

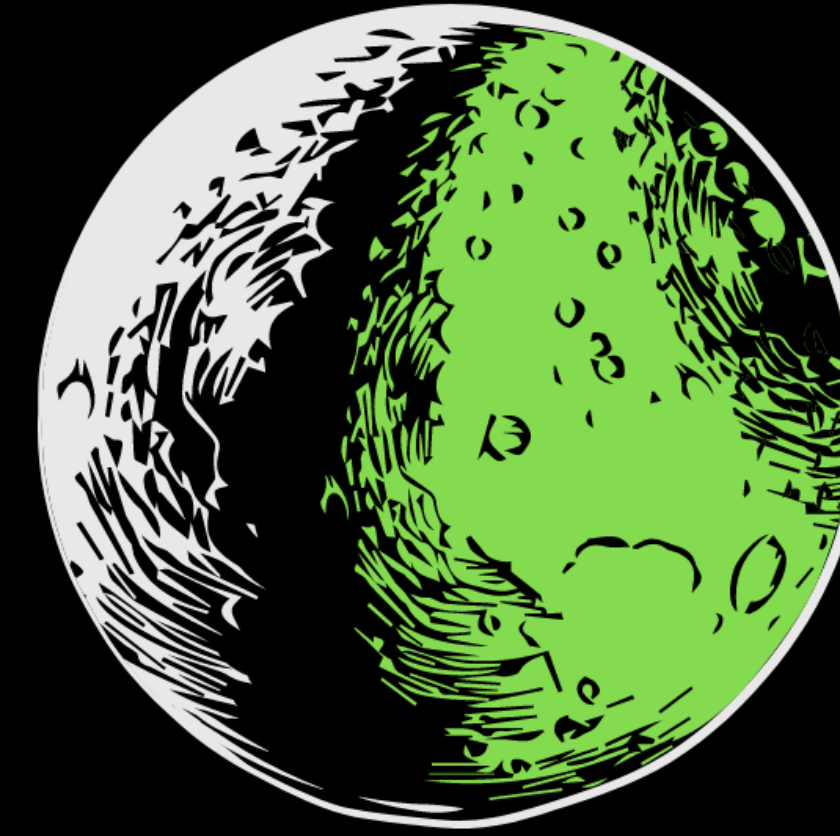
# MONSTER GESICHTET

## Frankenstein Escape Room

Christoph Schilling, Lisa Wagner, Thomas Reimann  
06.12.2020



# DIE STORY



Wir befinden uns in London, das Jahr ist 1875.

Augenzeugenberichte von Monstersichtungen häufen sich. Irgendetwas Merkwürdiges geht vor sich.

Das Dasein als Ermittler\*in ist nicht immer einfach, zu Weilen auch gefährlich. Du verspürst den fast schon gewohnten Drang. Dies ist Fall, den du einfach lösen musst.

Du machst dich auf den Weg mit den Augenzeugen zu reden und das Abenteuer beginnt...



ein



# UMSETZUNG

- High-Level Geschichte & Rätsel
- Aufteilung der Level, so dass der Schwierigkeitsgrad etwa gleich verteilt ist
  - 1 & 6 - Christoph
  - 2 & 5 - Thomas
  - 3 & 4 - Lisa



# UMSETZUNG

- Tools
  - Github + Github Desktop
  - VSCode
- Auf separaten Branches entwickelt
- Alles zusammen geführt
  - Coding Style beibehalten
- Finetuning der Story



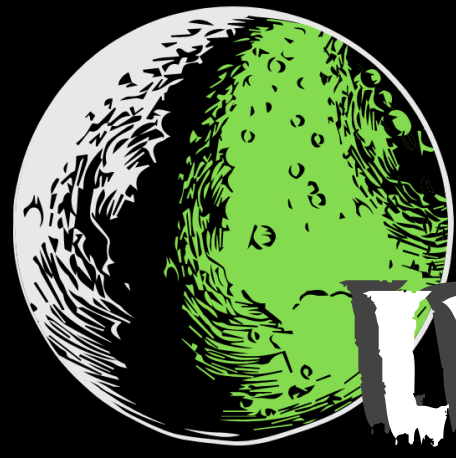


# LEVEL 1

## “DAS VERSTECK”

Schwierigkeitsgrad - Einfach





# LEVEL 1

## Rätsel

- ▶ Methoden
- ▶ list
- ▶ tuple
- ▶ Formeln/Algorithmen
- ▶ call
- ▶ append
- ▶ random

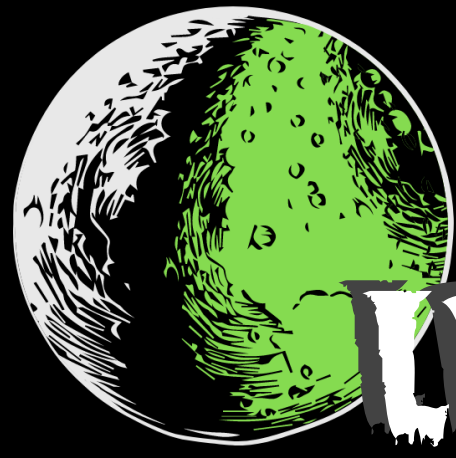
Erstellen eines Tupel/  
mehrere mit Random  
Aufruf

```
def create_level1(self):
    rectangle = random.choice([
        (1, 2, 6, 2, 6, 6, 1, 6),
        (3, 2, 7, 2, 7, 4, 3, 4),
        (3, 1, 9, 1, 9, 6, 3, 6)
    ])

    task_messages = [
        "Ein Monster wurde an vier Stellen in London gesichtet, die Menschen sind in Angst & Schrecken,",
        "du willst herausbekommen wo es herkommt.",
        "Dir fällt auf, dass die vier Orte ein Rechteck ergeben.",
        "Das Versteck des Monsters muss sich nahe des Mittelpunkts befinden",
        "Die Koordinaten sind: <b> " + str(rectangle)+"</b>",
        "Schreibe eine Methode",
        "die aus den Koordinaten den Mittelpunkt errechnet."
    ]
    hints = [
        "Mittelpunkt eines Rechtecks",
        "Gegeben ist ein Tuple der Form (x,y,x,y,x,y,x,y)",
        "Mögliche Formel:  $A+1/2AD$ ",
        "Erwartet wird eine List der Form [x,y]"
    ]

    return {"task_messages": task_messages, "hints": hints, "solution_function": self.solution_level1, "data": rectangle}
```

Aufruf des Tupels  
innerhalb des Textes  
als String



# LEVEL 1

## Lösung

- ▶ Methoden
- ▶ List
- ▶ Tupel
- ▶ Formeln/Algorithmen
- ▶ call
- ▶ append
- ▶ random

Aufruf der 2  
Variable „m“ als  
List

```
def solution_level1(self, rectangle):  
    m = [] # Mittelpunkt des Rechtecks  
    m.append(rectangle[0] + (2 / (rectangle[4] - rectangle[0])))  
    m.append(rectangle[1] + (2 / (rectangle[5] - rectangle[1])))  
  
    return m
```

```
def create_level1(self):  
    rectangle = random.choice([  
        (1, 2, 6, 2, 6, 6, 1, 6),  
        (3, 2, 7, 2, 7, 4, 3, 4),  
        (3, 1, 9, 1, 9, 6, 3, 6)
```

Anwendung der Formel mit  
den Werten des Tuple an  
das Ende der List (x,y)

Rückgabe von  
„m“ als List



# LEVEL 2

## “DAS KLINGELSCHILD”

Schwierigkeitsgrad - Einfach bis mittelschwer







## LEVEL 2

### Rätsel

Erstellen von zwei Stringvariablen mit dem Namen auf dem Klingelschild: einmal (doorbell) mit und einmal ohne Vokale (doorbell\_no\_vowels)

- ▶ Cast
- ▶ Listen/Arrays
- ▶ Tuple
- ▶ (Verschachtelte) Schleifen
- ▶ Libraries einbinden und anwenden

```
def create_level2(self):
    doorbell = "Dr. Viktor Frankenstein"
    doorbell_no_vowels = "Dr. VXktXr FrXnkXnstXXn"
    task_messages = [
        "Du findest an dem errechneten Ort eine verfallene Stadtvilla mit \
        einem unleserlichen Namen an der Klingel:",
        "<b>" + doorbell_no_vowels + "</b>",
        "Die Vokale sind unkenntlich. Nachdem Du geklingelt hast, antwortet eine mystische Stimme:",
        "<em>\"Wer klingelt an einer Klingel, ohne den Namen lesen zu können?\"</em> fragte die Stimme und fuhr nach einer kurzen Pause",
        "<em>\"Vielleicht öffne ich Dir die Tür. Doch dafür musst Du etwas tun. Erstelle eine Liste \
        mit allen möglichen Kombinationen. Gehe alphabetisch vor! Wie Du siehst fehlen \
        ja nur Vokale...aber es wird trotzdem eine lange Liste...</em>",
        "<em><b>...HARHARHAR!!...</b></em>",
        "<em>Wenn sie vollständig ist und der richtige Name an der gleichen Position wie auf meiner Liste steht, verrate ich ihn dir..."
    ]
    hints = [
        "Oje, so viele Kombinationen... probiere es trotzdem! Ersetze die X jeweils mit den \
        bekannten Vokalen \"a,e,i,o,u\"",
        "Mit jedem Einsetzen erhältst Du eine neue Kombination. Diese Kombination solltest Du \
        jeweils deiner Liste hinzufügen. ",
        "Irgendwann triffst Du auf die richtige Kombination der Vokale. An welcher Stelle steht diese Kombination?"
    ]
    return {"task_messages": task_messages, "hints": hints, "solution_function": self.get_combinations_and_position, "data": doorbell}
```

Ausgabe der Stringvariable ohne Vokale (soll das mystische Klingelschild repräsentieren)

Übergeben der Stringvariable mit dem vollständigen Namen an die Lösungsfunktion



## LEVEL 2

## Lösung

Erstellen:

- ▶ Array mit allen Vokalen in alphabetischer Reihenfolge
- ▶ Leeres Array für die Vokale in der richtigen Reihenfolge wie sie im Namen vorkommen

- ▶ Cast
- ▶ Listen/Arrays
- ▶ Tuple
- ▶ (Verschachtelte) Schleifen
- ▶ Libraries einbinden und anwenden

```
###Level 2###
def get_combinations_and_position(self, doorbell):
    vowels_alphabetical_order = ["a","e","i","o","u"]
    doorbell_solution_vowels = []
    ##Extract vowels|
    for character in doorbell:
        ##If vowel add to vowel list
        if character in vowels_alphabetical_order:
            doorbell_solution_vowels.append(character)
    ##Using itertools.product to generate a list of all possible combinations
    combination_list = list(product(vowels_alphabetical_order,repeat=len(doorbell_solution_vowels)))
    position_counter = 1
    ##Now finding the position of the correct vowel order/combination
    for c in combination_list:
        position_counter = position_counter + 1
        ##Parse as tuple because the the combinations are saved as tuple in list
        if tuple(doorbell_solution_vowels) == c:
            ##Print to check the result in console
            print(position_counter)
            return position_counter
    ###END Solution Level 2
```

Extrahieren der Vokale aus  
vollst. Namen (for-Schleife)

Alle Kombinationen  
(list(product...))erstellen und in Liste  
(combinations\_list) überführen

Die Liste durchiterieren (for-Schleife) und die  
Position (position\_counter), an der die richtige  
Kombination (aus Schritt 2) steht, ausgeben



# LEVEL 3

## “Die Metropolitan Police”

Schwierigkeitsgrad - Mittelschwer





## Rätsel & Lösung

Input: Random Name

Anforderungen:

- ▶ Startet mit den Initialen
- ▶ Quersumme des Dreier-Blocks zwischen 9 & 15
- ▶ Keine Ziffer doppelt innerhalb des Dreier-Blocks

Output: Badge Number

Format: XX-xxx-xxx-xxx

Beispiel: AC-425-428-263

- ▶ Nested functions
- ▶ Statements
- ▶ Use range
- ▶ String indexing & slicing
- ▶ Use random.seed()
- ▶ Use random.randint()
- ▶ Use join
- ▶ List comprehension

```
241 def create_badge_number(self, name):
242     """
243     Create a badge number with format XX-xxx-xxx-xxx.
244     Numbers are random from a given seed.
245     Requirements for the first block: Initials of the badge holders name.
246     Requirements per block of 3: cross sum is between 9 and 15, the digits are unique.
247     """
248     def get_initials(name):
249         initials = []
250         names = name.split()
251
252         for i in range(0,2):
253             initials.append(names[i][0])
254
255         return initials
256
257     initials = get_initials(name)
258
259     def create_seed(initials):
260         seed_digits = [ord(i) for i in str(initials)]
261
262         return sum(seed_digits)
263
264     seed = create_seed(initials)
265
266     def unique_digits(a):
267         digits = [int(i) for i in str(a)]
268
269         if digits[0] == digits[1]:
270             return False
271
272         elif digits[0] == digits[2]:
273             return False
274
275         elif digits[1] == digits[0]:
276             return False
277
278         else:
279             return True
280
281
282
```

```
284 def cross_sum(a):
285
286     cross_sum_digits = [int(i) for i in str(a)]
287
288     return sum(cross_sum_digits)
289
290 random.seed(seed)
291
292 numbers = []
293
294 while len(numbers) < 3:
295
296     x = random.randint(seed,988)
297
298     if unique_digits(x) == True:
299
300         if cross_sum(x) >= 9 and cross_sum(x) <= 15:
301
302             numbers.append(x)
303
304     else:
305         continue
306
307 badge_initials = ''.join(get_initials(name))
308 badge_nums = '-'.join(map(str, numbers))
309
310 badge_number = badge_initials + '-' + badge_nums
311
312 return badge_number
```



# LEVEL 4

## “DIE ASERVATENKAMMER”

Schwierigkeitsgrad - Mittelschwer bis Schwer





# LEVEL 4

## Rätsel & Lösung

- ▶ Use `itertools.permutations()`
- ▶ Use `list(string.ascii_uppercase)` → Liste des Alphabets
- ▶ List comprehension

```
def create_level4(self):  
    def create_random_code():  
        alphabet = list(string.ascii_uppercase)  
        code_numbers = [item + '1' for item in alphabet] + [item + '2' for item in alphabet]  
        random_code = []  
        for _ in range(0, 6):  
            random_code.append(random.choice(code_numbers))  
        return random_code
```

Erstellt jedes Mal eine  
Random Safe Kombination

Input: Random Safe Kombination,  
verschlüsselt

```
1  from itertools import permutations  
2  import string  
3  
4  def run(code):  
5      '''  
6          find the right 3 keys red, blue, yellow  
7          decode numbers from the note  
8          '''  
9      # Try out all combinations of the keys  
10  
11     def key_combinations():  
12  
13         colours = ['red', 'yellow', 'blue']  
14         all_possible_combinations = []  
15  
16         p = permutations(colours)  
17  
18         for p in list(p):  
19             all_possible_combinations.append(p)  
20  
21         print(all_possible_combinations)  
22         return all_possible_combinations  
23
```

3 Schlüssel → Alle Kombinationen

`list(string.ascii_uppercase)` → Liste des Alphabets erstellen  
& dann mit Zahlen kombinieren

```
26  # Decode the numbers to open the safe, 1-52 (A1 - Z2)  
27  
28  def decoding_numbers(code):  
29  
30      # Create the list of the uppercase letters of the alphabet with a 1 and then a 2 added  
31  
32      alphabet = list(string.ascii_uppercase)  
33  
34      key = [item + '1' for item in alphabet] + [item + '2' for item in alphabet]  
35  
36      decoded_numbers = []  
37  
38      for item in code:  
39          num = key.index(item)  
40  
41          decoded_numbers.append(num+1)  
42  
43      print(decoded_numbers)  
44      return decoded_numbers  
45  
46
```

Position in der Liste + 1 ist die Zahl für die Safe-Kombination.

Einfachere Alternative: Manuell ein Dictionary erstellen und die keys nutzen um den Random Code zu entschlüsseln.



# LEVEL 5

## “DAS BILD”

Schwierigkeitsgrad - Schwer





## Rätsel

Erstellen:

- ▶ Aktuelles Datum in TTMMJJJJ-Format
- ▶ Array, welches die jeweiligen Zahlen des Datums enthält
- ▶ Array mit den 8 Zeilen des Bildes

- ▶ Cast
- ▶ Listen/Arrays
- ▶ Tuple
- ▶ Schleifen

```
def create_level5(self):
    d = datetime.datetime.now()
    d_num_raw = d.strftime("%d" "%m" "%Y")
    d_num = list(map(int, str(d_num_raw)))

    static_chars = [
        "|-^^--^--^--^|",
        "|*==#==*|",
        "|" + "&nbsp;" * 17 + "|",
        "|< ⊗ : ⊗ >|",
        "|□--" + "&nbsp;" * 9 + "□" + "&nbsp;" * 9 + "--□|",
        "|" + "&nbsp;" * 14 + "|",
        "| □□□ |",
        "|_____|",
    ]

    task_messages = [
        "." * d_num[0] + "." * 6 + static_chars[0],
        "." * d_num[1] + "." * 6 + static_chars[1],
        "." * d_num[2] + "." * 6 + static_chars[2],
        "." * d_num[3] + "." * 6 + static_chars[3],
        "." * d_num[4] + static_chars[4],
        "." * d_num[5] + "." * 7 + static_chars[5],
        "." * d_num[6] + "." * 7 + static_chars[6],
        "." * d_num[7] + "." * 7 + static_chars[7],
        "&nbsp;",
        str(d_num_raw),
        "&nbsp;",
        "Was ist das für ein seltsames Bild? Sieht irgendwie aus, als wäre es zeilenweise verschoben...",
        "Und was hat diese Zahl in Reihe 10 zu bedeuten? Scheinbar ändert sie sich jeden Tag etwas... ",
        "Es sind 8 Zahlen und das Bild besteht aus 8 Reihen. Wie kannst Du es wieder richtig zusammensetzen?",
    ]

    hints = [
        "Wie hängt die 8-stellige Zahl mit den 8 Zeilen des Bildes zusammen? Um wieviele Positionen musst\
        Du die Zeilen jeweils verschieben, so dass das Bild wieder richtig zusammengesetzt ist?",
        "\"Verschieben\" heisst in dem Fall, dass Du etwas entfernen solltest. Gebe am Ende nur die 8 korrekt\
        ausgerichteten Zeilen aus.",
    ]

    return {"task_messages": task_messages, "hints": hints, "solution_function": self.realign_picture, "data": task_messages}
```

Überführen des Arrays mit dem Bild in die Task Messages und hinzufügen von Punkten abhängig von der Zahl je Element im Datums-Array → Verschieben wird „simuliert“

Task-Messages Array übergeben an die Lösungsfunktion





## Lösung

Erstellen:

- ▶ Array mit den Elementen 0-10, um das Bild und das Datum zu haben
- ▶ Leeres Array für die korrigierten Zeilen des Bildes
- ▶ Array, in dem das Datum überführt wird

- ▶ Cast
- ▶ Listen/Arrays
- ▶ Tuple
- ▶ Schleifen

```
###Level 5###
def realign_picture(self, task_messages):
    ###Put the misaligned picture into fresh array (only first 10 lines)
    picture_lines = task_messages[0:10]
    ###Create empty array for the realigned lines
    aligned_picture_lines = []
    ###Convert the number in line 9 to a list of numbers using "map"
    date_from_picture = list(map(int, str(picture_lines[9])))
    ###Iterate through the misaligned lines by using enumerate in order to match the index to Number index of line 9 (date)
    for i, elem in enumerate(picture_lines[0:8]):
        ###Convert each line to list to make sure single elements (characters) can be removed
        picture_line = list(picture_lines[i])
        ###Removing elements depending on the number which is given in number list from line 9
        del picture_line[:date_from_picture[i]]
        ###Add the aligned line to the realigned lines array after converting it back to string
        picture_line = ''.join(picture_line)
        aligned_picture_lines.append(str(picture_line))

    return aligned_picture_lines
###End solution Level 5###
```

Die ersten 8 Elemente des Arrays mit den verschobenen Linien durchgehen und je Element den Inhalt in eine Liste konvertieren und in Variable packen

Elemente Löschen (del) → so viele, wie in den jeweiligen Datums-Element (date\_from\_picture[i]) angeben

Zurückkonvertieren in String (.join) und erneute in Variable (picture\_line) packen

Zurückkonvertieren String dem Array mit den korrigieren Zeilen hinzufügen (append)



# Anpassung Notification und Ergebnisausgabe in escape.js

## Deine Lösung ist:

```
.....| - ^ ^ - - ^ - ^ - ^ |
.....| * = # # # = = * |
.....|
.....| < ⊗ : ⊗ > |
| □ --      □      -- □ |
.....|
.....| □ □ □ |
.....| _____ |
```

Juhu, das war richtig!

```
function notify(message, fadeout = true) {
  if (fadeout) {
    $("#message").html(message).fadeOut(500, 1).delay(3000).fadeOut(500, 0);
  }
  else {
    $("#message").html(message).fadeOut(500, 1);
  }
}
```

Ausgabe der Notification-Message in HTML zur korrekten Darstellung des Bildes → `.text(message)` zu `.html(message)`

```
function show_result(result) {
  var html_msg = "Deine Lösung ist:<br>"
  if (result.solution.constructor.name == "Array") {
    result.solution.forEach(function(m, index){
      html_msg = html_msg + m + "<br>";
    });
  }
  else {
    html_msg = html_msg + result.solution + "<br>"
  }

  if (result.correct) {
    notify(html_msg + "Juhu, das war richtig!");
    next_level();
  }
  else {
    notify(html_msg + "Das ist leider falsch.");
  }
}
```

Notification-Message mit html (`<br>`) versehen (var `html_msg`)

Verarbeitung von Arrays sicherstellen, d. h. wenn die zurückgegebene Lösung ein Array ist (if `... == Array`) sollen die Inhalte je Element an die Notification-Message angefügt werden (forEach)

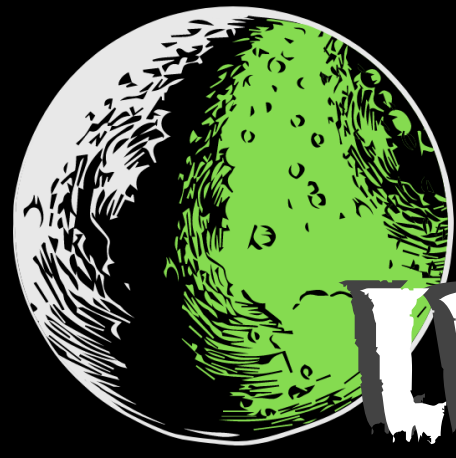


# LEVEL 6

“DER CODE”

Schwierigkeitsgrad - Schwer





## LEVEL 6

## Rätsel

- ▶ Methoden
- ▶ Modulo-Operator
- ▶ Euklidischer Algorithmus
- ▶ For-Schleife

```
def create_level6(self):
    task_messages = [
        "Du verstehst wie die Maschine funktioniert und stellst mit Grauen fest,",
        "dass Dr. Frankenstein hier das Monster erschaffen hat, dass in ganz London",
        "Angst und Schrecken verbreitet.",
        "Auf einem wüsten Schreibtisch findest du ein Notizbuch mit Einträgen,",
        "der letzte wurde vor einer Woche geschrieben,",
        "du kannst es aber nicht lesen, es scheint nur Buchstabensalat zu sein.",
        "Wenn du es nur entziffern könntest..."
    ]
    hints = [
        "Nutze den Code aus dem Safe um die Notiz zu entschlüsseln",
        "Schlüsselpaar",
        "inverses Element",
        "Euklidischer Algorithmus"
    ]

    return {"task_messages": task_messages, "hints": hints, "solution_function": self.solution_level6}
```





# LEVEL 6

## Lösung

- Methoden
- Modulo-Operator
- Euklidischer Algorithmus
- For-Schleife

Aufrufen der  
verschiedenen  
Funktionen

Euklidischer  
Algorithmus (Inverse)

Entschlüsselung (nur kleine  
Buchstaben und Sonderzeichen  
werden angehängen)

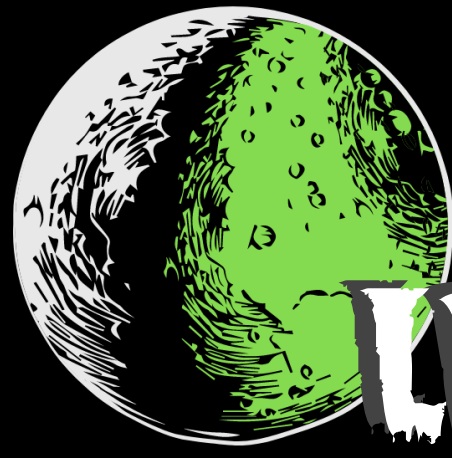
```
1  a = 25
2  b = 13
3  msg_crypt = "kju etahj cimu rfjkju. knv uzafdtb gnu njgzcija, vjfaja nhhwjvvsja kwnah ot cfakjwa, nmju jw rfwk sza kja jwfaajwtahja na vjfaj sjwhna"
4
5
6  def run(msg_encr, a, b):
7      def decrypt(y, a_inv, b):
8          return ((a_inv*(y - b)) % 26)
9
10     def euklid(a, m):
11         def ggt_euklid(a, b):
12             if a == 0:
13                 return (b, 0, 1)
14             else:
15                 ggt, y, x = ggt_euklid(b % a, a)
16                 return (ggt, x - (b // a) * y, y)
17
18         ggt, x, y = ggt_euklid(a, m)
19         if ggt != 1:
20             # Laufzeitfehler
21             raise Exception("Es gibt kein inverses zu a modulo m")
22         else:
23             return x % m
24
25     alphabet = "abcdefghijklmnopqrstuvwxyz"
26     msg = ""
27     for y in msg_encr:
28         if alphabet.find(y) == -1:
29             msg = msg + y
30         else:
31             a_inv = euklid(a, 26)
32             x = decrypt(alphabet.find(y), a_inv, b)
```



# Standalone LVL 6 –Part 1 Entschlüsselung

```
1  a = 25
2  b = 13
3  msg_crypt = "rjaa kt jv mfv gfjwgjw hjvlgniuu gnvu, mfvu kt tavjwj jfaofhj lgna!j czakza szw kjb bzavujw ot wjuuja!"
4
5
6  def run(msg_encr, a, b):
7      def decrypt(y, a_inv, b):
8          return ((a_inv*(y - b)) % 26)
9
10     def euklid(a, m):
11         def ggt_euklid(a, b):
12             if a == 0:
13                 return (b, 0, 1)
14             else:
15                 ggt, y, x = ggt_euklid(b % a, a)
16                 return (ggt, x - (b // a) * y, y)
17
18         ggt, x, y = ggt_euklid(a, m)
19         if ggt != 1:
20             raise Exception("Es gibt kein inverses zu a modulo m")
21         else:
22             return x % m
23
24     alphabet = "abcdefghijklmnopqrstuvwxyz"
25     msg = ""
26     for y in msg_encr:
27         if alphabet.find(y) == -1:
28             msg = msg + y
29         else:
30             a_inv = euklid(a, 26)
31             x = decrypt(alphabet.find(y), a_inv, b)
32             msg = msg + alphabet[x]
33
34     return msg
35
36 print(run(msg_crypt, a, b))
37
```





# LEVEL 6

## Lösung

```
1  def crypt(x, a, b):
2      return ((a*x+b) % 26)
3
4
5  msg = "wenn du es bis hierher geschafft hast, bist du unsere einzige chance london vor dem monster zu retten!"
6  msg_crypt = ""
7  a = 25
8  b = 13
9
10 alphabet = "abcdefghijklmnopqrstuvwxyz"
11
12 for x in msg:
13     if alphabet.find(x) == -1:
14         msg_crypt = msg_crypt + x
15     else:
16         y = crypt(alphabet.find(x), a, b)
17
18         msg_crypt = msg_crypt + alphabet[y]
19
20
21 print(msg_crypt)
22
```

- ▶ Methode zum Verschlüsseln Buchstaben Modulo-Operator
- ▶ Keys/Schlüsselpaar
- ▶ Check ob Alphabet-Zeichen
- ▶ Kein Alphabet-Zeichen, dann übernehme einfach
- ▶ Verschlüsselung
- ▶ Setze verschlüsselte Nachricht zusammen
- ▶ Ausgabe