



Verschlüsseln mit Python

oder

Warum wir zu wenige Buchstaben haben

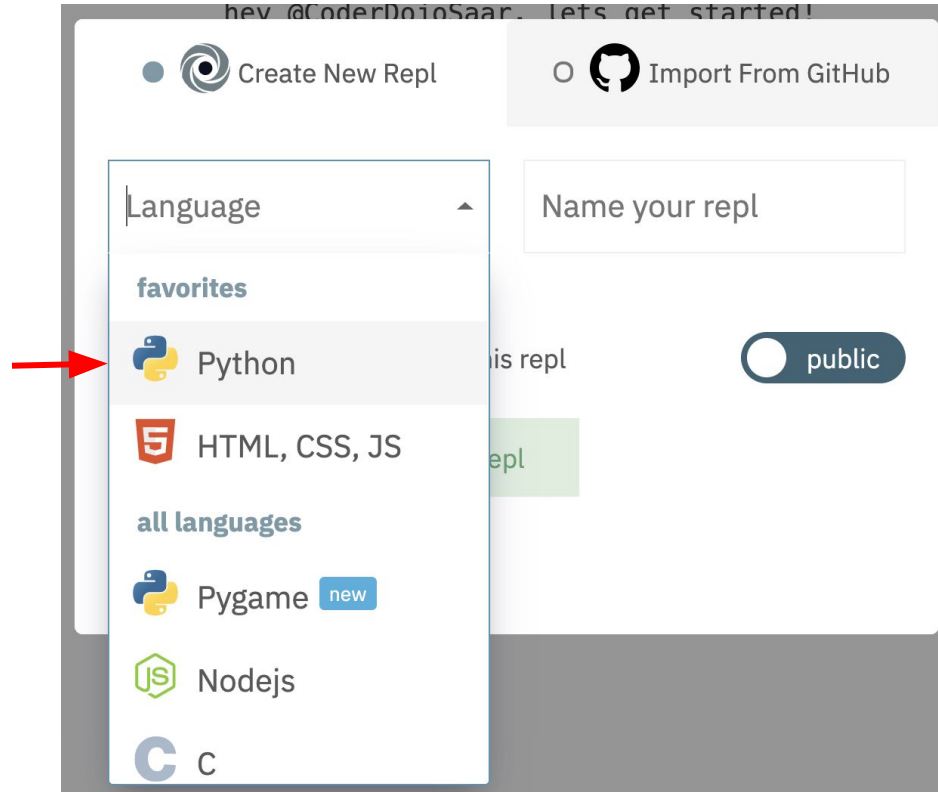
Anmelden bei



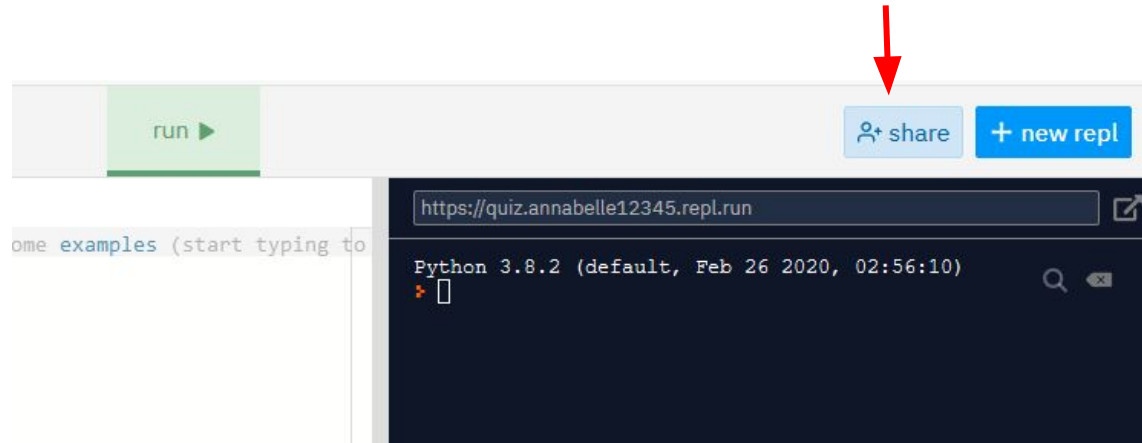
Neues repl erzeugen

The screenshot displays the Replit user interface. On the left is a sidebar with navigation options: Home (selected), My Repls, Talk, and Notifications. The top header includes a search bar labeled 'Search your repls' and a blue '+ new repl' button. A red arrow points to this button. Below the search bar, the 'create a repl' section features three language selection buttons: Python, HTML, CSS, JS, and Java, along with a link for 'all languages'.

Python wählen



Neues repl mit uns teilen



Freigeben für:

- **Annabelle12345**
- **CoderDojoSaar**

Emoji in Zoom



wenn ihr mit einer Aufgabe fertig seid

Wichtige Wörter

- Klartext plaintext
- Schlüssel key
- Geheimtext ciphertext
- verschlüsseln encrypt
- entschlüsseln decrypt



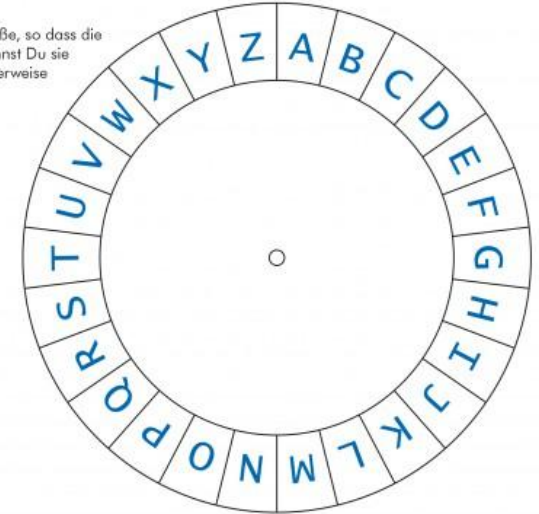
Cäsar Verschlüsselung



Der Cäsar-Code

Bastelanleitung

Schneide die beiden Scheiben aus. Lege die kleine Scheibe auf die große, so dass die Mittelpunkte genau übereinander liegen. In der Mitte der Scheiben kannst Du sie nun mit einem Clip oder einer Klammer verbinden, wie man sie üblicherweise zum Verschließen von Briefumschlägen nimmt. Jetzt solltest Du die Scheiben gut gegeneinander drehen können.



Ver- und Entschlüsseln von Nachrichten mit der Cäsar-Scheibe

Um mit der Cäsar-Scheibe zu verschlüsseln, musst Du zuerst die Scheiben einstellen. Der Buchstabe der inneren Scheibe, der dem Buchstaben **A** der großen Scheibe gegenüber steht, ist der „Schlüssel“. Dies kann zum Beispiel der Buchstabe **X** sein.

Jetzt kannst Du verschlüsseln. Wenn Du das Wort **MATHE** verschlüsseln willst, suchst Du die Buchstaben deines Wortes außen und ersetzt sie durch die entsprechenden Buchstaben innen. Aus **MATHE** wird **JXQEB**.

Das Entschlüsseln funktioniert genau umgekehrt: Du entschlüsselst von innen nach außen.

Wichtig ist, dass Du dabei den gleichen Schlüssel benutzt, der beim Verschlüsseln verwendet wurde. Für unser Beispiel musst Du die Scheiben also so einstellen, dass das **A** der großen Scheibe gegenüber dem **X** der kleinen Scheibe steht.

Ausprobieren!



Klartext: hallo
Schlüssel: 5
Geheimtext: ?



Geheimtext: ajwxhmqzjxxjqs
Schlüssel: 5
Klartext: ?



Ausprobieren!



Klartext: hallo

Schlüssel: 5

Geheimtext: **mfqqt**



Geheimtext: ajwxhmqzjxxjqs

Schlüssel: 5

Klartext: **verschluesseln**



Wie bringen wir das dem Computer bei?

1. Buchstaben werden Zahlen, damit wir rechnen können
2. Zahlen müssen auch wieder zu Buchstaben werden
3. berechnen von einzelnen Buchstaben
4. ganze Wörter verschlüsseln
5. und natürlich auch wieder entschlüsseln

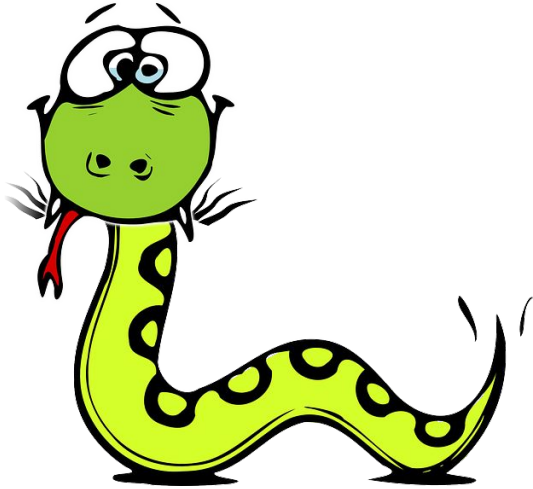
Aus Buchstaben werden Zahlen

Ascii Tabelle

Ascii = **A**merican **S**tandard **C**ode for
Information **I**nterchange

Dec	Hex	Char	Dec	Hex	Char
64		@	96		`
65		A	97		a
66		B	98		b
67		C	99		c
68		D	100		d
69		E	101		e
70		F	102		f
71		G	103		g
72		H	104		h
73		I	105		i
74		J	106		j
75		K	107		k
76		L	108		l
77		M	109		m
78		N	110		n
79		O	111		o
80		P	112		p
81		Q	113		q
82		R	114		r
83		S	115		s
84		T	116		t
85		U	117		u
86		V	118		v
87		W	119		w
88		X	120		x
89		Y	121		y
90		Z	122		z
91		[123		{
92		\	124		
93]	125		}
94		^	126		~
95		_	127		DEL

Aus Buchstaben werden Zahlen



`ord("a")` => 97 Anordnen

`chr(97)` => "a" character (engl .Buchstabe)

`print("Hallo")`

Finde zu den Zahlen die Buchstaben:

99

105

116

Finde zu den Buchstaben ihre Zahlen:

a

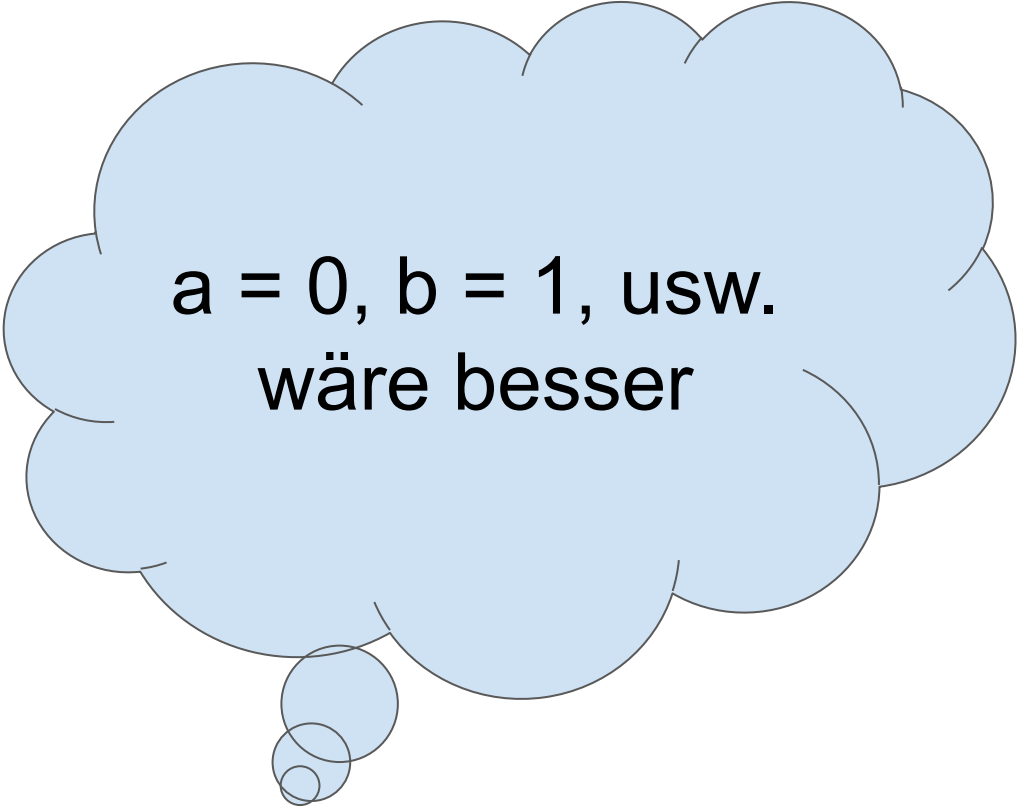
n

x

Ausprobieren!

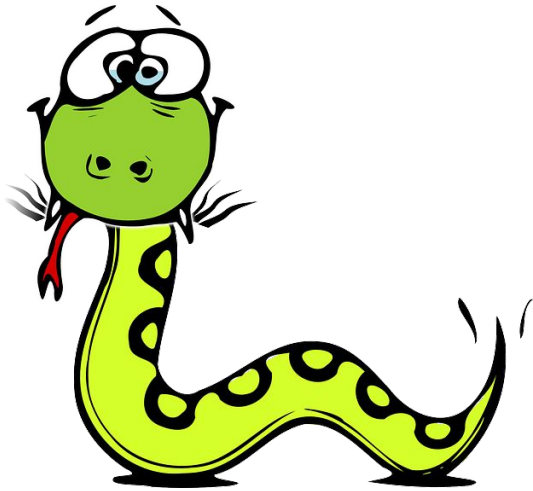
```
print(ord("a"))  
print(ord("n"))  
print(ord("x"))  
  
print(chr(99))  
print(chr(105))  
print(chr(116))
```

```
97  
110  
120  
c  
i  
t
```



**a = 0, b = 1, usw.
wäre besser**

Eigene Funktionen und Variablen



```
def meine_funktion(eingabe):
```

```
↔ return ausgabe
```

```
zahl = 1
```

```
text = "Hallo"
```


Aus Buchstaben werden Zahlen

```
def to_number(character):  
    number = ord(character) - ord("a")  
    return number
```

Testen:

```
print(to_number("a"))  
print(to_number("z"))
```

Ausprobieren!



Schreibe eine Funktion **to_character**, die eine Zahl zurück in einen Buchstaben verwandelt.

0 => a, 1 => b usw.

Teste deine Funktion mit mindestens 3 Zahlen.

Aus Zahlen werden Buchstaben

```
def to_character(number) :  
    character = chr(number + ord("a"))  
    return character
```

Testen:

```
print(to_character(0))  
print(to_character(25))
```

Ausprobieren!



Schreibe eine Funktion **encrypt_character**, die einen Buchstaben um den Wert des Schlüssels verschiebt:

- in Zahl umwandeln
- verschieben (addieren)
- zurück in Buchstabe

Teste deine Funktion mit mindestens 3 Beispielen.

```
def encrypt_character(character, key):  
    number_character = to_number(character)  
    number_new_character = number_character + key  
    new_character = to_character(number_new_character)  
    return new_character
```

Testen...

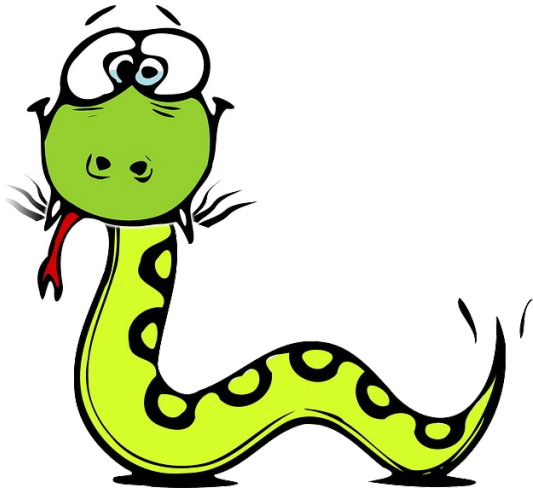
```
def encrypt_character(character, key):  
    number_character = to_number(character)  
    number_new_character = number_character + key  
    new_character = to_character(number_new_character)  
    return new_character
```

```
print(move("a", 1))  
print(move("b", 10))  
print(move("z", 1))
```



b
n
{

Modulo rechnen: Was ist der Rest?



rest = 25 % 3

```
def encrypt_character(character, key):  
    number_character = to_number(character)  
    number_new_character = number_character + key  
    number_new_character = number_new_character % 26  
    new_character = to_character(number_new_character)  
    return new_character
```

Testen...

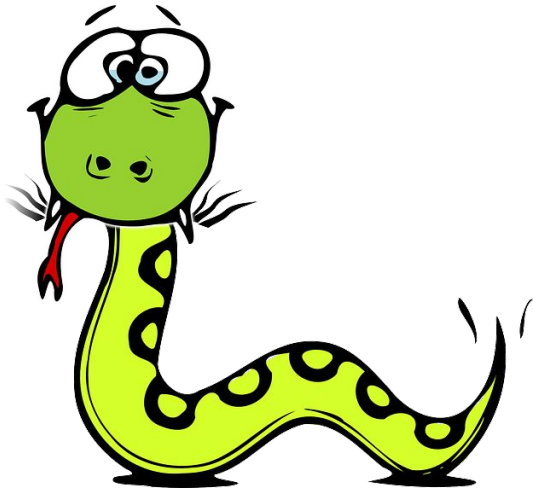

```
def encrypt_character(character, key):  
    number_character = to_number(character)  
    number_new_character = number_character + key  
    number_new_character = number_new_character % 26  
    new_character = to_character(number_new_character)  
    return new_character
```

```
print(verschieben("a", 1))  
print(verschieben("b", 10))  
print(verschieben("z", 1))
```



b
n
a

for - Schleife und Zeichenketten



for einheit **in** menge:

↔ machen

```
text = "Hallo"
```

```
text = text + "Du"
```

```
print(text)           # "HalloDu"
```

Ausprobieren!



Schreibe eine Funktion `encrypt`, die einen Klartext und einen Schlüssel bekommt:

- leeren Geheimtext erstellen
- alle Buchstaben des Klartexts nacheinander verschlüsseln und an den Geheimtext anhängen

Teste deine Funktion mit mindestens 3 Beispielen.

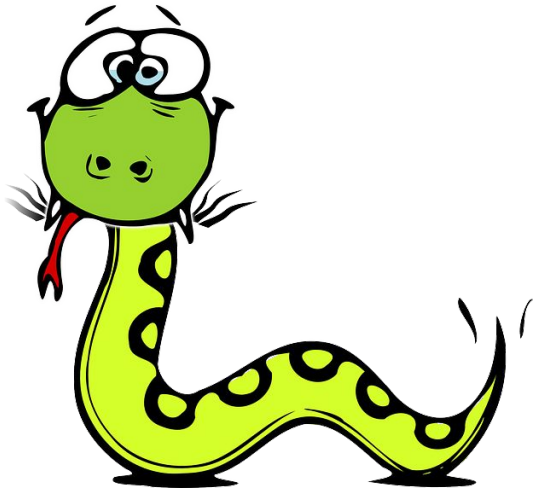
```
def encrypt(plaintext, key):  
    ciphertext = ""  
    for character in plaintext:  
        encrypted_character = encrypt_character(character, key)  
        ciphertext += encrypted_character  
    return ciphertext
```

Testen...

```
print(encrypt("Hallo", 0))  
print(encrypt("Hallo", 1))  
print(encrypt("Hallo", 100))  
print(encrypt("Hallo du da", 5))
```

```
hallo  
ibmmp  
dwhhk  
mfqqtizif
```

Bedingungen mit if



if wahrheitswert:

↔ machen

else:

↔ machen

```
def encrypt(plaintext, key):
```

```
    ciphertext = ""
```

```
    for character in plaintext:
```

```
        # Leerzeichen bleiben:
```

```
        if character == " ":
```

```
            ciphertext = ciphertext + " "
```

```
        else:
```

```
        ←→ ciphertext = ciphertext + encrypt_character(character, key)
```

```
    return ciphertext
```

Was ist der Klartext? Schlüssel: 17

jlgvi zyi yrsk vj
xvjtyrwwk

Wie entschlüsselt man eigentlich?



Ausprobieren!



Schreibe eine Funktion **decrypt**, die ein Wort und einen Schlüssel bekommt:

- “rückwärts verschlüsseln”

Teste deine Funktion mit mindestens 3 Beispielen.

Entschlüsseln

```
def decrypt(ciphertext, key):  
    # Rueckwaerts verschieben  
    plaintext = encrypt(ciphertext, -key)  
    return plaintext
```

Testen...

Was ist der Schlüssel?

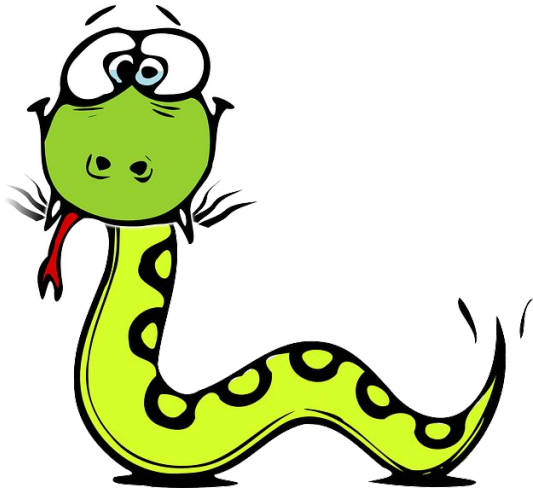
pl hxkk jxk gbabk qbuq
bkqpzeirbppbik rka axp
kro jfq cg sboprzebk axp
dbeq wr bfkcxze labo

A person wearing a brown hat and sunglasses is looking at a laptop screen. The background is filled with binary code (0s and 1s) in green and white. A dark grey banner is overlaid on the image, containing the text 'Brute Force Attack'. Below this banner is a light grey banner containing the text 'einfach alles ausprobieren...'.

Brute Force Attack

einfach alles ausprobieren...

while - Schleife und Zeichenketten



```
x = 0
```

```
while x < 10:
```

```
    x = x + 1
```

Ausprobieren!



Schreibe eine Funktion **brute_force_attack**, die einen Geheimtext bekommt:

- probiere alle Schlüssel aus
- lass jeden gefundenen Klartext auf der Konsole anzeigen
- zeige außerdem den verwendeten Schlüssel an

```
def brute_force_attack(ciphertext):  
    key = 0  
    while key < 27:  
        plaintext = decrypt(ciphertext, key)  
        key += 1  
        print("schluessel: " + str(key))  
        print("klartext: " + str(plaintext))
```


Links + Quellen

<https://www.elektronik-kompodium.de/sites/net/1907041.html>

<https://www.commfront.com/pages/ascii-chart>

<https://repl.it/>

<https://pixabay.com/de/illustrations/cyber-angriff-attacke-4444450/>

<https://de.wikipedia.org/wiki/Caesar-Verschl%C3%BCsselung>

<https://www.ziehenschule.de/news/mit-stift-und-papier-in-richtung-zukunft.html>

<https://emojiterra.com/de/daumen-hoch/> 👍