

Informatik Klasse 8

Unterrichtsbegleitende Materialien und Aufgaben

- Wie rechnet ein Computer?
 - Zahlendarstellung
 - Addition von Binärzahlen
 - Logische Schaltungen
 - Kodierung von Buchstaben
- Programmierung
 - Scratch
 - Turtle-Grafiken mit Python
 - Einführung in Python
 - Programmierung einfacher Spiele
- Kryptologie
 - Die CESAR-Verschlüsselung
 - CESAR-Verschlüsselung mit Python programmieren
- Bildverarbeitung
 - Das Portable Anymap Bildformat
 - Filter für das Portable Anymap Format mit python
 - Farbmischung
- Arbeiten mit Anwendungsprogrammen
- Empfohlene Software
 - Im Unterricht benutzte Software

Wie rechnet ein Computer?

Zahlendarstellung

Das Dezimalsystem

Das Dezimalsystem kennen wir alle. Das sind die Zahlen, die wir lesen und sofort verstehen können. Schauen wir noch einmal, wie diese Zahlen mathematisch funktionieren. Jede Ziffer bekommt eine unterschiedliche Bedeutung, je nachdem, wo sie in der Zahl steht. Ganz rechts sind die "Einer". Das heißt, eine 1 bedeutet auch 1. An der zweiten Stelle von rechts bedeutet die 1 schon 10 usw. Das lässt sich mathematisch so darstellen:

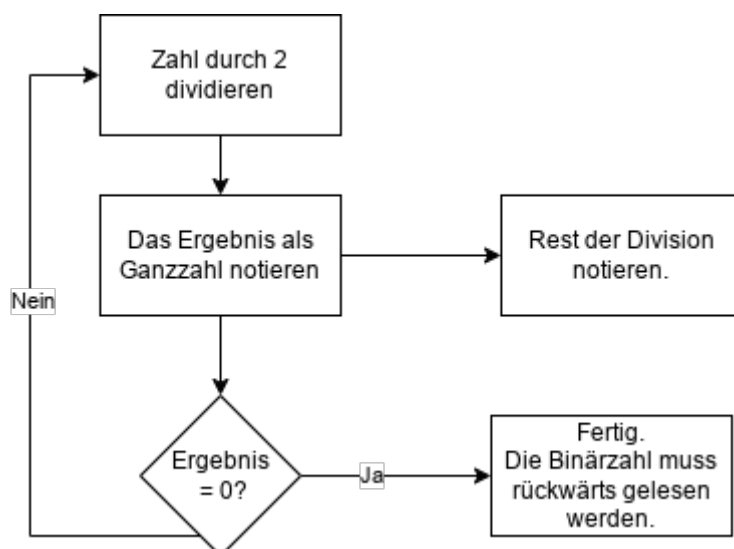
$$4711_{10} = 4 * 10^3 + 7 * 10^2 + 1 * 10^1 + 1 * 10^0$$

Dasselbe funktioniert auch, wenn wir weniger Ziffern zur Verfügung haben, z.B. nur zwei, nämlich 0 und 1:

$$1101_2 = 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 = 1 * 8 + 1 * 4 + 0 * 2 + 1 * 1 = 13_{10}$$

Auf diese Weise können Binärzahlen in Dezimalzahlen umgewandelt werden. Zur umgekehrten Rechnung kann man einen Algorithmus verwenden.

Algorithmus zum Berechnen einer Binärzahl



Beispiel:

$$190:2=95 \text{ Rest } 0$$

$$95:2=47 \text{ Rest } 1$$

$$47:2=23 \text{ Rest } 1$$

$$23:2=11 \text{ Rest } 1$$

$$11:2=5 \text{ Rest } 1$$

$$5:2=2 \text{ Rest } 1$$

$$2:2=1 \text{ Rest } 0$$

$$1:2=0 \text{ Rest } 1$$

Die Binärzahl lautet dann 10111110. Sie muss also von unten nach oben gelesen werden.

Addition von Binärzahlen

Addition von Dezimalzahlen

$$\begin{array}{r} 2752 \\ + 4261 \\ \hline 7013 \end{array}$$

Addition von Binärzahlen

$$\begin{array}{r} 10010 \\ + 100111 \\ \hline 111001 \end{array}$$

Die Addition funktioniert wie die Addition von Dezimalzahlen. Bei Dezimalzahlen stehen 10 Ziffern zur Verfügung. Ab der 10. Ziffer kommt es zu einem Überlauf, d. h., sie lässt sich nicht mehr darstellen. Daher muss die 10er-Stelle als Übertrag notiert werden und bei der Addition der nächsten Stelle berücksichtigt werden. Bei den Binärzahlen stehen nur zwei Ziffern zur Verfügung. Daher kommt es schon früher zu einem Überlauf. Dieser muss genauso berücksichtigt werden wie bei der Addition von Dezimalzahlen.

Wie rechnet ein Computer?

Logische Schaltungen

Ein Computer kann nur Nullen und Einsen. Das weiß mehr oder weniger jeder. Doch wie kann ein Computer damit rechnen? Darum geht es in diesem Kapitel.

Simulation von logischen Schaltungen

Zur Übung simulieren wir die Schaltungen, aus denen ein Computerchip aufgebaut ist, mit Digital Logic Sim. Die ZIP-Datei muss nur entpackt werden und das Programm wird dann mit einem Doppelklick auf `Digital LLogic Sim.exe` gestartet.

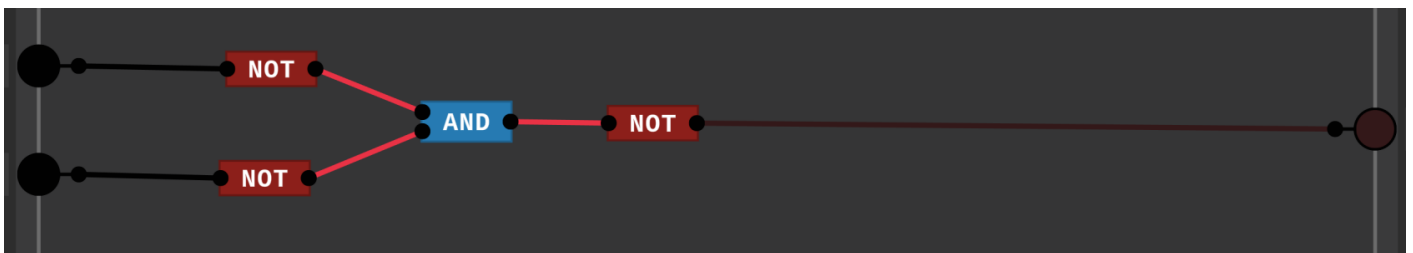
Mithilfe dieses Programms soll aus einer AND- und einer NOT-Schaltung ein Addierer aufgebaut werden, mit dem man zwei 1-Bit Binärzahlen addieren kann. Wer weitermachen möchte, kann damit natürlich auch einen 4-Bit Addierer bauen. Unter Windows werden die Daten unter diesem Verzeichnis gespeichert:

```
C:\Benutzer\BENUTZERNAME\AppData\LocalLow\SebastianLague\Digital Logic Sim\V1\Projects
```

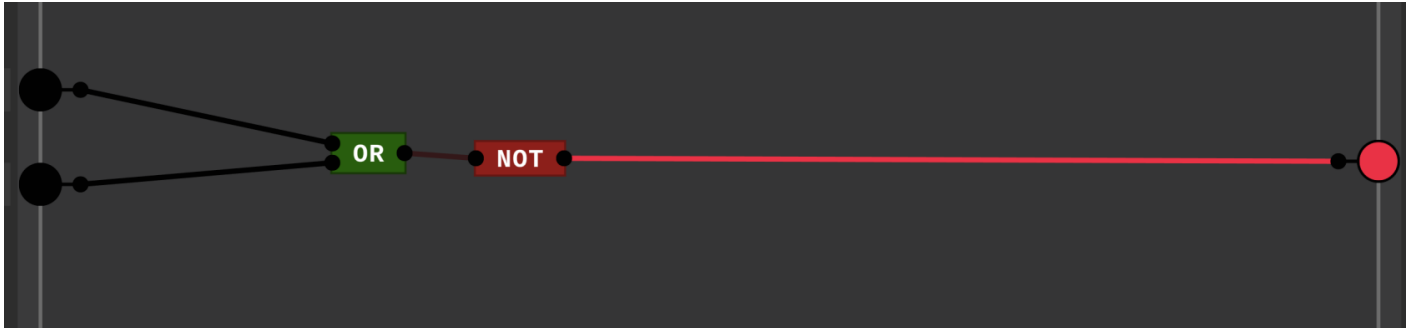
Um das Verzeichnis im *Windows Explorer* anzuzeigen muss unter *Ansicht* noch *Ausgeblendete Elemente* angewählt werden.

Hier sind die Schaltbilder der einzelnen Elemente. Diese müssen in dieser Reihenfolge angelegt werden.

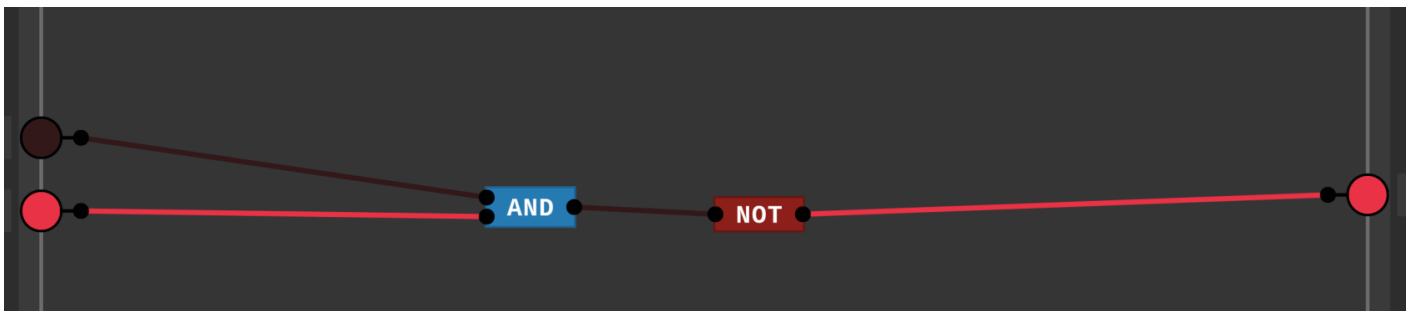
OR



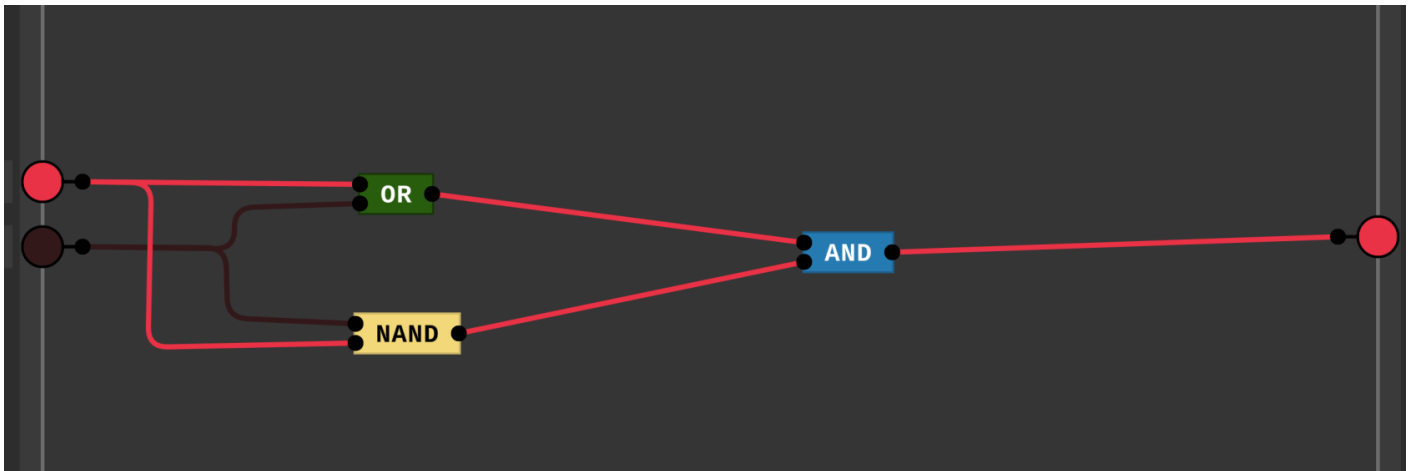
NOR



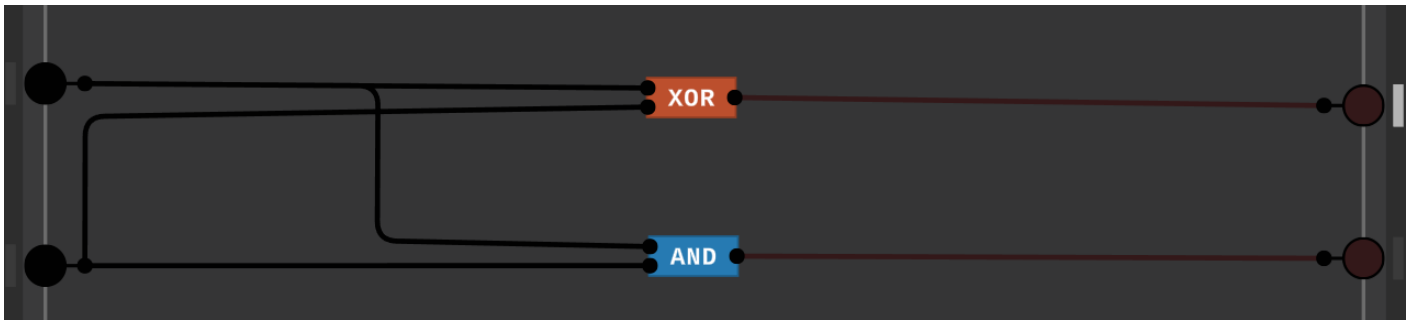
NAND



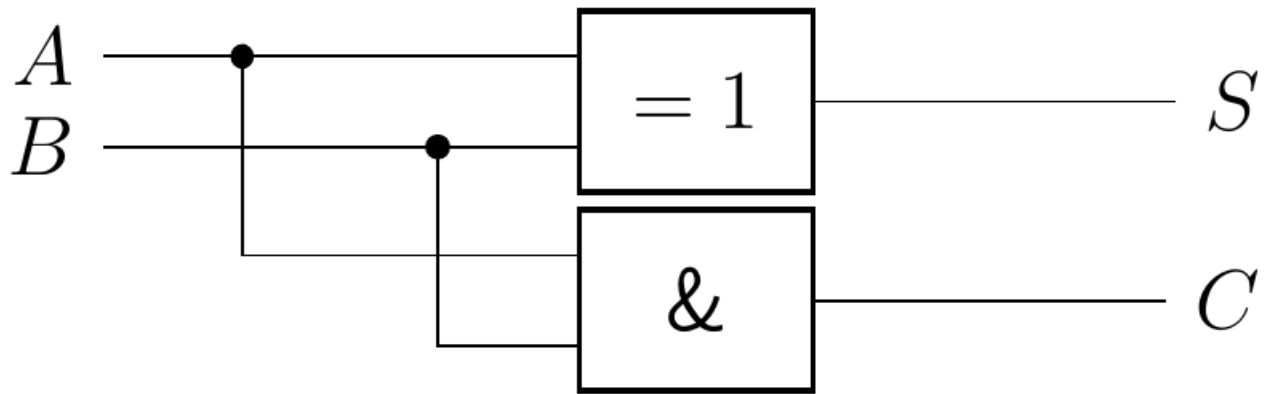
XOR



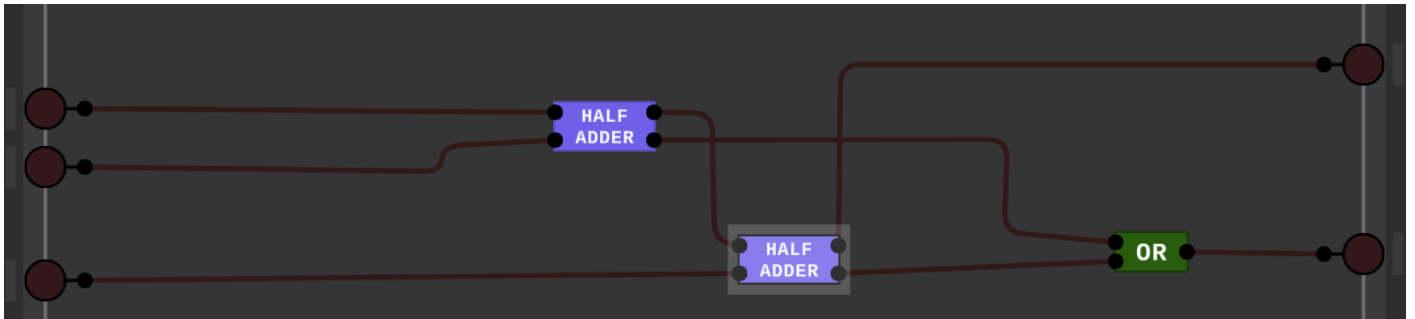
HALF ADDER



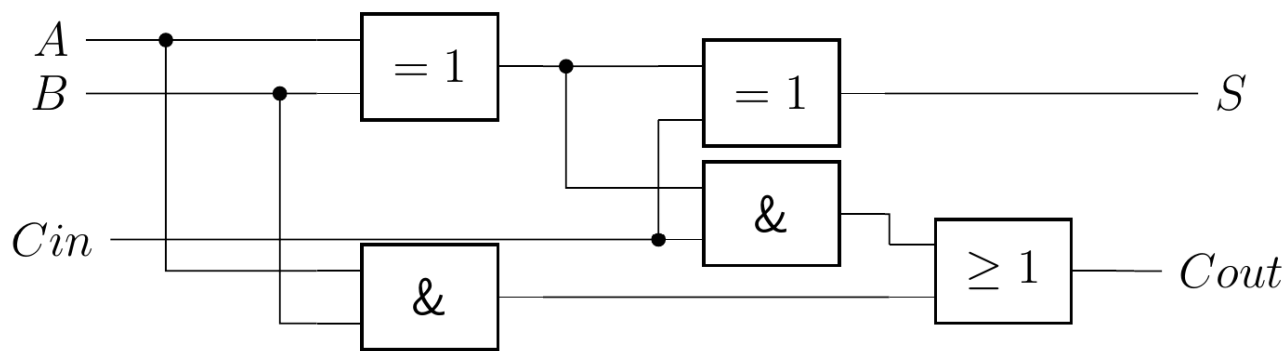
oder die Version mit Schaltsymbolen:



ADDER

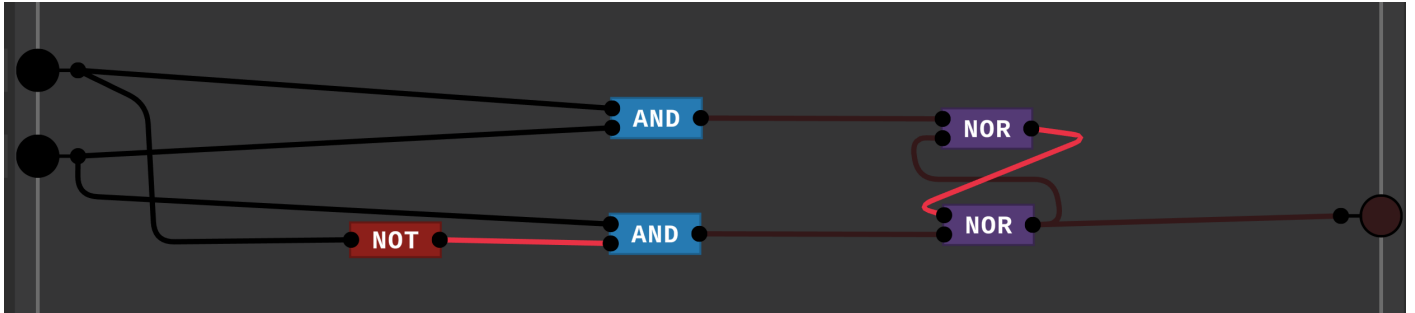


oder eine Version ohne den Haladdierer:

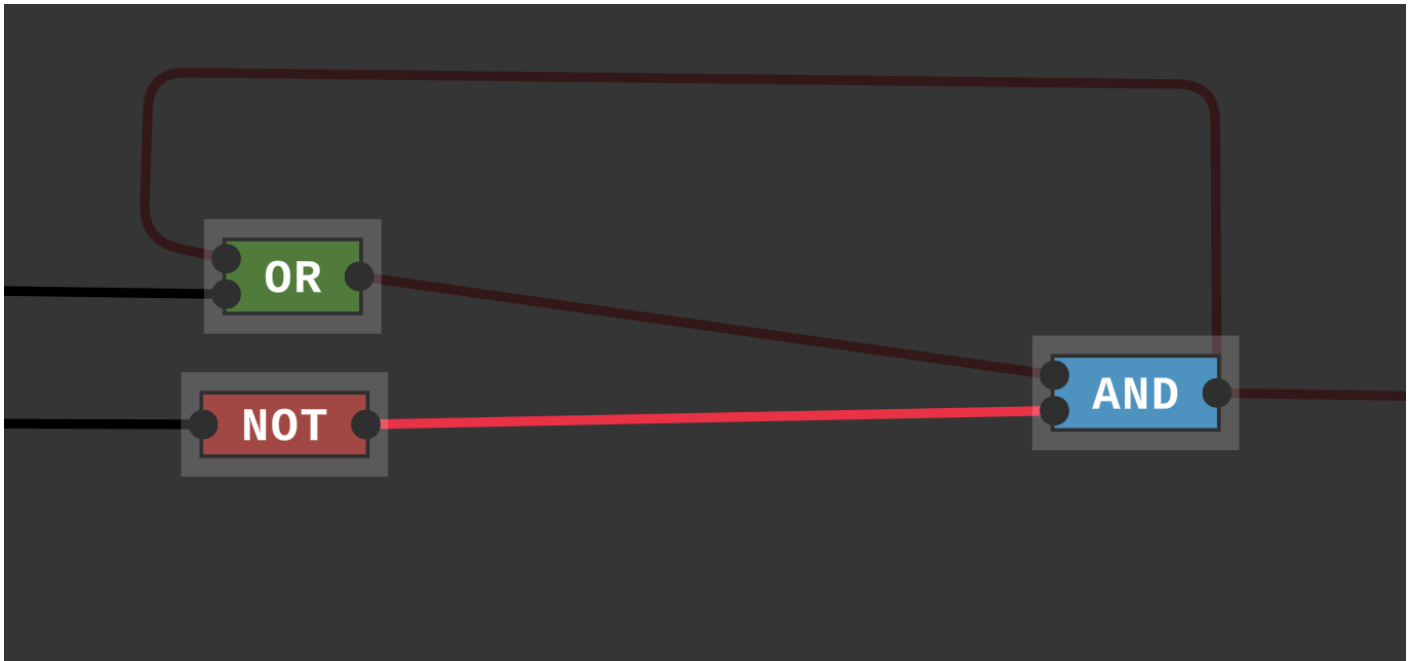


Simulation von Computerspeicher

D-Latch



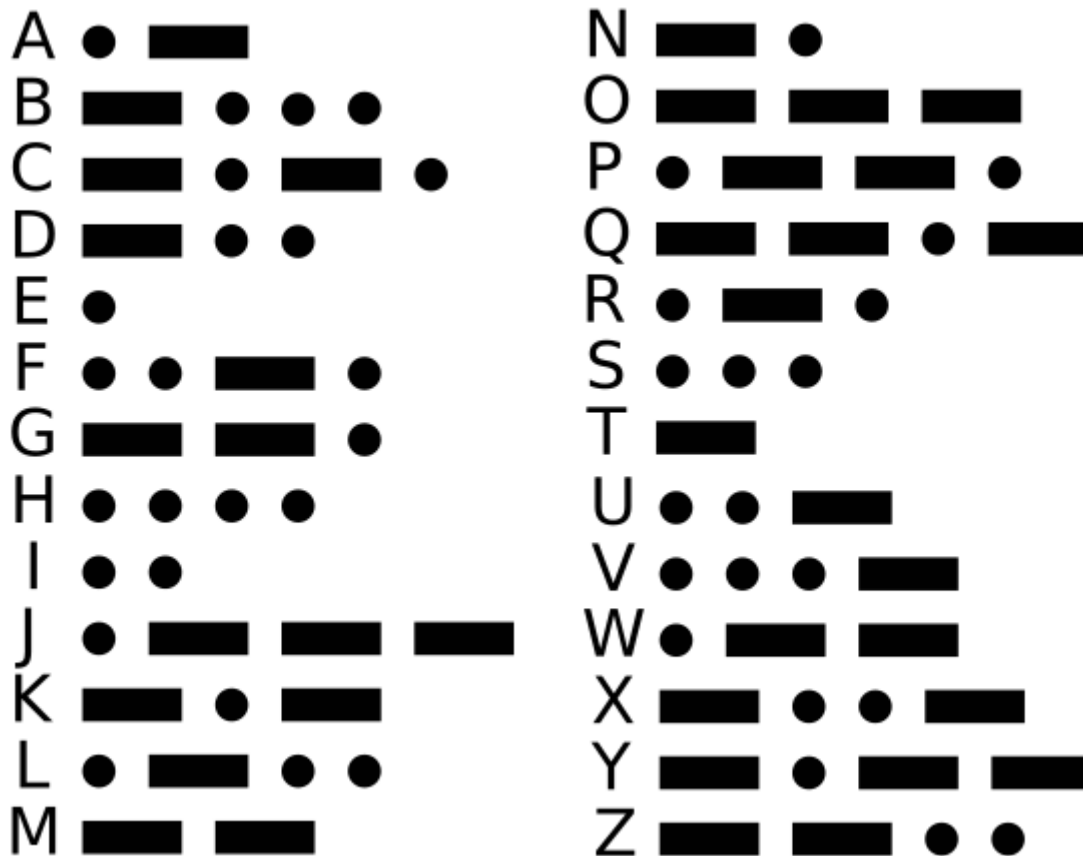
SR-Latch



Wie rechnet ein Computer?

Kodierung von Buchstaben

Ein Vorläufer der heutigen Kodierung von Buchstaben ist das Morsealphabet.



So wie ein Computerchip, konnte man mit Morsegeräten nur zwischen einem kurzen und einem langen Signal unterscheiden. Jeder Buchstabe bekam also eine Kombination aus kurzen und langen Impulsen zugewiesen.

Für die Kodierung von Buchstaben und anderen Zeichen auf dem Computer wurde auch eine Kodierungstabelle entwickelt. Die erste Tabelle hier ASCII (American Standard Code for Information Interchange)

<div><div><div><div><div>b₇</div><div>b₆</div><div>b₅</div></div><div><div><div><div>b₄</div><div>b₃</div><div>b₂</div><div>b₁</div></div><div>Bits</div></div></div><div><div>Column</div><div>Row</div></div></div></div></div>					0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1	
	0	1	2	3	4	5	6	7					
0	0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	0	1	0	0	4	EOT	DC4	\$	4	D	T	d	t
0	0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
0	1	0	0	0	8	BS	CAN	(8	H	X	h	x
0	1	0	0	1	9	HT	EM)	9	I	Y	i	y
0	1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
0	1	0	1	1	11	VT	ESC	+	;	K	[k	{
0	1	1	0	0	12	FF	FS	,	<	L	\	l	
0	1	1	0	1	13	CR	GS	—	=	M]	m	}
0	1	1	1	0	14	SO	RS	.	>	N	^	n	~
0	1	1	1	1	15	SI	US	/	?	O	_	o	DEL

Weitere Informationen

Programmierung

Scratch

Adresse zu Scratch

Eine Anmeldung ist nicht erforderlich, um mit Scratch zu programmieren. Die eigenen Programme können heruntergeladen und auch wieder hochgeladen werden. Damit ist eine Anmeldung auch nicht empfohlen.

Online Scratch Editor

Man kann Scratch auch als alleinstehendes Programm herunterladen. Das ist zurzeit nicht zu empfehlen, da die Version deutlich älter ist.

Offline Scratch Editor

Kurzanleitung

Das Prinzip von Scratch ist mit dem von Lego zu vergleichen. Verschiedene Bausteine der Programmierung können zusammengesteckt werden. Ob verschiedene Elemente zusammen passen wird durch die Form deutlich gemacht. Die Grundelemente von Programmiersprachen gibt es auch in Scratch. So gibt es Schleifen, Bedingungen, Variablen und auch Zufallszahlen. Besonders praktisch für die Einführung in die Robotik ist, dass es auch virtuelle Sensoren gibt. So kann z.b. der Bildschirmrand oder Hindernisse erkannt werden. Damit können wir viele Funktionen der Roboter hier virtuell darstellen.

Einführungskurs auf appcamps

Übungsprogramme

1. Schreibe ein Programm, damit sich der Sprite in einem Quadrat/Kreis bewegt. --> Schleifen
2. Definiere diesen Bewegungsablauf in einem eigenen Block. Rufe diesen Block in deinem Programm wiederholt auf. --> Schleifen und Definition von blöcken (Funktionen)

3. Überprüfe zu Beginn des Programms eine PIN ab. Ist die PIN falsch, wird das Programm nicht ausgeführt. --> Bedingung und Variable
4. Schreibe ein Programm, in dem sich die Figur zufällig bewegt. Die Figur muss Essen finden. Findet sie nicht rechtzeitig etwas zu Essen, muss sie verhungern. --> Schleifen, Bedingungen, Variablen, Zufallszahlen, Sensorik, Zähler

Turtle-Grafiken mit Python

Um mit Python Turtle-Grafiken zu zeichnen, muss man nur die Bibliothek "turtle" importieren: `from turtle import *` Beispiel für eine Grafik:

```
from turtle import *  
for i in range(0,4):  
    forward(100)  
    right(90)
```

Dies zeichnet ein Quadrat auf dem Bildschirm. Etwas interessanter ist die Spirale:

```
from turtle import *  
for steps in range(100):  
    for c in ('blue', 'red', 'green'):  
        color(c)  
        forward(steps)  
        right(30)
```

[Die offizielle Dokumentation der Turtle](#)

Programmierung

Einführung in Python

Die Einführung findet ihr [hier](#).

Programmierung einfacher Spiele

Wörterraten

Kopiere den Quellcode in Thonny und führe ihn aus. Bearbeite folgende Arbeitsaufträge:

1. Finde heraus, was du tun musst, um das Spiel zu spielen.
2. Schau dir den Quellcode an und vollziehe nach, wie das Programm abläuft.
3. Verändere das Programm nach deinen Vorstellungen.

```
# Quelle: https://www.geeksforgeeks.org/python-program-for-word-guessing-game/?ref=lbp
import random

# library that we use in order to choose
# random words from a list of words

name = input("What is your name? ")

# Here the user is asked to enter the name first

print("Good Luck ! ", name)

words = ['rainbow', 'computer', 'science', 'programming',
         'python', 'mathematics', 'player', 'condition',
         'reverse', 'water', 'board', 'geeks']

# Function will choose one random
# word from this list of words
word = random.choice(words)

print("Guess the characters")
```

```
guesses = ""

# any number of turns can be used here
turns = 12

while turns > 0:

    # counts the number of times a user fails
    failed = 0

    # all characters from the input
    # word taking one at a time.
    for char in word:

        # comparing that character with
        # the character in guesses
        if char in guesses:
            print(char, end=" ")

        else:
            print("_")

            # for every failure 1 will be
            # incremented in failure
            failed += 1

    if failed == 0:
        # user will win the game if failure is 0
        # and 'You Win' will be given as output
        print("You Win")

        # this print the correct word
        print("The word is: ", word)
        break

    # if user has input the wrong alphabet then
    # it will ask user to enter another alphabet
    print()
    guess = input("guess a character:")
```

```
# every input character will be stored in guesses
```

```
guesses += guess
```

```
# check input with the character in word
```

```
if guess not in word:
```

```
    turns -= 1
```

```
# if the character doesn't match the word
```

```
# then "Wrong" will be given as output
```

```
print("Wrong")
```

```
# this will print the number of
```

```
# turns left for the user
```

```
print("You have", + turns, 'more guesses')
```

```
if turns == 0:
```

```
    print("You Loose")
```

Erweiterungen

Wortliste in eigener Datei

```
# definiere die Datei, die du verwenden möchtest.
```

```
file = "wordlist.txt"
```

```
with open(file, "r") as f: # Öffne die Datei im Nur-Lesen-Modus
```

```
    wordlist = f.read().splitlines() # Teile die Datei zeilenweise.
```

```
print(wordlist)
```

Ausgabe der Wörter besser formatieren

Die Wörter werden in diesem Programm von oben nach unten geschrieben. Das liegt daran, dass der `print`-Befehl mit einem *carriage return* endet. Speichere die Lösungen zunächst in einer Variablen und gib diese am Schluss mit dem `print`-Befehl aus.

Anzahl Runden

Definiere die Anzahl der Fehlversuche in Abhängigkeit der Wortlänge.

Speichern der Gewinner

```
from datetime import datetime

with open("winner.txt", "a") as text_file:
    text_file.write(f"{name} hat gewonnen. {datetime.now()}\r\n") # oder nur \n, falls das nicht funktioniert.
```

Mastermind

Kopiere den Quellcode in Thonny und führe ihn aus. Bearbeite folgende Arbeitsaufträge:

1. Finde heraus, was du tun musst, um das Spiel zu spielen.
2. Schau dir den Quellcode an und vollziehe nach, wie das Programm abläuft.
3. Verändere das Programm nach deinen Vorstellungen.

```
# Quelle: https://www.geeksforgeeks.org/mastermind-game-using-python/?ref=lbp
import random

# the .randrange() function generates a
# random number within the specified range.
num = random.randrange(1000, 10000)

n = int(input("Guess the 4 digit number:"))

# condition to test equality of the
# guess made. Program terminates if true.
if (n == num):
    print("Great! You guessed the number in just 1 try! You're a Mastermind!")
else:
    # ctr variable initialized. It will keep count of
    # the number of tries the Player takes to guess the number.
    ctr = 0

    # while loop repeats as long as the
```

```

# Player fails to guess the number correctly.
while (n != num):
    # variable increments every time the loop
    # is executed, giving an idea of how many
    # guesses were made.
    ctr += 1

count = 0

# explicit type conversion of an integer to
# a string in order to ease extraction of digits
n = str(n)

# explicit type conversion of a string to an integer
num = str(num)

# correct[] list stores digits which are correct
correct = ['X']*4
print(correct)

# for loop runs 4 times since the number has 4 digits.
for i in range(0, 4):

    # checking for equality of digits
    if (n[i] == num[i]):
        # number of digits guessed correctly increments
        count += 1
        # hence, the digit is stored in correct[].
        correct[i] = n[i]
    else:
        continue

# when not all the digits are guessed correctly.
if (count < 4) and (count != 0): #- this condition is not needed as we are starting with the condition,
n!=num, which is, count<4

    print("Not quite the number. But you did get ",
          count, " digit(s) correct!")

# second code is not supposed to print the guessed numbers, from the sample output, here I get we are
not recording the position of the guess, but count. But as per the explanation, the code should not print the
guessed numbers, rather give their count.

```

```

        # print("Also these numbers in your input were correct.")
        # for k in correct:
        # print(k, end=' ')

print('\n')
print('\n')
n = int(input("Enter your next choice of numbers: "))

# when none of the digits are guessed correctly.
elif (count == 0):
    print("None of the numbers in your input match.")
    n = int(input("Enter your next choice of numbers: "))

# condition for equality.
if n == num:
    # ctr must be incremented when the n==num gets executed as we have the other incrementation in the
n!=num condition
    ctr+=1
    print("You've become a Mastermind!")
    print("It took you only", ctr, "tries.")

```

Siegerehrung

Die Siegerehrung kann in jedes Spiel eingebaut werden. Der Import muss natürlich an den Anfang der Datei.

```

import time
width=70
speed=0.015
for x in range(0,3):
    for i in range(width,0,-1):
        out = ""
        for j in range(width,i,-1):
            out += " "
        out += "You win"

```

```
print(out)
time.sleep(speed)
```

```
for i in range(0,width):
    out = ""
    for j in range(i,width):
        out += " "
    out += "You win"
    print(out)
    time.sleep(speed)
```

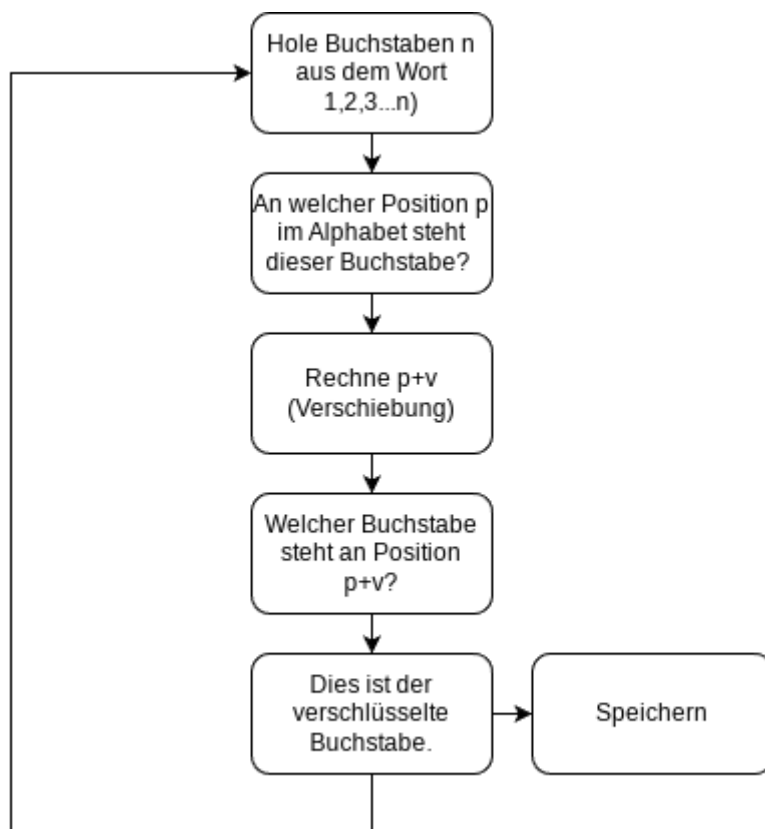
Kryptologie

Die CESAR-Verschlüsselung

Die Verschlüsselungsscheibe

CESAR-Verschlüsselung mit Python programmieren

Algorithmus für die Verschlüsselung mit CESAR



Für die Programmierung einer Ver- und Entschlüsselung mit der CAESAR-Verschlüsselung benötigt man folgende Elemente:

Beispiel für eine Modularechnung

```
for i in range(100):  
    x=i%5  
    print(x)
```

Die Modulorechnung benötigt man für die Verschiebung, wenn die Rechnung $p+v$ (siehe Algorithmus) über den letzten Buchstaben hinausgeht.

Funktionen zu Strings

- `lower()` gibt einen String in Kleinbuchstaben aus. Der entschlüsselte Text ist immer in Kleinbuchstaben.
- `len()` gibt die Länge eines Strings (oder einer Liste) zurück.
- `index()` gibt die Position des Zeichens in einem String zurück.

Iteration über einen String

Eine Iteration ist, wenn man in einem Algorithmus eine bestimmte Anweisung in einer Schleife immer wieder ausführt, wobei sich mindestens ein Parameter mit jedem Durchgang ändert. Dies ist häufig ein Zähler. In dem folgenden Beispiel wird mit jedem Durchgang der Schleife ein neuer Buchstabe aus dem String `s` geholt.

```
s = "Theodor-Heuss-Schule"  
print(s.lower())  
for c in s:  
    a = s.index(c)  
    print("Buchstabe ", c, " taucht erstmalig an Position ", a, " auf.")  
  
print(f"Der erste Buchstabe in {s}: {s[0]}")  
for i in range(len(s)):  
    print(f"Dies ist der {i+1}. Buchstabe: {s[i]}")
```

CESAR Verschlüsselung

Dies hier ist das Grundgerüst für ein Skript, das die CESAR-Verschlüsselung realisiert. Es muss nur noch ein wenig Code ergänzt werden. Natürlich gibt es auch andere Lösungen.

```
ALPHABET = "abcdefghijklmnopqrstuvwxyz"
```

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

```
    encryptedText = ""
```

```
    text = text.lower()
```

```
    # Hier wird der Text verschlüsselt.
```

```
    return encryptedText.upper()
```

```
def decrypt(text, key):
```

```
    return ""
```

```
text = input("Den Text eingeben: ")
```

```
key = int(input("Den Schlüssel eingeben: "))
```

```
geheim = encrypt(text, key)
```

```
klar = decrypt(geheim, key)
```

```
print("Der verschlüsselte Text: ", geheim)
```

```
print("Der aus dem Geheimtext entschlüsselte Text: ", klar)
```

Bildverarbeitung

Das Portable Anymap Bildformat

Das Portable Anymap Bildformat gibt es in drei Versionen. Diese Bilder lassen sich mit einem Texteditor erstellen. Jedes Bildformat beginnt mit einer „magischen Zahl“ zur Identifizierung des Bildtyps, gefolgt von der Auflösung des Bildes und evtl. der Anzahl der Grau-, bzw. der Farbstufen. Die drei Bildarten lassen sich so definieren:

Schwarz-Weiß Bild

```
P1
11 7
1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0
```

Dieses Bild stellt ein schwarzes T auf weißem Hintergrund dar.

Graustufenbild

```
P2
11 7
15
15 15 15 15 15 15 15 15 15 15 15
0 0 0 0 0 13 0 0 0 0 0
0 0 0 0 0 11 0 0 0 0 0
0 0 0 0 0 9 0 0 0 0 0
0 0 0 0 0 8 0 0 0 0 0
```

0 0 0 0 0 4 0 0 0 0 0

Farbbild

Dieses Bild stellt ein farbiges T dar.

Filter für das Portable Anymap Format mit python

```
from tkinter import *
from PIL import Image, ImageTk
# Ändere nichts oberhalb dieser Zeile

# Schreibe hier den Filter für P1 Dateien
def filter_p1(n):
    pass

# Schreibe hier den Filter für P2 Dateien
def filter_p2(n):
    pass

# Schreibe hier den Filter für P3 Dateien
def filter_p3(n):
    pass

# Ändere nichts unterhalb dieser Zeile.
new_image = ""
filename= input("Bitte Dateinamen angeben: ")
file = filename
newfile = file.split(".")[0]+ "_filtered." + file.split(".")[1]

with open(file, "r") as f:
    file = f.read()
    success = False
    if "P1" in file:
        success = True
        filetype = "P1"
    elif "P2" in file:
```



```

    success = True
    filetype = "P2"
elif "P3" in file:
    success = True
    filetype = "P3"
if not success:
    print("Could not determine filetype. Abort.")
    quit()
file_content = file.split("\n")
image_data = []
value = 0
for l in file_content:
    if not l.startswith("#"):
        match value:
            case 0:
                new_image = new_image + l + "\n"
                value += 1
            case 1:
                new_image = new_image + l + "\n"
                resolution = l
                value += 1
            case 2:
                if filetype != "P1":
                    new_image = new_image + l + "\n"
                    depth = l
                else:
                    image_data += filter(lambda x: x != "", l.split(" "))
                    value += 1
            case _:
                image_data += filter(lambda x: x != "", l.split(" "))

#print(image_data)
columns = int(resolution.split(" ")[0])
if filetype == "P3":
    columns = columns * 3
count = 1
for i in image_data:
    if filetype == "P1":
        if i != "":
            new_image = new_image + str(filter_p1(int(i))) + ""

```

```

elif filetype == "P2":
    if i != "":
        if "\n" in i:
            print("I: "+i)
            new_image = new_image + str(filter_p2(int(i))) + " "
elif filetype == "P3":
    if i != "":
        new_image = new_image + str(filter_p3(int(i))) + " "
if count%columns == 0:
    new_image = new_image + "\n"
count += 1
f.close()

```

```

with open(newfile, "w") as nf:

```

```

    nf.write(new_image)

```

```

    nf.close()

```

```

root = Tk()

```

```

# Read the Image

```

```

original_image = Image.open(filename)

```

```

filtered_image = Image.open(newfile)

```

```

# Resize the image using resize() method

```

```

w = int(resolution.split(" ")[0])

```

```

h = int(resolution.split(" ")[1])

```

```

scale=10

```

```

if w < 300:

```

```

    original_image = original_image.resize((w*scale, h*scale))

```

```

    filtered_image = filtered_image.resize((w*scale, h*scale))

```

```

oimg = ImageTk.PhotoImage(original_image)

```

```

fimg = ImageTk.PhotoImage(filtered_image)

```

```

# create label and add resize image

```

```

label1 = Label(root, image=oimg)

```

```

label1.image = oimg

```

```

label1.pack()

```

```

label2 = Label(root, image=fimg)

```

```

label2.image = fimg

```

```

label2.pack()

```

```

# Execute Tkinter

```

```
root.mainloop()  
print("Bye, bye")
```

Bildverarbeitung

Farbmischung

Farbpalettengenerator

Arbeiten mit Anwendungsprogrammen

Empfohlene Software

Empfohlene Software

Im Unterricht benutzte Software

Thonny

Installiere auch die Erweiterung "notebook".

Das Notebook wird mit dem Befehl `!jupyter notebook` gestartet.

Digital Logic Sim