DS 13 – Teamprojekt: Verschlüsselung mit der Caesar-Chiffre (Teil 1)

Informatik 10 – MINT-Zweig | Projektarbeit

Lernziele

Ich kann

- das Verfahren der Caesar-Chiffre beschreiben und von Hand anwenden,
- die Position eines Buchstabens im Alphabet bestimmen und mit `% 26` am Ende des Alphabets vorne weiterzählen,
- Funktionen zum Verschlüsseln eines Buchstabens und eines ganzen Textes implementieren und systematisch testen,
- ein Programmierprojekt im Team planen, Teilaufgaben verteilen und die Ergebnisse zusammenführen.

🗣️ Sprachhilfe

- „Der Buchstabe … hat im Alphabet die Position … . Mit dem Schlüssel … ergibt sich die Position …, also der Buchstabe … .“
- „Am Ende des Alphabets rechnen wir `% 26`, damit nach Z wieder A kommt: … % 26 ergibt … .“
- „Die Funktion … bekommt die Parameter … und gibt … zurück. Das ist ihre Schnittstelle.“
- „Wir haben mit dem Randfall … getestet, weil dort am ehesten ein Fehler steckt.“

1 Wie die Caesar-Chiffre funktioniert

Wie verschickt man eine Nachricht so, dass nur Eingeweihte sie lesen können? Julius Caesar soll ein einfaches Verfahren benutzt haben: Jeder Buchstabe wird im Alphabet um eine feste Anzahl von Stellen nach hinten verschoben. Diese Anzahl ist der **Schlüssel**. Am Ende des Alphabets geht es vorne weiter.

Schlüssel 3:

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

So wird aus dem **Klartext** `HALLO` der **Geheimtext** `KDOOR`: H → K, A → D, L → O, L → O, O → R. Wer den Schlüssel kennt, entschlüsselt, indem er jeden Buchstaben um 3 Stellen **zurück** verschiebt.

> 📌 **Definition:** Beim **Verschlüsseln** wird ein lesbarer **Klartext** mithilfe eines **Schlüssels** in einen **Geheimtext** umgewandelt. Beim **Entschlüsseln** wird aus dem Geheimtext mit dem Schlüssel wieder der Klartext.

✏️ Übung 1 – Von Hand verschlüsseln

Benutze die Tabelle oben.

1. Verschlüssele das Wort `INFORMATIK`.
2. Entschlüssele den Geheimtext `ZDVVHU`.
3. Beschreibe in einem Satz, wie du beim Entschlüsseln vorgegangen bist.

I	N	F	O	R	M	A	T	I	K

Meine Antworten zu 2 und 3:

...

2 Das Projekt im Team planen

Professionelle Softwareteams tippen nicht alle gleichzeitig drauflos. Sie gehen in Schritten vor:

1. **Verstehen und planen:** Was soll das Programm können? Wie löst man das Problem erst einmal ohne Computer?
2. **Zerlegen und verteilen:** Für jede Teilaufgabe wird eine Funktion geplant, jede Person übernimmt eine.
3. **Umsetzen und testen:** Jede Funktion wird mit vorher festgelegten Beispielen getestet.
4. **Zusammenführen:** Die Funktionen werden zu einem Programm zusammengesetzt und gemeinsam getestet.
5. **Dokumentieren und vorstellen.**

Damit ihr gleichzeitig arbeiten könnt, legt ihr **vorher** fest, wie jede Funktion heißt, welche Parameter sie bekommt und was sie zurückgibt.

> 📌 **Definition:** Die **Schnittstelle** einer Funktion legt fest, wie sie heißt, welche Parameter sie erwartet und welchen Rückgabewert sie liefert. Wer die Funktion benutzt, muss nur die Schnittstelle kennen.

Teil	Funktion mit Parametern	Rückgabewert
A	`verschluessle_buchstabe(buchstabe, schluessel)`	der verschobene Buchstabe
B	`verschluessle_text(text, schluessel)`	der Geheimtext
C	Hauptprogramm (keine Funktion)	– (fragt ab und gibt aus)

B benutzt A. Trotzdem kann man B schon programmieren, bevor A fertig ist: Man kennt ja die Schnittstelle von A.

✏️ Übung 2 – Projektplan

Plant gemeinsam auf Papier, **bevor** ihr programmiert:

1. Wie ermittelt man zu einem Buchstaben seine Position im Alphabet?
2. Wie kommt man von der Position zurück zum (verschobenen) Buchstaben?
3. Was passiert am Ende des Alphabets, z. B. bei `Z` mit dem Schlüssel 3?

Verteilt dann die Teilaufgaben. Niemand testet die eigene Funktion allein.

Teilaufgabe	programmiert von	getestet von	fertig?
A: einen Buchstaben verschlüsseln			
B: einen Text verschlüsseln			
C: interaktives Programm			

Unsere Antworten zu den Fragen 1 bis 3:

...

3 Werkzeuge für das Projekt

Das Alphabet als Zeichenkette

Den Index eines Zeichens findet ihr mit `.index()`, das kennt ihr von den Listen.

```python
alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
print(alphabet)
print("Position von C:", alphabet.index("C"))
print("Buchstabe an Position 5:", alphabet[5])
```

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
Position von C: 2
Buchstabe an Position 5: F
```

`alphabet.index("C")` geht vom **Zeichen** zur **Position** (2), `alphabet[5]` von der **Position** zum **Zeichen** (F).

⚠️ Stolperstein: Zeichen, die nicht im Alphabet stehen

Führe die Zelle aus.

```python
alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
print(alphabet.index("a"))
```

```
Traceback (most recent call last):
  File "<Zelle 2>", line 2, in <module>
ValueError: substring not found
```

`ValueError: substring not found` heißt „Teilzeichenkette nicht gefunden“. Prüft deshalb vorher mit `zeichen in alphabet`, ob das Zeichen ein Großbuchstabe ist.

Am Ende des Alphabets: der Rest-Operator

X hat die Position 23, um 3 verschoben ergibt das 26. Die gibt es nicht. Nach Z soll es wieder bei A weitergehen, wie auf einer Uhr nach 12 wieder die 1 kommt. Das leistet der **Rest-Operator** `%`: Für Zahlen von 0 bis 25 ändert sich nichts, aus 26 wird 0, aus 27 wird 1.

🔮 **Vorhersage:** Welche vier Zeilen gibt die Schleife aus?

```python
for position in range(22, 26):
    neue_position = (position + 3) % 26
    print(position, "->", neue_position)
```

```
22 -> 25
23 -> 0
24 -> 1
25 -> 2
```

W (22) wird zu Z (25), X (23) zu A (0), Y zu B, Z zu C. Die Klammern sind wichtig: erst addieren, dann den Rest bilden.

> 📌 **Merksatz:** Mit `(position + schluessel) % 26` erhält man immer eine gültige Position von 0 bis 25. Am Ende des Alphabets geht es dadurch vorne weiter.

> 🧰 **Zweiter Weg (freiwillig):** Der Computer speichert jedes Zeichen als Zahl. `ord("A")` ist 65, `chr(68)` ist `"D"`. Die Position im Alphabet ist `ord(buchstabe) - ord("A")`, zurück geht es mit `chr(position + ord("A"))`. Entscheidet euch im Team für **einen** Weg.

4 Teilaufgabe A – einen Buchstaben verschlüsseln (Lücken füllen)

Die Teilaufgaben werden schrittweise selbstständiger: A ist ein Lückencode, B hat ein Gerüst aus Kommentaren, C setzt ihr allein zusammen.

Schreibt eine Funktion `verschluessle_buchstabe(buchstabe, schluessel)`, die **einen** Großbuchstaben um `schluessel` Stellen verschiebt und den neuen Buchstaben zurückgibt. Die drei Schritte kennt ihr aus Übung 2 (Projektplan): Position finden, verschieben und mit `% 26` umlaufen lassen, Buchstaben ablesen. Ersetzt jedes `___`.

> 💡 **Tipp:** Gute Tests prüfen nicht nur den **Normalfall** (A → D), sondern auch die **Randfälle**: das Ende des Alphabets (X, Y, Z) und besondere Schlüssel wie 0 oder 25.

Aufruf	erwartet	tatsächlich	✓ oder ✗
`verschluessle_buchstabe("A", 3)`	`D`		
`verschluessle_buchstabe("X", 3)`	`A`		
`verschluessle_buchstabe("Z", 1)`	`A`		
`verschluessle_buchstabe("M", 0)`	`M`		
`verschluessle_buchstabe("B", 25)`	`A`		

Lückencode

```
alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

def verschluessle_buchstabe(buchstabe, schluessel):
    position = alphabet.index(___)
    neue_position = (___ + ___) % 26
    return alphabet[___]

print(verschluessle_buchstabe("A", 3))    # erwartet: D
print(verschluessle_buchstabe("X", 3))    # erwartet: A
print(verschluessle_buchstabe("Z", 1))    # erwartet: A
print(verschluessle_buchstabe("M", 0))    # erwartet: M
print(verschluessle_buchstabe("B", 25))   # erwartet: A
```

5 Teilaufgabe B – einen ganzen Text verschlüsseln (mit Gerüst)

Ihr braucht das Muster *Einen neuen Text aufbauen* aus DS 10. Es funktioniert wie die Summenbildung, nur mit Text: Man beginnt mit `""` und hängt in der Schleife mit `+` Zeichen an.

```
text = "ABC"
ergebnis = ""
for zeichen in text:
    ergebnis = ergebnis + zeichen + zeichen
print(ergebnis)
```

```
AABBCC
```

1. Schreibt eine Funktion `verschluessle_text(text, schluessel)`, die in einer Schleife jeden Buchstaben mit `verschluessle_buchstabe` verschlüsselt und den Geheimtext zurückgibt. Kontrolle: `verschluessle_text("HALLO", 3)` ergibt `KDOOR`.
2. **Zusatz-Herausforderung:** Was passiert bei `"Hallo Welt!"`? Überlegt euch im Team eine Lösung, z. B. Leerzeichen und Satzzeichen unverändert lassen und den Text vorher mit `.upper()` umwandeln. Kontrolle: `KDOOR ZHOW!`.

Die Zelle benutzt eure Funktion aus Teilaufgabe A. Führt die Zelle oben deshalb zuerst aus.

```python
def verschluessle_text(text, schluessel):
    # ergebnis beginnt als leere Zeichenkette
    # für jedes Zeichen im Text:
    #     das Zeichen mit verschluessle_buchstabe verschlüsseln
    #     und an ergebnis anhängen
    # ergebnis zurückgeben
    pass

print(verschluessle_text("HALLO", 3), "erwartet: KDOOR")
# Zusatz-Herausforderung:
# print(verschluessle_text("Hallo Welt!", 3), "erwartet: KDOOR ZHOW!")
```

```
None erwartet: KDOOR
```

6 Teilaufgabe C – das interaktive Programm (selbstständig)

Setzt beide Funktionen zu einem vollständigen Programm zusammen, in dieser Reihenfolge: Alphabet, Funktion A, Funktion B, Hauptprogramm. Das Hauptprogramm fragt Klartext und Schlüssel mit `input()` ab und gibt den Geheimtext aus. Testet mit mindestens drei Eingaben. Kontrolle: „Treffen um acht“ mit Schlüssel 5 ergibt `YWJKKJS ZR FHMY`.

Schreibt an den Anfang einen **Kommentar** mit dem Programmnamen, eurem Team und einem Satz dazu, was das Programm tut.

```python
# Caesar-Chiffre: Texte verschlüsseln
# Team:
# Was das Programm tut:

# TODO: Teilaufgabe C – Alphabet, Funktion A, Funktion B, Hauptprogramm
```

📒 Projekttagebuch

Tragt am Ende der Arbeitsphase ein, was ihr geschafft habt, welche Probleme aufgetreten sind und was als Nächstes kommt. Ihr braucht es in DS 14.

Datum	erledigt	Probleme und wie wir sie gelöst haben	nächster Schritt

📌 Das Wichtigste in Kürze

- Die **Caesar-Chiffre** verschiebt jeden Buchstaben um eine feste Anzahl von Stellen (den **Schlüssel**) im Alphabet. Aus dem **Klartext** wird der **Geheimtext**.
- Die Position eines Buchstabens ist sein Index in `"ABCDEFGHIJKLMNOPQRSTUVWXYZ"` (oder `ord(buchstabe) - ord("A")`).
- `(position + schluessel) % 26` sorgt dafür, dass es nach Z wieder bei A weitergeht.
- Einen neuen Text baut man auf, indem man mit `""` beginnt und in einer Schleife Zeichen anhängt.
- Im Team zerlegt man das Problem in Teilaufgaben, legt vorher die **Schnittstellen** fest, testet jede Funktion mit Normal- und Randfällen und führt die Teile am Ende zusammen.

> **In DS 14** entwickelt ihr die passende **Entschlüsselung** und recherchiert Berufsfelder rund um die Softwareentwicklung. Zum Weiterüben: `DS13_Weitere_Uebungen.ipynb`.