Aufgabe 1

Zeichne mit print-Befehlen und dem Zeichen *:

a) Zeichne ein Rechteck
b) Zeichne ein Haus
c) Zeichne ein Balkendiagramm

Aufgabe 2

Zeichne mit print-Befehlen und dem Zeichen * oder ● in eine 5 x 7 Matrix:

a) Zeichne eine Null
b) Zeichne eine Acht
c) Zeichne eine Eins
d) Zeichne eine Ziffer deiner Wahl

Hinweis:
Das Zeichen ● hat den Unicode 25CF. Es kann über die Windows-Zeichentabelle eingegeben werden.

Aufgabe 3

Gegeben ist die Liste:

- handzeichen = ['Schere', 'Stein', 'Papier']

Schreibe ein Programm, das ein zufälliges Handzeichen ausdruckt.

Aufgabe 4

Gegeben sind zwei Listen. Eine Liste enthält 5 englische Wörter. Die andere Liste enthält 5 dazu passende deutsche Wörter. Dein Programm soll die Überschrift „Stimmt das?“ drucken. Danach soll dein Programm ein Durcheinander zeigen, indem es ein zufällig gewähltes englisches Wort und ein zufällig gewähltes deutsches Wort nebeneinander ausdruckt.

Aufgabe 5

Erweitere das Dictionary englisch_deutsch mit den Schlüsseln:

- 'mouse', 'rat', 'parrot', 'elephant'

Aufgabe 6

Es gibt eine andere Möglichkeit, das Dictionary aufzubauen. Beispiel:

- englisch_deutsch = {'cat': 'Katze', 'dog': 'Hund', 'cow': 'Kuh', 'bird': 'Vogel'}

Baue auf diese Weise das Dictionary italienisch_deutsch mit den Zahlen Eins bis Vier.

Aufgabe 7

Wir haben das Dictionary englisch_deutsch erstellt. Nun wollen wir das Dictionary deutsch_englisch erstellen. Schreibe das Programm dazu.

Aufgabe 8

Schreibe ein Programm, das den Benutzer nach zwei Zahlen fragt und anschließend die Summe ausgibt.

Aufgabe 9

Schreibe ein Programm, das den Benutzer nach seinem Namen fragt und anschließend den Namen in Großbuchstaben ausgibt.
Hinweis: Die Methode heißt upper()

Aufgabe 10
Das Dictionary englisch_deutsch soll erweitert werden. Schreibe ein Programm, das zuerst nach einem englischen Wort – dem Schlüssel – fragt. Danach fragt das Programm nach dem passenden deutschen Wort – dem Wert. Anschließend erweitert das Programm das Dictionary mit Schlüssel und Wert. Was passiert, wenn der Schlüssel bereits vorhanden ist?

Aufgabe 11
Das Dictionary englisch_deutsch soll geprüft werden. Schreibe ein Programm, das nach einem englischen Wort – dem Schlüssel – fragt. Anschließend prüft das Programm, ob der Schlüssel im Dictionary englisch_deutsch bereits vorhanden ist. Abhängig vom Ergebnis gibt das Programm die Meldung aus: 'Schlüssel ist nicht frei' oder 'Schlüssel ist frei'.

Aufgabe 12
Schreibe ein Programm, das den Benutzer nach zwei Zahlen fragt. Anschließend soll das Programm die Summe ausgeben und 'Fertig?' fragen. Wenn der Benutzer 'j' oder 'J' antwortet, ist das Programm zu Ende. Andernfalls fragt das Programm erneut nach zwei Zahlen.

Aufgabe 13
Der Computer soll eine Zahl zwischen 1 und 1000 erraten. Ändere Computer_erraet_die_Zahl.py entsprechend. Wie viele Schritte benötigt der Computer zum Ergebnis?

Aufgabe 14
Erweitere ein_einfacher_Chatbot.py mit deinen eigenen Zufallsantworten und Reaktionsantworten.

Aufgabe 14a
Gegeben sind drei Listen:
subjekt = ["Der Hund", "Die Journalistin", "Der Maler"]
prädikat = ["vergräbt", "interviewt", "malt"]
objekt = ["den Knochen", "den Bürgermeister", "ein Bild"]

a) Schreibe ein Programm, das ein zufälliges Subjekt und ein zufälliges Prädikat und ein zufälliges Objekt hintereinander stellt und den zufälligen Satz ausgibt.
b) Schreibe nun eine Funktion, die den zufälligen Satz baut.

Aufgabe 14b
Gegeben sind 3 Sätze:
original1 = "Wie wird das Wetter?"
original2 = "Wie Programmieren geht, weiß ich."
original3 = "Heute gab es etwas Gutes zu essen!"
Das Programm ein_einfacher_chatbot.py erkennt nur Wörter ohne die Satzzeichen ? ! . ,

a) Schreibe eine Funktion, die diese 4 Satzzeichen aus einem Originalsatz entfernt.
b) Teste deine Funktion mit den 3 Sätzen.

Aufgabe 15

	Grundgebühr pro Monat	Verbrauchspreis pro kWh
Stromtarif "Watt für wenig"	13,50 €	0,75 €
Stromtarif "Billig Strom"	9,20 €	0,81 €

Schreibe eine Funktion, die den Monatsverbrauch in Kilowattstunden akzeptiert und den monatlichen Rechnungsbetrag beim Tarif "Billig Strom" berechnet. Prüfe die Ergebnisse.

Aufgabe 16

Schreibe eine Funktion, die Grundgebühr, Verbrauchspreis und Verbrauch eines Stromtarifs akzeptiert und den monatlichen Rechnungsbetrag für diesen Stromtarif berechnet. Prüfe die Ergebnisse.

Aufgabe 17

Wir wollen in Großbritannien einkaufen. Die Preise sind dort in britischen Pfund (GBP) angegeben. Wir müssen also umrechnen.

a) Schreibe eine Funktion, die britische Pfund in Euro umrechnet.
   (Kurs: 1 GBP = 1,21 EUR)
   1. Schreibe eine Kurzbeschreibung
   2. Mit welchen Daten soll die Funktion arbeiten?
   3. Definiere einen Namen für die Funktion, gib Eingabewert und Rückgabewert an
   4. ~~Gerüst~~
   5. Rumpf
b) Prüfe die Ergebnisse mit Hilfe einer Tabelle von 0 GBP bis 10 GBP in Schritten von 0.50 GBP.

Aufgabe 18

Wir haben in Großbritannien eingekauft. Die Preise unserer Einkäufe stehen in einer Liste:

Book: Guinness World Records	09.00 GBP
CD: Adele – 30	11.99 GBP
Poster: Butterflies	12.99 GBP
Bicycle Reflective Gilet	24.99 GBP

a) Schreibe eine Funktion, die die Preise der Einkäufe addiert und die Summe in Euro umrechnet. (Kurs: 1 GBP = 1,21 EUR)
   1. Schreibe eine Kurzbeschreibung
   2. Mit welchen Daten soll die Funktion arbeiten?
   3. Definiere einen Namen für die Funktion, gib Eingabewert und Rückgabewert an
   4. ~~Gerüst~~
   5. Rumpf
b) Drucke die Summe für alle 4 Einkäufe.
c) Die Weste (Gilet) ist zu teuer. Drucke die Summe für die ersten 3 Einkäufe.

Aufgabe 19

Schreibe ein Programm, das die Kosten für die beiden Tarife "Watt für wenig" und "Billig Strom"

a) berechnet
b) nebeneinander ausdruckt
c) in ein gemeinsames Diagramm plottet

Aufgabe 20

Laura prüft zwei Angebote für mobiles Internet.

Flatrate: Unbegrenztes Surfen für 8,99 € pro Monat

Volumentarif: Preis pro GB: 6,50 €

In den letzten 6 Monaten hat Laura ihren Verbrauch notiert:

Januar	1,2 GB
Februar	1,5 GB
März	2 GB
April	1,2 GB
Mai	1 GB
Juni	2 GB

Schreibe ein Programm, das die aufgelaufenen Kosten für die beiden Tarife "Flatrate" und "Volumentarif"

a) berechnet
b) nebeneinander ausdruckt
c) in ein gemeinsames Diagramm plottet

Aufgabe 21

Herr Häberle hat ein neues Elektroauto. Nach der täglichen Nutzung lädt Herr Häberle den Akku des Elektroautos wieder auf, zu Hause an der Steckdose oder an einer Ladestation. Herr Häberle hat an 3 Tagen notiert, wie lange er gefahren ist und wie lange er geladen hat.

Tag	Was	Dauer in Stunden	Änderung des Ladezustands pro Stunde	Kommentar
Montag	Fahren	2	-33 %	Hohes Tempo
	Laden	13	+5 %	Steckdose
Dienstag	Fahren	5	-18 %	Mittleres Tempo
	Laden	16	+5 %	Steckdose
Mittwoch	Fahren	2	-33 %	Hohes Tempo
	Laden	1	+40 %	Ladestation

Am Montagmorgen ist Herr Häberle mit 100 % Ladezustand losgefahren. Erstelle ein Diagramm, das den Ladezustand des Akkus in Abhängigkeit von der Zeit darstellt.

a) Schreibe eine Funktion mit den Eingangswerten: alter Ladezustand, Dauer, Änderung und dem Rückgabewert: neuer Ladezustand.
   1. Schreibe eine Kurzbeschreibung
   2. Mit welchen Daten soll die Funktion arbeiten?
   3. Definiere einen Namen für die Funktion, gib Eingabewert und Rückgabewert an
   4. ~~Gerüst~~
   5. Rumpf
b) Übertrage die Dauer aus der Tabelle in eine Liste.
c) Beschreibe die Zeitachse durch eine Liste. Die Liste beginnt mit 0. Die folgenden Zeitpunkte berechnest du aus der angegebenen Dauer.
d) Übertrage die Änderung aus der Tabelle in eine Liste.
e) Beschreibe die Ladezustands-Achse durch eine Liste. Die Liste beginnt mit 100. Die folgenden Ladezustände berechnest du mit Hilfe der Funktion aus Teilaufgabe a).
f) Erstelle das Diagramm mit Hilfe der Funktionen des Moduls matplotlib.

Aufgabe 22

a) Erstelle die Klasse "Fahrrad" mit den Eigenschaften:
   - Besitzer
   - Farbe
   - Typ (Touring, Renn, Mountain o. ä.)
   - Anzahl Gänge

b) Erstelle eine Instanz der Klasse "Fahrrad" und gib die Eigenschaften aus.

Aufgabe 23

a) Erweitere die Klasse "Fahrrad". Erstelle die Methoden:
   - fahren
   - klingeln
   - abstellen
   - kilometerstand

b) Erstelle eine Instanz der Klasse "Fahrrad" und rufe die Methoden auf.

Aufgabe 24

a) Erstelle die Eltern-Klasse "Zweirad" mit den Eigenschaften:
   - Besitzer
   - Farbe
   - Typ
   - Anzahl Gänge

   und den Methoden:
   - fahren
   - klingeln
   - abstellen
   - kilometerstand

b) Erstelle die Kind-Klassen "Fahrrad" und "Pedelec", die alle Eigenschaften und Methoden der Eltern-Klasse "Zweirad" erben.

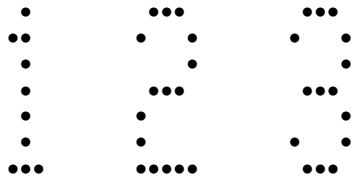
c) Die Klasse "Pedelec":
   - hat zusätzlich die Eigenschaft "Kapazität" (Wattstunden)
   - überschreibt die Methode "abstellen" (warum?)
   - hat zusätzlich die Methode "aufladen"

d) Erstelle eine Instanz der Klasse "Fahrrad" und eine Instanz der Klasse "Pedelec" und rufe alle Methoden auf.

e) Ausgabe der Eigenschaft "Kapazität" der Instanz des "Pedelec".

Aufgabe 25

Die Ziffern auf einem Kassenzettel sind aus Punkten zusammengesetzt. Mit 5x7 Punkten können die Ziffern 0 bis 9 gut lesbar dargestellt werden. Wir beschränken uns auf die Ziffern 1, 2 und 3.



Nicht alle Ziffern auf dem Kassenzettel sehen so aus wie oben. Gelegentlich fehlen Punkte!
Schreibe ein Programm, das erkennt, ob es sich bei einem Muster mit 5x7 Punkten um eine der Ziffern 1, 2 oder 3 handelt. Wenige fehlende Punkte sollen toleriert werden.
Das Programm soll die erkannte Ziffer (oder die erkannten Ziffern) angeben. Wird keine Ziffer erkannt, soll „keine“ angegeben werden.

a) Definiere die (ungestörten) Muster der Ziffern 1, 2 und 3 durch Listen 7 Strings. Jeder String enthält das Muster einer Zeile.
b) Schreibe eine Funktion b), die zwei beliebige Muster akzeptiert, und die passenden Zeilen zählt.
c) Schreibe eine Funktion c), die ein fragliches Muster und das Minimum der passenden Zeilen akzeptiert. Funktion c) soll Funktion b) aufrufen und das fragliche Muster mit dem ungestörten Muster der Ziffern 1, 2 und 3 vergleichen. Funktion c) soll einen String mit den erkannten Ziffern zurückgeben.
d) Definiere ein gestörtes Muster der Ziffer 2. Ein Punkt fehlt.
e) Schreibe ein Programm, das Funktion c) aufruft, und ein gestörtes Muster prüft.

Aufgabe 26

Schreibe einen Unittest für die Funktionen von Aufgabe 25.

Aufgabe 27

Schreibe einen Unittest für die Funktionen von Aufgabe 19.

Aufgabe 28

Schreibe einen Unittest für die Funktionen von Aufgabe 20.

Aufgabe 29

Schreibe ein Programm, das die Ziffern 1, 2 und 3 durch Grauwerte in einem numpy Array darstellt. Grauwert 15 bedeutet *weiß*. Grauwert 0 bedeutet *schwarz*.

a) Erstelle drei numpy Arrays mit 7 Zeilen und 5 Spalten. Die Ziffern 1, 2 und 3 werden mit dem Wert *schwarz* auf dem Hintergrund *weiß* dargestellt.
b) Plotte die drei numpy Arrays mit Hilfe der Funktion imshow() aus der Bibliothek matplotlib.pyplot.
c) Gib den Ziffern 1, 2 und 3 an bestimmten Stellen *hellgraue* Werte
    1. Zahl Eins mit einem *hellen* senkrechten Strich
    2. Zahl Zwei mit einem *hellen* waagerechten Fuß
    3. Zahl Drei mit einem *hellen* waagerechten Mittelstrich
d) Plotte die drei numpy Arrays mit Hilfe der Funktion imshow() aus der Bibliothek matplotlib.pyplot.
e) Gib den Ziffern 1, 2 und 3 an bestimmten Stellen *Farbverläufe*
    1. Zahl Eins mit einem senkrechten *Farbverlauf*
    2. Zahl Zwei mit einem waagerechten *Farbverlauf*
    3. Zahl Drei mit einem waagerechten *Farbverlauf*
f) Plotte die drei numpy Arrays mit Hilfe der Funktion imshow() aus der Bibliothek matplotlib.pyplot.
g) Baue in jede Ziffer einen *Fehler* ein
h) Plotte die drei numpy Arrays mit Hilfe der Funktion imshow() aus der Bibliothek matplotlib.pyplot.

Aufgabe 30

Löse Aufgaben durch gezielte Adressierung von Zeilen und Spalten des numpy Arrays.

a) Erstelle ein numpy Array mit 7 Zeilen und 5 Spalten.
   Fülle das numpy Array mit dem Grauwert 15 für weiß.
b) Setze einen schwarzen Punkt in die Mitte des numpy Arrays.
c) Umrahme den schwarzen Punkt mit einem Quadrat mit dem Grauwert 4
d) Umrahme das Quadrat mit einem weiteren Quadrat mit dem Grauwert 8
e) Prüfe jede Lösung mit Hilfe der Funktion imshow() aus der Bibliothek matplotlib.pyplot.

Aufgabe 31

Ein einfacher Saugroboter hat drei Zustände: STANDBY – GERADEAUS_FAHREN – DREHEN.
Die Bedingungen für die Zustandsübergänge sind: an_aus – kollision_erkannt – genug_gedreht.
Zu Beginn soll der Saugroboter im Zustand STANDBY sein.

a) Welche Zustandsübergänge gibt es?
b) Welche Bedingung gehört zu welchem Zustandsübergang?
c) Zeichne ein Zustandsübergangsdiagramm mit den gegebenen Zuständen und Übergängen.
d) Schreibe ein Programm, das den Saugroboter steuert. Die Zustandsübergänge werden durch die Tasten a – k – g der PC-Tastatur gesteuert. Verwende die Bibliothek StateMachine.
    1. Schreibe eine Kurzbeschreibung
    2. Importiere die notwendigen Bibliotheken
    3. Schreibe eine Begrüßung mit einer Erklärung des Gebrauchs der Tasten
    4. Erzeuge das Objekt state_machine
    5. Vorbelegung des Zeichens von der Tastatur
    5a. Timer für periodische Ausführung
    6. Schreibe die Funktionen mit den print-Ausgaben:
        - standby()
        - geradeaus_fahren()
        - drehen()
    7. Verwende die Funktion taste_gedrueckt(zeichen) von Ampel.py
    8. Schreibe die Funktionen zur Steuerung der Zustandsübergänge:
        - an_aus()
        - kollision_erkannt()
        - genug_gedreht()
    9. Definiere die Zustände STANDBY – GERADEAUS_FAHREN – DREHEN
    10. Zustandsübergänge hinzufügen
    11. Loop

Aufgabe 32

Schreibe ein Programm mit GUI, das drei verschiedene Smileys untereinander anzeigt. Smileys findest du in folgenden Dateien: slightly_smiling_face.png, neutral_face.png, slightly_frowning_face.png.

Aufgabe 33

Schreibe ein Programm mit GUI, das die drei Smileys (von Aufgabe 32) nebeneinander anzeigt.

Aufgabe 34

Schreibe ein Programm mit GUI, das die drei Smileys (von Aufgabe 32) diagonal anzeigt.
Die Bilderzeugung und die Platzierung des Smileys sollen in eine Funktion ausgelagert werden.
Hinweis: Der Aufruf tk.PhotoImage() muss im Hauptprogramm stehen.

Aufgabe 35

Schreibe ein Programm mit GUI, das auf Knopfdruck ein Smiley (von Aufgabe 32) anzeigt.
Hinweis: Der Aufruf tk.PhotoImage() muss im Hauptprogramm stehen.

Aufgabe 36

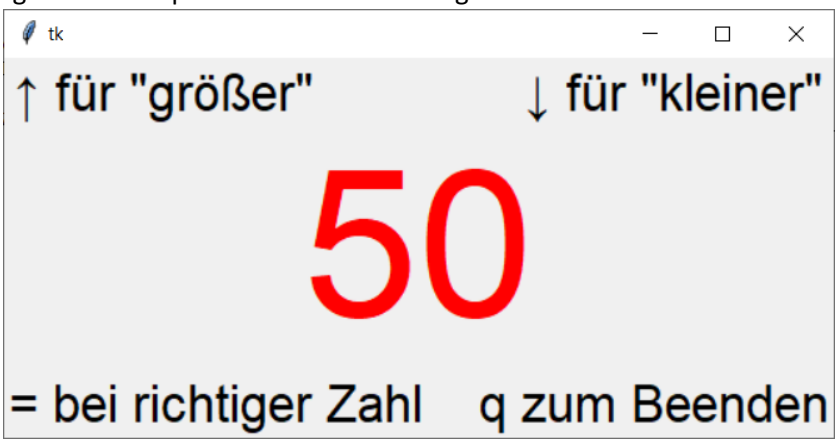
Schreibe ein Programm mit GUI, das drei Smileys in einem Gitter von 7 Zeilen und 11 Spalten anzeigt. Platziere:

- das erste Smiley in der oberen linken Ecke
- das zweite Smiley im Zentrum
- das dritte Smiley in der unteren rechten Ecke

Hinweis: Die Smileys sind 72 Pixel breit und 72 Pixel hoch.

Aufgabe 37

Schreibe das Programm "Computer errät Zahl" mit folgendem GUI:

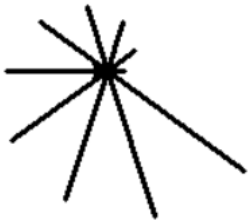


a) Erzeuge ein Fenster und platziere Bedienungsanleitung-1 und -2.
b) Erzeuge eine Steuervariable für die geratene Zahl. Platziere die Ausgabe der geratenen Zahl zwischen Bedienungsanleitung-1 und -2.
c) Binde die Betätigung einer Computertaste an das Fenster. Wenn eine Taste betätigt wird, soll die Funktion onKeyPress aufgerufen werden.
d) Erzeuge die globalen Variablen:
untere
obere
versuche
e) Schreibe die Funktion rate_zahl() zur Berechnung der geratenen Zahl.
f) Schreibe die Funktion onKeyPress. Die Funktion soll auf die Tasten ↓ ↑ = q reagieren. Arbeite mit den globalen Variablen.
    1. Bei ↓ die obere Grenze anpassen, versuche hochzählen
    2. Bei ↑ die untere Grenze anpassen, versuche hochzählen
    3. Bei = einen Kommentar zeigen
    4. Bei q das Fenster schließen
g) Schreibe die Funktion zeige_Kommentar(). Die Funktion gibt die Anzahl der Versuche aus, die der Computer benötigt hat.

Aufgabe 38
Zeichne ein Sechseck. Verwende Turtle.

Aufgabe 39
Zeichne einen Stern, der nur aus Strahlen besteht. Verwende Turtle.

a) Beginne mit 10 Strahlen
b) Definiere eine Variable für die Anzahl der Strahlen. Berechne daraus den Winkel für den Stern.
c) Zeichne einen Stern mit Strahlen wachsender Länge.



Aufgabe 40
Zeichne ein Vieleck. Verwende Turtle.

a) Hauptprogramm: Frage den Benutzer nach der Anzahl der Ecken, 1 bis 10 sind erlaubt. Hinweis: Verwende die Methode numinput().
b) Funktion: Schreibe die Funktion vieleck. Eingabewert ist die Anzahl der Ecken.
c) Hauptprogramm: Rufe die Funktion vieleck auf.
d) Erweitere b): Der berechnete Winkel soll auf der Zeichenfläche erscheinen.

Aufgabe 41
Teste die Codierung der Textdatei.

a) Schreibe ein paar Zeilen Text mit ä, ö und ü in eine Textdatei. Verwende den Editor.
b) Speichere den Text in zwei Dateien: Einmal UTF-8 codiert, einmal ANSI-codiert.
c) Ändere das Programm zwei_textcodierungen.py so, dass es deine Dateien liest.

Aufgabe 42
Gib den Dateipfad an.

a) Starte das Programm datei_liegt_woanders.py. Was bedeutet die Fehlermeldung?
b) Korrigiere das Programm.

Aufgabe 43
Ändere den Inhalt der Datei.

a) Editiere das Programm lesen_ersetzen_schreiben.py. Ersetze ein weiteres Wort im ursprünglichen Text.
b) Ersetze mehrfach vorkommende Worte im ursprünglichen Text.

Aufgabe 44
Das Programm move_player.py bewegt ein Raumschiff vor dem Hintergrund "liquid pygame".
Ändere das Programm:

a) Bewege eine Biene vor dem Hintergrund "bluete". Was muss im Programm geändert werden, damit das ganze Blütenbild zu sehen ist?
b) Bewege einen Dinosaurier vor dem Hintergrund "fluss". Was muss im Programm geändert werden, damit der Dinosaurier das Fenster nicht "halb" verlassen kann?

Aufgabe 45

Das Programm mein_perzeptron.py arbeitet mit sortierten Trainingsdaten und immer gleichen "zufälligen" Gewichten. Das Testdatum ist ein "Hund". Ändere das Programm:

a) Kommentiere die Zeile aus, die für immer gleiche "zufällige" Gewichte sorgt.
   - Wie viele Epochen werden nun benötigt?
   - Haben die gelernten Gewichte andere Zahlenwerte?
   - Bleiben die Ergebnisse gleich, wenn du das Programm 2x hintereinander startest?

b) Ändere das Testdatum. Gib die Größe und Breite eines "nicht Hundes" an.
   Teste das Ergebnis.

c) Bringe die Trainingsdaten und die Label in dieselbe "zufällige" Reihenfolge.
   Hinweis: Die Befehle dafür lauten:

```
rng = np.random.default_rng(seed)  # seed ist eine beliebige Zahl
rng.shuffle(array)                 # array ist feature oder labels
```

   - Wie viele Epochen werden nun benötigt?
   - Haben die gelernten Gewichte andere Zahlenwerte?

Aufgabe 46

Das Programm KI_Beziehung_zwischen_Zahlenfolgen_finden.py findet Beziehungen zwischen Zahlenfolgen. Ändere das Programm:

a) Gesucht ist die Beziehung zwischen den Zahlenfolgen [1, 2, 3, 4, 5, 6, 7, 8, 9] und [1, 3, 5, 7, 9, 11, 13, 15, 17].
   - Vergleiche Loss mit dem Loss bei den ursprünglichen Zahlenfolgen.
   - Ist die vorhergesagte Zahl brauchbar?
   - Erhöhe die Anzahl der Epochen auf 1000 und beobachte Loss und vorhergesagte Zahl.

b) Gesucht ist die Beziehung zwischen den Zahlenfolgen [1, 2, 3, 4, 5, 6, 7, 8, 9] und [1, 4, 9, 16, 25, 36, 49, 64, 81].
   - Vergleiche den Loss mit den bisher erreichten Werten.
   - Ist die vorhergesagte Zahl brauchbar?
   - Finde eine Erklärung für das Ergebnis.

Aufgabe 47

Das Programm mein_Klassifikator.py lernt von 15 Bildern mit 256 Grauwerten.
Das Modell wird mit 10 Bildern trainiert und mit 5 Bildern getestet.

Teste verschiedene Einstellungen im Programm:

a)	Mit 6 Bildern testen	Zeile 21: test_size=0.4 *gilt für a), b) und c)*
b)	Klassifikator beschränken	Zeile 24: clf = RandomForestClassifier(n_estimators=10)
c)	Klassifikator tauschen	Zeile 24: clf = DecisionTreeClassifier()

