . . . . .               | 16       |
| 2.6      | Bazeries . . . . .               | 18       |
| 2.6.1    | Examples . . . . .               | 19       |
| 2.7      | Beaufort . . . . .               | 20       |
| 2.7.1    | Examples . . . . .               | 20       |
| 2.8      | Bifid . . . . .                  | 22       |
| 2.8.1    | Examples . . . . .               | 23       |
| 2.9      | Caesar . . . . .                 | 24       |
| 2.9.1    | Examples . . . . .               | 25       |
| 2.10     | Caesar Progressive . . . . .     | 26       |
| 2.10.1   | Examples . . . . .               | 27       |
| 2.11     | Chao . . . . .                   | 29       |
| 2.11.1   | Examples . . . . .               | 29       |
| 2.12     | Columnar Transposition . . . . . | 30       |
| 2.12.1   | Examples . . . . .               | 31       |
| 2.13     | Four Square . . . . .            | 32       |
| 2.13.1   | Examples . . . . .               | 33       |

|          |                                   |           |
|----------|-----------------------------------|-----------|
| 2.14     | Gronsfeld . . . . .               | 34        |
| 2.14.1   | Examples . . . . .                | 35        |
| 2.15     | Keyword . . . . .                 | 36        |
| 2.15.1   | Examples . . . . .                | 37        |
| 2.16     | MyszkowskiTransposition . . . . . | 39        |
| 2.16.1   | Examples . . . . .                | 39        |
| 2.17     | Nihilist . . . . .                | 41        |
| 2.17.1   | Examples . . . . .                | 41        |
| 2.18     | Playfair . . . . .                | 43        |
| 2.18.1   | Examples . . . . .                | 44        |
| 2.19     | Polybius . . . . .                | 45        |
| 2.19.1   | Examples . . . . .                | 46        |
| 2.20     | Porta . . . . .                   | 47        |
| 2.20.1   | Examples . . . . .                | 47        |
| 2.21     | Rot13 . . . . .                   | 49        |
| 2.21.1   | Examples . . . . .                | 49        |
| 2.22     | Rot5 . . . . .                    | 50        |
| 2.22.1   | Examples . . . . .                | 51        |
| 2.23     | Rot18 . . . . .                   | 52        |
| 2.23.1   | Examples . . . . .                | 52        |
| 2.24     | Rot47 . . . . .                   | 53        |
| 2.24.1   | Examples . . . . .                | 54        |
| 2.25     | Scytale . . . . .                 | 54        |
| 2.25.1   | Examples . . . . .                | 55        |
| 2.26     | Simple Substitution . . . . .     | 56        |
| 2.26.1   | Examples . . . . .                | 57        |
| 2.27     | Three Square . . . . .            | 59        |
| 2.27.1   | Examples . . . . .                | 59        |
| 2.28     | Trifid . . . . .                  | 60        |
| 2.28.1   | Examples . . . . .                | 61        |
| 2.29     | Two Square . . . . .              | 62        |
| 2.29.1   | Examples . . . . .                | 63        |
| 2.30     | Vic . . . . .                     | 64        |
| 2.30.1   | Examples . . . . .                | 64        |
| 2.31     | Vigenere . . . . .                | 65        |
| 2.31.1   | Examples . . . . .                | 66        |
| 2.32     | Zigzag . . . . .                  | 67        |
| 2.32.1   | Examples . . . . .                | 68        |
| <b>3</b> | <b>Indices and tables</b>         | <b>71</b> |
|          | <b>Index</b>                      | <b>73</b> |

**Download:**

<https://pypi.org/project/secretpy>

**Documentation:**

<https://secretpy.readthedocs.io>

**Source code & Development:**

<https://github.com/tigertv/secretpy>

## 1.1 Description

SecretPy is a cryptographic Python package. It uses the following classical cipher algorithms:

- Affine
- Atbash
- Bazeries
- Beaufort
- Caesar, Caesar Progressive
- Chaocipher
- Keyword
- Playfair, Two Square(Double Playfair), Three Square, Four Square
- Polybius, ADFGX, ADFGVX, Bifid, Trifid, Nihilist
- Rot13, Rot5, Rot18, Rot47
- Simple Substitution

- Transposition: Columnar, Scytale, Spiral, Myszkowski, Zigzag(Railfence)
- Vic
- Vigenere, Autokey, Gronsfeld, Porta

## 1.2 Installation

To install this library, you can use pip:

```
pip install secretpy
```

Alternatively, you can install the package using the repo's cloning and the make:

```
git clone https://github.com/tigertv/secretpy
cd secretpy
make install
```

## 1.3 Usage

### 1.3.1 Direct way

The cipher classes can encrypt only characters which exist in the alphabet, and they don't have a state.

```
from secretpy import Caesar, alphabets as al

def encdec(cipher, plaintext, key, alphabet=al.ENGLISH):
    print(
↳ '=====
↳ ')
    print(plaintext)
    enc = cipher.encrypt(plaintext, key, alphabet)
    print(enc)
    print(cipher.decrypt(enc, key, alphabet))

key = 3
cipher = Caesar()

plaintext = u"thequickbrownfoxjumpsoverthelazydog"
encdec(cipher, plaintext, key)

alphabet = al.GERMAN
plaintext = u"schweißgequältvomödentextzürnttypografjakob"
encdec(cipher, plaintext, key, alphabet)

alphabet = al.SWEDISH
plaintext = u"faqomschweizklövdutrångpjäxby"
encdec(cipher, plaintext, key, alphabet)

'''
Output:
```

(continues on next page)

(continued from previous page)

```

=====
thequickbrownfoxjumpsoverthelazydog
wkhtxlfneurzqiramxpsvryhuwkhodcbgrj
thequickbrownfoxjumpsoverthelazydog
=====
schweißgequältvomödentextzürnttypografjakob
vfkzhlcjhtxßowyrpaghqwhäwübuqwwösrjudimdnre
schweißgequältvomödentextzürnttypografjakob
=====
faqomschweizklövdutrångpjäxby
idtrpvfkzhlöncygxwuaqjsmbåä
faqomschweizklövdutrångpjäxby
'''

```

### 1.3.2 CryptMachine

CryptMachine saves a state. There are alphabet, key and cipher, they can be changed at anytime. In the previous example, plaintext contains only characters existing in the alphabet i.e. without spaces and etc. To change the behaviour, you can use CryptMachine and decorators(SaveAll, Block), so it's a preferred way to do encryption/decryption:

```

from secretpy import Caesar, CryptMachine, alphabets as al
from secretpy.cmdecorators import SaveAll, Block

def encdec(machine, plaintext):
    print("-----")
    print(plaintext)
    enc = machine.encrypt(plaintext)
    print(enc)
    print(machine.decrypt(enc))

key = 3
cipher = Caesar()
cm0 = CryptMachine(cipher, key)

cm = cm0
cm.set_alphabet(al.ENGLISH)
plaintext = "I don't love non-alphabet characters. I will remove all of them: ^,&@$~
↳(*;?&#. Great!"
encdec(cm, plaintext)

cm = Block(cm, length=5, sep="-")
plaintext = "This text is divided by blocks of length 5!"
encdec(cm, plaintext)

cm = SaveAll(cm0)
plaintext = "I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!"
encdec(cm, plaintext)

cm.set_alphabet(al.ENGLISH_SQUARE_IJ)
plaintext = "Jj becomes Ii because we use ENGLISH_SQUARE_IJ!"
encdec(cm, plaintext)

```

(continues on next page)

(continued from previous page)

```

cm.set_alphabet(al.JAPANESE_HIRAGANA)
cm.set_key(1)
plaintext = u"text  "
encdec(cm, plaintext)

'''
Output:

-----
I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
lgrqworyhqrqdoskdehwfkdudfwhuvlzlouhpryhdooriwkhpjuhdw
idontlovenonalphabetcharactersiwillremoveallofthemgreat
-----

This text is divided by blocks of length 5!
wklvw-hawlv-glylg-hgebe-orfnv-riohq-jwk
thistextisdividedbyblocksoflength

-----

I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
L oryh qrq-doskdehw fkdudfwhuv. Wkhvh duh : ^,&@$~(*;?&#. Wkdw'v lw!
I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
-----

Jj becomes Ii because we use ENGLISH_SQUARE_IJ!
Mm ehfrphv Mm ehfdxvh zh xvh HQKOMVL_VTXDUH_MM!
Ii becomes Ii because we use ENGLISH_SQUARE_II!
-----

text
text
text
'''

```

### 1.3.3 CompositeMachine

Combining several ciphers to get more complex cipher, you can use CompositeMachine:

```

from secretpy import Rot13, Caesar, CryptMachine, CompositeMachine
from secretpy.cmdecorators import SaveAll

def encdec(machine, plaintext):
    print("=====")
    print(plaintext)
    enc = machine.encrypt(plaintext)
    print(enc)
    dec = machine.decrypt(enc)
    print(dec)

key = 5
plaintext = u"Dog jumps four times and cat six times"
print(plaintext)

cm1 = SaveAll(CryptMachine(Caesar(), key))
enc = cm1.encrypt(plaintext)
print(enc)

```

(continues on next page)

(continued from previous page)

```

cm2 = SaveAll(CryptMachine(Rot13()))
enc = cm2.encrypt(enc)
print(enc)

print("=====")

cm = CompositeMachine(cm1)
cm.add_machines(cm2)
enc = cm.encrypt(plaintext)
print(enc)
encdec(cm, plaintext)

cm.add_machines(cm1, cm2)
encdec(cm, plaintext)

'''
Output:

Dog jumps four times and cat six times
Itl ozrux ktzw ynrjx fsi hfy xnc ynrjx
Vgy bmehk xgmj laewk sfv usl kap laewk
=====
Vgy bmehk xgmj laewk sfv usl kap laewk
=====
Dog jumps four times and cat six times
Vgy bmehk xgmj laewk sfv usl kap laewk
Dog jumps four times and cat six times
=====
Dog jumps four times and cat six times
Nyq tewzc pyeb dswoc kxn mkd csh dswoc
Dog jumps four times and cat six times

'''

```

## 1.4 Maintainers

- @tigertv (Max Vetrov)



## 2.1 ADFGX

**class** `secretpy.ADFGX`

The ADFGX Cipher

**decrypt** (*text*, *key*, *alphabet*=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'))

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key is in ENGLISH
- **alphabet** (*string*) – Alphabet which will be used(length=25), if there is no a value, ENGLISH\_SQUARE\_IJ is used

**Returns** `text`

**Return type** `string`

**encrypt** (*text*, *key*, *alphabet*=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'))

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key is in ENGLISH
- **alphabet** (*string*) – Alphabet which will be used(length=25), if there is no a value, ENGLISH\_SQUARE\_IJ is used

**Returns** `text`

**Return type** `string`

## 2.1.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import ADFGX, CryptMachine, alphabets as al
5  from secretpy.cmdecorators import SaveAll, Block, UpperCase
6
7
8  alphabet = (
9      u"b", u"t", u"a", u"l", u"p",
10     u"d", u"h", u"o", u"z", u"k",
11     u"q", u"f", u"v", u"s", u"n",
12     u"g", u"ij", u"c", u"u", u"x",
13     u"m", u"r", u"e", u"w", u"y"
14 )
15 key = "cargo"
16 cipher = ADFGX()
17 plaintext = u"attackatonce"
18
19 print(plaintext)
20 enc = cipher.encrypt(plaintext, key, alphabet)
21 print(enc)
22 print(cipher.decrypt(enc, key, alphabet))
23 print("-----")
24
25
26 def encdec(machine, plaintext):
27     print(plaintext)
28     enc = machine.encrypt(plaintext)
29     print(enc)
30     print(machine.decrypt(enc))
31     print("-----")
32
33
34 cm0 = CryptMachine(cipher, key)
35 cm = UpperCase(Block(cm0))
36 cm.set_alphabet(alphabet)
37 plaintext = u"Attack at once!"
38 encdec(cm, plaintext)
39
40 cm1 = SaveAll(cm0)
41 plaintext = u"Attack at once!"
42 encdec(cm1, plaintext)
43
44 alphabet = al.GERMAN_SQUARE
45 key = "german"
46 cm.set_key(key)
47 cm.set_alphabet(alphabet)
48 plaintext = u"Schweißgequält vom öden Text zürnt Typograf Jakob"
49 encdec(cm, plaintext)
50
51 alphabet = (
52     u"úû", u"áá", u"b", u"cč", u"dd'",
53     u"f", u"g", u"h", u"ííj", u"zž",
54     u"k", u"l", u"m", u"nň", u"oó",
55     u"p", u"q", u"rř", u"sš", u"tt'",

```

(continues on next page)

(continued from previous page)

```

56     u"v", u"w", u"x", u"yý", u"eěě",
57 )
58 key = "czech"
59 cm.set_key(key)
60 cm.set_alphabet(alphabet)
61 plaintext = u"Necht' již hříšné saxofony d'áblů rozezvučí síň úděsnými tóny waltzu,
↳ tanga a quickstepu."
62 encdec(cm, plaintext)
63
64 alphabet = (
65     u"zž", u"aää", u"b", u"cč", u"dd'",
66     u"eé", u"f", u"g", u"h", u"iij",
67     u"k", u"líl'", u"m", u"oóô", u"p",
68     u"q", u"rř", u"sš", u"tt'", u"uú",
69     u"v", u"w", u"x", u"yý", u"nň",
70 )
71 key = "slovak"
72 cm.set_key(key)
73 cm.set_alphabet(alphabet)
74 plaintext = u"Vypätá dcéra grófa Maxwella s IQ nižším ako kôň núti čel'ad' hrýzt'
↳ hřbu jabík."
75 encdec(cm, plaintext)
76
77 alphabet = (
78     u"α", u"β", u"γ", u"δ", u"ε",
79     u"ζ", u"η", u"θ", u"ι", u"κ",
80     u"λ", u"μ", u"ν", u"ξ", u"ο",
81     u"π", u"ρ", u"σ", u"τ", u"υ",
82     u"φ", u"χ", u"ψ", u"ω", u" ",
83 )
84 key = "greek"
85 cm.set_key(key)
86 cm.set_alphabet(alphabet)
87 plaintext = u"Ξεσκεπζω την ψυχοφθρα σα βδελυγμα."
88 encdec(cm, plaintext)
89
90 '''
91 attackatonce
92 faxdfaddgdgfffaifaxafx
93 attackatonce
94 -----
95 Attack at once!
96 FAXDF ADDDG DGFFF AFAXA FAFX
97 ATTACKATONCE
98 -----
99 Attack at once!
100 Faxdfa dd dgdg!fffaifaxafx
101 Attack at once!
102 -----
103 Schweißgequält vom öden Text zürnt Typograf Jakob
104 DDAAX FFXGG FGDFE DFAAG GGGDG GAADG XGGFF AGGGG FAAAF XDXGD XXXFG DAXFG XAAGF FXGXD
↳ GGAAD GGFAA XFXDD D
105 SCHWEISGEQUALTOMODENTEXTZURNNTYPOGRAFIAKOB
106 -----
107 Necht' již hříšné saxofony d'áblů rozezvučí síň úděsnými tóny waltzu, tanga a
↳ quickstepu.
108 FGDxD GAXFX FFXAD GAGFX XDDxD DDAxX XGGGG GFFGA ADXAG AXXGF DGAfD AGGAX FDFGX XAXDA
↳ XDAGG XGDXX DADAD AGGAX DFFGF XAFGX XGDAG GGGAX GGAAF XAGDG DGXDD GADFX AGGAX FFXAD
↳ ADGGG X

```

(continues on next page)

(continued from previous page)

```

109 NECHTIIZHRISNESAXOFONYDABLUROZEZVUCISINUDESNYMITONYWALTZUTANGAAQUICKSTEPU
110 -----
111 Vypätá dcéra grófa Maxwella s IQ nižším ako kôň núti čel'ad' hrýzt' hrbu jabík.
112 FADDD ADAGA FFXGD AXDGA XDAFD DADAA FGXGA XGGXF ADXDD DFDFX FDAXX DGADX DXGAA FFXFD
113 ↪DDFFG AAGGA AFXAA GGAXF GXGAF XDFDA GDFGG GDGFD DXXXA GXGDD GFDA
114 VYPATADCERAGROFAMAXWELLASIQNIZSIMAKOKONNUTICELADHRYZTHRBUIABLK
115 -----
116 Ξεσκεπζω την ψυχοφθρα σα βδελυγμα.
117 AXAGF XXXGF AXDXA AGFXA GFAXA GFFGA DFFFA AAFA GXDGX DDDAD FFAGD AXDGX FAGGG D
118 ΞΕΣΚΕΠΑΖΩ ΤΗΝ ΨΥΧΟΦΘΟΡΑΣΑΒΔΕΛΥΓΜΙΑ
119 -----
120 '''

```

## 2.2 ADFGVX

**class** secretpy.**ADFGVX**

The ADFGVX Cipher

**decrypt** (*text*, *key*, *alphabet=None*)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key*, *alphabet=None*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English with numbers is used

**Returns** text

**Return type** string

### 2.2.1 Examples

```

1 #!/usr/bin/python
2 # -*- encoding: utf-8 -*-
3
4 from secretpy import ADFGVX, CryptMachine
5 from secretpy.cmdecorators import Block, UpperCase

```

(continues on next page)

(continued from previous page)

```

6
7
8 def encdec(machine, plaintext):
9     print(plaintext)
10    enc = machine.encrypt(plaintext)
11    print(enc)
12    print(machine.decrypt(enc))
13    print("-----")
14
15
16 key = "privacy"
17 cm = UpperCase(Block(CryptMachine(ADFGVX(), key), 4))
18
19 alphabet = (
20     u"n", u"a", u"l", u"c", u"3", u"h",
21     u"8", u"t", u"b", u"2", u"o", u"m",
22     u"e", u"5", u"w", u"r", u"p", u"d",
23     u"4", u"f", u"6", u"g", u"7", u"i",
24     u"9", u"j", u"0", u"k", u"1", u"q",
25     u"s", u"u", u"v", u"x", u"y", u"z",
26 )
27 cm.set_alphabet(alphabet)
28 plaintext = "Attack at 12.00am!"
29 encdec(cm, plaintext)
30
31 key = "pangram"
32 cm.set_key(key)
33 plaintext = "The quick brown fox jumps over the lazy dog."
34 encdec(cm, plaintext)
35
36 alphabet = (
37     u"5", u"", u"", u"", u"", u"",
38     u"", u"", u"", u"", u"", u"",
39     u"", u"", u"", u"", u"", u"",
40     u"", u"", u"", u"", u"", u"",
41     u"", u"", u"", u"", u"", u"",
42     u"0", u"1", u"2", u"3", u"4", u"",
43 )
44 cm.set_alphabet(alphabet)
45 key = "russian"
46 cm.set_key(key)
47 plaintext = u", ! ? ."
48 encdec(cm, plaintext)
49
50 '''
51 Attack at 12.00am!
52 DGDD DAGD DGAF ADDF DADV DVFA ADVX
53 ATTACKAT1200AM
54 -----
55 The quick brown fox jumps over the lazy dog.
56 DXGF VDVD VFAA GGDY AFXG XGFA GFFA DDVG DDXA FAXG ADDF XXXD AXDX VVDD DGVV FXFA VVFX
57 ↪ XV
58 THEQUICKBROWNFOXJUMPSOVERTHELAZYDOG
59 -----
60 , ! ? .
61 AAXF FGXG GDDF FVDA FDDG GGVF DGXV VAAV VFXA XDDA DGAV AGDF AXFV VFDX GFVD XFVV FG

```

(continues on next page)

(continued from previous page)

```

62 -----
63 '''

```

## 2.3 Affine

**class** `secretpy.Affine`

The Affine Cipher

**decrypt** (*text*, *key*, *alphabet*=`'abcdefghijklmnopqrstuvwxyz'`)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*tuple of 2 integers*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** `text`

**Return type** `string`

**encrypt** (*text*, *key*, *alphabet*=`'abcdefghijklmnopqrstuvwxyz'`)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*tuple of 2 integers*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** `text`

**Return type** `string`

### 2.3.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Affine, CryptMachine, alphabets as al
5  from secretpy.cmdecorators import UpperCase, Block, SaveAll
6
7
8  alphabet = al.ENGLISH
9  plaintext = u"thequickbrownfoxjumpsoverthelazydog"
10 key = (7, 8)
11
12 cipher = Affine()
13 print(plaintext)
14
15 enc = cipher.encrypt(plaintext, key, alphabet)

```

(continues on next page)

(continued from previous page)

```

16 print(enc)
17 dec = cipher.decrypt(enc, key, alphabet)
18 print(dec)
19
20
21 def encdec(machine, plaintext):
22     print("=" * 80)
23     print(plaintext)
24     enc = machine.encrypt(plaintext)
25     print(enc)
26     print(machine.decrypt(enc))
27
28
29 cm0 = CryptMachine(cipher, key)
30
31 cm = cm0
32 cm.set_alphabet(al.ENGLISH)
33 cm.set_key(key)
34 plaintext = "I don't love non-alphabet characters. I will remove all of them: ^,&@$~
35 ↳(*;?&#. Great!"
36 encdec(cm, plaintext)
37
38 cm = Block(cm, length=4, sep="/=/")
39 cm.set_alphabet(al.SWEDISH + al.DECIMAL)
40 plaintext = "This text is divided by blocks of length 4!"
41 encdec(cm, plaintext)
42
43 cm = SaveAll(cm0)
44 plaintext = "I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!"
45 encdec(cm, plaintext)
46
47 cm.set_alphabet(al.ENGLISH_SQUARE_IJ)
48 plaintext = "Jj becomes Ii because we use ENGLISH_SQUARE_IJ!"
49 encdec(cm, plaintext)
50
51 cm.set_alphabet(al.JAPANESE_HIRAGANA)
52 plaintext = u"text !"
53 encdec(cm, plaintext)
54
55 cm = UpperCase(cm)
56 alphabet = al.GREEK
57 cm.set_alphabet(alphabet)
58 plaintext = u"Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)"
59 encdec(cm, plaintext)
60
61 '''
62 thequickbrownfoxjumpsoverthelazydog
63 lfkqsmwapxcgvrntsojeczkxlfkhibudcy
64 thequickbrownfoxjumpsoverthelazydog
65 =====
66 I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
67 mdcvlnhczkvcihjfkplwfixiwlkxemgmhxxkoczkihhcrlfkoyxkil
68 idontlovenonalphabetcharactersiwillremoveallofthemgreat
69 =====
70 This text is divided by blocks of length 4!
71 yszr/=y7ny/=/zr0z/=/9z07/=/0pup/=/höwa/=/röeh/=/7vly/=/sf
72 thistextisdividedbyblocksoflength4

```

(continues on next page)

(continued from previous page)

```

72 =====
73 I love non-alphabet characters. These are : ^,&@$(*;?&#. That's it!
74 Z hÖ97 vÖv-ih6sip7y wsikiwy7kr. Ys7r7 ik7 : ^,&@$(*;?&#. Ysiy'r zy!
75 I love non-alphabet characters. These are : ^,&@$(*;?&#. That's it!
76 =====
77 Jj becomes Ii because we use ENGLISH_SQUARE_IJ!
78 Pp qmxzlmc Pp qmxircm fm rcm MSADPCH_CORIVM_PP!
79 Ii becomes Ii because we use ENGLISH_SQUARE_II!
80 =====
81 text      !
82 text      !
83 text      !
84 =====
85 Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)
86 ΛΟΘΙΡ ΗΦΙΝΖ ΓΗΡ ΝΚΘΞΑ Α ΙΘΙΣΛΙΦΧΗ. (ΗΤΔΦΟΗ ΓΜΘΣ)
87 ΘΛΕΙ ΑΡΕΤ ΚΑΙ ΤΛΜΗ Η ΕΛΕΒΘΕΡΑ. (ΑΝΔΡΑΣ ΚΛΒΟΣ)
88 '''

```

## 2.4 Atbash

**class** secretpy.**Atbash**

The Atbash Cipher

**decrypt** (*text*, *key=None*, *alphabet='abcdefghijklmnopqrstuvwxyz'*)

Decryption method

### Parameters

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – is not used
- **alphabet** (*string or tuple of strings*) – Alphabet which will be used, if there is no a value, English is used

**Returns** *text*

**Return type** *string*

**encrypt** (*text*, *key=None*, *alphabet='abcdefghijklmnopqrstuvwxyz'*)

Encryption method

### Parameters

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – is not used
- **alphabet** (*string or tuple of strings*) – Alphabet which will be used, if there is no a value, English is used

**Returns** *text*

**Return type** *string*

## 2.4.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Atbash, CryptMachine, alphabets
5  from secretpy.cmdecorators import UpperCase, SaveAll
6
7
8  def encdec(machine, plaintext):
9      print("-" * 80)
10     print(plaintext)
11     enc = machine.encrypt(plaintext)
12     print(enc)
13     dec = machine.decrypt(enc)
14     print(dec)
15
16
17  cm = UpperCase(CryptMachine(Atbash()))
18
19  plaintext = u"attackatdawn"
20  encdec(cm, plaintext)
21
22  plaintext = u""
23  cm.set_alphabet(alphabets.HEBREW)
24  encdec(cm, plaintext)
25
26  cm = SaveAll(cm)
27  plaintext = u"The quick brown fox jumps over the lazy dog."
28  cm.set_alphabet(alphabets.GERMAN)
29  encdec(cm, plaintext)
30
31  plaintext = u"Achtung Minen!!!"
32  encdec(cm, plaintext)
33
34  cm.set_alphabet(alphabets.ARABIC)
35  plaintext = u" : "
36  encdec(cm, plaintext)
37
38  '''
39  -----
40  attackatdawn
41  ZGGZXPZGWZDM
42  ATTACKATDAWN
43  -----
44
45
46
47  -----
48  The quick brown fox jumps over the lazy dog.
49  KWZ NJVÖT ÜMPHQ YPG UJROL PIZM KWZ SBEF ÄPX.
50  THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG.
51  -----
52  Achtung Minen!!!
53  BÖWKJQX RVQZQ!!!
54  ACHTUNG MINEN!!!
55  -----

```

(continues on next page)

(continued from previous page)

```

56 :
57 :
58 :
59 '''

```

## 2.5 Autokey

**class** `secretpy.Autokey`

The Autokey Cipher

**decrypt** (*text*, *key*, *alphabet*=`'abcdefghijklmnopqrstuvwxyz'`)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*string*) – Decryption key
- **alphabet** (*string or tuple of strings*) – Alphabet which will be used, if there is no a value, English is used

**Returns** `text`

**Return type** `string`

**encrypt** (*text*, *key*, *alphabet*=`'abcdefghijklmnopqrstuvwxyz'`)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*string*) – Encryption key
- **alphabet** (*string or tuple of strings*) – Alphabet which will be used, if there is no a value, English is used

**Returns** `text`

**Return type** `string`

### 2.5.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Autokey, CryptMachine, alphabets as al
5  from secretpy.cmdecorators import UpperCase, Block, SaveAll
6
7
8  alphabet = al.GERMAN
9  plaintext = u"thequickbrownfoxjumpsoverthelazydog"
10 key = "queenly"
11
12 cipher = Autokey()
13 print(plaintext)

```

(continues on next page)

(continued from previous page)

```

14
15 enc = cipher.encrypt(plaintext, key, alphabet)
16 print(enc)
17 dec = cipher.decrypt(enc, key, alphabet)
18 print(dec)
19
20
21 def encdec(machine, plaintext):
22     print("-----")
23     print(plaintext)
24     enc = machine.encrypt(plaintext)
25     print(enc)
26     print(machine.decrypt(enc))
27
28
29 cm0 = CryptMachine(cipher, key)
30
31 cm = cm0
32 cm.set_alphabet(al.ENGLISH + al.DECIMAL)
33 cm.set_key("keys")
34 plaintext = "I don't love non-alphabet characters. I will remove all of them: ^,&@$~
35 ↳(*;?&#. Great!"
36 encdec(cm, plaintext)
37
38 cm = Block(cm, length=5, sep="-")
39 plaintext = "This text is divided by blocks of length 5!"
40 encdec(cm, plaintext)
41
42 cm = SaveAll(cm0)
43 plaintext = "I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!"
44 encdec(cm, plaintext)
45
46 cm.set_alphabet(al.ENGLISH_SQUARE_IJ)
47 plaintext = "Jj becomes Ii because we use ENGLISH_SQUARE_IJ!"
48 encdec(cm, plaintext)
49
50 cm.set_alphabet(al.JAPANESE_HIRAGANA)
51 cm.set_key(u"")
52 plaintext = u"text !"
53 encdec(cm, plaintext)
54
55 cm = UpperCase(cm)
56 alphabet = al.GREEK
57 cm.set_alphabet(alphabet)
58 cm.set_key(u"κλειδ")
59 plaintext = u"Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)"
60 encdec(cm, plaintext)
61
62 '''
63 thequickbrownfoxjumpsoverthelazydog
64 föiudtäßivamvhyyäeeüxüonhbwzvb1wvk
65 thequickbrownfoxjumpsoverthelazydog
66 -----
67 I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
68 shc5lo28xy28ey3uamt0cieacjtvru10z3tdmxzcmz6sf4ssrzyimz
69 idontlovenonalphabetcharactersiwillremoveallofthemgreat
70 -----

```

(continues on next page)

(continued from previous page)

```

70 This text is divided by blocks of length 5!
71 3l6ac-15blw-0130g-myjlf-op0l3-2hvw1-14li
72 thistextisdividedbyblocksoflength5
73 -----
74 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
75 S pcdm y28-ey3uamt0 cieacjtvru. Clvax hvw : ^,&@$~(*;?&#. Xhrx'b pt!
76 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
77 -----
78 Jj becomes Ii because we use ENGLISH_SQUARE_IJ!
79 Sn zwlwniu Wu fwlvwg wy mwa IGYPNEO_CYMHIU_CI!
80 Ii becomes Ii because we use ENGLISH_SQUARE_II!
81 -----
82 text          !
83 text          !
84 text          !
85 -----
86 Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)
87 ΣΠΝΜ ΙΛ ΤΑΑ ΨΙΣΧΗ ΨΗΟΚΕ. (ΥΘΟΑΣ ΨΕΓΗΟΛ)
88 ΛΕΙ ΑΣΕΤ ΚΑΘ ΤΚΜΗ ΕΛΥΘΔΡ. (ΑΝΕΡΣ ΚΑΛΒΕΤ)
89 '''

```

## 2.6 Bazeries

**class** secretpy.Bazeries

The Bazeries Cipher

**decrypt** (*text*, *key*, *alphabet*=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'))

Decryption method

### Parameters

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key*, *alphabet*=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'))

Encryption method

### Parameters

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

## 2.6.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Bazeries, CryptMachine, alphabets as al
5  from secretpy.cmdecorators import UpperCase, SaveAll
6
7
8  def encdec(machine, plaintext):
9      print("=" * 80)
10     print(plaintext)
11     enc = machine.encrypt(plaintext)
12     print(enc)
13     dec = machine.decrypt(enc)
14     print(dec)
15
16
17 cm = UpperCase(CryptMachine(Bazeries()))
18 cm.set_alphabet(al.ENGLISH_SQUARE_IJ)
19
20 key = (81257, u"eightyonethousandtwohundredfiftyseven")
21 cm.set_key(key)
22 plaintext = u"Whoever has made a voyage up the Hudson" \
23            u" must remember the Kaatskill mountains"
24 encdec(cm, plaintext)
25
26 cm = SaveAll(cm)
27 encdec(cm, plaintext)
28
29 cm = UpperCase(SaveAll(CryptMachine(Bazeries()), key))
30 cm.set_alphabet(al.ENGLISH_SQUARE_IJ)
31 plaintext = u"The quick brown fox jumps over the lazy dog!"
32 encdec(cm, plaintext)
33
34 cm.set_alphabet(al.GREEK_SQUARE)
35 cm.set_key((2358, u"κλειδ"))
36 plaintext = u"Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)"
37 encdec(cm, plaintext)
38
39 '''
40 =====
41 Whoever has made a voyage up the Hudson must remember the Kaatskill mountains
42 DUMTMCDSENRTMEVQXMOELCCRVXDMDKWXNNMUKRDKUMYNMBPRKEEPMGNGEKWXCWB
43 WHOEVERHASMADEAVOYAGEUPTHEHUDSONMUSTREMEMBERTHEKAATSKILLMOUNTAINS
44 =====
45 Whoever has made a voyage up the Hudson must remember the Kaatskill mountains
46 DUMTMCD SEN RTE M V EQXMOE LC CRV XDMDKW XNNM UKRDKUMY NMB PRKEEPMGN GEKWXCWB
47 WHOEVER HAS MADE A VOYAGE UP THE HUDSON MUST REMEMBER THE KAATSKILL MOUNTAINS
48 =====
49 The quick brown fox jumps over the lazy dog!
50 PAB XHMDK YCUFC IWS TCRQN XBZE GMD KUML CVO!
51 THE QUICK BROWN FOX IUMPS OVER THE LAZY DOG!
52 =====
53 Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)
54 ΥΜΡΥΕ ΒΨΥΖΚ ΒΓΕ ΧΨΡΚΦ Υ ΒΔΥΕΚΡΖΥΜ. (ΦΣΚΥΖΠΝ ΚΕΚΣΧΑ)
55 ΘΕΛΕΙ ΑΡΕΤΗ ΚΑΙ ΤΟΛΜΗ Η ΕΛΕΥΘΕΡΙΑ. (ΑΝΔΡΕΑΞ ΚΑΛΒΟΞ)
56 '''

```

## 2.7 Beaufort

**class** secretpy.**Beaufort**

The Beaufort Cipher

**decrypt** (*text*, *key*, *alphabet*=*'abcdefghijklmnopqrstuvwxyz'*)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*string*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key*, *alphabet*=*'abcdefghijklmnopqrstuvwxyz'*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*string*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

### 2.7.1 Examples

```
1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Beaufort, CryptMachine, alphabets as al
5  from secretpy.cmdecorators import SaveAll
6
7
8  def encdec(cipher, plaintext, key, alphabet=al.ENGLISH):
9      print(
10         ↪ '=====
11         ↪ ')
12         print(plaintext)
13         enc = cipher.encrypt(plaintext, key, alphabet)
14         print(enc)
15         print(cipher.decrypt(enc, key, alphabet))
16
17 key = "key"
18 cipher = Beaufort()
19 plaintext = u"thequickbrownfoxjumpsoverthelazydog"
```

(continues on next page)

(continued from previous page)

```

20 encdec(cipher, plaintext, key)
21
22 alphabet = al.GERMAN
23 plaintext = u"schweißgequältvomödentextzürnttypografjakob"
24 encdec(cipher, plaintext, key, alphabet)
25
26 alphabet = al.SWEDISH
27 plaintext = u"faqomschweizklövdutrångpjäxby"
28 encdec(cipher, plaintext, key, alphabet)
29
30 # using cryptmachine
31
32
33 def encdec(machine, plaintext):
34     print("-----")
35     print(plaintext)
36     enc = machine.encrypt(plaintext)
37     print(enc)
38     print(machine.decrypt(enc))
39
40
41 cm0 = CryptMachine(cipher, key)
42
43 plaintext = "I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!"
44 cm = SaveAll(cm0)
45 encdec(cm, plaintext)
46
47 cm.set_alphabet(al.ENGLISH_SQUARE_IJ)
48 plaintext = "Jj becomes Ii because we use ENGLISH_SQUARE_IJ!"
49 encdec(cm, plaintext)
50
51 cm.set_alphabet(al.JAPANESE_HIRAGANA)
52 cm.set_key(u"")
53 plaintext = u"text  "
54 encdec(cm, plaintext)
55
56 plaintext = "I don't love non-alphabet characters. I will remove all of them: ^,&@$~
57 ↳(*;?&#. Great!"
58 cm = cm0
59 cm.set_alphabet(al.ENGLISH)
60 cm.set_key(key)
61 encdec(cm, plaintext)
62
63 '''
64 Output:
65
66 =====
67 thequickbrownfoxjumpsoverthelazydog
68 rxuukqiuxtqcxzknveypgwjutlrgtylgvwy
69 thequickbrownfoxjumpsoverthelazydog
70 =====
71 schweißgequältvomödentextzürnttypografjakob
72 wcrsaqlüuyoußpdäwöhalvabvjäxvfvkjäühkßpkykj
73 schweißgequältvomödentextzürnttypografjakob
74 =====
75 faqomschweizklövdutrångpjäxby
76 feizvgiäcgzöawzsbeuqäääjbgbjj

```

(continues on next page)

(continued from previous page)

```

76 faqomschweizklövdutrångpjäxby
77 -----
78 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
79 C tkpa lwr-yzprkdur crknyilutm. Fdagg ehg : ^,&@$~(*;?&#. Lrkl'g cl!
80 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
81 -----
82 Jj becomes Ii because we use ENGLISH_SQUARE_IJ!
83 Bw xfcliyag Bw xfcyqnu oa esa UXYOBNR_SPEKOU_BW!
84 Ii becomes Ii because we use ENGLISH_SQUARE_II!
85 -----
86 text
87 text
88 text
89 -----
90 I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
91 cbkxlnwjuxqlktjdexglwdehkcfcgngciqzthgskpayztkflrgsstayr
92 idontlovenonalphabetcharactersiwillremoveallofthemgreat
93 '''

```

## 2.8 Bifid

**class** secretpy.**Bifid**

The Bifid Cipher

**decrypt** (*text*, *key=None*, *alphabet=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z')*)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string or tuple or list*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key=None*, *alphabet=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z')*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string or tuple or list*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

## 2.8.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Bifid, CryptMachine, alphabets as al
5  from secretpy.cmdecorators import UpperCase, SaveAll, Block
6
7
8  def encdec(machine, plaintext):
9      print(plaintext)
10     enc = machine.encrypt(plaintext)
11     print(enc)
12     print(machine.decrypt(enc))
13     print("-----")
14
15
16  key = 5
17  cm = CryptMachine(Bifid(), key)
18  alphabet = [
19      u"", u"", u"", u"", u"", u"",
20      u"", u"", u"", u"", u"", u"",
21      u"", u"", u"", u"", u"", u"",
22      u"", u"", u"", u"", u"", u"",
23      u"", u"", u"", u"", u"", u"",
24      u"1", u"2", u"3", u"4", u"5", u"6"
25  ]
26  cm.set_alphabet(alphabet)
27  plaintext = u" , !!!"
28  encdec(cm, plaintext)
29
30  cm1 = Block(cm)
31
32  alphabet = [
33      u"p", u"h", u"q", u"g", u"m",
34      u"e", u"a", u"y", u"l", u"n",
35      u"o", u"f", u"d", u"x", u"k",
36      u"r", u"c", u"v", u"s", u"z",
37      u"w", u"b", u"u", u"t", u"ij"
38  ]
39  cm1.set_alphabet(alphabet)
40  plaintext = u"Defend the East Wall of the Castle!"
41  encdec(cm1, plaintext)
42
43  cm1 = UpperCase(cm1)
44  alphabet = [
45      u"b", u"g", u"w", u"k", u"z",
46      u"q", u"p", u"n", u"d", u"s",
47      u"ij", u"o", u"a", u"x", u"e",
48      u"f", u"c", u"l", u"u", u"m",
49      u"t", u"h", u"y", u"v", u"r"
50  ]
51  cm1.set_alphabet(alphabet)
52  cm1.set_key(10)
53  plaintext = "flee at once"
54  encdec(cm1, plaintext)
55

```

(continues on next page)

(continued from previous page)

```

56 cm = SaveAll(cm)
57
58 alphabet = al.GREEK_SQUARE
59 cm.set_alphabet(alphabet)
60 plaintext = u"Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)"
61 encdec(cm, plaintext)
62
63 '''
64 , !!!
65 4
66
67 -----
68 Defend the East Wall of the Castle!
69 ffyhk khycp liash adtrl hcchl blr
70 defendtheeastwallofthecastle
71 -----
72 flee at once
73 UAEOL WRINS
74 FLEEATONCE
75 -----
76 Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)
77 Ζλζπρ οεπκφ ζιν μζυτυ η κλφδζγεγφ. (Πγδαργγ Ρυτακ)
78 Θελει αρετη και τολμη η ελευθερια. (Ανδρεα Καλβο)
79 -----
80 '''

```

## 2.9 Caesar

**class** secretpy.Caesar

The Caesar Cipher

**decrypt** (*text*, *key*=3, *alphabet*='abcdefghijklmnopqrstuvwxyz')

Decryption method

### Parameters

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** decrypted text

**Return type** string

**encrypt** (*text*, *key*=3, *alphabet*='abcdefghijklmnopqrstuvwxyz')

Encryption method

### Parameters

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** encrypted text

**Return type** string

## 2.9.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Caesar, CryptMachine, alphabets as al
5  from secretpy.cmdecorators import SaveAll, Block
6
7
8  def encdec(cipher, plaintext, key, alphabet=al.ENGLISH):
9      print(
10         ↪ '=====
11         ↪ ')
12         print(plaintext)
13         enc = cipher.encrypt(plaintext, key, alphabet)
14         print(enc)
15         print(cipher.decrypt(enc, key, alphabet))
16
17 key = 3
18 cipher = Caesar()
19
20 alphabet = al.SWEDISH
21 plaintext = u"faqomschweizklövdutrångpjäxby"
22 encdec(cipher, plaintext, key, alphabet)
23
24 # using cryptmachine
25
26 def encdec(machine, plaintext):
27     print("-----")
28     print(plaintext)
29     enc = machine.encrypt(plaintext)
30     print(enc)
31     print(machine.decrypt(enc))
32
33
34 cm0 = CryptMachine(cipher, key)
35
36 cm = cm0
37 cm.set_alphabet(al.ENGLISH)
38 plaintext = "I don't love non-alphabet characters. I will remove all of them: ^,&@$~
39 ↪ (*;?&#. Great!"
40 encdec(cm, plaintext)
41
42 cm = Block(cm, length=5, sep="-")
43 plaintext = "This text is divided by blocks of length 5!"
44 encdec(cm, plaintext)
45
46 cm = SaveAll(cm0)
47 plaintext = "I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!"
48 encdec(cm, plaintext)

```

(continues on next page)

(continued from previous page)

```

49 cm.set_alphabet(al.ENGLISH_SQUARE_IJ)
50 plaintext = "Jj becomes Ii because we use ENGLISH_SQUARE_IJ!"
51 encdec(cm, plaintext)
52
53 cm.set_alphabet(al.JAPANESE_HIRAGANA)
54 cm.set_key(1)
55 plaintext = u"text "
56 encdec(cm, plaintext)
57
58 alphabet = u"abcdeABCDEFghijFGHIJ"
59 cm.set_alphabet(alphabet)
60 cm.set_key(3)
61 plaintext = u"Text aBcdeHijf"
62 encdec(cm, plaintext)
63
64 '''
65 Output:
66
67 =====
68 faqomschweizklövdutrångpjäxby
69 idtrpvfkzhlönocygxwuaqjsmbåä
70 faqomschweizklövdutrångpjäxby
71 =====
72 I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
73 lgrqworyhqrqdoskdehwfkdudfwhuvlzloouhpryhdooriwxhpjuhdw
74 idontlovenonalphabetcharactersiwillremoveallofthemgreat
75 =====
76 This text is divided by blocks of length 5!
77 wklvw-hawlv-glylg-hgebe-orfnv-riohq-jwk
78 thistextisdividedbyblocksoflength
79 =====
80 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
81 L oryh qrq-doskdehw fkdudfwhuv. Wkhvh duh : ^,&@$~(*;?&#. Wkdw'v lw!
82 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
83 =====
84 Jj becomes Ii because we use ENGLISH_SQUARE_IJ!
85 Mm ehfrphv Mm ehfdxvh zh xvh HQKOMVL_VTXDUH_MM!
86 Ii becomes Ii because we use ENGLISH_SQUARE_II!
87 =====
88 text
89 text
90 text
91 =====
92 Text aBcdeHijf
93 TCxt dEABCaGHi
94 Text aBcdeHijf
95 '''

```

## 2.10 Caesar Progressive

**class** secretpy.CaesarProgressive

The Caesar Progressive Cipher

**decrypt** (text, key=3, alphabet='abcdefghijklmnopqrstuvwxyz')

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** decrypted text**Return type** string**encrypt** (*text*, *key*=3, *alphabet*='abcdefghijklmnopqrstuvwxyz')

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** encrypted text**Return type** string

## 2.10.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import CaesarProgressive, CryptMachine, alphabets as al
5  from secretpy.cmdecorators import SaveAll
6
7
8  def encdec(cipher, plaintext, key, alphabet=al.ENGLISH):
9      print(
10         ↪ '=====
11         ↪ ')
12      print(plaintext)
13      enc = cipher.encrypt(plaintext, key, alphabet)
14      print(enc)
15      print(cipher.decrypt(enc, key, alphabet))
16
17 plaintext = u"thequickbrownfoxjumpsoverthelazydog"
18 key = 3
19 cipher = CaesarProgressive()
20 encdec(cipher, plaintext, key)
21
22 alphabet = al.GERMAN
23 plaintext = u"scheißgequältvomödentextzürnttypografjakob"
24 encdec(cipher, plaintext, key, alphabet)
25
26 alphabet = al.SWEDISH
27 plaintext = u"faqomschweizklövdutrångpjäxby"

```

(continues on next page)

(continued from previous page)

```

28 encdec(cipher, plaintext, key, alphabet)
29
30 # using cryptmachine
31
32
33 def encdec(machine, plaintext):
34     print("-----")
35     print(plaintext)
36     enc = machine.encrypt(plaintext)
37     print(enc)
38     print(machine.decrypt(enc))
39
40
41 cm0 = CryptMachine(cipher, key)
42
43 plaintext = "I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!"
44 cm = SaveAll(cm0)
45 encdec(cm, plaintext)
46
47 cm.set_alphabet(al.ENGLISH_SQUARE_IJ)
48 plaintext = "Jj becomes Ii because we use ENGLISH_SQUARE_IJ!"
49 encdec(cm, plaintext)
50
51 cm.set_alphabet(al.JAPANESE_HIRAGANA)
52 cm.set_key(1)
53 plaintext = u"text "
54 encdec(cm, plaintext)
55
56 plaintext = "I don't love non-alphabet characters. I will remove all of them: ^,&@$~
57 ↳(*;?&#. Great!"
58 cm = cm0
59 cm.set_alphabet(al.ENGLISH)
60 encdec(cm, plaintext)
61
62 '''
63 Output:
64
65 =====
66 thequickbrownfoxjumpsoverthelazydog
67 wljwbqlumdbkcvfpcohlpmesvkiqqgggmyr
68 thequickbrownfoxjumpsoverthelazydog
69 =====
70 schweißgequältvomödentextzürnttypografjakob
71 vgmülqiqpüdkäfibryägnßtqxörovwüunzjpumxüq
72 schweißgequältvomödentextzürnttypografjakob
73 =====
74 faqomschweizklövdutrångpjäxby
75 ievutålreqvkzäqkwlkuicmhåxcå
76 faqomschweizklövdutrångpjäxby
77 =====
78 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
79 L ptbl vxx-lxcvprvl vbvnxasesu. Wljyl iao : ^,&@$~(*;?&#. Etnh'h yk!
80 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
81 =====
82 Jj becomes Ii because we use ENGLISH_SQUARE_IJ!
83 Mn glkwvpd Vw qutsnmz sb sre FPKPOYP_AZEMDS_XY!
84 Ii becomes Ii because we use ENGLISH_SQUARE_II!

```

(continues on next page)

(continued from previous page)

```

84 -----
85 text
86 text
87 text
88 -----
89 I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
90 jffrryrvdnxzznzexrtxnxdpxcuguwncptubpybjtqcdhzodbkfrfcw
91 idontlovenonalphabetcharactersiwillremoveallofthemgreat
92 '''

```

## 2.11 Chao

**class** secretpy.Chao

The Chaocipher

**decrypt** (*text*, *key*, *alphabet=None*)

Decryption method

### Parameters

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** decrypted text

**Return type** string

**encrypt** (*text*, *key*, *alphabet=None*)

Encryption method

### Parameters

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** encrypted text

**Return type** string

### 2.11.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Chao, CryptMachine, alphabets
5  from secretpy.cmdecorators import UpperCase, SaveAll
6
7
8  def encdec(machine, plaintext):

```

(continues on next page)

(continued from previous page)

```

9     print(plaintext)
10    enc = machine.encrypt(plaintext)
11    print(enc)
12    print(machine.decrypt(enc))
13    print("-----")
14
15
16    alphabet = "ptlnbqdeoyfsfavzkgjrihwxumc" # RIGHT WHEEL PT
17    key = "hxuczvamdslkpefjrigtwobnyq"      # LEFT WHEEL CT
18
19    cm = UpperCase(SaveAll(CryptMachine(Chao(), key)))
20    cm.set_alphabet(alphabet)
21
22    plaintext = "well done is better than well said"
23    encdec(cm, plaintext)
24
25    plaintext = "plaintext"
26    encdec(cm, plaintext)
27
28    cm.set_alphabet(alphabets.ENGLISH)
29    cm.set_key(alphabets.ENGLISH)
30    plaintext = "do not use pc"
31    encdec(cm, plaintext)
32
33    '''
34    Output:
35
36    well done is better than well said
37    OAHQ HCNV NX TSZJRR HJBY HQKS OUJY
38    WELL DONE IS BETTER THAN WELL SAID
39    -----
40    plaintext
41    HULROKQUA
42    PLAINTEXT
43    -----
44    do not use pc
45    DN LLQ QYM MW
46    DO NOT USE PC
47    -----
48    '''

```

## 2.12 Columnar Transposition

**class** secretpy.ColumnarTransposition

The Columnar Transposition Cipher

**decrypt** (text, key, alphabet='abcdefghijklmnopqrstuvwxyz')

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*string*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is

used

**Returns** text

**Return type** string

**encrypt** (*text*, *key*, *alphabet*=*'abcdefghijklmnopqrstuvwxyz'*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*string*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

### 2.12.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import ColumnarTransposition, CryptMachine, alphabets as al
5  from secretpy.cmdecorators import SaveAll, Block
6
7
8  cipher = ColumnarTransposition()
9  alphabet = al.GERMAN
10 plaintext = u"schweißgequält vom ödentext zur nttypograf jakob"
11 key = u"schlüssel"
12
13 print(plaintext)
14 enc = cipher.encrypt(plaintext, key, alphabet)
15 print(enc)
16 dec = cipher.decrypt(enc, key, alphabet)
17 print(dec)
18
19
20 def encdec(machine, plaintext):
21     print("=" * 80)
22     print(plaintext)
23     enc = machine.encrypt(plaintext)
24     print(enc)
25     print(machine.decrypt(enc))
26
27
28 key = "manykeys"
29 cm0 = CryptMachine(cipher, key)
30
31 cm = cm0
32 cm.set_alphabet(al.ENGLISH)
33 plaintext = "I don't love non-alphabet characters and uppercase. I will remove all of
↳ them: ^, & @ $ % ~ ( * ; ? & # . "
34 encdec(cm, plaintext)

```

(continues on next page)

(continued from previous page)

```

35 cm = Block(cm, length=5, sep=" ")
36 plaintext = "This text is divided by blocks of length 5!"
37 encdec(cm, plaintext)
38
39
40 cm = SaveAll(cm0)
41 plaintext = "I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!"
42 encdec(cm, plaintext)
43
44 cm.set_alphabet(al.JAPANESE_HIRAGANA)
45 cm.set_key(u"")
46 plaintext = u"text "
47 encdec(cm, plaintext)
48
49 cm.set_alphabet(al.ENGLISH_SQUARE_IJ)
50 cm.set_key(u"keyj")
51 plaintext = "ENGLISH_SQUARE_IJ doesn't change Ii to Jj because it's a transposition_
↳ cipher!"
52 encdec(cm, plaintext)
53
54 '''
55 schweißgequält vom ödentext zur nttypograf jakob
56 cuenfgmzghänt jwl ttae öürsqdraivxpoß otobeteyk
57 schweißgequält vom ödentext zur nttypograf jakob
58 =====
59 I don't love non-alphabet characters and uppercase. I will remove all of them: ^,&@$~
↳ (*;?&#.
60 dnbcuemfllhsrlamtacreieeieaadseoooetpiotvhrnarlnntepwvhopaac11
61 idontlovenonalphabetcharactersanduppercaseiwillremoveallofthem
62 =====
63 This text is divided by blocks of length 5!
64 hsboe iontv letid shidy ftekt siblx dcg
65 thistextisdividedbyblocksoflength
66 =====
67 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
68 L lhse nbc-steaaeai acrrtopatt. Nteai vhr : ^,&@$~(*;?&#. Hhoe't es!
69 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
70 =====
71 text
72 text
73 text
74 =====
75 ENGLISH_SQUARE_IJ doesn't change Ii to Jj because it's a transposition cipher!
76 NSUIECG_TBUTRP_TC elsr d'n aijcea Ns op Ei qeotnij ai't s inhghajsheoes saoiir!
77 ENGLISH_SQUARE_IJ doesn't change Ii to Jj because it's a transposition cipher!
78 '''

```

## 2.13 Four Square

**class** secretpy.FourSquare

The Four-Square Cipher

**decrypt** (text, key=None, alphabet=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'))

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*tuple of two strings*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text**Return type** string

**encrypt** (*text*, *key=None*, *alphabet=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z')*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*tuple of two strings*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text**Return type** string

## 2.13.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import FourSquare, CryptMachine, alphabets
5  from secretpy.cmdecorators import UpperCase
6
7
8  def encdec(machine, plaintext):
9      print(plaintext)
10     enc = machine.encrypt(plaintext)
11     print(enc)
12     dec = machine.decrypt(enc)
13     print(dec)
14     print("-----")
15
16
17 alphabet = alphabets.ENGLISH_SQUARE_OQ
18
19 key = (u"example", u"keyword")
20
21 cm = UpperCase(CryptMachine(FourSquare()))
22
23 cm.set_alphabet(alphabet)
24 cm.set_key(key)
25 plaintext = u"Help me Obi wan Kenobi"
26 encdec(cm, plaintext)
27
28 plaintext = u"Help me Obi wan Kenobi a"
```

(continues on next page)

(continued from previous page)

```

29 encdec(cm, plaintext)
30
31 alphabet = alphabets.ENGLISH_SQUARE_IJ
32 cm.set_alphabet(alphabet)
33 key = (u"criptog", u"segurt")
34 cm.set_key(key)
35 plaintext = u"Attack at dawn!"
36 encdec(cm, plaintext)
37
38
39 '''
40 Help me Obi wan Kenobi
41 FYGMKYHOBXMFKKKIMD
42 HELPMEOWIWANKENOBI
43 -----
44 Help me Obi wan Kenobi a
45 FYGMKYHOBXMFKKKIMDPT
46 HELPMEOWIWANKENOBIAZ
47 -----
48 Attack at dawn!
49 PMMUTBPMCUXH
50 ATTACKATDAWN
51 -----
52 '''

```

## 2.14 Gronsfeld

**class** secretpy.Gronsfeld

The Gronsfeld Cipher

**decrypt** (*text*, *key*, *alphabet*=*'abcdefghijklmnopqrstuvwxyz'*)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*tuple of integers*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** *text*

**Return type** *string*

**encrypt** (*text*, *key*, *alphabet*=*'abcdefghijklmnopqrstuvwxyz'*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*tuple of integers*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** *text*

Return type string

## 2.14.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Gronsfeld, CryptMachine, alphabets as al
5  from secretpy.cmdecorators import UpperCase, Block, SaveAll
6
7
8  alphabet = al.GERMAN
9  plaintext = u"schweißgequält vom ödentext zur nttypograf jakob"
10 key = (4, 17, 9)
11
12 cipher = Gronsfeld()
13
14 print(plaintext)
15 enc = cipher.encrypt(plaintext, key, alphabet)
16 print(enc)
17 dec = cipher.decrypt(enc, key, alphabet)
18 print(dec)
19
20 #####
21
22
23 def encdec(machine, plaintext):
24     print("-----")
25     print(plaintext)
26     enc = machine.encrypt(plaintext)
27     print(enc)
28     print(machine.decrypt(enc))
29
30
31 key = (14, 2, 11)
32 cm0 = CryptMachine(cipher, key)
33
34 cm = cm0
35 cm.set_alphabet(al.ENGLISH)
36 plaintext = "I don't love non-alphabet characters. I will remove all of them: ^,&@$~
37 ↳(*;?&#. Great!"
38 encdec(cm, plaintext)
39
40 cm = Block(cm, length=5, sep=" ")
41 cm.set_key((1, 12, 7, 2))
42 plaintext = "This text is divided by blocks of length 5!"
43 encdec(cm, plaintext)
44
45 cm = SaveAll(cm0)
46 plaintext = "I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!"
47 encdec(cm, plaintext)
48
49 cm.set_alphabet(al.ENGLISH_SQUARE_IJ)
50 plaintext = "Jj becomes Ii because we use ENGLISH_SQUARE_IJ!"
51 encdec(cm, plaintext)

```

(continues on next page)

(continued from previous page)

```

52 cm.set_alphabet(al.JAPANESE_HIRAGANA)
53 plaintext = u"text          !"
54 encdec(cm, plaintext)
55
56 cm = UpperCase(cm)
57 alphabet = al.GREEK
58 cm.set_alphabet(alphabet)
59 plaintext = u"Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)"
60 encdec(cm, plaintext)
61
62 '''
63 schweißgequältvomödentextzürnttypografjakob
64 wtqävrdxnufpgasßghvwxvcxmhvaüxlysxäewseöxf
65 schweißgequältvomödentextzürnttypografjakob
66 -----
67 I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
68 wfbzvwxcxbqyonavcmsvnnvccoeestdwytzncsozjglznztvssorfglh
69 idontlovenonalphabetcharactersiwillremoveallofthemgreat
70 -----
71 This text is divided by blocks of length 5!
72 utpuu qevje kkwuk genfd majmt amnfz nvi
73 thistextisdividedbyblocksoflength
74 -----
75 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
76 J xvxf zvp-bxwjbnlv dthtboagse. Ajfel csq : ^,&@$~(*;?&#. Ajbf'z ku!
77 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
78 -----
79 Jj becomes Ii because we use ENGLISH_SQUARE_IJ!
80 Kv igdatgt Vq dfphwtr dg vem GOTSITU_ZSVNYG_KV!
81 Ii becomes Ii because we use ENGLISH_SQUARE_II!
82 -----
83 text          !
84 text          !
85 text          !
86 -----
87 Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)
88 IOPZ XZΥΡ ΠΒ ΔΝΝΠ Μ ΖΜΞΑΒ. (ΧΙΣΖΩ ΜΒΠΤ)
89 ΘΛΕΙ ΑΡΕΤ ΚΑΙ ΤΛΜΗ Η ΕΛΕΥΘΕΡΑ. (ΑΝΔΡΑΣ ΚΛΒΟΣ)
90 '''

```

## 2.15 Keyword

**class** secretpy.**Keyword**

The Keyword Cipher

**decrypt** (*text*, *key*, *alphabet*=*'abcdefghijklmnopqrstuvwxyz'*)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key*, *alphabet*='abcdefghijklmnopqrstuvwxyz')

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

## 2.15.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3  from secretpy import Keyword, CryptMachine, alphabets as al
4  from secretpy.cmdecorators import UpperCase, Block, SaveAll
5
6
7  alphabet = al.GERMAN
8  plaintext = u"thequickbrownfoxjumpsoverthelazydog"
9  key = "queenly"
10
11 cipher = Keyword()
12 print(plaintext)
13
14 enc = cipher.encrypt(plaintext, key, alphabet)
15 print(enc)
16 dec = cipher.decrypt(enc, key, alphabet)
17 print(dec)
18
19
20 def encdec(machine, plaintext):
21     print("-----")
22     print(plaintext)
23     enc = machine.encrypt(plaintext)
24     print(enc)
25     print(machine.decrypt(enc))
26
27
28 cm0 = CryptMachine(cipher, key)
29
30 cm = cm0
31 cm.set_alphabet(al.DECIMAL + al.ENGLISH)
32 cm.set_key("manykeys")
33 plaintext = "I don't love non-alphabet characters. I will remove all of them: ^,&@$~
34 ↪(*;?&#. Great!"
35 encdec(cm, plaintext)
36 cm = UpperCase(Block(cm, length=5, sep="-"))

```

(continues on next page)

(continued from previous page)

```

37 plaintext = "This text is divided by blocks of length 5!"
38 encdec(cm, plaintext)
39
40 cm = SaveAll(cm0)
41 plaintext = "I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!"
42 encdec(cm, plaintext)
43
44 cm.set_alphabet(al.ENGLISH_SQUARE_IJ)
45 plaintext = "Jj becomes Ii because we use ENGLISH_SQUARE_IJ!"
46 encdec(cm, plaintext)
47
48 cm.set_alphabet(al.JAPANESE_HIRAGANA)
49 cm.set_key(u"")
50 plaintext = u"text                !"
51 encdec(cm, plaintext)
52
53 cm = UpperCase(cm)
54 alphabet = al.GREEK
55 cm.set_alphabet(alphabet)
56 cm.set_key(u"κλειδ")
57 plaintext = u"Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)"
58 encdec(cm, plaintext)
59
60 '''
61 thequickbrownfoxjumpsoverthelazydog
62 rblmscefuojviyjdshkpktlorblggqzxnja
63 thequickbrownfoxjumpsoverthelazydog
64 -----
65 I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
66 c6jirgju7iji3glb347r5b3p35r7pqcvcggp7hju73ggj8rb7h9p73r
67 idontlovenonalphabetcharactersiwillremoveallofthemgreat
68 -----
69 This text is divided by blocks of length 5!
70 RBCQR-7WRCQ-6CUC6-764X4-GJ5FQ-J8G7I-9RBE
71 THISTEXTISDIVIDEDBYBLOCKSOFLNGTH5
72 -----
73 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
74 C gju7 iji-3glb347r 5b3p35r7pq. Rb7q7 3p7 : ^,&@$~(*;?&#. Rb3r'q cr!
75 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
76 -----
77 Jj becomes Ii because we use ENGLISH_SQUARE_IJ!
78 Cc aknigkq Cc aknmtqk vk tqk KHSFCQB_QOTMPK_CC!
79 Ii becomes Ii because we use ENGLISH_SQUARE_II!
80 -----
81 text                !
82 text                !
83 text                !
84 -----
85 Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)
86 ΑΘΖ ΚΡΤΓ ΚΖ ΤΘΜΒ Β ΘΥΡΗΚ. (ΚΝΔΡΑΚΣ ΛΘΕΟΣ)
87 ΘΛΕΙ ΑΡΕΤ ΚΑΙ ΤΛΜΗ Η ΕΛΕΥΘΕΡΑ. (ΑΝΔΡΑΣ ΚΛΒΟΣ)
88 '''

```

## 2.16 MyszowskiTransposition

**class** secretpy.MyszowskiTransposition

The Myszowski Transposition Cipher

**decrypt** (*text*, *key*, *alphabet*=*'abcdefghijklmnopqrstuvwxyz'*)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key*, *alphabet*=*'abcdefghijklmnopqrstuvwxyz'*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

### 2.16.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import MyszowskiTransposition, CryptMachine, alphabets as al
5  from secretpy.cmdecorators import UpperCase, Block, SaveAll
6
7  alphabet = al.GERMAN
8  plaintext = u"schweißgequält vom ödentext zur nttypograf jakob"
9  key = u"schlüssel"
10
11  cipher = MyszowskiTransposition()
12  print(plaintext)
13
14  enc = cipher.encrypt(plaintext, key, alphabet)
15  print(enc)
16  dec = cipher.decrypt(enc, key, alphabet)
17  print(dec)
18
19
20  def encdec(machine, plaintext):
21      print("-----")

```

(continues on next page)

(continued from previous page)

```

22     print(plaintext)
23     enc = machine.encrypt(plaintext)
24     print(enc)
25     print(machine.decrypt(enc))
26
27
28 cm0 = CryptMachine(cipher, key)
29
30 cm = cm0
31 cm.set_alphabet(al.ENGLISH)
32 cm.set_key("tomatokey")
33 plaintext = "I don't love non-alphabet characters. I will remove all of them: ^,&@$~
34 ↪(*;?&#. Great!"
35 encdec(cm, plaintext)
36
37 cm = Block(cm, length=5, sep="-")
38 plaintext = "This text is divided by blocks of length 5!"
39 encdec(cm, plaintext)
40
41 cm = SaveAll(cm0)
42 plaintext = "I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!"
43 encdec(cm, plaintext)
44
45 cm.set_alphabet(al.JAPANESE_HIRAGANA)
46 cm.set_key(u"")
47 plaintext = u"text          !"
48 encdec(cm, plaintext)
49
50 cm = UpperCase(cm)
51 alphabet = al.GREEK
52 cm.set_alphabet(alphabet)
53 cm.set_key(u"κλειδ")
54 plaintext = u"Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)"
55 encdec(cm, plaintext)
56
57 cm.set_key(u"kryptonyckeln")
58 cm.set_alphabet(al.SWEDISH)
59 plaintext = u"FAQ om Schweiz: Klöv du trång pjäxby?"
60 encdec(cm, plaintext)
61
62 '''
63 schweißgequältvomödentextzürnttypografjakob
64 cuenfgmzghäntjwelötütrasißqvodxtrpoaobeteyk
65 schweißgequältvomödentextzürnttypografjakob
66 -----
67 I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
68 nahivevaclleohallroncsohdloptrrimatgitnleaeweefmtebtroa
69 idontlovenonalphabetcharactersiwillremoveallofthemgreat
70 -----
71 This text is divided by blocks of length 5!
72 svogt-doxes-iilnh-eddbk-ehtts-iyclt-ibf
73 thistextisdividedbyblocksoflength
74 -----
75 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
76 V acei ncs-totreoha sslnpereer. Tielb ath : ^,&@$~(*;?&#. Aata'h th!
77 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
78 -----

```

(continues on next page)

(continued from previous page)

```

78 text      !
79 text      !
80 text      !
81 -----
82 Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)
83 ΙΚΜΥΑ ΣΣΛΤ Λ ΒΕΛΕ Α ΑΟΑΑΗΘΝΚΘ. (ΡΙΗΕΔ ΕΤΕΡΡΛ)
84 ΘΛΕΙ ΑΡΕΤ ΚΑΙ ΤΛΜΗ Η ΕΛΕΥΘΕΡΑ. (ΑΝΔΡΑΣ ΚΛΒΟΣ)
85 -----
86 FAQ om Schweiz: Klöv du trång pjäxby?
87 WNI PF ELGXZJC: KRÄS TO DAÖBM UQHVÅY?
88 FAQ OM SCHWEIZ: KLÖV DU TRÅNG PJÄXBY?
89 '''

```

## 2.17 Nihilist

**class** secretpy.Nihilist

The Nihilist Cipher

**decrypt** (*text*, *key*=None, *alphabet*=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'))

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key*=None, *alphabet*=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'))

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

### 2.17.1 Examples

```

1 #!/usr/bin/python
2 # -*- encoding: utf-8 -*-
3

```

(continues on next page)

(continued from previous page)

```

4  from secretpy import Nihilist, CryptMachine, alphabets as al
5
6
7  def encdec(cipher, plaintext, key, alphabet=al.ENGLISH):
8      print(plaintext)
9      enc = cipher.encrypt(plaintext, key, alphabet)
10     print(enc)
11     print(cipher.decrypt(enc, key, alphabet))
12     print("-----")
13
14
15  key = "english"
16  plaintext = u"thequickbrownfoxjumpsoverthelazydog"
17  cipher = Nihilist()
18  encdec(cipher, plaintext, key)
19
20
21  def encdec(machine, plaintext):
22      print(plaintext)
23      enc = machine.encrypt(plaintext)
24      print(enc)
25      print(machine.decrypt(enc))
26      print("-----")
27
28
29  key = "mykey"
30  cm = CryptMachine(cipher, key)
31
32  alphabet = (
33      u"z", u"e", u"b", u"r", u"a",
34      u"s", u"c", u"d", u"f", u"g",
35      u"h", u"i", u"j", u"k", u"l", u"m",
36      u"n", u"o", u"p", u"q", u"t",
37      u"u", u"v", u"w", u"x", u"y"
38  )
39  cm.set_alphabet(alphabet)
40  plaintext = u"Waltz, bad nymph, for quick jigs vex."
41  encdec(cm, plaintext)
42
43  alphabet = (
44      u"a", u"b", u"c", u"d", u"e", u"f",
45      u"g", u"h", u"i", u"j", u"k", u"l",
46      u"m", u"n", u"o", u"p", u"q", u"r",
47      u"s", u"t", u"u", u"v", u"w", u"x",
48      u"y", u"z", u"0", u"1", u"2", u"3",
49      u"4", u"5", u"6", u"7", u"8", u"9",
50  )
51  cm.set_alphabet(alphabet)
52  key = "freedom"
53  cm.set_key(key)
54  plaintext = u"Meet thursday 2300hr!"
55  encdec(cm, plaintext)
56
57  alphabet = (
58      u" ", u" ", u" ", u" ", u" ", u" ",
59      u" ", u" ", u" ", u" ", u" ", u" ",
60      u" ", u" ", u" ", u" ", u" ", u" ",

```

(continues on next page)

(continued from previous page)

```

61     u"", u"", u"", u"", u"", u"",
62     u"", u"", u"", u"", u"", u"",
63     u"1", u"2", u"3", u"4", u"5", u"6"
64 )
65 cm.set_alphabet(alphabet)
66 key = u""
67 cm.set_key(key)
68 plaintext = u""
69 encdec(cm, plaintext)
70
71 alphabet = al.GREEK
72 cm.set_alphabet(alphabet)
73 plaintext = u"ΠΙΝΑΚΑΣ"
74 encdec(cm, plaintext)
75
76 '''
77 Output:
78
79 thequickbrownfoxjumpsoverthelazydog
80 57 54 36 61 66 64 35 40 44 57 59 68 73 38 48 78 45 69 54 75 63 48 76 36 62 65 63 37_
81 ↪41 43 73 77 37 74 43
82 thequickbrownfoxjumpsoverthelazydog
83 -----
84 Waltz, bad nymph, for quick jigs vex.
85 88 70 67 57 66 48 70 56 53 110 70 98 64 36 97 49 99 84 44 77 68 87 65 37 76 87 67 87
86 waltzbadnymphforquicksjigs vex
87 -----
88 Meet thursday 2300hr!
89 47 51 30 57 56 55 74 52 77 29 26 65 88 87 69 89 37 51
90 meetthursday2300hr
91 -----
92 60 41 79 80 60
93
94 -----
95 ΠΙΝΑΚΑΣ
96 95 78 87 65 79 65 97
97 πινακας
98 -----
99 '''

```

## 2.18 Playfair

**class** secretpy.**Playfair**

The Playfair Cipher

**decrypt** (*text*, *key*=", *alphabet*=(*'a'*, *'b'*, *'c'*, *'d'*, *'e'*, *'f'*, *'g'*, *'h'*, *'ij'*, *'k'*, *'l'*, *'m'*, *'n'*, *'o'*, *'p'*, *'q'*, *'r'*,  
*'s'*, *'t'*, *'u'*, *'v'*, *'w'*, *'x'*, *'y'*, *'z'*))

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key

- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, ENGLISH\_SQUARE\_IJ is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key*=", *alphabet*=(*'a'*, *'b'*, *'c'*, *'d'*, *'e'*, *'f'*, *'g'*, *'h'*, *'ij'*, *'k'*, *'l'*, *'m'*, *'n'*, *'o'*, *'p'*, *'q'*, *'r'*, *'s'*, *'t'*, *'u'*, *'v'*, *'w'*, *'x'*, *'y'*, *'z'*))  
Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, ENGLISH\_SQUARE\_IJ is used

**Returns** text

**Return type** string

## 2.18.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Playfair, CryptMachine
5  from secretpy.cmdecorators import UpperCase
6
7
8  def encdec(machine, plaintext):
9      print(plaintext)
10     enc = machine.encrypt(plaintext)
11     print(enc)
12     dec = machine.decrypt(enc)
13     print(dec)
14     print("-----")
15
16
17 cm = UpperCase(CryptMachine(Playfair()))
18 alphabet = [
19     u"p", u"l", u"a", u"y", u"f",
20     u"i", u"r", u"e", u"x", u"m",
21     u"b", u"c", u"d", u"g", u"h",
22     u"k", u"n", u"o", u"q", u"s",
23     u"t", u"u", u"v", u"w", u"z",
24 ]
25 cm.set_alphabet(alphabet)
26 plaintext = u"Hide the gold in the tree stump"
27 encdec(cm, plaintext)
28
29 plaintext = "sometext"
30 encdec(cm, plaintext)
31
32 plaintext = "this is a secret message"
33 encdec(cm, plaintext)

```

(continues on next page)

(continued from previous page)

```

34 '''
35
36 Hide the gold in the tree stump
37 BMODZBXDNABEKUDMUIXMMOUVIF
38 HIDETHEGOLDINTHETREESTUMP
39 -----
40 sometext
41 KQIXVIIW
42 SOMETEXT
43 -----
44 this is a secret message
45 ZBMKMKFORDEXZIMOOFDX
46 THISISASECRETMESSAGE
47 -----
48 '''

```

## 2.19 Polybius

**class** secretpy.Polybius

The Polybius Cipher

**decrypt** (*text*, *key*=None, *alphabet*=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'))

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key*=None, *alphabet*=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'))

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

## 2.19.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Polybius, CryptMachine, alphabets as al
5  from secretpy.cmdecorators import SaveAll
6
7
8  def encdec(machine, plaintext):
9      print(plaintext)
10     enc = machine.encrypt(plaintext)
11     print(enc)
12     dec = machine.decrypt(enc)
13     print(dec)
14     print("-----")
15
16
17  key = "mykey"
18  cm = CryptMachine(Polybius(), key)
19
20  plaintext = u"Defend the east wall of the castle"
21  encdec(cm, plaintext)
22
23  cm = SaveAll(cm)
24
25  alphabet = [
26      u"p", u"h", u"q", u"g", u"m",
27      u"e", u"a", u"y", u"l", u"n",
28      u"o", u"f", u"d", u"x", u"k",
29      u"r", u"c", u"v", u"s", u"z",
30      u"w", u"b", u"u", u"t", u"ij"
31  ]
32  cm.set_alphabet(alphabet)
33  encdec(cm, plaintext)
34
35  plaintext = "thisisasecretmessage"
36  encdec(cm, plaintext)
37
38  cm.set_alphabet(al.GREEK)
39  cm.set_key(u"πινακας")
40  plaintext = u"Ξεσκεπζω την ψυχοφθρα σα βδελυγμα."
41  encdec(cm, plaintext)
42
43  '''
44  Defend the east wall of the castle
45  22142314332243251414154243461532323423432514211542433214
46  defendtheeastwallofthecastle
47  -----
48  Defend the east wall of the castle
49  341433 143 1345 4211 41 424 4454512425253233542114422444542514
50  defend the east wall of the castle
51  -----
52  thisisasecretmessage
53  5421554455442444144241145411144444242314
54  thisisasecretmessage
55  -----

```

(continues on next page)

(continued from previous page)

```

56  Ξεσκεπζω την ψυχοφθρα σα βδελυγμα.
57  422516152 511 213161513 213 565255435.434444514161446222425365223413514
58  ξεσκεπζω την ψυχοφθρα σα βδελυγμα.
59  -----
60  '''

```

## 2.20 Porta

**class** secretpy.Porta

The Porta Cipher

**decrypt** (*text*, *key*, *alphabet*=*'abcdefghijklmnopqrstuvwxyz'*)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*string*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key*, *alphabet*=*'abcdefghijklmnopqrstuvwxyz'*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*string*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

### 2.20.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3  from secretpy import Porta, CryptMachine, alphabets as al
4  from secretpy.cmdecorators import UpperCase, Block, SaveAll
5
6
7  alphabet = al.GERMAN
8  plaintext = u"schweißgequält vom ödentext zur nttypograf jakob"
9  key = u"schlüssel"
10
11  cipher = Porta()
12  print(plaintext)

```

(continues on next page)

(continued from previous page)

```

13
14 enc = cipher.encrypt(plaintext, key, alphabet)
15 print(enc)
16 dec = cipher.decrypt(enc, key, alphabet)
17 print(dec)
18
19 #####
20
21
22 def encdec(machine, plaintext):
23     print("-----")
24     print(plaintext)
25     enc = machine.encrypt(plaintext)
26     print(enc)
27     print(machine.decrypt(enc))
28
29
30 cm0 = CryptMachine(cipher, key)
31
32 cm = cm0
33 cm.set_alphabet(al.ENGLISH)
34 cm.set_key("keys")
35 plaintext = "I don't love non-alphabet characters. I will remove all of them: ^,&@$~
36 ↳(*;?&#. Great!"
37 encdec(cm, plaintext)
38
39 cm = Block(cm, length=4, sep="::")
40 plaintext = "This text is divided by blocks of length 5!"
41 encdec(cm, plaintext)
42
43 cm = SaveAll(cm0)
44 plaintext = "I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!"
45 encdec(cm, plaintext)
46
47 cm.set_alphabet(al.ENGLISH_SQUARE_IJ)
48 plaintext = "Jj becomes Ii because we use ENGLISH_SQUARE_IJ!"
49 encdec(cm, plaintext)
50
51 cm.set_alphabet(al.JAPANESE_HIRAGANA)
52 cm.set_key(u"")
53 plaintext = u"text !"
54 encdec(cm, plaintext)
55
56 cm = UpperCase(cm)
57 alphabet = al.GREEK
58 cm.set_alphabet(alphabet)
59 cm.set_key(u"κλειδ")
60 plaintext = u"Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο) "
61 encdec(cm, plaintext)
62
63 '''
64 '''

```

## 2.21 Rot13

**class** secretpy.Rot13

The Rot13 Cipher (Half)

**decrypt** (*text*, *key=None*, *alphabet='abcdefghijklmnopqrstuvwxyz'*)

Decryption method

### Parameters

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – not used
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key=None*, *alphabet='abcdefghijklmnopqrstuvwxyz'*)

Encryption method

### Parameters

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – not used
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

### 2.21.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Rot13, CryptMachine, alphabets as al
5  from secretpy.cmdecorators import SaveAll, Block
6
7
8  def encdec(machine, plaintext):
9      print("-----")
10     print(plaintext)
11     enc = machine.encrypt(plaintext)
12     print(enc)
13     dec = machine.decrypt(enc)
14     print(dec)
15
16
17  cm = CryptMachine(Rot13())
18
19  plaintext = u"The quick brown fox jumps over the lazydog."
20  encdec(cm, plaintext)
21

```

(continues on next page)

(continued from previous page)

```

22 cm = SaveAll(cm)
23
24 plaintext = u"Why did the chicken cross the road? Gb trg gb gur bgure fvqr"
25 encdec(cm, plaintext)
26
27 plaintext = u"Die heiÙe Zypernsonne quälte Max und Victoria ja böse auf dem Weg bis_
↪zur Küste."
28 cm.set_alphabet(al.GERMAN)
29 encdec(cm, plaintext)
30
31 plaintext = u"FAQ om Schweiz: Klöv du trång pjäxby?"
32 cm.set_alphabet(al.SWEDISH)
33 encdec(cm, plaintext)
34
35 cm.set_alphabet(al.JAPANESE_HIRAGANA)
36 cm.set_key(1)
37 plaintext = u""
38 encdec(cm, plaintext)
39
40 cm = Block(cm)
41 encdec(cm, plaintext)
42
43 '''
44 -----
45 The quick brown fox jumps over the lazydog.
46 gurdhvpxoebjasbkwhzcfbiregurylnlqbt
47 thequickbrownfoxjumpsoverthelazydog
48 -----
49 Why did the chicken cross the road? Gb trg gb gur bgure fvqr
50 Jul qvq gur puvpxra pebff gur ebnq? To get to the other side
51 Why did the chicken cross the road? Gb trg gb gur bgure fvqr
52 -----
53 Die heiÙe Zypernsonne quälte Max und Victoria ja böse auf dem Weg bis zur Küste.
54 Sxt wtxot Kjatcüdßüüt bfläet Öpi füs Gxreßcxp yp qmdt pfu stö Htv qxd kfc Zndet.
55 Die heiÙe Zypernsonne quälte Max und Victoria ja böse auf dem Weg bis zur Küste.
56 -----
57 FAQ om Schweiz: Klöv du trång pjäxby?
58 UPB oä Drwhtxk: Zång sf eclöv aymiqj?
59 FAQ om Schweiz: Klöv du trång pjäxby?
60 -----
61
62
63
64 -----
65
66
67
68 '''

```

## 2.22 Rot5

**class** secretpy.Rot5

The Rot5 Cipher

**decrypt** (text, key=None, alphabet=None)

Decryption method

#### Parameters

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key=None*, *alphabet=None*)

Encryption method

#### Parameters

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

## 2.22.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Rot5, alphabets, CryptMachine
5
6
7  def encdec(machine, plaintext):
8      print("-----")
9      print(plaintext)
10     enc = machine.encrypt(plaintext)
11     print(enc)
12     dec = machine.decrypt(enc)
13     print(dec)
14
15
16  cm = CryptMachine(Rot5())
17
18  plaintext = "My text " + alphabets.DECIMAL + " your text "
19  encdec(cm, plaintext)
20
21  '''
22  -----
23  My text 0123456789 your text
24  5678901234
25  0123456789
26  '''

```

## 2.23 Rot18

**class** `secretpy.Rot18`

The Rot18 Cipher

**decrypt** (*text*, *key=None*, *alphabet=None*)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – is not used
- **alphabet** (*string*) – is not used

**Returns** `text`

**Return type** `string`

**encrypt** (*text*, *key=None*, *alphabet=None*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – is not used
- **alphabet** (*string*) – is not used

**Returns** `text`

**Return type** `string`

### 2.23.1 Examples

```
1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Rot18, CryptMachine
5  from secretpy.cmdecorators import SaveAll, UpperCase, Block
6
7
8  def encdec(machine, plaintext):
9      print("-----")
10     print(plaintext)
11     enc = machine.encrypt(plaintext)
12     print(enc)
13     dec = machine.decrypt(enc)
14     print(dec)
15
16
17  plaintext = u"The man has 536 dogs! How many dogs do you have?"
18
19  cm = CryptMachine(Rot18())
20  encdec(cm, plaintext)
21
22  cm1 = Block(cm)
23  encdec(cm1, plaintext)
```

(continues on next page)

(continued from previous page)

```

24 cm = SaveAll(cm)
25 encdec(cm, plaintext)
26
27 cm = UpperCase(cm)
28 encdec(cm, plaintext)
29
30
31 '''
32 -----
33 The man has 536 dogs! How many dogs do you have?
34 gurznaunf081qbt fubjz nalqb tfqblbhunir
35 themanhas536dogshowmanydogsdoyouhave
36 -----
37 The man has 536 dogs! How many dogs do you have?
38 gurzn aunf0 8lqbt fubjz nalqb tfqbl bhuni r
39 themanhas536dogshowmanydogsdoyouhave
40 -----
41 The man has 536 dogs! How many dogs do you have?
42 Gur zna unf 081 qbt f! Ubj znal qbt f qb lbh unir?
43 The man has 536 dogs! How many dogs do you have?
44 -----
45 The man has 536 dogs! How many dogs do you have?
46 GUR ZNA UNF 081 QBTF! UBJ ZNAL QBTF QB LBH UNIR?
47 THE MAN HAS 536 DOGS! HOW MANY DOGS DO YOU HAVE?
48 '''

```

## 2.24 Rot47

**class** secretpy.**Rot47**

The Rot47 Cipher

**decrypt** (*text*, *key=None*, *alphabet=None*)

Decryption method

### Parameters

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key=None*, *alphabet=None*)

Encryption method

### Parameters

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

### 2.24.1 Examples

```
1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Rot47, CryptMachine
5  from secretpy.cmdecorators import SaveAll
6
7
8  def encdec(machine, plaintext):
9      print("-----")
10     print(plaintext)
11     enc = machine.encrypt(plaintext)
12     print(enc)
13     dec = machine.decrypt(enc)
14     print(dec)
15
16
17 plaintext = u"The Quick Brown Fox Jumps Over The Lazy Dog."
18
19 cm = CryptMachine(Rot47())
20 encdec(cm, plaintext)
21
22 cm = SaveAll(cm)
23 encdec(cm, plaintext)
24
25 '''
26 -----
27 The Quick Brown Fox Jumps Over The Lazy Dog.
28 %96"F:4<qC@H?u@IyF>AD~G6C%96{2KJs@8]
29 TheQuickBrownFoxJumpsOverTheLazyDog.
30 -----
31 The Quick Brown Fox Jumps Over The Lazy Dog.
32 %96 "F:4< qC@H? u@I yF>AD ~G6C %96 {2KJ s@8]
33 The Quick Brown Fox Jumps Over The Lazy Dog.
34 '''
```

## 2.25 Scytale

**class** secretpy.**Scytale**

The Scytale Cipher

**decrypt** (*text*, *key*, *alphabet=None*)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key - Number of windings
- **alphabet** – Alphabet is not used

**Returns** decrypted text

**Return type** string

**encrypt** (*text*, *key*, *alphabet=None*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key - Number of windings
- **alphabet** – Alphabet is not used

**Returns** encrypted text

**Return type** string

## 2.25.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Scytale, CryptMachine, alphabets as al
5  from secretpy.cmdecorators import UpperCase, Block, SaveAll
6
7
8  alphabet = al.GERMAN
9  plaintext = u"thequickbrownfoxjumpsoverthelazydog"
10 key = 3
11 cipher = Scytale()
12
13 print(plaintext)
14 enc = cipher.encrypt(plaintext, key, alphabet)
15 print(enc)
16 dec = cipher.decrypt(enc, key, alphabet)
17 print(dec)
18
19
20 def encdec(machine, plaintext):
21     print("-----")
22     print(plaintext)
23     enc = machine.encrypt(plaintext)
24     print(enc)
25     print(machine.decrypt(enc))
26
27
28 cm0 = CryptMachine(cipher, key)
29 cm0.set_alphabet(al.ENGLISH + al.DECIMAL)
30
31 cm = cm0
32 plaintext = "I don't love non-alphabet characters. I will remove all of them: ^,&@$~
33 ↳(*;?&#. Great!"
34 encdec(cm, plaintext)
35
36 cm = UpperCase(Block(cm, length=5, sep="-"))
37 plaintext = "This text is divided by blocks of length 5!"
38 encdec(cm, plaintext)

```

(continues on next page)

(continued from previous page)

```

38
39 cm = SaveAll(cm0)
40 plaintext = "I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!"
41 encdec(cm, plaintext)
42
43 cm.set_alphabet(al.JAPANESE_HIRAGANA)
44 plaintext = u"text !"
45 encdec(cm, plaintext)
46
47 cm = UpperCase(cm)
48 alphabet = al.GREEK
49 cm.set_alphabet(alphabet)
50 plaintext = u"Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)"
51 encdec(cm, plaintext)
52
53 '''
54 thequickbrownfoxjumpsoverthelazydog
55 tqcrnxmorezohukofjpvltlygeibwousehad
56 thequickbrownfoxjumpsoverthelazydog
57 -----
58 I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
59 inonahaeilevlfertdtvolatacrwlmeltmeolenpbcrtsiroaohga
60 idontlovenonalphabetcharactersiwillremoveallofthemgreat
61 -----
62 This text is divided by blocks of length 5!
63 TSXSV-EYOSL-G5HTT-DIDBC-OETIE-IIDBL-KFNH
64 THISTEXTISDIVIDEDBYBLOCKSOFLNGTH5
65 -----
66 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
67 I vola tac-rheeaile npbcrtsseat. Ttona heh : ^,&@$~(*;?&#. Aets'r hs!
68 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
69 -----
70 text !
71 text !
72 text !
73 -----
74 Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)
75 ΘΕΡΙ ΛΗΕΕΑ ΔΑ ΟΙΕΚ Τ ΜΕΥΡΑΡΣΛΣ. (ΛΑΤΑΗΛ ΘΝΚΒ)
76 ΘΛΕΙ ΑΡΕΤ ΚΑΙ ΤΛΜΗ Η ΕΛΕΥΘΕΡΑ. (ΑΝΔΡΑΣ ΚΛΒΟΣ)
77 '''

```

## 2.26 Simple Substitution

**class** secretpy.SimpleSubstitution

The Simple Substitution Cipher

**decrypt** (*text*, *key*, *alphabet*=*'abcdefghijklmnopqrstuvwxyz'*)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*string*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is

used

**Returns** text

**Return type** string

**encrypt** (*text*, *key*, *alphabet*='abcdefghijklmnopqrstuvwxyz')

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*string*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

### 2.26.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import SimpleSubstitution, CryptMachine, alphabets as al
5  from secretpy.cmdecorators import Block, SaveAll
6
7
8  cipher = SimpleSubstitution()
9  alphabet = al.GERMAN
10 plaintext = u"schweißgequält vom ödentext zur nttypograf jakob"
11 key = u"fwxyäöeßü d a b c g l m n p q r s t u v h j o i k z"
12
13 print(plaintext)
14 enc = cipher.encrypt(plaintext, key, alphabet)
15 print(enc)
16 dec = cipher.decrypt(enc, key, alphabet)
17 print(dec)
18
19 #####
20
21
22 def encdec(machine, plaintext):
23     print("-----")
24     print(plaintext)
25     enc = machine.encrypt(plaintext)
26     print(enc)
27     print(machine.decrypt(enc))
28
29
30 cm0 = CryptMachine(cipher, key)
31
32 cm = cm0
33 cm.set_alphabet(al.ENGLISH)
34 cm.set_key("yzabchijkdeflmntuvwxopqrsg")
35 plaintext = "I don't love non-alphabet characters. I will remove all of them: ^,&@$~
    ↳ (*;?&#. Great!"

```

(continues on next page)

(continued from previous page)

```

36 encdec(cm, plaintext)
37
38 cm = Block(cm, length=5, sep="")
39 plaintext = "This text is divided by blocks of length 5!"
40 encdec(cm, plaintext)
41
42 cm = SaveAll(cm0)
43 plaintext = "I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!"
44 encdec(cm, plaintext)
45
46 cm.set_alphabet(al.ENGLISH_SQUARE_IJ)
47 key = (
48     "n", "g", "a", "b", "l",
49     "s", "t", "u", "v", "c",
50     "m", "o", "p", "q", "h",
51     "ij", "k", "w", "x", "y",
52     "r", "d", "e", "f", "z"
53 )
54 cm.set_key(key)
55 plaintext = "Jj becomes Ii because we use ENGLISH_SQUARE_IJ!"
56 encdec(cm, plaintext)
57
58 alphabet = u"abcdABCDEFghijFGHIJ"
59 key = u"dABDFIJEfgCHabchijG"
60 cm.set_alphabet(alphabet)
61 cm.set_key(key)
62 plaintext = u"Text aBcdHijf"
63 encdec(cm, plaintext)
64
65 '''
66 schweißgequältvomödentextzürnttypografjakob
67 qxßuäüzeänsobrtlcüyägrävrjkgprrhmlepfdöfalw
68 schweißgequältvomödentextzürnttypografjakob
69 -----
70 I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
71 kbnmxfnpcmnmyftjyzcxajvyaxcvwkqkffvclnpcyffnhxjclivcyx
72 idontlovenonalphabetcharactersiwillremoveallofthemgreat
73 -----
74 This text is divided by blocks of length 5!
75 xjkwx-crxxw-bkpkb-cbzzs-fnaew-nhfcm-ixj
76 thistextisdividedbyblocksoflength
77 -----
78 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
79 K fnpc mnm-yftjyzcx ajvyaxcvw. Xjwc yvc : ^,&@$~(*;?&#. Xjyx'w kx!
80 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
81 -----
82 Jj becomes Ii because we use ENGLISH_SQUARE_IJ!
83 Vv glaqolw Vv glanywl dl ywl LPTMVWU-WIYNKL_VV!
84 Ii becomes Ii because we use ENGLISH_SQUARE_II!
85 -----
86 Text aBcdHijf
87 Text dIBDiabg
88 Text aBcdHijf
89 '''

```

## 2.27 Three Square

**class** secretpy.ThreeSquare

The Three Square Cipher

**decrypt** (*text*, *key*, *alphabet*=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'))

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*tuple of 3 strings*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key*, *alphabet*=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'))

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*tuple of 3 strings*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

### 2.27.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import ThreeSquare, CryptMachine, alphabets as al
5  from secretpy.cmdecorators import UpperCase, SaveAll, Block
6
7
8  def encdec(machine, plaintext):
9      print("=" * 80)
10     print(plaintext)
11     enc = machine.encrypt(plaintext)
12     print(enc)
13     dec = machine.decrypt(enc)
14     print(dec)
15
16
17  key = (u"example", u"keyword", u"third")
18  cm = UpperCase(CryptMachine(ThreeSquare()))
19  cm.set_alphabet(al.ENGLISH_SQUARE_QQ)

```

(continues on next page)

(continued from previous page)

```

20 cm.set_key(key)
21 plaintext = u"The quick brown fox jumps over the lazy dog!"
22 encdec(cm, plaintext)
23
24 cm = SaveAll(cm)
25 encdec(cm, plaintext)
26
27 cm = SaveAll(CryptMachine(ThreeSquare()))
28 cm.set_alphabet(al.ENGLISH_SQUARE_IJ)
29 key = (u"criptog", u"segurt", u"mars")
30 cm.set_key(key)
31 plaintext = u"Attack at dawns!!!"
32 encdec(cm, plaintext)
33
34 cm.set_alphabet(al.GREEK_SQUARE)
35 key = (u"κλειδ", u"κλειδ", u"λειδκ")
36 cm.set_key(key)
37 plaintext = u"Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)"
38 encdec(cm, plaintext)
39
40 cm = UpperCase(Block(CryptMachine(ThreeSquare())))
41 cm.set_alphabet(al.GREEK_SQUARE)
42 key = (u"κλειδ", u"κλειδ", u"λειδκ")
43 cm.set_key(key)
44 encdec(cm, plaintext)
45
46 '''
47 =====
48 The quick brown fox jumps over the lazy dog!
49 TPGEDELYICAWBACHSYNNJOSUJJTYRPSUWHWYINTBJELCBTXYSFONMV
50 THEOUICKBROWNFOXJUMPSOVERTHELAZYDOGZ
51 =====
52 The quick brown fox jumps over the lazy dog!
53 TPJ LDONY IAAEX ARB SOUNH OSXS JVM RMMU EHW!OWNZBJEGCAKXKSFYUMU
54 THE OUICK BROWN FOX JUMPS OVER THE LAZY DOG!Z
55 =====
56 Attack at dawns!!!
57 Actuac vs ihctz!!!ddnwosuz
58 Attack at dawns!!!z
59 =====
60 Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)
61 Ινιχι υλβρ σερ τβκλβ θ κσωμεμογο. (Γκνιπυ ρζεσθλ)ζπγξδεσξηληπλαβιβηνετψ
62 Θελει αρετη και τολμη η ελευθερια. (Ανδρεα Καλβο)ω
63 =====
64 Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)
65 ΠΝΚΛΙ ΕΙΛΒΒ ΣΙΗΤΓ ΜΛΓΑΚ ΣΔΜΕΚ ΟΗΩΓΙ ΣΙΚΠΥ ΜΖΕΠΥ ΛΒΠΓΘ ΤΕΡΞΛ ΖΕΠΕΘ ΒΚΚΗΞ ΞΤΨ
66 ΘΕΛΕΙΑΡΕΤΗΚΑΙΤΟΛΜΗΗΕΛΕΥΘΕΡΙΑΑΝΔΡΕΑΣΚΑΛΒΟΣΩ
67 '''

```

## 2.28 Trifid

```
class secretpy.Trifid
```

The Trifid Cipher

**decrypt** (*text*, *key*=None, *alphabet*=None)

Decryption method

#### Parameters

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English with ‘.’ is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key=None*, *alphabet=None*)

Encryption method

#### Parameters

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English with ‘.’ is used

**Returns** text

**Return type** string

## 2.28.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Trifid, CryptMachine
5  from secretpy.cmdecorators import SaveAll
6
7
8  def encdec(machine, plaintext):
9      print(plaintext)
10     enc = machine.encrypt(plaintext)
11     print(enc)
12     print(machine.decrypt(enc))
13     print("-----")
14
15
16  key = 5
17  cm = CryptMachine(Trifid(), key)
18
19  alphabet = u"epsducvwym.zlksnbtfgorijhaq" # 27 characters
20  cm.set_alphabet(alphabet)
21
22  plaintext = u"defendtheeastwallofthecastle"
23  encdec(cm, plaintext)
24
25  cm1 = cm
26  alphabet = (
27      u"aaa", u"b", u"c",
28      u"d", u"e", u"f",

```

(continues on next page)

(continued from previous page)

```

29     u"g", u"h", u"i",
30
31     u"j", u"k", u"l",
32     u"m", u"n", u"oö",
33     u"p", u"q", u"r",
34
35     u"s", u"t", u"u",
36     u"v", u"w", u"x",
37     u"y", u"z", u"+",
38 )
39 cm1.set_alphabet(alphabet)
40
41 plaintext = u"Flygande bäckasiner söka hwila på mjuka tuvor!"
42 encdec(cm1, plaintext)
43
44 cm2 = SaveAll(cm)
45 alphabet = "felimardstbcghjknopquvwyz+"
46 cm2.set_alphabet(alphabet)
47
48 plaintext = u"Aide-toi, le ciel t'aidera"
49 encdec(cm2, plaintext)
50
51
52 '''
53 defendtheeastwallofthecastle
54 suefecphsegyyjiximfofocejlrf
55 defendtheeastwallofthecastle
56 -----
57 Flygande bäckasiner söka hwila på mjuka tuvor!
58 fbiiajbmdmdsazckwpnujshvokdgpaggackzkri
59 flygandebäckasinersokahwilapamjukatuvoor
60 -----
61 Aide-toi, le ciel t'aidera
62 Fmjf-voi, ss uftf p'ufeqqc
63 Aide-toi, le ciel t'aidera
64 -----
65 '''

```

## 2.29 Two Square

**class** secretpy.**TwoSquare**

The Two-Square Cipher, also called Double Playfair

**decrypt** (*text*, *key*=None, *alphabet*=('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'))

Decryption method

### Parameters

- **text** (*string*) – Text to decrypt
- **key** (*integer*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key*=None, *alphabet*=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'))

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

## 2.29.1 Examples

```

1 #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import TwoSquare, CryptMachine, alphabets
5  from secretpy.cmdecorators import UpperCase
6
7
8  def encdec(machine, plaintext):
9      print(plaintext)
10     enc = machine.encrypt(plaintext)
11     print(enc)
12     dec = machine.decrypt(enc)
13     print(dec)
14     print("-----")
15
16
17 alphabet = alphabets.ENGLISH_SQUARE_OO
18
19 key = (u"example", u"keyword")
20
21 cm = UpperCase(CryptMachine(TwoSquare()))
22
23 cm.set_alphabet(alphabet)
24 cm.set_key(key)
25 plaintext = u"Help me Obi wan Kenobi"
26 encdec(cm, plaintext)
27
28 plaintext = u"Help me Obi wan Kenobi y"
29 encdec(cm, plaintext)
30
31 '''
32 Help me Obi wan Kenobi
33 XGDLXWSDJYRYHOTKDG
34 HELPMEOWIWANKENOB
35 -----
36 Help me Obi wan Kenobi y
37 XGDLXWSDJYRYHOTKDGZX
38 HELPMEOWIWANKENOBIZ

```

(continues on next page)

(continued from previous page)

```

39 -----
40 ' ' '

```

## 2.30 Vic

**class** secretpy.Vic

The Vic Cipher

**decrypt** (*text*, *key=None*, *alphabet=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z')*)

Decryption method

**Parameters**

- **text** (*string*) – Text to decrypt
- **key** (*number string*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (*text*, *key=None*, *alphabet=('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'ij', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z')*)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*number string*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

### 2.30.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Vic, CryptMachine
5  from secretpy.cmdecorators import SaveAll
6
7
8  def encdec(machine, plaintext):
9      print("-----")
10     print(plaintext)
11     enc = machine.encrypt(plaintext)
12     print(enc)
13     print(machine.decrypt(enc))

```

(continues on next page)

(continued from previous page)

```

14
15
16 key = "0452"
17 cm = SaveAll(CryptMachine(Vic(), key))
18 alphabet = [
19     u"e", u"t", u" ", u"a", u"o", u"n", u" ", u"r", u"i", u"s",
20     u"b", u"c", u"d", u"f", u"g", u"h", u"j", u"k", u"l", u"m",
21     u"p", u"q", u"/", u"u", u"v", u"w", u"x", u"y", u"z", u".",
22 ]
23 plaintext = u"Attack at dawn!"
24 cm.set_alphabet(alphabet)
25 encdec(cm, plaintext)
26
27 '''
28 Output:
29
30 -----
31 Attack at dawn!
32 Anwhrs an roae!er
33 Attack at dawn!
34 '''

```

## 2.31 Vigenere

**class** secretpy.Vigenere

The Vigenere Cipher

**decrypt** (text, key, alphabet='abcdefghijklmnopqrstuvwxyz')

Decryption method

### Parameters

- **text** (*string*) – Text to decrypt
- **key** (*string*) – Decryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

**encrypt** (text, key, alphabet='abcdefghijklmnopqrstuvwxyz')

Encryption method

### Parameters

- **text** (*string*) – Text to encrypt
- **key** (*string*) – Encryption key
- **alphabet** (*string*) – Alphabet which will be used, if there is no a value, English is used

**Returns** text

**Return type** string

## 2.31.1 Examples

```

1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Vigenere, CryptMachine, alphabets as al
5  from secretpy.cmdecorators import UpperCase, Block, SaveAll
6
7  alphabet = al.GERMAN
8  plaintext = u"schweißgequält vom ödentext zur nttypograf jakob"
9  key = u"schlüssel"
10
11 cipher = Vigenere()
12 print(plaintext)
13
14 enc = cipher.encrypt(plaintext, key, alphabet)
15 print(enc)
16 dec = cipher.decrypt(enc, key, alphabet)
17 print(dec)
18
19
20 def encdec(machine, plaintext):
21     print("-----")
22     print(plaintext)
23     enc = machine.encrypt(plaintext)
24     print(enc)
25     print(machine.decrypt(enc))
26
27
28 cm0 = CryptMachine(cipher, key)
29
30 cm = cm0
31 cm.set_alphabet(al.ENGLISH)
32 cm.set_key("keys")
33 plaintext = "I don't love non-alphabet characters. I will remove all of them: ^,&@$~
34 ↳(*;?&#. Great!"
35 encdec(cm, plaintext)
36
37 cm = Block(cm, length=5, sep="-")
38 plaintext = "This text is divided by blocks of length 5!"
39 encdec(cm, plaintext)
40
41 cm = SaveAll(cm0)
42 plaintext = "I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!"
43 encdec(cm, plaintext)
44
45 cm.set_alphabet(al.ENGLISH_SQUARE_IJ)
46 plaintext = "Jj becomes Ii because we use ENGLISH_SQUARE_IJ!"
47 encdec(cm, plaintext)
48
49 cm.set_alphabet(al.JAPANESE_HIRAGANA)
50 cm.set_key(u"")
51 plaintext = u"text          !"
52 encdec(cm, plaintext)
53
54 cm = UpperCase(cm)
55 alphabet = al.GREEK

```

(continues on next page)

(continued from previous page)

```

55 cm.set_alphabet(alphabet)
56 cm.set_key(u"κλειδ")
57 plaintext = u"Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)"
58 encdec(cm, plaintext)
59
60 '''
61 schweißgequältvomödentextzürnttypografjakob
62 geodcärkpewdwrjcqivguac lhßjfpääwdcküshqlict
63 schweißgequältvomödentextzürnttypografjakob
64 -----
65 I don't love non-alphabet characters. I will remove all of them: ^,&@$~(*;?&#. Great!
66 shmfdpnmnormf kpnzkfclmlyjkgrwbwospjjjoqmnoejdyjrzoqejjoer
67 idontlovenonalphabetcharactersiwillremoveallofthemgreat
68 -----
69 This text is divided by blocks of length 5!
70 dlqkd-ivlsw-bafmb-wnfwt-vsacc-sddor-elr
71 thistextisdividedbyblocksoflength
72 -----
73 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
74 S pmno rmf-kpnzkfcl mlyjkgrwbw. Rzowc sbi : ^,&@$~(*;?&#. Rzkx'q ad!
75 I love non-alphabet characters. These are : ^,&@$~(*;?&#. That's it!
76 -----
77 Jj becomes Ii because we use ENGLISH_SQUARE_IJ!
78 Sn zwmskwb Ng togymbi uw dwc WWLIABM_QHDEPW_SN!
79 Ii becomes Ii because we use ENGLISH_SQUARE_II!
80 -----
81 text          !
82 text          !
83 text          !
84 -----
85 Θλει αρετ και τλμη η ελευθερα. (Ανδρα Κλβο)
86 ΣΠΝΜ ΒΨΠ ΞΤ ΤΥΠ Ρ ΝΨΣΟΓΞΙ. (ΔΧΞΓΙΙΦ ΥΛΧΖΨΦ)
87 ΛΕΙ ΑΡΕΤ ΚΑΙ ΤΛΜΗ Η ΕΛΕΥΘΕΡΑ. (ΑΝΔΡΑΣ ΚΛΒΟΣ)
88 '''

```

## 2.32 Zigzag

**class** secretpy.Zigzag

The Zigzag Cipher (Rail-Fence)

**decrypt** (text, key, alphabet=None)

Decryption method

**Parameters**

- **text** (string) – Text to decrypt
- **key** (integer) – Decryption key
- **alphabet** (string) – unused

**Returns** text

**Return type** string

**encrypt** (text, key, alphabet=None)

Encryption method

**Parameters**

- **text** (*string*) – Text to encrypt
- **key** (*integer*) – Encryption key
- **alphabet** (*string*) – unused

**Returns** text**Return type** string

## 2.32.1 Examples

```
1  #!/usr/bin/python
2  # -*- encoding: utf-8 -*-
3
4  from secretpy import Zigzag
5
6
7  plaintext = u"thequickbrownfoxjumpsoverthelazydog"
8  plaintext = u"thequick"
9  key = 3
10
11  chipher = Zigzag()
12  print(plaintext)
13
14  enc = chipher.encrypt(plaintext, key)
15  print(enc)
16  dec = chipher.decrypt(enc, key)
17  print(dec)
18
19  #####
20
21  print("-----")
22
23  plaintext = u"wearediscoveredfleeatonce"
24
25  print(plaintext)
26  enc = chipher.encrypt(plaintext, key)
27  print(enc)
28  dec = chipher.decrypt(enc, key)
29  print(dec)
30
31  #####
32
33  print("-----")
34
35  plaintext = u"defendtheeastwallofthecastle"
36  key = 4
37
38  print(plaintext)
39  enc = chipher.encrypt(plaintext, key)
40  print(enc)
41  dec = chipher.decrypt(enc, key)
42  print(dec)
43
44  '''
```

(continues on next page)

(continued from previous page)

```
45 thequickbrownfoxjumpsoverthelazydog
46 tubnjsrldhqikrwxupoeteayoecoomvhzg
47 thequickbrownfoxjumpsoverthelazydog
48 -----
49 wearediscoveredfleeatonce
50 wecrlteerdsoeefaocaivden
51 wearediscoveredfleeatonce
52 -----
53 defendtheeastwallofthecastle
54 dttfsedhswotatfneaalhcleee
55 defendtheeastwallofthecastle
56 '''
```



## CHAPTER 3

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



## A

ADFGVX (*class in secretpy*), 10  
 ADFGX (*class in secretpy*), 7  
 Affine (*class in secretpy*), 12  
 Atbash (*class in secretpy*), 14  
 Autokey (*class in secretpy*), 16

## B

Bazeries (*class in secretpy*), 18  
 Beaufort (*class in secretpy*), 20  
 Bifid (*class in secretpy*), 22

## C

Caesar (*class in secretpy*), 24  
 CaesarProgressive (*class in secretpy*), 26  
 Chao (*class in secretpy*), 29  
 ColumnarTransposition (*class in secretpy*), 30

## D

decrypt () (*secretpy.ADFGVX method*), 10  
 decrypt () (*secretpy.ADFGX method*), 7  
 decrypt () (*secretpy.Affine method*), 12  
 decrypt () (*secretpy.Atbash method*), 14  
 decrypt () (*secretpy.Autokey method*), 16  
 decrypt () (*secretpy.Bazeries method*), 18  
 decrypt () (*secretpy.Beaufort method*), 20  
 decrypt () (*secretpy.Bifid method*), 22  
 decrypt () (*secretpy.Caesar method*), 24  
 decrypt () (*secretpy.CaesarProgressive method*), 26  
 decrypt () (*secretpy.Chao method*), 29  
 decrypt () (*secretpy.ColumnarTransposition method*), 30  
 decrypt () (*secretpy.FourSquare method*), 32  
 decrypt () (*secretpy.Gronsfeld method*), 34  
 decrypt () (*secretpy.Keyword method*), 36  
 decrypt () (*secretpy.MyszkowskiTransposition method*), 39  
 decrypt () (*secretpy.Nihilist method*), 41  
 decrypt () (*secretpy.Playfair method*), 43

decrypt () (*secretpy.Polybius method*), 45  
 decrypt () (*secretpy.Porta method*), 47  
 decrypt () (*secretpy.Rot13 method*), 49  
 decrypt () (*secretpy.Rot18 method*), 52  
 decrypt () (*secretpy.Rot47 method*), 53  
 decrypt () (*secretpy.Rot5 method*), 50  
 decrypt () (*secretpy.Scytale method*), 54  
 decrypt () (*secretpy.SimpleSubstitution method*), 56  
 decrypt () (*secretpy.ThreeSquare method*), 59  
 decrypt () (*secretpy.Trifid method*), 60  
 decrypt () (*secretpy.TwoSquare method*), 62  
 decrypt () (*secretpy.Vic method*), 64  
 decrypt () (*secretpy.Vigenere method*), 65  
 decrypt () (*secretpy.Zigzag method*), 67

## E

encrypt () (*secretpy.ADFGVX method*), 10  
 encrypt () (*secretpy.ADFGX method*), 7  
 encrypt () (*secretpy.Affine method*), 12  
 encrypt () (*secretpy.Atbash method*), 14  
 encrypt () (*secretpy.Autokey method*), 16  
 encrypt () (*secretpy.Bazeries method*), 18  
 encrypt () (*secretpy.Beaufort method*), 20  
 encrypt () (*secretpy.Bifid method*), 22  
 encrypt () (*secretpy.Caesar method*), 24  
 encrypt () (*secretpy.CaesarProgressive method*), 27  
 encrypt () (*secretpy.Chao method*), 29  
 encrypt () (*secretpy.ColumnarTransposition method*), 31  
 encrypt () (*secretpy.FourSquare method*), 33  
 encrypt () (*secretpy.Gronsfeld method*), 34  
 encrypt () (*secretpy.Keyword method*), 37  
 encrypt () (*secretpy.MyszkowskiTransposition method*), 39  
 encrypt () (*secretpy.Nihilist method*), 41  
 encrypt () (*secretpy.Playfair method*), 44  
 encrypt () (*secretpy.Polybius method*), 45  
 encrypt () (*secretpy.Porta method*), 47  
 encrypt () (*secretpy.Rot13 method*), 49  
 encrypt () (*secretpy.Rot18 method*), 52

`encrypt()` (*secretpy.Rot47 method*), [53](#)  
`encrypt()` (*secretpy.Rot5 method*), [51](#)  
`encrypt()` (*secretpy.Scytale method*), [55](#)  
`encrypt()` (*secretpy.SimpleSubstitution method*), [57](#)  
`encrypt()` (*secretpy.ThreeSquare method*), [59](#)  
`encrypt()` (*secretpy.Trifid method*), [61](#)  
`encrypt()` (*secretpy.TwoSquare method*), [63](#)  
`encrypt()` (*secretpy.Vic method*), [64](#)  
`encrypt()` (*secretpy.Vigenere method*), [65](#)  
`encrypt()` (*secretpy.Zigzag method*), [67](#)

## F

`FourSquare` (*class in secretpy*), [32](#)

## G

`Gronsfeld` (*class in secretpy*), [34](#)

## K

`Keyword` (*class in secretpy*), [36](#)

## M

`MyszkowskiTransposition` (*class in secretpy*), [39](#)

## N

`Nihilist` (*class in secretpy*), [41](#)

## P

`Playfair` (*class in secretpy*), [43](#)

`Polybius` (*class in secretpy*), [45](#)

`Porta` (*class in secretpy*), [47](#)

## R

`Rot13` (*class in secretpy*), [49](#)

`Rot18` (*class in secretpy*), [52](#)

`Rot47` (*class in secretpy*), [53](#)

`Rot5` (*class in secretpy*), [50](#)

## S

`Scytale` (*class in secretpy*), [54](#)

`SimpleSubstitution` (*class in secretpy*), [56](#)

## T

`ThreeSquare` (*class in secretpy*), [59](#)

`Trifid` (*class in secretpy*), [60](#)

`TwoSquare` (*class in secretpy*), [62](#)

## V

`Vic` (*class in secretpy*), [64](#)

`Vigenere` (*class in secretpy*), [65](#)

## Z

`Zigzag` (*class in secretpy*), [67](#)