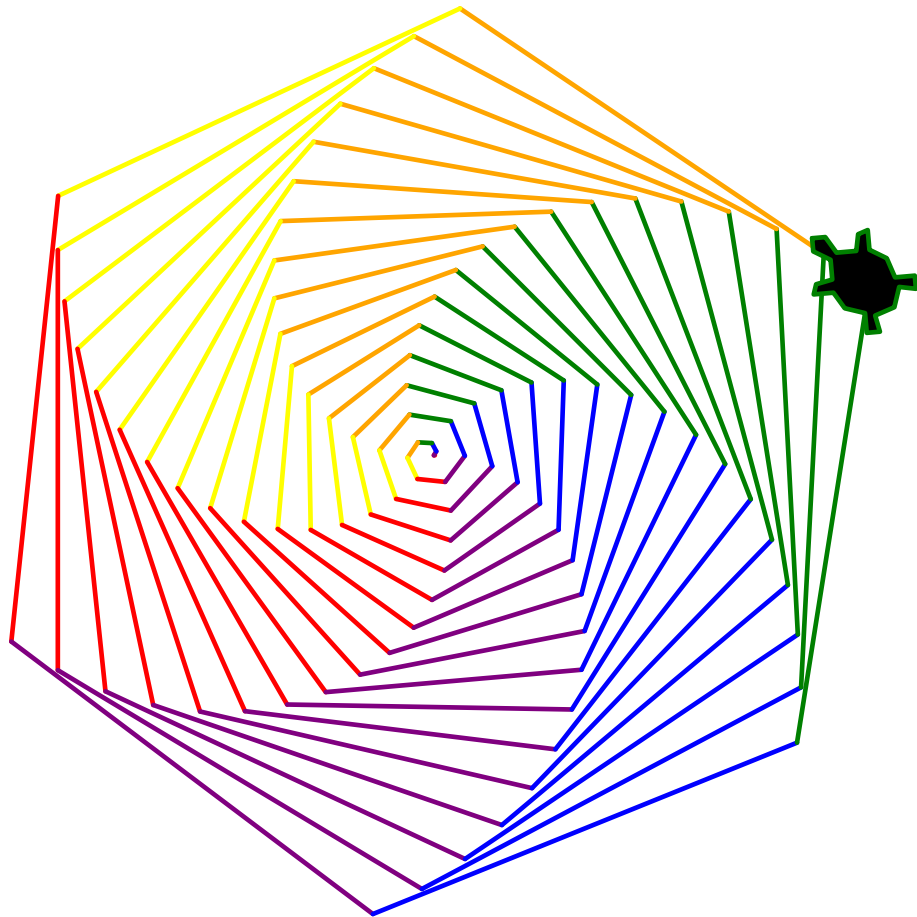


Informatik Klasse 7



Unterrichtsskript

2024

Author	Lukas Meyer-Hilberg
Titel	Informatik Klasse 7
Stand	26.07.2024
Version	2024-07-26v9
ID	2d270f7f2c2f2cf48c0d6c16563110bb56ec825a
Bundesland	Baden-Württemberg
Schulform	Gymnasium
Klasse	7
Lizenz	CC BY-NC-SA 4.0
Website	https://s.hilberg.eu
Direktlink	https://s.hilberg.eu/2024-07-26v9



Dieses Werk ist lizenziert unter einer Creative Commons “Namensnennung – Nicht-kommerziell – Weitergabe unter gleichen Bedingungen 4.0 International” Lizenz.

<https://creativecommons.org/licenses/by-nc-sa/4.0/deed.de>

Sollte trotz sorgfältiger Prüfung in diesem Werk ein Inhalt gefunden werden, dessen Lizenz nicht zu den genannten Punkten der Lizenz passt, eine Quellenangabe oder eine Namensnennung vergessen worden sein, so bittet der Autor über eine kurze Nachricht an code@hilberg.eu, um diesen Mangel schnellstmöglich zu beheben.

Code des Titelbilds:^[12]

Vielen Dank an








Thomas Claus für die Rückmeldung im Unterrichtseinsatz und konstruktive Verbesserungsvorschläge.

Inhaltsverzeichnis

1	Daten und Codierung	5
1.1	Textcodierung und Binärcode	5
1.2	Bildcodierung	6
2	Algorithmen und Programmierung	8
2.1	Variablen in einem Algorithmus	8
2.2	Den Wert von Variablen vergleichen	10
2.3	Visuelles Programmieren und textuelles Programmieren	12
2.4	Mit <code>for</code> - und <code>while</code> -Schleifen Programmcode vereinfachen	13
2.5	Mit Bedingungen und Verzweigungen entscheiden	16
2.6	Mit Schleifen, Bedingungen und Variablen programmieren	19
2.7	Mit Struktogrammen Algorithmen entwerfen	24
3	Informationsgesellschaft und Datensicherheit^[7]	26
3.1	Cäsar-Verschlüsselung ^[7]	26
3.2	Cäsar-Verschlüsselung knacken	27
3.2.1	Cäsar-Verschlüsselung mit einer Häufigkeitsanalyse knacken	27
3.2.1	Cäsar-Verschlüsselung mit der Brute-Force-Methode knacken	29
3.3	Monoalphabetische Verschlüsselung	30

Was bedeuten die Symbole?

In diesem Skript lernst du kennen, wie mit Hilfe von Algorithmen programmiert werden kann. Dabei arbeitest du mit Online-Inhalten die teilweise auch interaktiv bedient werden können. In Tabelle 1 auf Seite 3 siehst du alle Links aufgelistet, die du im Verlauf von dieser Einheit brauchst.

-  Wenn du dieses Symbol siehst, dann gibt es einen Inhalt, den du über einen Link aus Tabelle 1 aufrufen kannst.
-  Wenn du mit der vorangegangenen Aufgabe gut klargekommen bist, dann kannst du diese Aufgabe im Expertenmodus probieren.
-  Unter diesen Links findest du weiteres Infomaterial.
-  Hier sollst du im Heft/Ordner bearbeiten und dokumentieren.
-  Denke daran deinen Arbeitsfortschritt selber regelmäßig zu speichern und abzugeben. **Datenverlust zählt als „nicht erledigt“!**
-  Auf Seite 4 wird der Python-Arbeitsbereich beschrieben, den wir in diesem Skript nutzen. Wie du den Arbeitsbereich aufrufst, siehst du in Tabelle 1.
-  Ein zweiter Python-Arbeitsbereich steht zur Verfügung: JupyterLite umfasst weniger Funktionen, lässt sich jedoch **schneller starten**. Wie du den Arbeitsbereich aufrufst, siehst du in Tabelle 1. Hier sind game-Notebooks von jupyterlite^[11] enthalten.

Welche Links rufe ich auf?

Die Links aus der folgenden Tabelle sind unter der entsprechenden Version unter <https://s.hilberg.eu> abrufbar.

Tabelle 1: Übersicht der Materialquellen, abrufbar unter <https://s.hilberg.eu/2024-07-26v9/links> .

Verweis	Beschreibung
s1	Scratch Editor
s2	Blockly Editor
s3	Blockly Editor Demo
s4	Blockly Game
s5	Chiffrierscheibe Cäsar-Verschlüsselung
s6	Buchstabenhäufigkeit
v1	Erklärvideo Algorithmus
lab	Python Lab
lablite	PythonLite Lab
c0	Textcodierung
c1	Bildcodierung
c2	Algorithmus
p0	Intro Python Notebooks
p1	Python Turtle
p2	Variablen
p3	Bedingungen
p4	Schildkrötenterrarium
p5	Vokabeln
k1	Cäsar-Verschlüsselung
k2	Cäsar-Verschlüsselung knacken
k3	Monoalphabetische Verschlüsselung
b	Programmier-Bausteine

Arbeiten mit Python-Notebooks

Deinen *Python-Arbeitsbereich* rufst du im Browser auf. Bei jedem neuen Öffnen wird dein Arbeitsfortschritt gegebenenfalls zurückgesetzt - denke daran deine Arbeit regelmäßig **herunterzuladen**.

Du arbeitest mit **Python-Notebooks** die in Zellen aufgebaut sind. In jeder Zelle kannst du entweder einfach nur Text (Beschreibungen, Erklärungen, Bilder, ...) speichern oder Programmcode der ausgeführt werden kann.

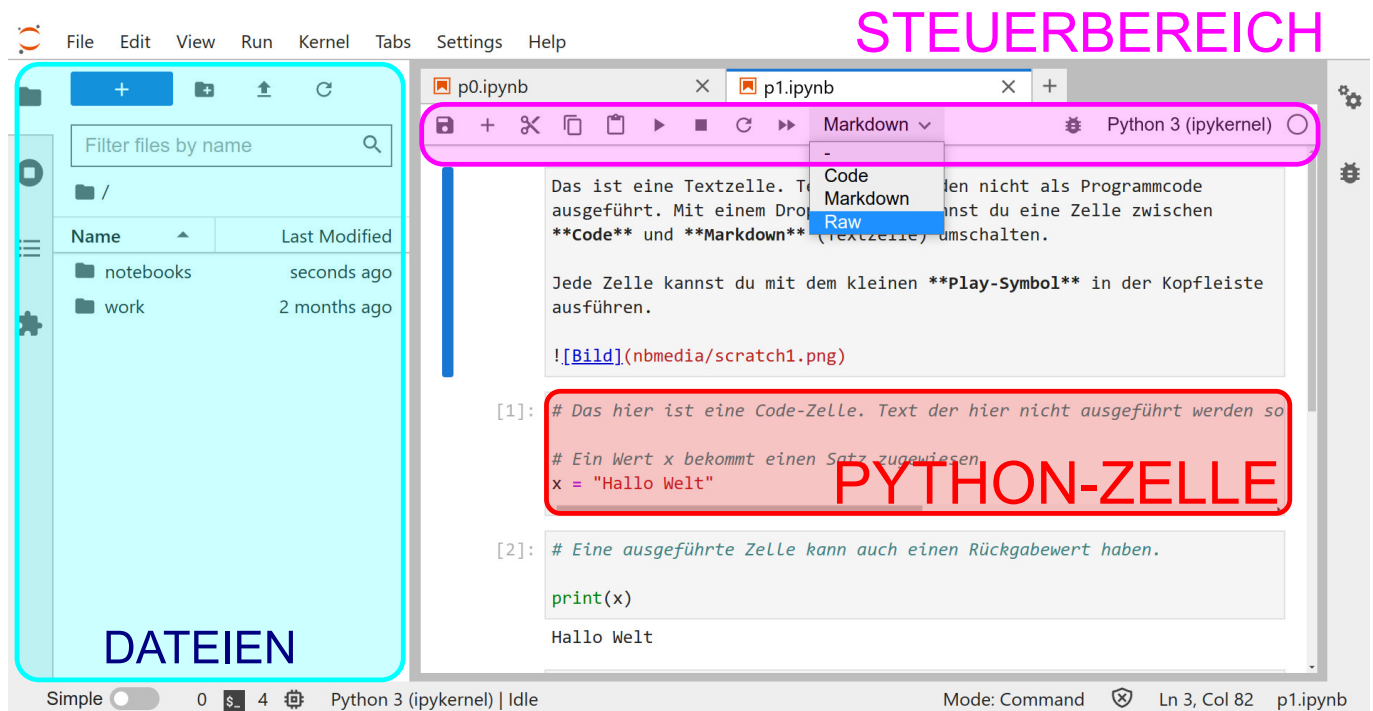


Abbildung 1: Eine Übersicht des „Python-Labs“.

DATEIEN In diesem Bereich kannst du mit Dateien arbeiten. Lade gespeicherte Dateien herunter, lade bestehende Dateien hoch, erstelle Ordner oder benenne sie um.

PYTHON-ZELLE Ein Python-Notebook (Dateiendung `.ipynb`) besteht aus einer oder mehreren Zellen. Jede Zelle kann ausgeführt werden.

STEUERBEREICH Hier findest du Steuerelemente die eine Zelle ausführen, den Zelltyp setzen oder das ganze Notebook neustartet. Hat eine Zelle den Zelltyp **Markdown**, dann wird der Inhalt nicht als Programm ausgeführt sondern als einfacher Text angezeigt. Wenn der Zelltyp auf **Code** gesetzt ist, dann wird der Programmcode mit Python ausgeführt.

1 Daten und Codierung

1.1 Textcodierung und Binärcode

Aufgabe 1



Erstelle bei dir im Heft eine ASCII-Tabelle. Diese soll die Spalten enthalten: ASCII-Code, Zeichen, Binärcode. Überlege dir, welche Zeichen du brauchst, um einen Satz zu codieren. Öffne dir als Hilfsmittel das Notebook c0.

Python-Notebook: Textcodierung und Binärcode

Mit diesen Befehlen kannst du umwandeln:

Binär -> Dezimal

```
[1]: int('101', 2)
```

```
[1]: 5
```

Dezimal -> Binär (hier z.B. 7-Bit)

```
[2]: format(18, '07b')
```

```
[2]: '0010010'
```

Zeichen -> ASCII-Code

```
[3]: ord('B')
```

```
[3]: 66
```

ASCII-Code -> Zeichen

```
[4]: chr(97)
```

```
[4]: 'a'
```

Bearbeite im Heft/Ordner als: Aufgabe 1

Aufgabe 2



Wandle mit Hilfe deiner ASCII-Tabelle den folgenden Satz in eine Bitfolge (7-Bit) um: Informatik ist toll!

👉 Wenn du schnell fertig bist, wandle ein eigenes Wort in eine Bitfolge (8-Bit) um.

Bearbeite im Heft/Ordner als: Aufgabe 2

1.2 Bildcodierung

Aufgabe 3



Codiere ein einfaches Pixelbild als Bitfolge. Dabei wird zuerst die Breite und dann die Höhe des Bildes als Binärzahl codiert. Anschließend folgt der Farbwert der einzelnen Pixel zeilenweise. Eine 1 entspricht beispielsweise einem schwarzen Pixel.

Entwirf zuerst ein eigenes Bild bei dir im Heft. Zeichne die Abmessungen ein und wandle in eine Binärzahl um. Gib dann, mit Hilfe des Notebooks c1 die Bitfolge für das Bild an.

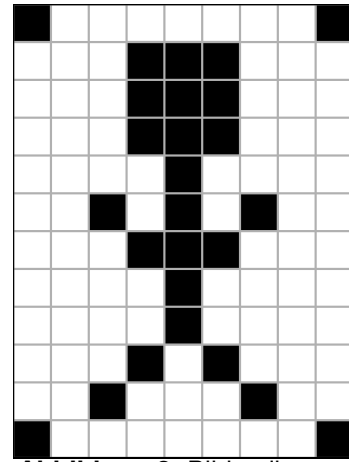
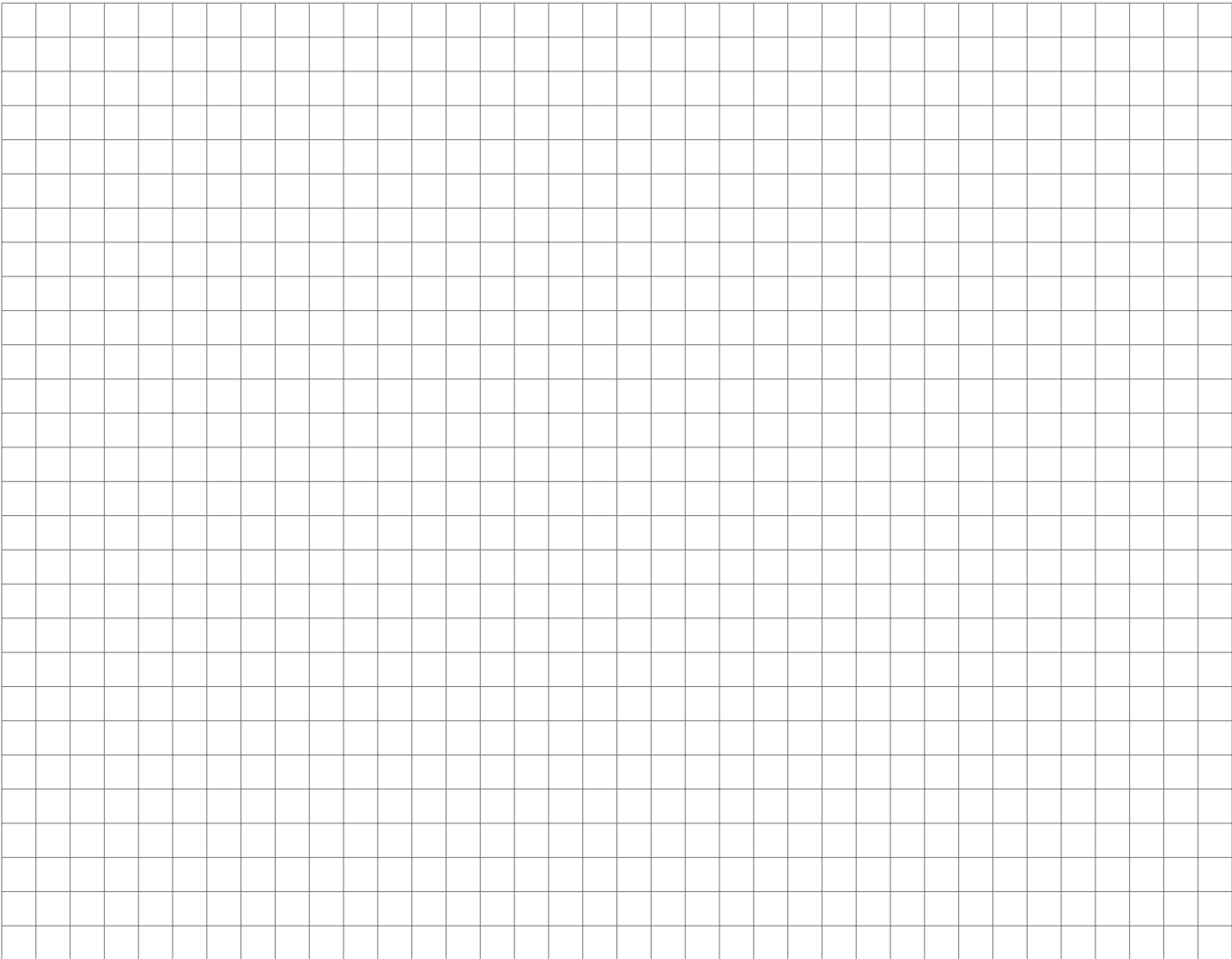


Abbildung 2: Bildcodierung

Bearbeite im Heft/Ordner als: Aufgabe 3



2 Algorithmen und Programmierung

2.1 Variablen in einem Algorithmus

Im Folgenden werden Variablen zuerst initialisiert und anschließend mit ihnen gerechnet. Kannst du erklären, warum in manchen Zellen ein Fehler gemeldet wird?

Python-Notebook: Variablen zuweisen / Operationen mit Variablen

```
[1]: zahl1 = 3
    zahl2 = 12
    zahl3 = '5'
    wort1 = 'Hallo'
    wort2 = 'Informatik'
```

```
[2]: zahl1
```

```
[2]: 3
```

```
[3]: zahl1+zahl3
```

```
-----
TypeError                                Traceback (most recent call last)
Cell In[3], line 1
----> 1 zahl1+zahl3

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

```
[4]: zahl1+int(zahl3)
```

```
[4]: 8
```

```
[5]: zahl4
```

```
-----
NameError                                Traceback (most recent call last)
Cell In[5], line 1
----> 1 zahl4

NameError: name 'zahl4' is not defined
```

```
[6]: wort1+wort2
```

```
[6]: 'HalloInformatik'
```

```
[7]: wort1+zahl1
```

```
-----
TypeError                                Traceback (most recent call last)
Cell In[7], line 1
----> 1 wort1+zahl1

TypeError: can only concatenate str (not "int") to str
```


```
[8]: wort1+str(zahl1)
```

```
[8]: 'Hallo3'
```

Aufgabe 4



lablite

 **lablite** Formuliere für beide Zellen ein passendes Zahlenrätsel in Worten. Bestimme danach das Ergebnis der beiden Print-Ausgaben. Kontrolliere das Ergebnis, indem du den Algorithmus in einem eigenen Notebook ausführst.

Python-Notebook: Zahlenrätsel

```
[ ]: zahl = 5
      zahl = zahl * 2
      zahl = zahl + 3
      print(zahl)
```

```
[ ]: zahl = 5
      zahl = zahl + 3
      zahl = zahl * 2
      print(zahl)
```

[illegible]

Aufgabe 5



lablite

Erstelle zu dem folgenden Zahlenrätsel ein Algorithmus in einem Notebook und bestimme damit das Ergebnis.

1. Ich denke mir eine Zahl.
2. Ich multipliziere meine Zahl mit 2.
3. Ich multipliziere meine neue Zahl mit 5.
4. Ich teile meine derzeitige Zahl durch die ursprüngliche Zahl ganz zu Beginn.
5. Ich subtrahiere 7 von meiner momentanen Zahl.

Speichere als: python5.ipynb

[illegible]

2.2 Den Wert von Variablen vergleichen

Python-Notebook: Variablen vergleichen

```
[1]: zahl1 = 3
    zahl2 = 12
    wort1 = 'Hallo'
    wort2 = 'Informatik'

[2]: zahl1 == zahl2
[2]: False

[3]: zahl1 != zahl2
[3]: True

[4]: zahl2/4 == zahl1
[4]: True

[5]: wort1 == 'hallo'

[5]: False

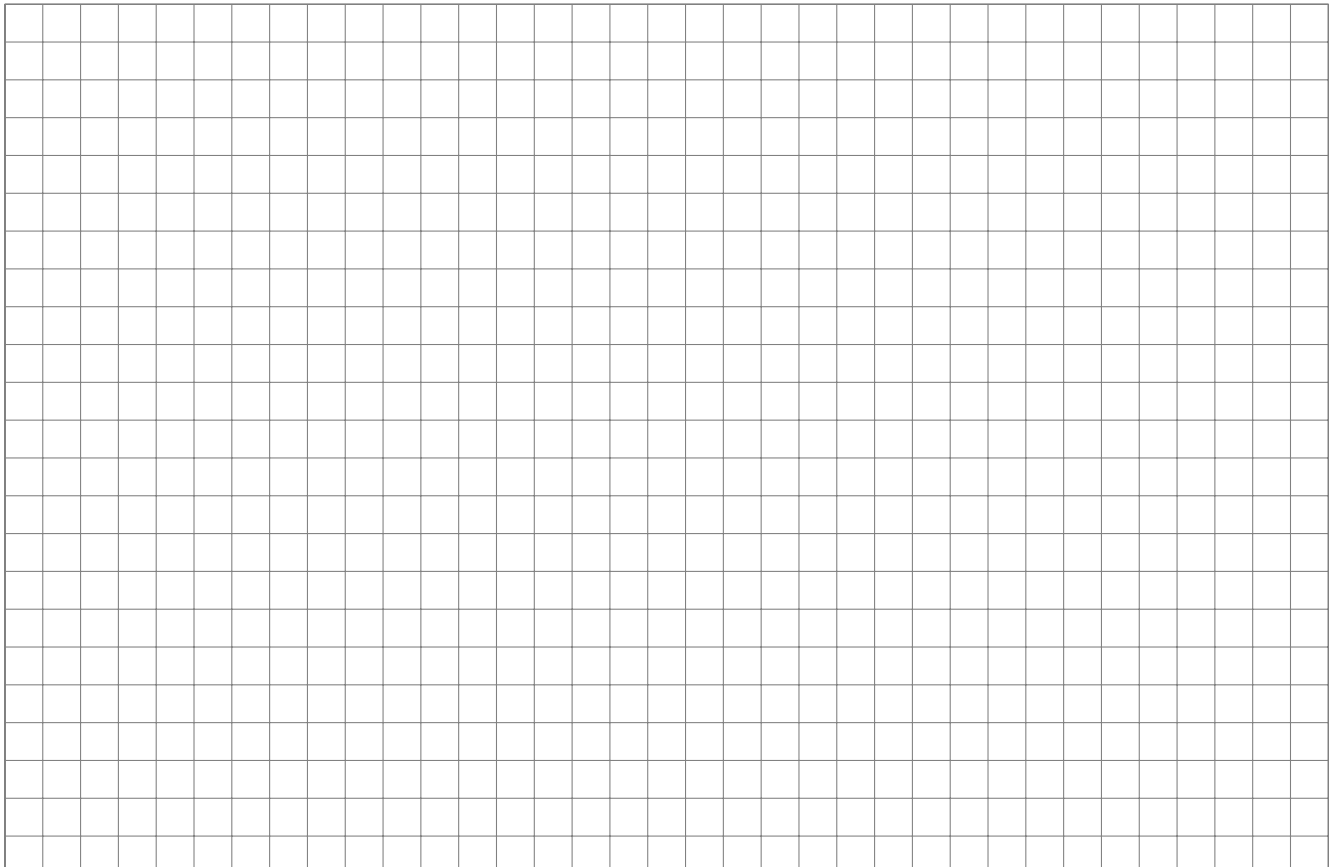
[6]: zahl1 > zahl2
[6]: False

[7]: zahl1 < 3
[7]: False


[8]: zahl1 <= 3
[8]: True

[9]: zahl1 == 3
[9]: True
```

Es werden Variablen verglichen. Notiere dir jeweils die Bedeutung von den Vergleichsoperatoren `==`, `!=`, `>=`, `<=`, `>` und `<`. Erstelle weitere Beispiele für Vergleiche mit den Variablen aus dem Beispiel.



Aufgabe 6

 c3 lablite

Die Zellen werden der Reihe nach ausgeführt. Notiere die Rückgabewerte für jede Zelle. Führe danach das Notebook c3.ipynb aus und korrigiere deine Lösungen.

Python-Notebook: Operationen und Vergleiche mit Variablen

```
[ ]: zahl1 = 5
    zahl2 = 15
    wort1 = 'Guten'
    wort2 = 'Tag'
```

```
[ ]: zahl2/'3' == 5
```

```
[ ]: wort1+wort2 == 'Hallo Informatik'
```

```
[ ]: 17 >= zahl1+zahl2
```

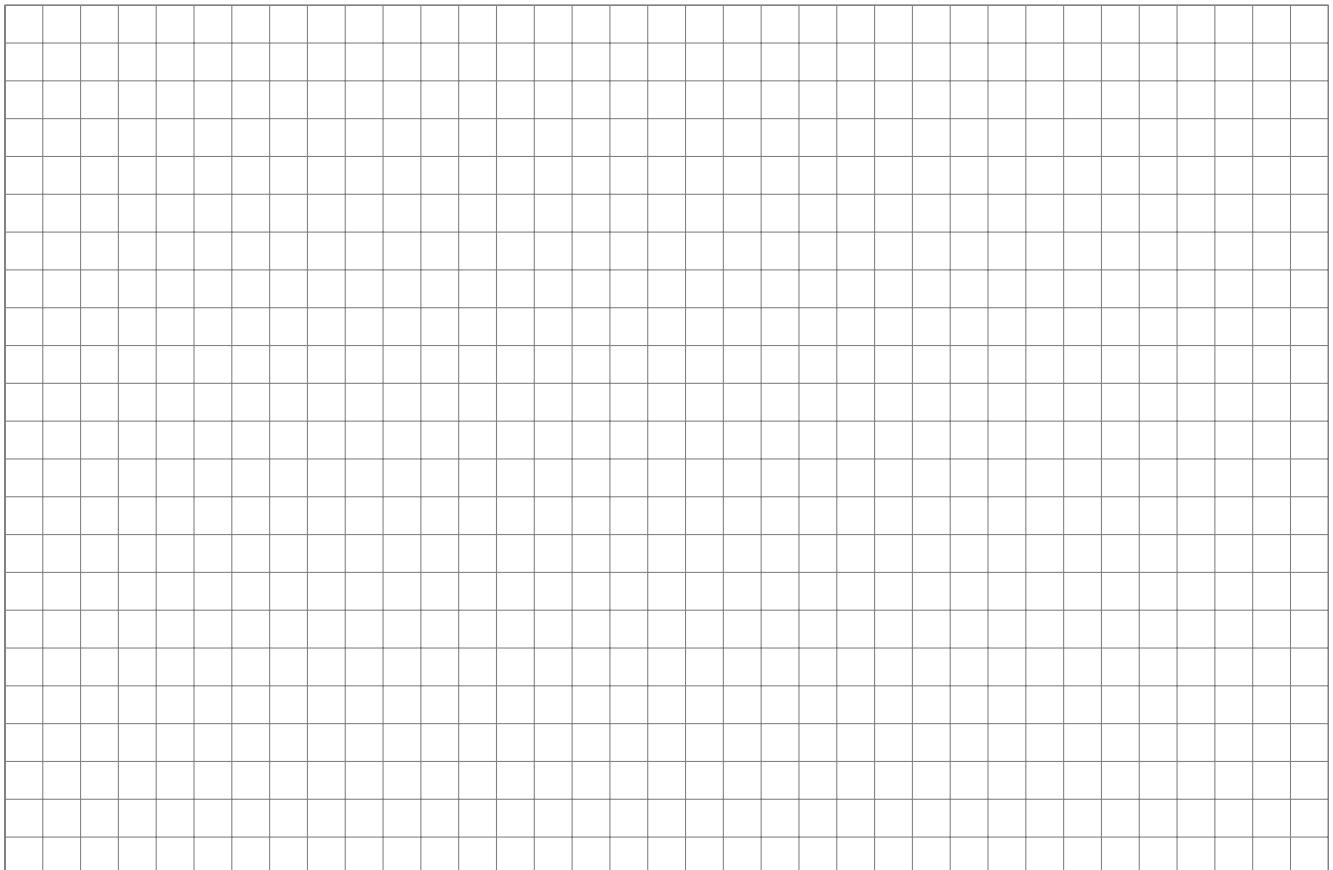
```
[ ]: zahl4+3
```

Aufgabe 7

 c2

In Aufgabe 2 hast du den Satz „Informatik ist toll!“ in eine Bitfolge umgewandelt. Schreibe einen Algorithmus, der diesen Satz als Bitfolge darstellt und überprüfe damit deine Lösung von Aufgabe 2 indem du Variablen vergleichst.

Speichere als: python7.ipynb



2.3 Visuelles Programmieren und textuelles Programmieren

 s3

Mit Blockly kannst du Programmblöcke zusammenstellen und eine Figur (Schildkröte) auf einer Bühne bewegen. Die visuelle Programmierung mit den Blöcken wird auch als Programm-Code (Textzeichen) angezeigt.

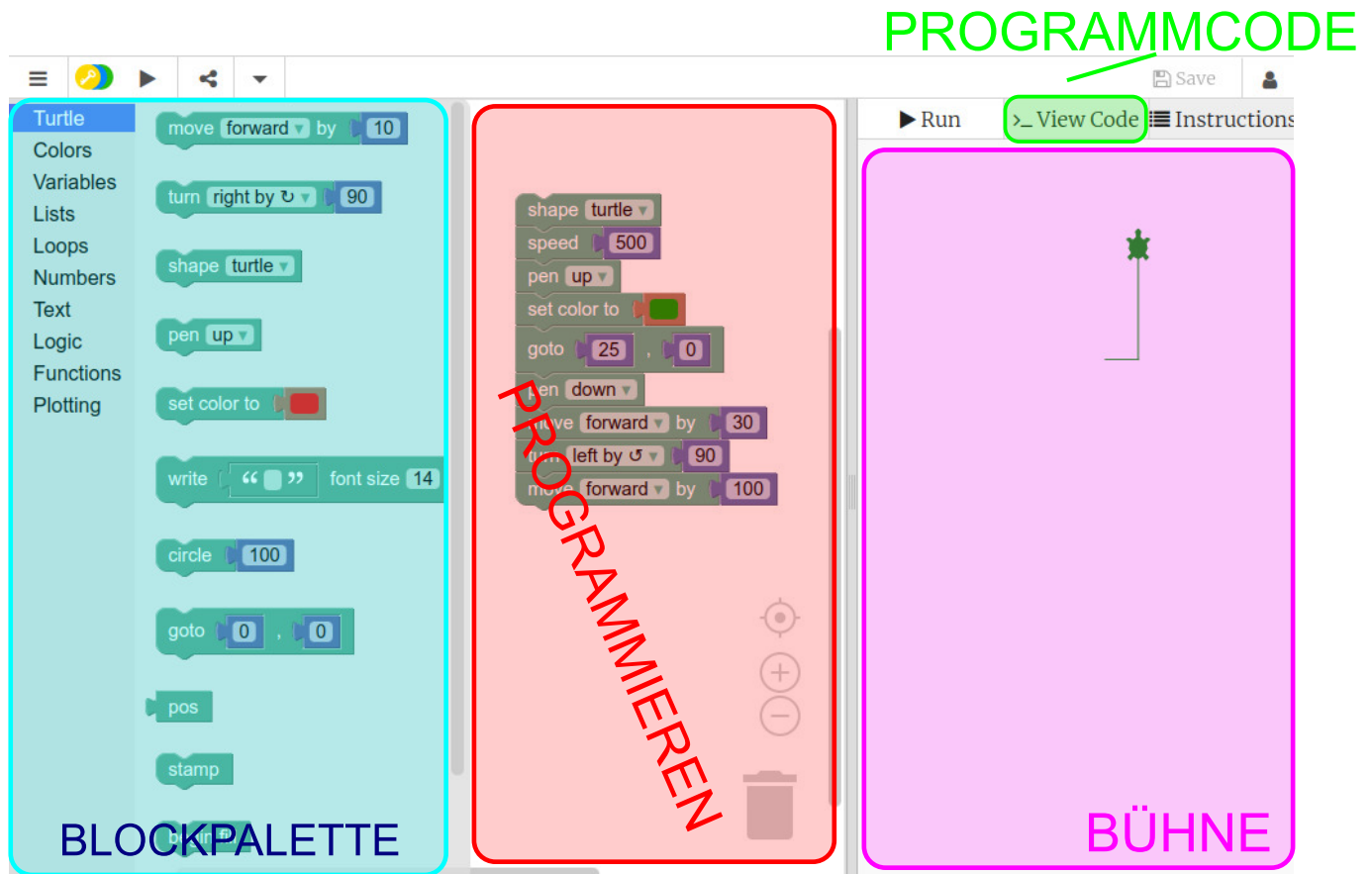


Abbildung 3: Mit Blockly wird visuell programmiert. Der passende Quellcode wird daneben angezeigt.

BÜHNE Auf der Bühne läuft alles ab, was du programmierst. Dort machen deine Figuren das, was du ihnen im Programmierbereich aufträgst.

BLOCKPALETTE In der Blockpalette findest du die Blöcke, die du zum Programmieren brauchst. Die Funktionalität ist ähnlich zu Scratch.

PROGRAMMCODE Hier kannst du die Programmierung deiner Figur in Python-Programmcode anzeigen lassen.

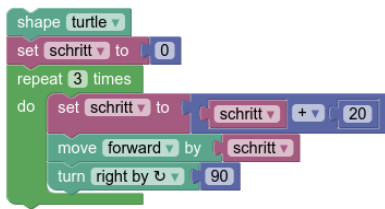
PROGRAMMIEREN Genauso wie in Scratch werden deine Anweisungen in Blöcken programmiert.

2.4 Mit for- und while-Schleifen Programmcode vereinfachen

visuelles Programmieren

textuelles Programmieren

Ausgabe des Programms



```

turtle.shape("turtle")
schritt = 0
for count in range(3):
    schritt = schritt + 20
    turtle.forward(schritt)
    turtle.right(90)
  
```

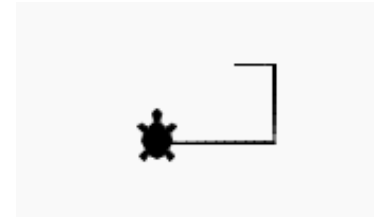
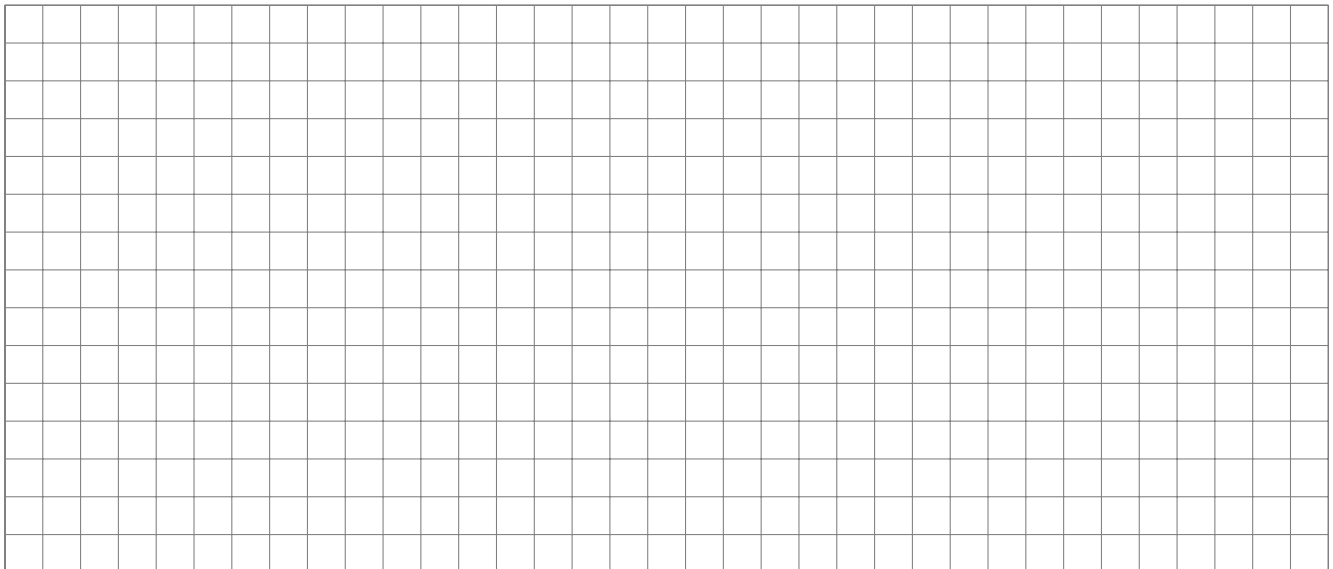


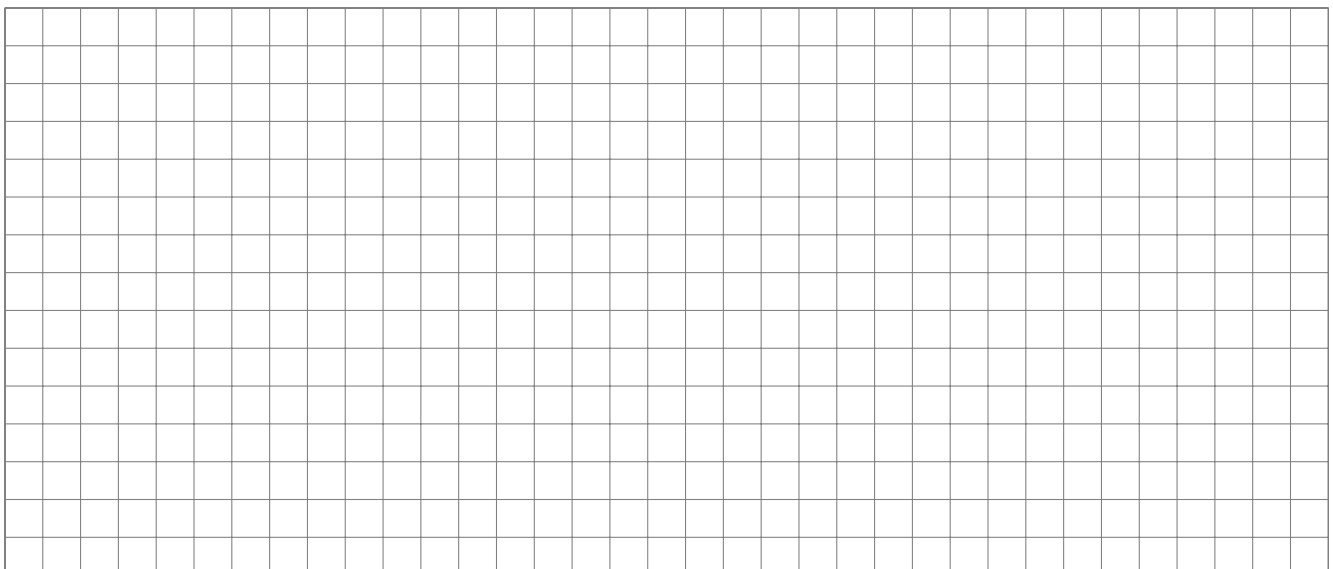
Abbildung 4: Visuelles und textuelles Programmieren von



Beschreibe, was die `for`-Schleife für eine Funktion hat. Markiere jeweils an den Ecken des Laufbildes der Schildkröte die Werte der **Laufvariable** `count`.



Skizziere den Laufweg der Schildkröte (20 Schritte entsprechen einem Kästchen) wenn der Block in der `for`-Schleife vier Mal wiederholt wird.



In dem nächsten Beispiel siehst du zwei `for`-Schleifen und zwei `while`-Schleifen. Dabei wird in jedem Durchgang der Wert der Laufvariable `i` angezeigt.

Python-Notebook: Schleifen

```
[1]: for i in range(3):
      print(f'i={i}')
```

```
i=0
i=1
i=2
```

```
[2]: i=0
      while i<3:
          print(f'i={i}')
```

```
i=0
i=1
i=2
```


```
[3]: i=5
      for _ in range(3):
          i = i + 1
          print(f'i={i}')
```

```
i=6
i=7
i=8
```

```
[4]: i=5
      while i < 8:
          i = i + 1
          print(f'i={i}')
```

```
i=6
i=7
i=8
```

Aufgabe 8

 c4 lablite

Notiere dir die Werte der Variable bei jedem Schleifendurchgang. Führe danach das Notebook `c4.ipynb` aus und korrigiere deine Lösungen.

Python-Notebook: Schleifen

```
[ ]: for i in range(4):
      print(f'i={i}')
```

```
[ ]: i=0
      while i<=3:
          i = i + 1
          print(f'i={i}')
```

```
[ ]: i=5
      for _ in range(3):
          print(f'i={i}')
```

```
i = i + 1
```

```
[ ]: i=5
      while i <= 8:
          print(f'i={i}')
```

```
i = i + 1
```


Aufgabe 9



Der Befehl `binaer` wandelt ein Zeichen in die Binärdarstellung (ASCII) um. Begründe schriftlich im Heft, welchen Wert die Variable `bitfolge` am Ende der Sequenz hat. Überprüfe deine Antwort indem du den Programmcode in einem Notebook ausführst.

```
1 bitfolge=''
2 for buchstabe in 'Schule':
3     bitfolge = binaer(buchstabe)
4
5 bitfolge
```

This image shows a full page of blank graph paper. The grid consists of small, equal-sized squares formed by thin gray lines. There are 20 columns and 20 rows of these squares, creating a total area of 400 small squares. The grid is perfectly aligned and covers the entire page without any margins or additional markings.

2.5 Mit Bedingungen und Verzweigungen entscheiden

Mit einer bedingten Anweisung^[1] können wir in einem Programmcode festlegen welcher Codeabschnitt ausgeführt wird.

In einem `if...else...` Block wird überprüft **ob** (engl. `if`) eine Bedingung zutrifft. **Falls** diese Bedingung **nicht** (engl. `else`) zutrifft, wird anderer Code ausgeführt.


Es können auch mehrere `if`-Anweisungen hintereinander ausgeführt werden - dabei nutzt man ab der zweiten Bedingung die überprüft wird die Anweisung `elif` bzw. `else if`. Wird mehr als nur eine `if`-Anweisung verwendet, dann spricht man von einer *Verzweigung*.

Python-Notebook: Bedingungen und Verzweigungen

```
[1]: for i in range(5):  
    if i > 3:  
        print(f'i={i} ist größer als 3')  
    elif i < 3:  
        print(f'i={i} ist kleiner als 3')  
    else:  
        print(f'i={i} ist 3')
```

```
i=0 ist kleiner als 3  
i=1 ist kleiner als 3  
i=2 ist kleiner als 3  
i=3 ist 3  
i=4 ist größer als 3
```

Aufgabe 10

 c5 lablite


Notiere die Ausgabe wenn der Programmcode ausgeführt wird. Führe danach das Notebook `c5.ipynb` aus und korrigiere deine Lösungen.

Python-Notebook: Schleifen

```
[ ]: for i in range(5):  
    if i+i == 4:  
        print(f'{i} Hallo')  
  
    if i < 3:  
        print(f'{i} Schule')  
    elif i > 3:  
        print(f'{i} Technik')  
    else:  
        print(f'{i} Informatik')
```

Aufgabe 11



Benutze Blockly  als Hilfestellung für diese Aufgabe.

Verändere den Quellcode so, dass die Schildkröte im Uhrzeigersinn ein Quadrat mit der Seitenlänge 150 abläuft.

Python-Notebook: Wir lassen die Schildkröte laufen

Mit dem Python Modul **Turtle** kannst du eine Schildkröte bewegen.

```
[ ]: # Hier werden die Module importiert
from infomodules.turtleinit import *

# Wir erschaffen eine neue Schildkröte
turtle = newturtle()

# Ab hier fangen die Laufanweisungen an
turtle.shape("turtle")
turtle.speed(100)
turtle.penup()
turtle.goto(-100,-50)
turtle.setheading(0)
turtle.pendown()
turtle.forward(200)
turtle.left(90)
turtle.forward(100)
turtle.left(90)
turtle.forward(200)
turtle.left(90)
turtle.forward(100)
```

Speichere als: python11.ipynb

Aufgabe 12

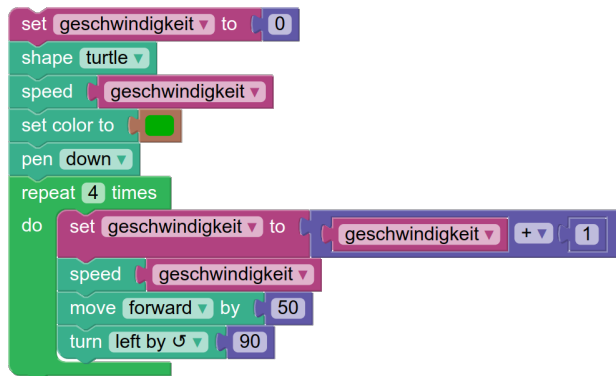


Abbildung 5: Eine for-Schleife.

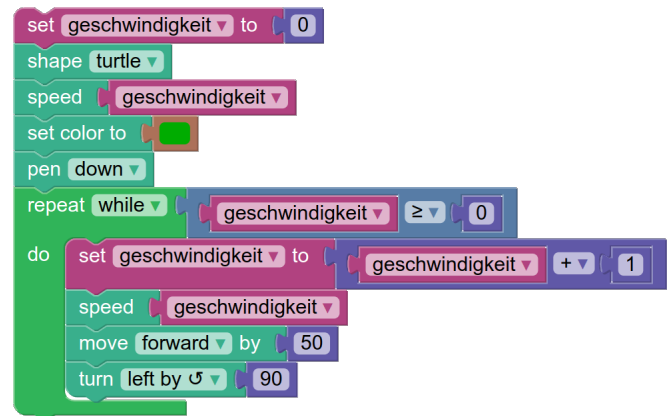
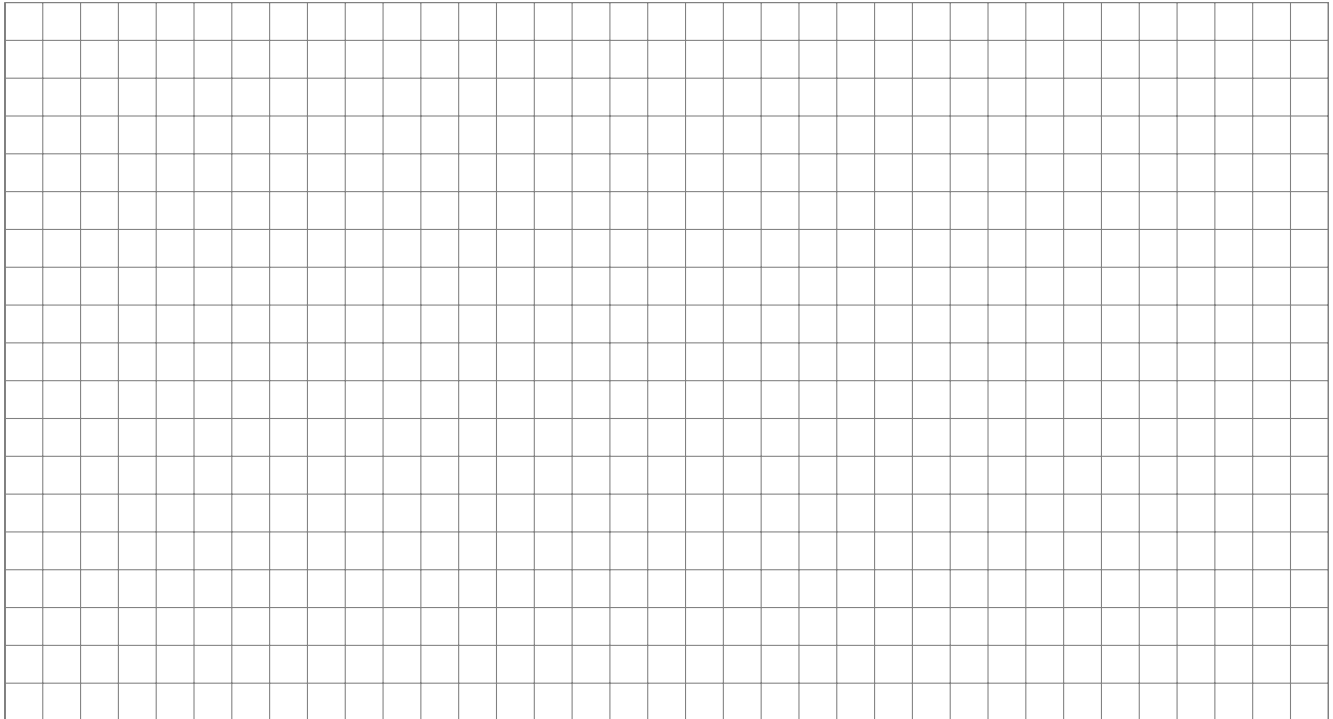


Abbildung 6: Eine while-Schleife.

In den Abbildungen 5 und 6 wird eine Schildkröte einmal mit einer `for`-Schleife und einmal mit einer `while`-Schleife bewegt. Bewegen sich beide Schildkröten gleich?



Aufgabe 13



Öffne das *Blockly-Turtle-Spiel* und löse die Aufgaben mit Hilfe von Schleifen.

Übertrage die ersten beiden Aufgaben in Python-Programmcode und verwende dabei einmal eine `for` und einmal eine `while`-Schleife.

Speichere als: python13.ipynb

Aufgabe 14



Erweitere den Code so, dass die Schildkröte in dem Terrarium von oben nach unten läuft.

Python-Notebook: Schildkrötenterrarium

Mit der Anweisung `turtle.heading()` wird die aktuelle Ausrichtung der Schildkröte als Winkel (in Grad) abgefragt. Bewegt sie sich gerade nach rechts, dann ist die Ausrichtung 0° . Mit `turtle.setheading(90)` wird die Ausrichtung z.B. auf 90° gesetzt - die Schildkröte läuft gerade nach oben.

```
[ ]: # Hier werden die Module importiert
from infomodules.turtleinit import *

# Wir erschaffen eine neue Schildkröte
breite = 250
hoehe = 200

turtle = newturtle(width=breite, height=hoehe)
turtle.screen.delay(0)
turtle.speed(100)

# Die Schildkröte bewegt sich zum Startpunkt gerade nach rechts.
turtle.setheading(0)

# Laufanweisungen
turtle.shape("turtle")

max_distance = 800
distance = 0
step = 1

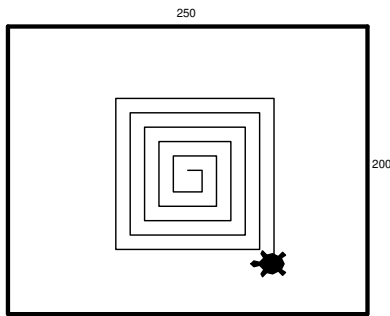
while distance < max_distance:
    turtle.forward(step)
    x, y = turtle.position()

    if (x > breite/2) or (x < -breite/2):
        turtle.setheading(180-turtle.heading())

    distance = distance + step
```

Speichere als: python14.ipynb

Aufgabe 15



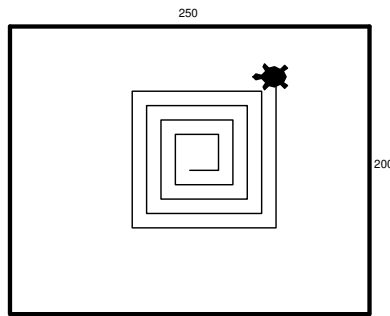
(a)

(I)

```

1  step = 20
2  turtle.penup()
3  turtle.setpos(-100,-80)
4  turtle.pendown()
5
6  for count in range(7):
7      turtle.forward(step)
8      turtle.left(90)
9      turtle.forward(step)
10     turtle.right(90)

```



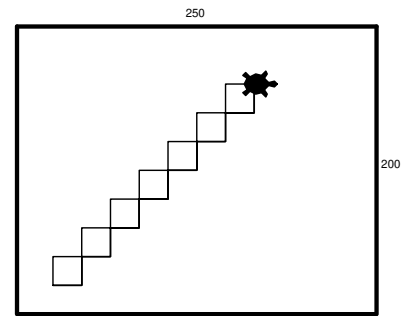
(b)

(II)

```

1  step = 20
2  turtle.penup()
3  turtle.setpos(0,0)
4  turtle.pendown()
5
6  for count in range(9):
7      turtle.forward(step)
8      step = step + 5
9      turtle.left(90)
10     turtle.forward(step)
11     step = step + 5
12     turtle.left(90)

```



(c)

(III)

```

1  step = 10
2  turtle.penup()
3  turtle.setpos(0,0)
4  turtle.pendown()
5
6  while step < 120:
7      turtle.forward(step)
8      step = step + 5
9      turtle.right(90)
10     turtle.forward(step)
11     step = step + 5
12     turtle.right(90)

```

Ordne zwei Bilder aus (a), (b) und (c) zwei Programmcodes (I), (II) und (III) zu. Ein Paar passt nicht zusammen. Begründe deine Zuordnung.

Zeichne danach die Spur der Schildkröte von dem nicht zugeordneten Programmcode auf. Verwende für einen 20er-Schritt ein Kästchen.



p1

Erstelle einen Programmcode, der das nicht-zugeordnete Bild erstellt.



Aufgabe 16



Python-Notebook: Vokabelabfrage

Mit dem Modul `num2words` kann man eine Zahl in Text ausgeben lassen - in verschiedenen Sprachen. Mit `input()` kann eine Nutzereingabe übergeben werden-

```
[ ]: # Hier werden die Module importiert
from num2words import num2words
import random

points = 0
print("Gib die Zahl ein:")
for count in range(5):
    zahl = random.randint(0,100)
    print(num2words(zahl,lang='fr'))
    eingabe = input()
    if int(eingabe)==zahl:
        print("Richtig :)")
        points = points + 1
    else:
        print("Falsch :(. Richtig war:")
        print(zahl)

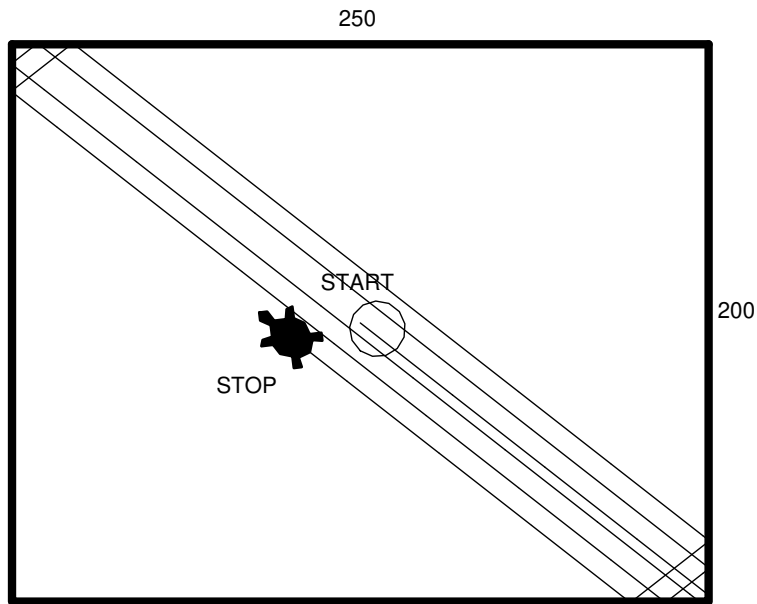
print("Deine Punkte:")
print(points)
```

Schreibe das Programm so um, dass für falsche Antworten ein Punkt abgezogen wird.

✈ Ändere den Programmcode so, dass das Programm eine Addition oder Subtraktion zweier Zahlen in einer Fremdsprache abfragt.

Speichere als: `python16.ipynb`

Aufgabe 17



In der Mitte des Terrariums steht ein Futtertrog. Wenn Schildi einen Abstand von 10cm oder weniger hat, dann kann sie bei jedem Schritt so viel Nahrung aufnehmen, dass sie 20cm weiter laufen kann. Ihre Geschwindigkeit bleibt dabei unverändert.

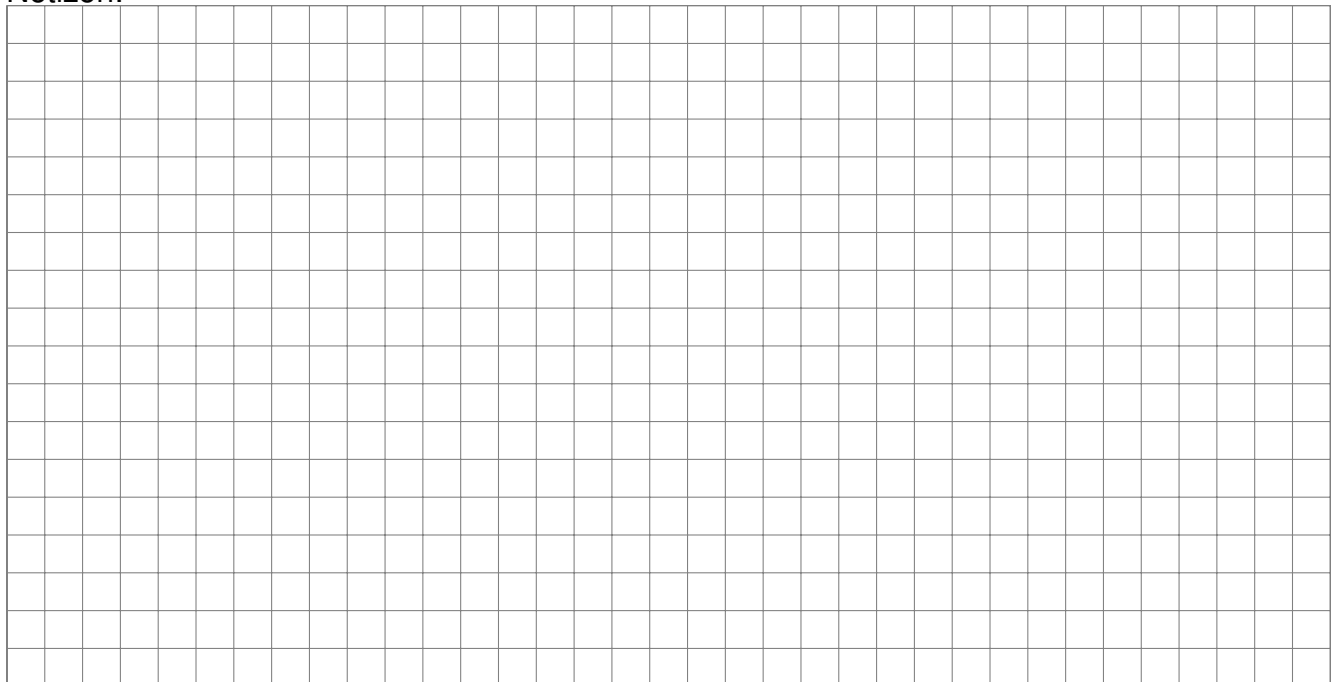
Abbildung 8: Schildi und der Futtertrog.

Passe den Code aus Aufgabe 14 so an, dass die beschriebene Situation zutrifft. Dabei ist die Ausrichtung beim Start nicht ausschlaggebend.

🐢 Mit `random.randint(-180,180)` kannst du eine zufällige Zahl zwischen -180 und $+180$ würfeln. Lasse deine Schildkröte in eine zufällige Richtung starten.

Speichere als: `python17.ipynb`

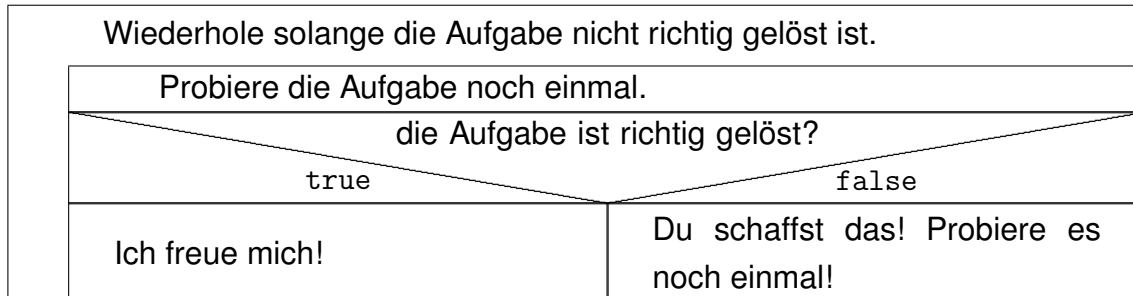
Notizen:



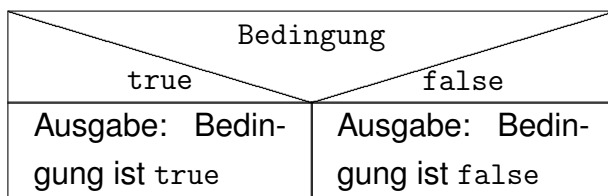
2.7 Mit Struktogrammen Algorithmen entwerfen

Mit einem **Struktogramm**^[4] kann man einen Algorithmus zuerst auf dem Papier entwerfen. Dafür ein Beispiel:

„Solange ich die Aufgabe nicht richtig habe, probiere ich sie noch einmal. Falls die Aufgabe richtig gelöst ist, freue ich mich. Falls nicht, motiviere ich mich die Aufgabe noch einmal zu probieren.“

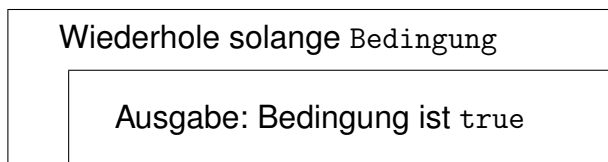


Schreibweisen im Struktogramm^[5,6]



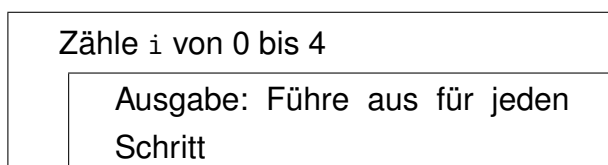
Bedingung/Verzweigung

```
1 if Bedingung == True:
2     print('Ausgabe: Bedingung ist True')
3 else:
4     print('Ausgabe: Bedingung ist False')
```



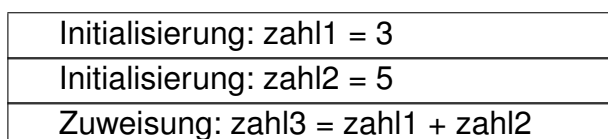
while-Schleife

```
1 while Bedingung:
2     print('BEDINGUNG ist True')
```



for-Schleife

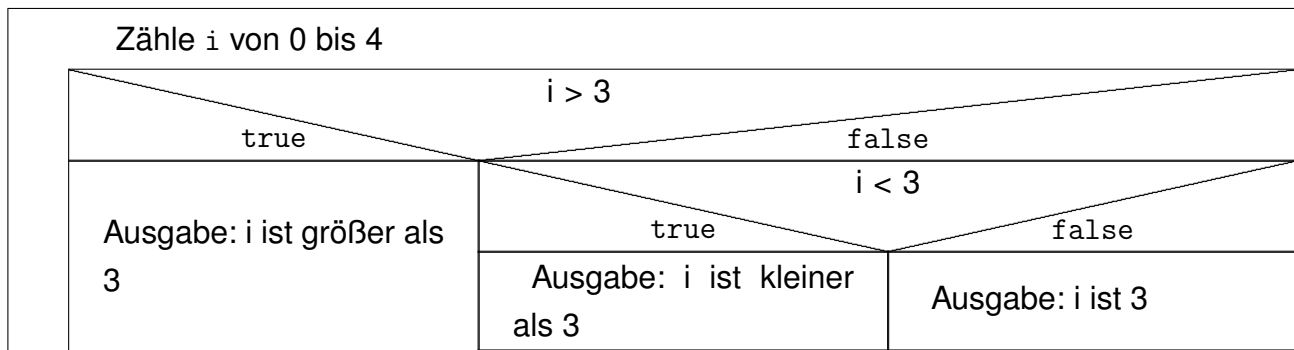
```
1 for i in range(4):
2     print('Fuehre aus fuer jeden Schritt')
```



Einzelne Sequenzschritte im Programm

```
1 zahl1 = 3
2 zahl2 = 5
3 zahl13 = zahl1 + zahl2
```

Der Programmcode auf Seite 16 lässt sich ebenso als Struktogramm darstellen:



Python-Notebook: Bedingungen und Verzweigungen

```
[1]: for i in range(5):
    if i > 3:
        print(f'i={i} ist größer als 3')
    elif i < 3:
        print(f'i={i} ist kleiner als 3')
    else:
        print(f'i={i} ist 3')
```

```
i=0 ist kleiner als 3
i=1 ist kleiner als 3
i=2 ist kleiner als 3
i=3 ist 3
i=4 ist größer als 3
```

Aufgabe 18

Erstelle zu dem folgenden Programmcode ein Struktogramm (den Programmcode kennst du schon aus Aufgabe 10)

Python-Notebook: Schleifen

```
[ ]: for i in range(5):
    if i+i == 4:
        print(f'{i} Hallo')

    if i < 3:
        print(f'{i} Schule')
    elif i > 3:
        print(f'{i} Technik')
    else:
        print(f'{i} Informatik')
```

3 Informationsgesellschaft und Datensicherheit^[7]

3.1 Cäsar-Verschlüsselung^[7]

^[3,9] Gaius Julius Caesar, der römische Feldherr, soll laut dem römischen Schriftsteller Sueton für vertrauliche Mitteilungen eine besondere Verschlüsselungsmethode angewendet haben, die später als Caesar-Verschlüsselung bekannt wurde. In dieser Technik verschob Caesar das Alphabet um drei Buchstaben. Sueton beschreibt das Verfahren folgendermaßen:

„Wenn er etwas vertraulich Übermittelndes hatte, schrieb er in Geheimschrift. Das bedeutet, dass er die Buchstaben in einer Reihenfolge anordnete, die es unmöglich machte, ein Wort zu erkennen. Um dies zu entschlüsseln, musste man den vierten Buchstaben, also D für A, austauschen, und dasselbe mit den übrigen Buchstaben tun.“




Abbildung 9: eine gebastelte Chiffrierscheibe für die Cäsar-Verschlüsselung^[9].

Wie weit dreht Cäsar seine Chiffrierscheibe?

Aufgabe 19



Verwende eine Chiffrierscheibe wie in Abbildung 9 um die folgende Nachricht von Cäsar zu entschlüsseln. Bastel dir dazu die Scheibe entweder aus Papier (Vorlage^[10] „Die Caesar-Chiffre“ am Ende des Skriptes), oder rufe  auf.

Python-Notebook: Cäsar-Verschlüsselung

```
[2]: caesar_verschlüsseln('ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'D')
```

```
[2]: 'DEFGHIJKLMNOPQRSTUVWXYZABC'
```

```
[3]: caesar_entschlüsseln('DEFGHIJKLMNOPQRSTUVWXYZABC', 'D')
```

```
[3]: 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

1) Verschlüssele das Wort 'Kryptographie' mit dem Schlüssel 'P'

```
[ ]: caesar_verschlüsseln('Kryptographie', 'P')
```

2) Entschlüssele das Wort 'LJMJNRSFHMWNHMY' mit dem Schlüssel 'F'

```
[ ]: caesar_entschlüsseln('LJMJNRSFHMWNHMY', 'F')
```

3) Schreibe eine kurze Nachricht mit höchstens drei Wörtern und verschlüssele diese mit einem von dir gewählten Schlüssel. Tausche anschließend die verschlüsselte Nachricht und den zugehörigen Schlüssel mit deinem Sitznachbarn und entschlüssele diese.

Die Buchstabenhäufigkeit, auch als Graphemhäufigkeit bekannt, ist eine Zahl, die uns sagt, wie oft bestimmte Buchstaben in einem Text oder einer Gruppe von Texten (Korpus) auftreten. Man kann sie entweder als absolute Anzahl oder im Verhältnis zur Gesamtanzahl der Buchstaben im Text angeben. Wie oft bestimmte Buchstaben auftauchen, hängt von der Sprache ab, die verwendet wird. Man beschäftigt sich schon seit dem frühen 19. Jahrhundert mit dem Zählen der Buchstabenhäufigkeit in Texten oder Textsammlungen^[2]. In Tabelle 2 siehst du eine Auflistung der Buchstaben e, n, i, s, r und a mit der jeweiligen Häufigkeit in verschiedenen Sprachen.

Tabelle 2: Prozentsatz der Buchstaben e, n, i, s, r und a in verschiedenen Sprachen^[2].

Buchstabe	Deutsch	Englisch	Französisch	Spanisch	Esperanto	Italienisch	Schwedisch	Polnisch
e	17,40%	12,702%	14,715%	13,68%	8,99%	11,79%	9,9%	6,9%
n	9,78%	6,749%	7,095%	6,71%	7,96%	6,88%	8,8%	4,7%
i	7,55%	6,966%	7,529%	6,25%	10,01%	11,28%	5,1%	7,0%
s	7,27%	6,327%	7,948%	7,98%	6,09%	4,98%	6,3%	3,8%
r	7,00%	5,987%	6,553%	6,87%	5,91%	6,37%	8,3%	3,5%
a	6,51%	8,167%	7,636%	12,53%	12,12%	11,74%	9,3%	8,0%

Aufgabe 20



Diese Nachricht wurde mit der Cäsar-Verschlüsselung verschlüsselt: „*Ui yij uyd abuydui Kfiy fqi-iyuhj, kdt tuh lxxbkuiub vph tyuiud Junj yij luhbehudwuwqdwud. Tyu Setuadqsauh qki tuh Abqiiu 7 aeuddud ruijycj xubvud!*“

- Erstelle eine Tabelle mit der relativen Buchstabenhäufigkeit der Geheimnachricht.
- Finde den Schlüssel des verschlüsselten Textes mit Hilfe von Tabelle 2 heraus.
- Entschlüssele die Geheimnachricht.
- 🔑 Warum ist das Entschlüsseln schwerer, wenn nicht bekannt ist in welcher Sprache die Nachricht verfasst ist?

Bearbeite im Heft/Ordner als: Aufgabe 20

3.2.1 Cäsar-Verschlüsselung mit der Brute-Force-Methode knacken

Python-Notebook: Cäsar-Verschlüsselung mit der Brute-Force-Methode knacken

```
[2]: for schlüssel in 'ABCDEFGHIJKLMNOPQRSTUVWXYZ':
    entschlüsselte_nachricht = caesar_entschlüsseln("RKVVY NKC SCD OSXO QOROSWO_
    ↳QOROSWXXMRBSMRD", schlüssel)
    print(f'Schlüssel: {schlüssel} \t Nachricht: {entschlüsselte_nachricht}')
```

Schlüssel: A	Nachricht: RKV VY NKC SCD OSXO QOROSWO QOROSWXKMRBSMRD
Schlüssel: B	Nachricht: QJUUX MJB RBC NRWN PNQNRVN PNQNRVWJLQARLQC
Schlüssel: C	Nachricht: PITTW LIA QAB MQVM OMPQMUM OMPMQUVIK PZQKPB
Schlüssel: D	Nachricht: OHSSV KHZ PZA LPUL NLOLPTL NLOLPTU HJOYPJOA
Schlüssel: E	Nachricht: NGRRU JGY OYZ KOTK MKNKOSK MKNKOSTG INXOINZ
Schlüssel: F	Nachricht: MFQQT IFX NXY JNSJ LJMJNRJ LJMJNRSFHMWNHMY
Schlüssel: G	Nachricht: LEPPS HEW MWX IMRI KILIMQI KILIMQREG LVMGLX
Schlüssel: H	Nachricht: KDOOR GDV LVW HLQH JHKHLP HJKHLPQDFKULFKW
Schlüssel: I	Nachricht: JCNNQ FCU KUV GKPG IGJGKOG IGJGKOPCEJTKEJV
Schlüssel: J	Nachricht: IBMMP EBT JTU FJOF HFIFJNF HFIFJNOBDISJDIU
Schlüssel: K	Nachricht: HALLO DAS IST EINE GEHEIME GEHEIMNACHRICHT
Schlüssel: L	Nachricht: GZKKN CZR HRS DHMD FDGDHLD FDGDHLMZBGQHBGS
Schlüssel: M	Nachricht: FYJJM BYQ GQR CGLC ECFCGKC ECFCGKLYAFPGAFR
Schlüssel: N	Nachricht: EXIIL AXP FPQ BFKB DBEBFJB DBEBFJKXZE OFZEQ
Schlüssel: O	Nachricht: DWHHK ZWO EOP AEJA CADA EIA CADA EIJWYDNEYDP
Schlüssel: P	Nachricht: CVGGJ YVN DNO ZDIZ BZCZDHZ BZCZDHIVXCMDXCO
Schlüssel: Q	Nachricht: BUFFI XUM CMN YCHY AYBYCGY AYBYCGHUWBLCWBN
Schlüssel: R	Nachricht: ATEEH WTL BLM XBGX ZXAXBFX ZXAXBFGTVAKBVAM
Schlüssel: S	Nachricht: ZSDDG VSK AKL WAFW YWZWA EW YWZWA EFSUZJAUZL
Schlüssel: T	Nachricht: YRCCF URJ ZJK VZEV XVYVZDV XVYVZDERTYIZTYK
Schlüssel: U	Nachricht: XQBBE TQI YIJ UYDU WUXUYCU WUXUYCDQSXHYSXJ
Schlüssel: V	Nachricht: WPAAD SPH XHI TXCT VTW TXBT VTW TXBCPRWGXRWI
Schlüssel: W	Nachricht: VOZZC ROG WGH SWBS USVSWAS USVSWABOQVFWQVH
Schlüssel: X	Nachricht: UNYYB QNF VFG RVAR TRURVZR TRURVZANPUEVPUG
Schlüssel: Y	Nachricht: TMXXA PME UEF QUZQ SQTQUYQ SQTQUYZMOTDUOTF
Schlüssel: Z	Nachricht: SLWWZ OLD TDE PTYP RPSPTXP RPSPTXYLNSCTNSE

Erkläre, warum die Cäsar-Verschlüsselung auch ohne Häufigkeitsanalyse leicht geknackt werden kann.

[illegible]

3.3 Monoalphabetische Verschlüsselung

Bei der Caesar-Verschlüsselung wird jeder Buchstabe im Text um eine bestimmte Anzahl von Schritten im Alphabet verschoben. Das ist wie ein **Geheimschritt** nach links oder rechts.

Die *monoalphabetische Verschlüsselung* ist wie ein **Tauschspiel**. Jeder Buchstabe im Klartext wird durch einen anderen Buchstaben oder ein anderes Zeichen im Geheimtext ersetzt. Dieser Tausch bleibt für den gesamten Text gleich. Es ist so, als würde jedes Mal, wenn du ein „A“ siehst, es durch ein bestimmtes anderes Zeichen ersetzt. Das Klartextalphabet wird also einmal durcheinandergewürfelt und es gibt für jeden Buchstaben im Alphabet eine Entsprechung im Geheimalphabet.

Aufgabe 21

- Verwende die Kopiervorlage „Mono-Chiffre“ und trage das Geheimalphabet „*skhmyfradbniugwecxtopjqzvl*“ ein.
- Verschlüssele die Nachricht „*Diese Verschlüsselung ist etwas sicherer.*“ mit Hilfe des Schlüssels. Überprüfe, ob du das gleiche Ergebnis wie in dem Notebook erhältst.

Python-Notebook: monoalphabetische Verschlüsselung

```
[2]: MV('skhmyfradbniugwecxtopjqzvl').verschluessele('Diese Verschlüsselung ist etwas_
→sicherer.')
```


```
[2]: 'Mdyty Jyxthaipyttypgr dto yoqst tdhayxyx.'
```

```
[3]: MV('skhmyfradbniugwecxtopjqzvl').entschluessele('Mdyty Jyxthaipyttypgr dto yoqst_
→tdhayxyx.')
```

```
[3]: 'Diese Verschlüsselung ist etwas sicherer.'
```

Bearbeite im Heft/Ordner als: Aufgabe 21

Aufgabe 22

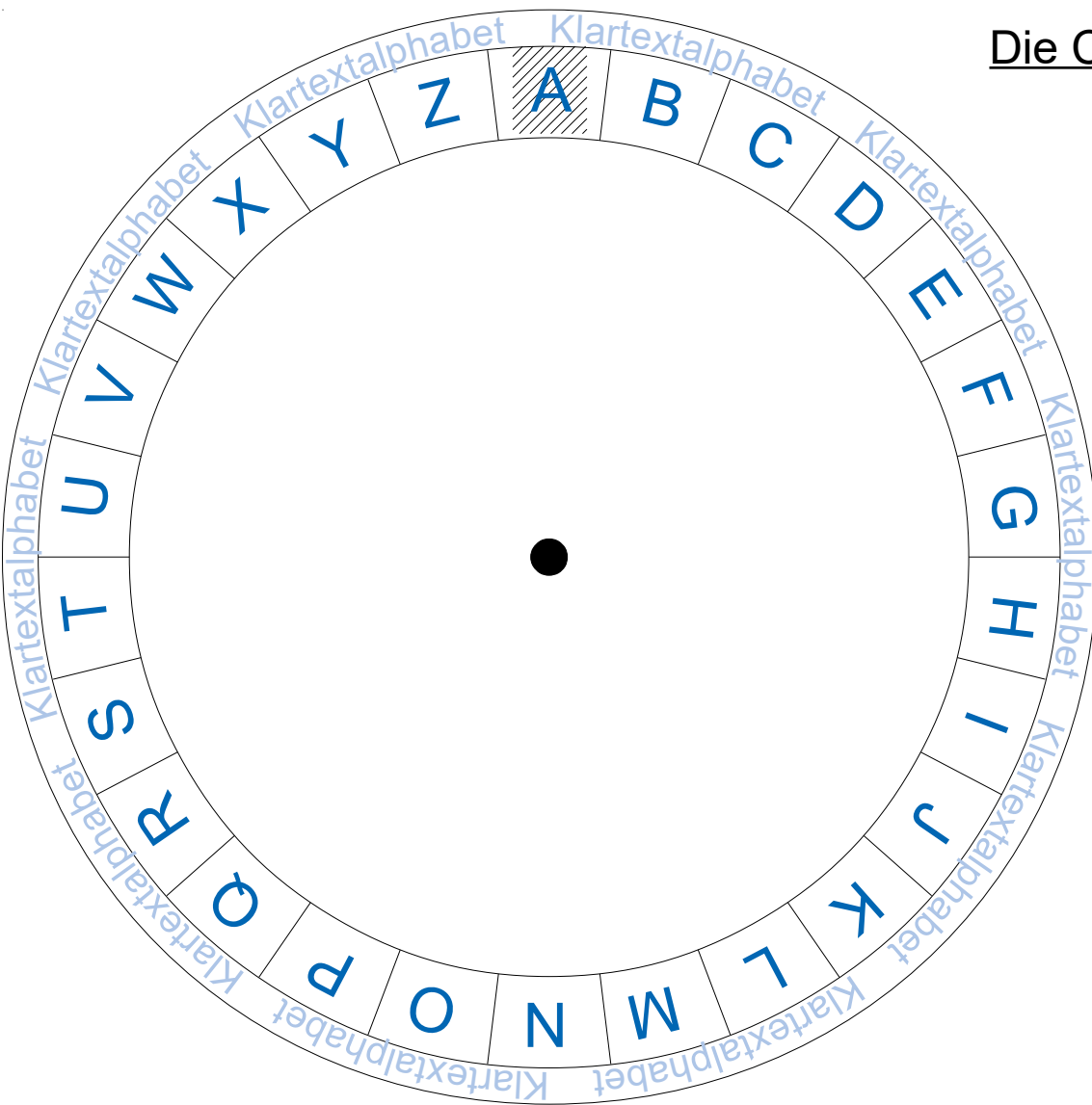
- a) Führe eine Häufigkeitsanalyse der Buchstaben in der Nachricht „Jxdyl yld Sabmulsslm gu yxlslw Diabwxabz. Fl mildnlw ylw Zloz xsz, ylsze nldiulw xsz iuab yxl Biujxnclxzsidiimtsl ylw Quabsziql!“ durch.  k3
- b) Begründe, warum sich diese Nachricht nicht mit der Cäsar-Verschlüsselung entschlüsseln lässt.
- c) Begründe, ob die monoalphabetische Verschlüsselung gegen Brute-Force-Angriffe weniger anfällig ist.
- d) 🦋 Gib für die monoalphabetische Verschlüsselung und für die Cäsar-Verschlüsselung die Schlüssellänge in Bit an.

Bearbeite im Heft/Ordner als: Aufgabe 22

Quellen

- [1] Bedingte Anweisung und Verzweigung. URL https://de.wikipedia.org/w/index.php?title=Bedingte_Anweisung_und_Verzweigung&oldid=228255344.
- [2] Buchstabenhäufigkeit – Wikipedia — de.wikipedia.org. https://de.wikipedia.org/w/index.php?title=Buchstabh%C3%A4ufigkeit&oldid=239394531#Buchstabh%C3%A4ufigkeit_in_ausgew%C3%A4hlten_Sprachen. [Accessed 25-11-2023].
- [3] Caesar-Verschlüsselung – Wikipedia — de.wikipedia.org. <https://de.wikipedia.org/w/index.php?title=Caesar-Verschl%C3%BCsslung&oldid=238950976>. [Accessed 24-11-2023].
- [4] Nassi-Shneiderman-Diagramm – Wikipedia — de.wikipedia.org. <https://de.wikipedia.org/w/index.php?title=Nassi-Shneiderman-Diagramm&oldid=239335123>. [Accessed 24-11-2023].
- [5] Operatorenliste für Struktogramme — schule-bw.de. <https://www.schule-bw.de/faecher-und-schularten/mathematisch-naturwissenschaftliche-faecher/informatik/material/materialien-zum-neuen-bildungsplan-informatik-an-den-nichtgewerblichen-beruf/operatorenliste-fuer-struktogramme-v2-1-1.pdf/view>. [Accessed 26-11-2023].
- [6] Informatik (Aufbaukurs Informatik Klasse 7) Bildungsplan, 5/29/17 5:28 PM. URL https://bildungsplaene-bw.de/site/bildungsplan/get/documents/lsw/export-pdf/depot-pdf/ALLG/BP2016BW_ALLG_GYM_INF7.pdf.
- [7] Jungblut, D. Informatik Klasse 7 – Informationsgesellschaft und Datensicherheit, 24.02.2022.
- [8] Jungblut, D. Programmieren mit Scratch, 30.01.2022.
- [9] Schaller. lehrerfortbildung-bw.de. https://lehrerfortbildung-bw.de/u_matnatech/informatik/gym/bp2016/fb1/3_rechner_netze/2_kopier/5_caesar/05_run_ab_caesar.pdf, 22.11.2016. [Accessed 24-11-2023].
- [10] Zechnall, D. lehrerfortbildung-bw.de. https://lehrerfortbildung-bw.de/u_matnatech/informatik/gym/bp2016/fb1/3_rechner_netze/2_kopier/5_caesar/05_run_Caesar_und_Mono_AB_Zec_Farbe.pdf, 2019. [Accessed 24-11-2023].
- [11] jupyterlite. GitHub - jupyterlite/demo: JupyterLite demo deployed to GitHub Pages — github.com. <https://github.com/jupyterlite/demo>. [Accessed 25-Mar-2023].
- [12] williamnavaraj. Williamnavaraj/lpyturtle3, 2022.12.19. URL <https://github.com/williamnavaraj/ipyturtle3>.

Die Caesar-Chiffre



Bitte ausschneiden

Verschlüsseln und
Entschlüsseln
mit dem Caesar-Verfahren:

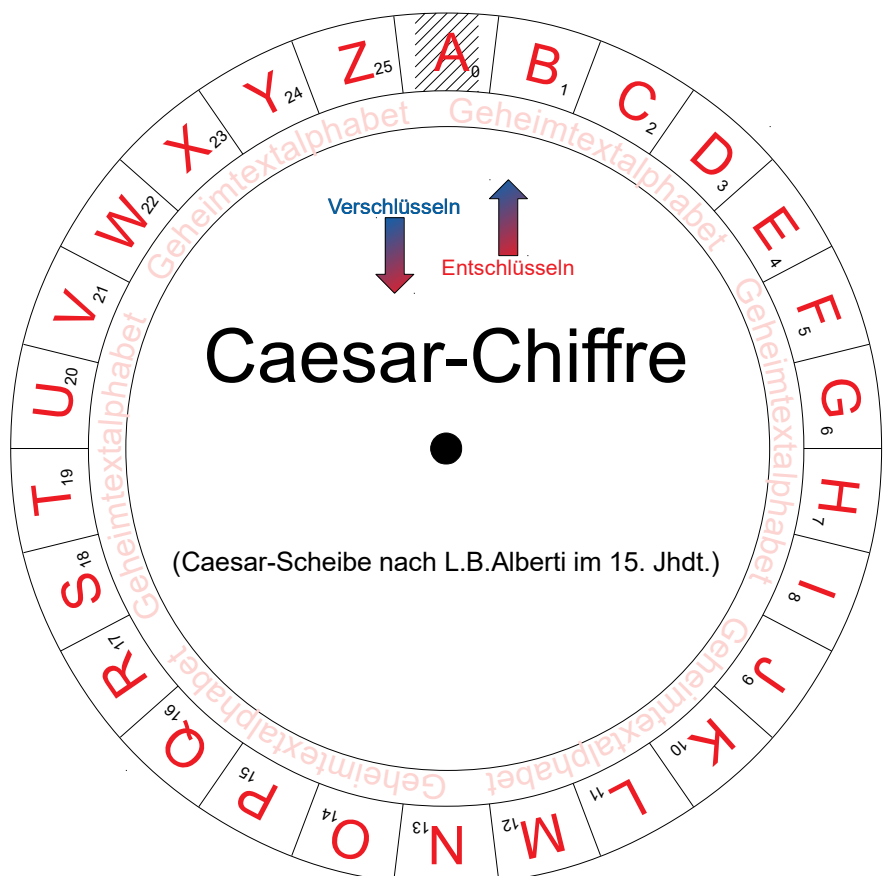
Schlüssel:

Schlüssellänge:

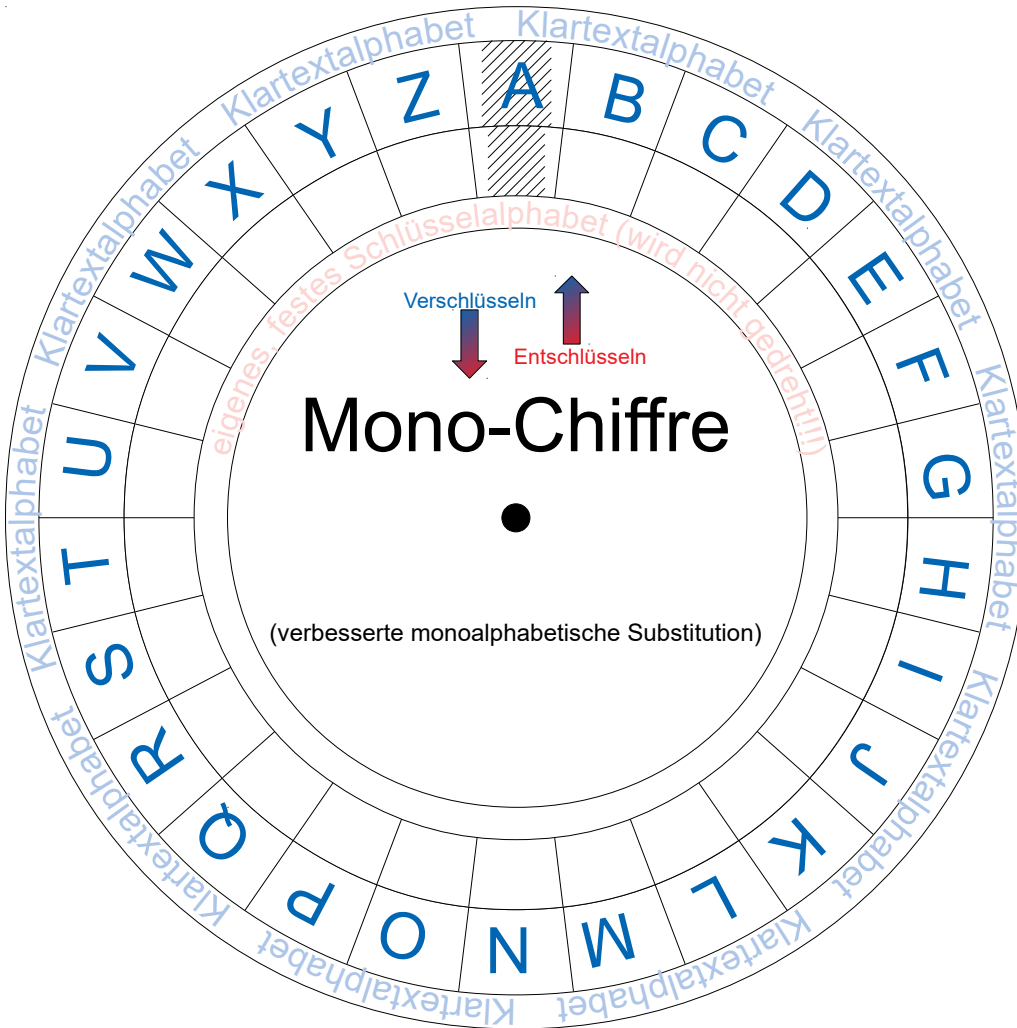
Schlüssellänge (in Bit):

Vorteile:

Nachteile:



Die „Mono-Chiffre“



1. Idee:

Schlüssel:

Schlüssellänge:

Schlüssellänge (in Bit):

Vorteile:

Nachteile:

2. Idee:

Schlüssel:

Schlüssellänge:

Schlüssellänge (in Bit):

Vorteile:

Nachteile:

