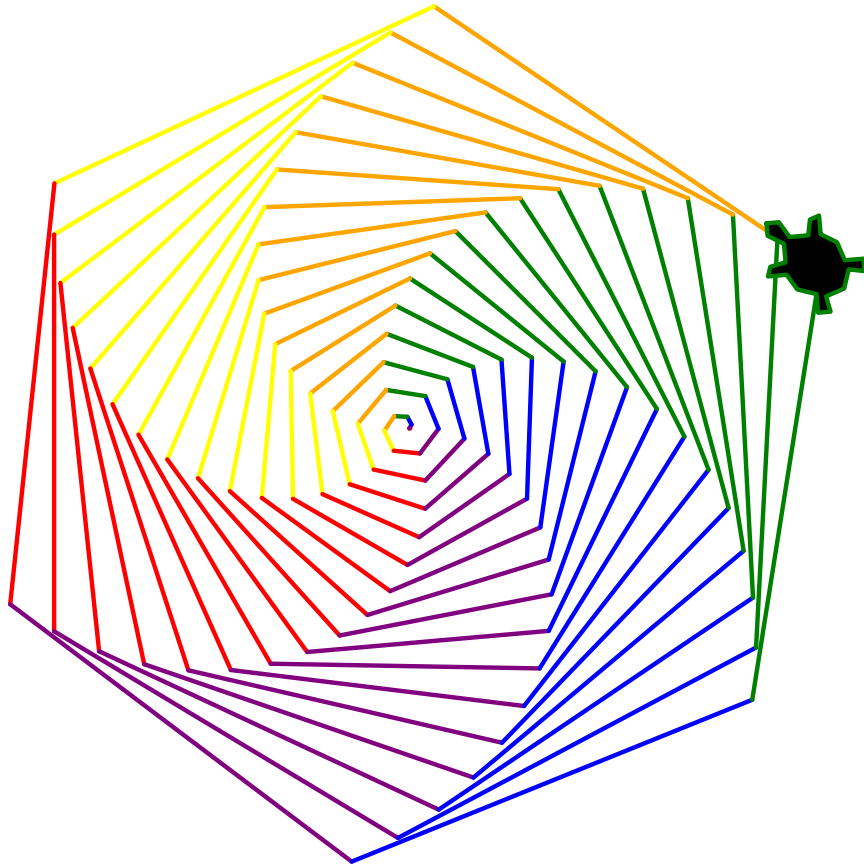


Informatik Klasse 7



Unterrichtsskript

2023

Author	Lukas Meyer-Hilberg
Titel	Informatik Klasse 7
Stand	07.04.2023
Bundesland	Baden-Württemberg
Klasse	7
Lizenz	CC BY-NC-SA 4.0
Website	https://it.hilberg.eu



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:^[5]

Vielen Dank an








Thomas Claus für die Rückmeldung im Unterrichtseinsatz und konstruktive Verbesserungsvorschläge.

Inhaltsverzeichnis

1	Daten und Codierung	4
1.1	Textcodierung und Binärcode	4
1.2	Bildcodierung	5
2	Algorithmen und Programmierung	6
2.1	Variablen in einem Algorithmus	6
2.2	Den Wert von Variablen vergleichen	6
2.3	Mit <code>for</code> -Schleifen Programmcode vereinfachen	8
2.4	Visuelles Programmieren	9
2.5	Mit Bedingungen und Verzweigungen entscheiden	11
2.6	Mit <code>while</code> -Schleifen Programmcode vereinfachen	12
2.7	Mit Schleifen, Bedingungen und Variablen programmieren	13

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 2 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 3 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^[4] enthalten.

Welche Links rufe ich auf?

Tabelle 1: Linkübersicht

Verweis	URL	Beschreibung
s1	https://url.hilberg.eu/s1	Scratch Editor
s2	https://url.hilberg.eu/s2	Blockly Editor
s3	https://url.hilberg.eu/s3	Blockly Editor Demo
s4	https://url.hilberg.eu/s4	Blockly Game
v1	https://url.hilberg.eu/v1	Erklärvideo Algorithmus
skript	https://url.hilberg.eu/skript	Dieses Informatik Skript
lab	https://url.hilberg.eu/lab	Python Lab
lablite	https://url.hilberg.eu/lablite	PythonLite Lab
c0	https://url.hilberg.eu/c0	Textcodierung
c1	https://url.hilberg.eu/c1	Bildcodierung
c2	https://url.hilberg.eu/c2	Algorithmus
p0	https://url.hilberg.eu/p0	Intro Python Notebooks
p1	https://url.hilberg.eu/p1	Python Turtle
p2	https://url.hilberg.eu/p2	Variablen
p3	https://url.hilberg.eu/p3	Bedingungen
p4	https://url.hilberg.eu/p4	Schildkrötenterrarium
p5	https://url.hilberg.eu/p5	Vokabeln

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.

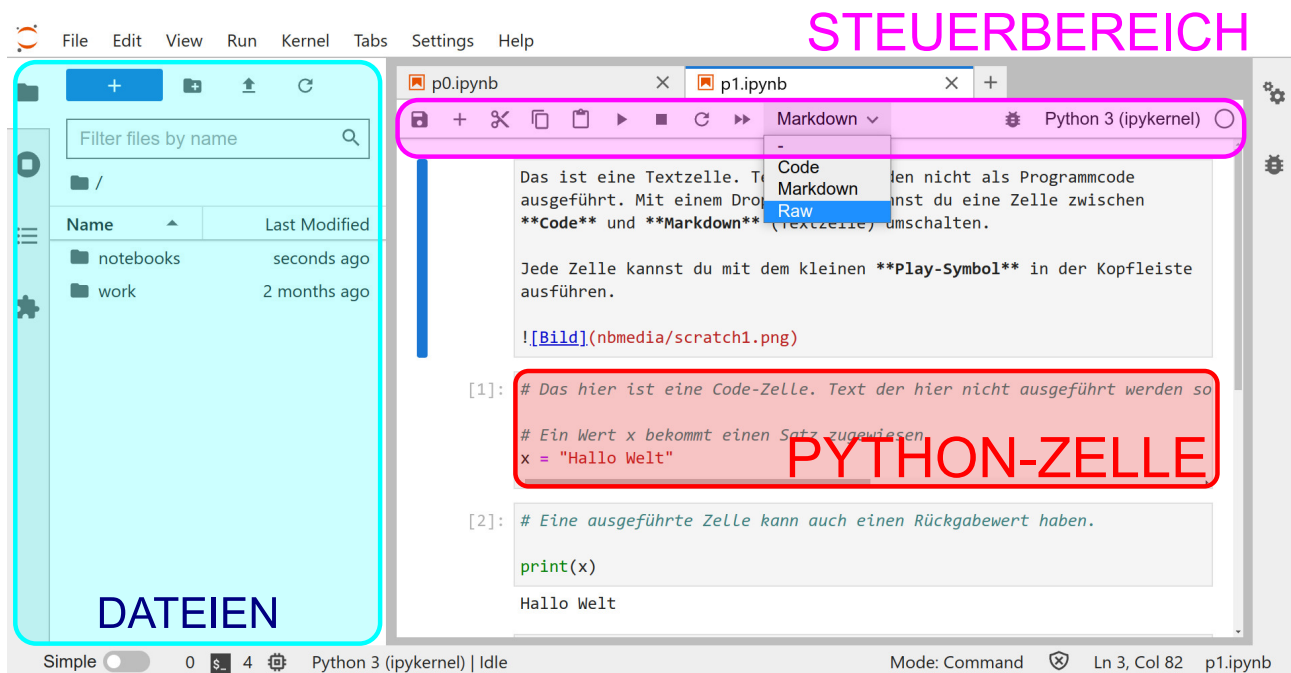


Abbildung 1: Eine Übersicht des „Python-Labs“.

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.

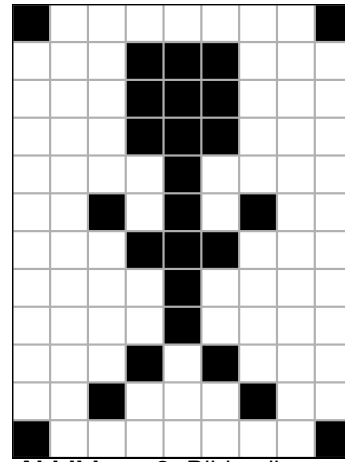


Abbildung 2: Bildcodierung

Bearbeite im Heft/Ordner als: Aufgabe 3



2 Algorithmen und Programmierung

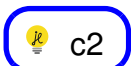
2.1 Variablen in einem Algorithmus

Das Wort „Schule“ wird, genauso wie in Aufgabe 2, in eine Bitfolge umgewandelt. Dabei wandelt der Befehl `binaer` ein Zeichen in die Binärdarstellung (ASCII) um.

In der **Variable** `bitfolge` wird der Stand der Umwandlung gespeichert. Gib in jeder Zeile den Wert von `bitfolge` an.

```
1 bitfolge = ''
2 bitfolge = bitfolge + binaer('S')
3 bitfolge = bitfolge + binaer('c')
4 bitfolge = bitfolge + binaer('h')
5 bitfolge = bitfolge + binaer('u')
6 bitfolge = bitfolge + binaer('l')
7 bitfolge = bitfolge + binaer('e')
8 bitfolge
```

Aufgabe 4



c2

Erstelle einen eigenen Algorithmus der folgenden Satz in eine Bitfolge umwandelt:
Informatik ist toll!

Speichere als: `python4.ipynb`

Aufgabe 5



c2

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 bitfolge = bitfolge + binaer('S')
3 bitfolge = bitfolge + binaer('c')
4 bitfolge = binaer('h')
5 bitfolge = bitfolge + binaer('u')
6 bitfolge = bitfolge + binaer('l')
7 bitfolge = bitfolge + binaer('e')
8 bitfolge
```

Bearbeite im Heft/Ordner als: Aufgabe 5

2.2 Den Wert von Variablen vergleichen

Das Wort „Schule“ wird mit einem Algorithmus als Bitfolge umgewandelt. Danach wird der Wert der Variable `bitfolge` mit zwei anderen Bitfolgen verglichen.

Python-Notebook: Werte vergleichen

```
[1]: bitfolge = ''  
      bitfolge = bitfolge + format(ord('S'), '07b')  
      bitfolge = bitfolge + format(ord('c'), '07b')  
      bitfolge = bitfolge + format(ord('h'), '07b')  
      bitfolge = bitfolge + format(ord('u'), '07b')  
      bitfolge = bitfolge + format(ord('l'), '07b')  
      bitfolge = bitfolge + format(ord('e'), '07b')  
      bitfolge
```

```
[1]: '101001111000111101000111010111011001100101'
```

Hier wurde wohl ein Fehler gemacht in der Umwandlung:

```
[2]: bitfolge == '101001111010111101000111010111011001100101'
```

```
[2]: False
```

Diese Bitfolgen stimmen überein:

```
[3]: bitfolge == '101001111000111101000111010111011001100101'
```

```
[3]: True
```

Eine Bitfolge stimmt nicht mit der Variable `bitfolge` überein. Findest du den Fehler? Welches Wort wird in dem Fall als Bitfolge dargestellt?

[illegible]

Aufgabe 6



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.

2.3 Mit for-Schleifen Programmcode vereinfachen

Python-Notebook: Eine Schleife verkürzt Programmcode

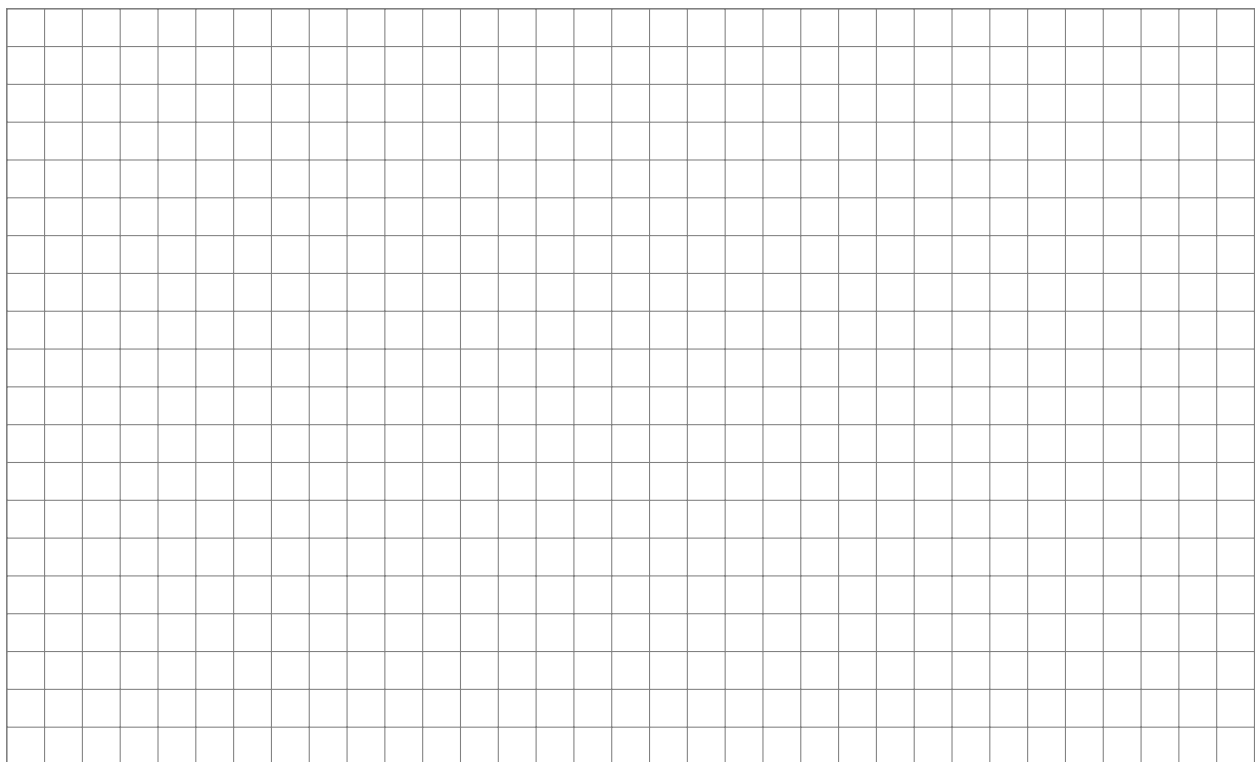
```
[1]: #Wir definieren eine kleine Funktion um eine Dezimalzahl in eine
      ↪ 7-Bit-Binärzahl umzuwandeln.
```

```
def binaer(buchstabe): return format(ord(buchstabe), '07b')
```

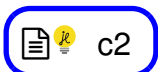
```
[2]: bitfolge=''
      for buchstabe in 'Schule':
          bitfolge = bitfolge + binaer(buchstabe)

      bitfolge
```

```
[2]: '101001111000111101000111010111011001100101'
```



Aufgabe 7



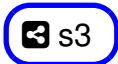
c2

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
```

Bearbeite im Heft/Ordner als: Aufgabe 7

2.4 Visuelles Programmieren



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.

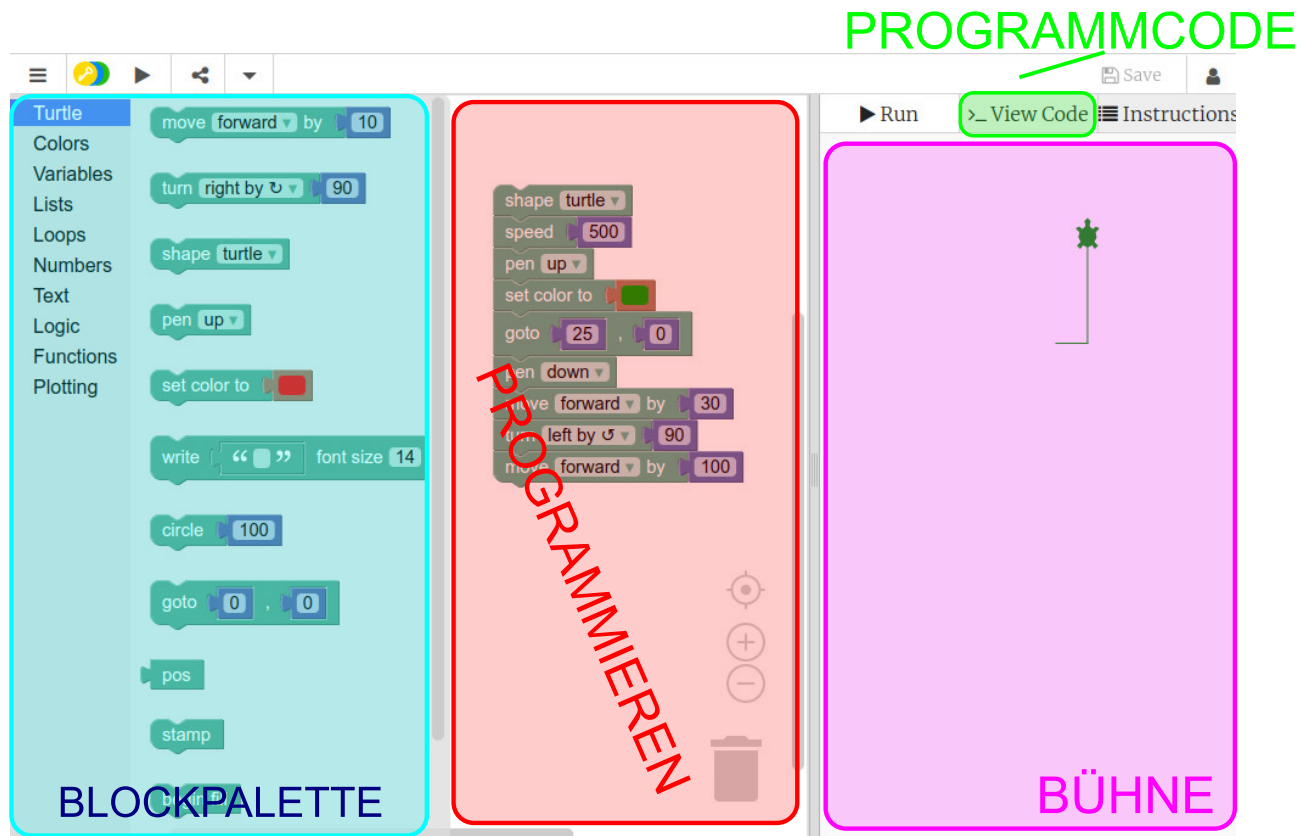


Abbildung 3: Mit Blockly wird visuell programmiert. Der passende Quellcode wird daneben angezeigt.

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.

Aufgabe 8



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: python8.ipynb

2.5 Mit Bedingungen und Verzweigungen entscheiden

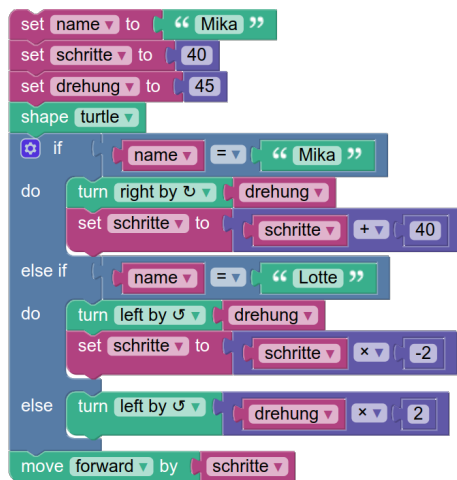


Abbildung 4: Bedingungen und Verzweigungen.



Mit einer bedingten Anwei-

sung^[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: Programmieren mit Bedingungen

Jede Schildkröte entscheidet sich unterschiedlich zu laufen.

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

# Wir erschaffen eine neue Schildkröte
turtle = newturtle()

# Variablen werden angelegt
name = 'Mika'
schritte = 40
drehung = 45
turtle.shape("turtle")

# Die erste Bedingung
if name == 'Mika':
    turtle.right(drehung)
    schritte = schritte + 40

# Die zweite Bedingung
elif name == 'Lotte':
    turtle.left(drehung)
    schritte = schritte * -2

# Führe aus, wenn keine der beiden Bedingungen zutrifft.
else:
    turtle.left(drehung * 2)

turtle.forward(schritte)
```

Aufgabe 9



Öffne das Notebook aus dem Einstiegsbeispiel in Abbildung 4 und ändere den Wert der ersten Festlegung der Variablen `schritte` und `drehung` genau so, dass die Schildkröten Lotte und Mika nach dem Laufen an der gleichen Position stehen.

Speichere als: `python9.ipynb`

2.6 Mit `while`-Schleifen Programmcode vereinfachen

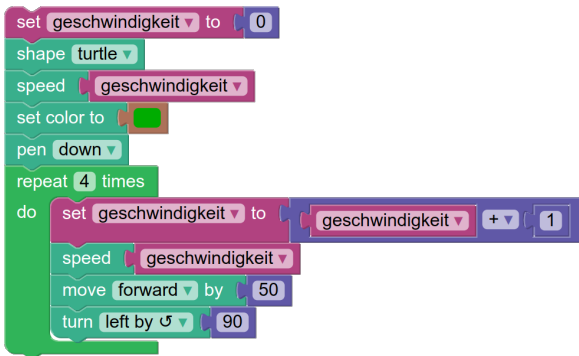


Abbildung 5: Eine `for`-Schleife.

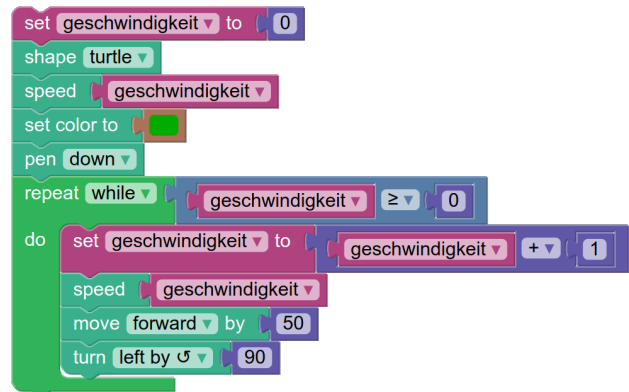
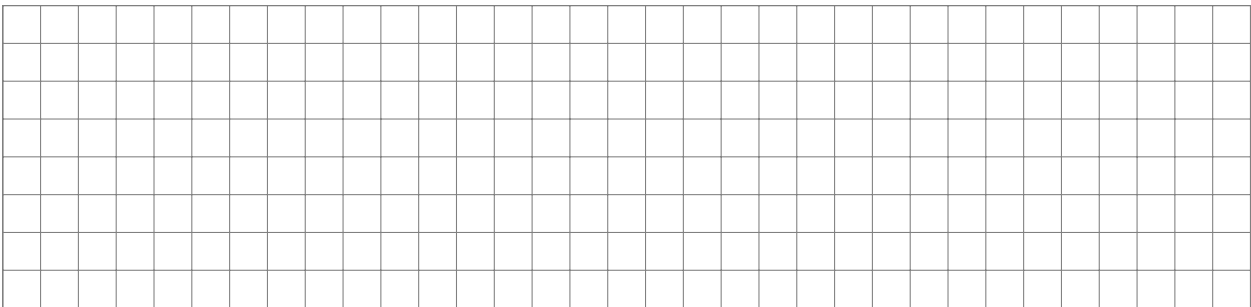


Abbildung 6: Eine `while`-Schleife.

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 10



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: `python10.ipynb`

Aufgabe 11



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

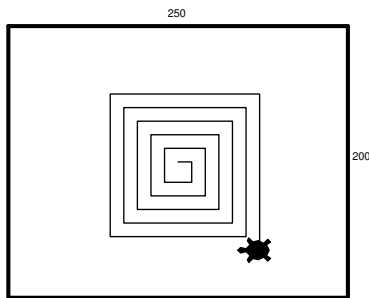
while distance < max_distance:
    turtle.forward(step)
    x, y = turtle.position()

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

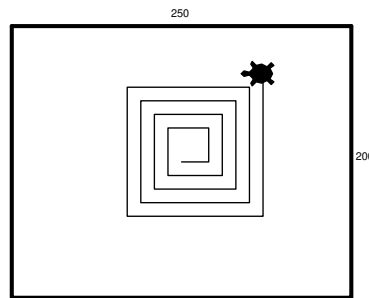
    distance = distance + step
```

Speichere als: python11.ipynb

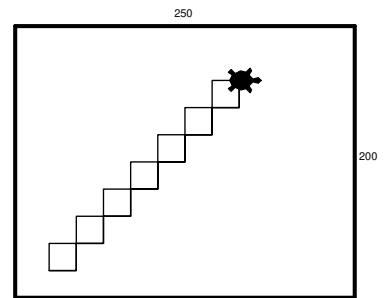
Aufgabe 12



(a)



(b)



(c)

(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)
11
12

```

(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)

```

(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)

```

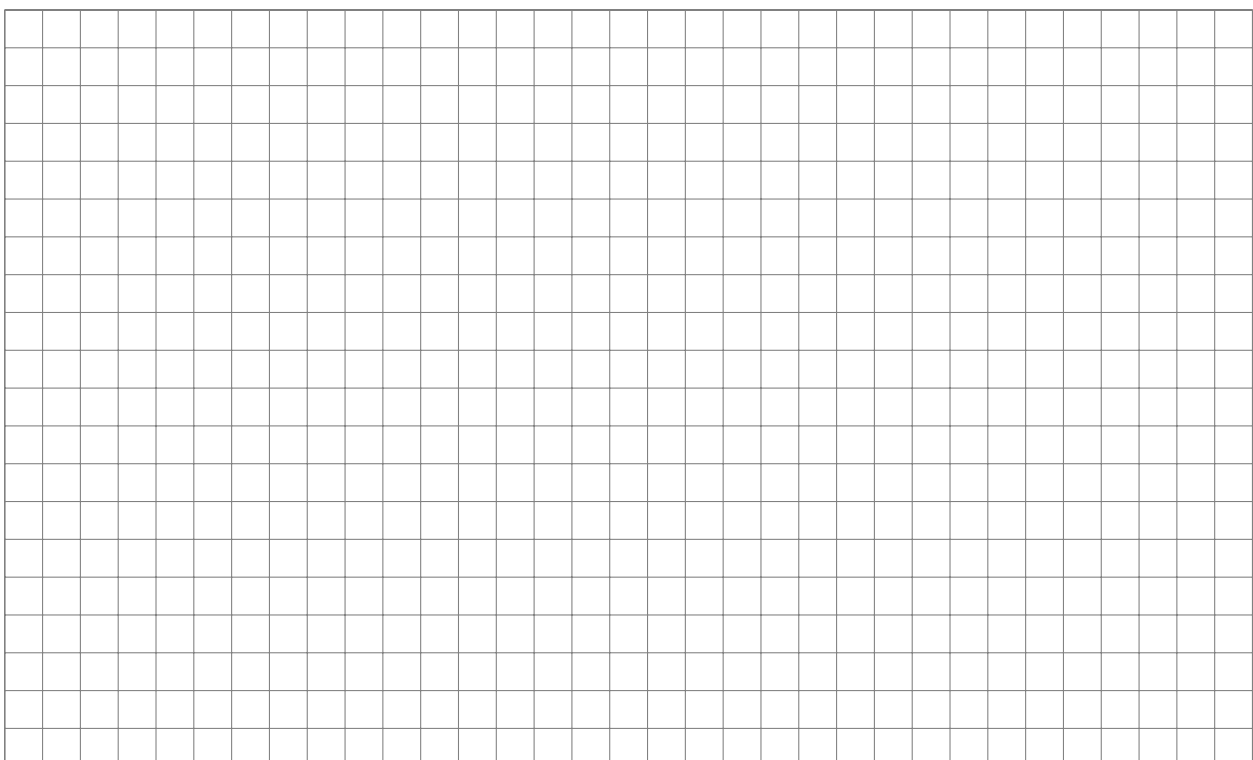
Ordne zwei Bilder aus (a), (b) und (c) zwei Programmcodes (I), (II) und (III) zu. Ein Paar passt nicht zusammen. Begründe deine Zuordnung.

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 13

 p5

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)

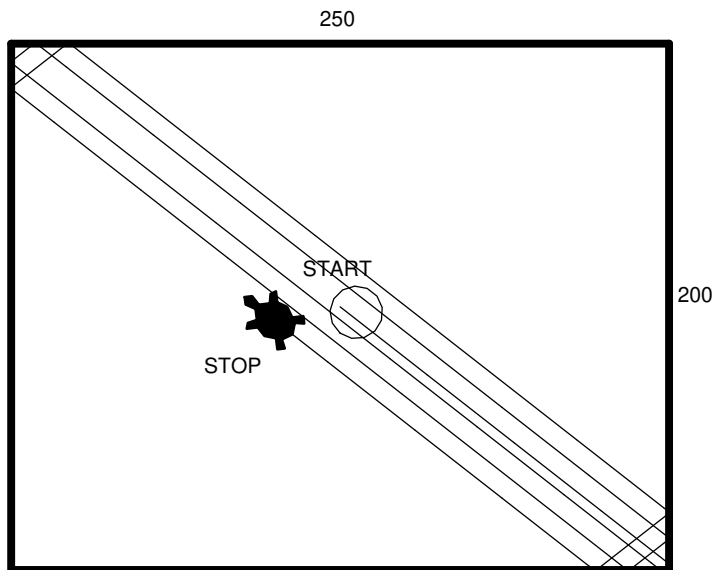
print("Deine Punkte:")
print(points)
```

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: python13.ipynb

Aufgabe 14



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.

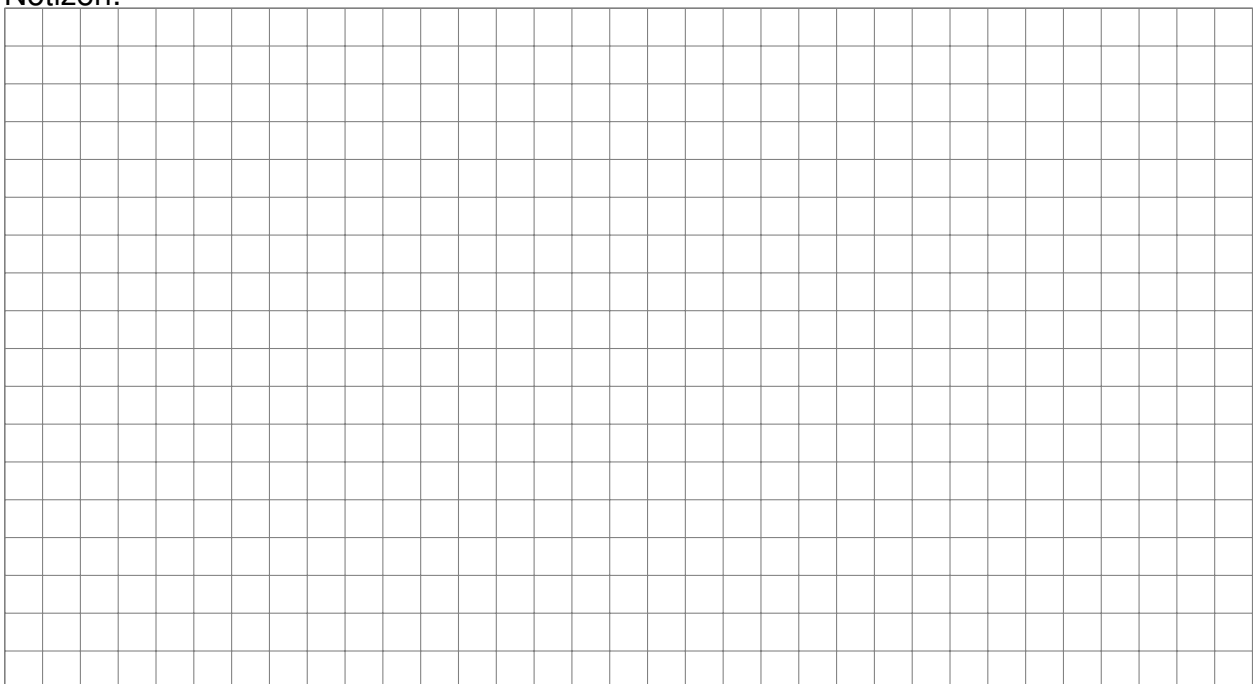
Abbildung 8: Schildi und der Futtertrog.

Passe den Code aus Aufgabe 11 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: `python14.ipynb`

Notizen:



Quellen

- [1] Bedingte Anweisung und Verzweigung. URL https://de.wikipedia.org/w/index.php?title=Bedingte_Anweisung_und_Verzweigung&oldid=228255344.
- [2] 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.
- [3] Jungblut, D. Programmieren mit Scratch, 30.01.2022.
- [4] jupyterlite. GitHub - jupyterlite/demo: JupyterLite demo deployed to GitHub Pages — github.com. <https://github.com/jupyterlite/demo>. [Accessed 25-Mar-2023].
- [5] williamnavaraj. Williamnavaraj/lpyturtle3, 2022.12.19. URL <https://github.com/williamnavaraj/ipyturtle3>.