

Math-Trainer Dokumentation

Featureliste, Ablaufbeispiele mit Belegungstabellen, Quellcode, Struktogramme

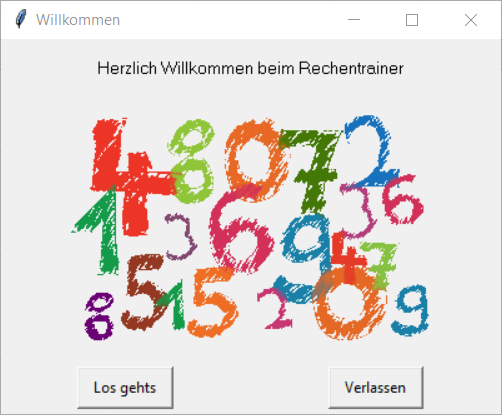


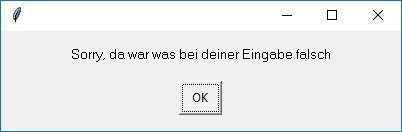
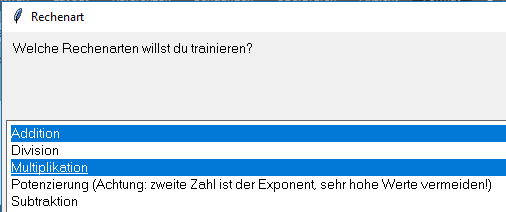
12/2017-01/2018

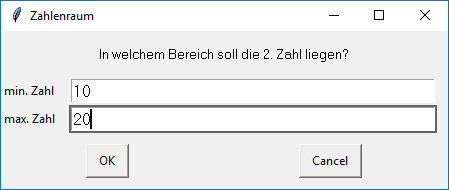
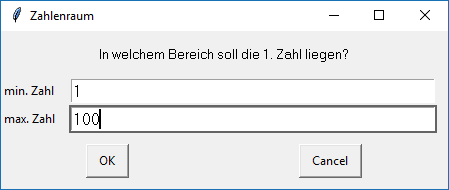
von Valentin Herrmann

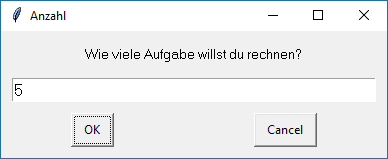
Erstellt im Rahmen des Informatik-Unterricht

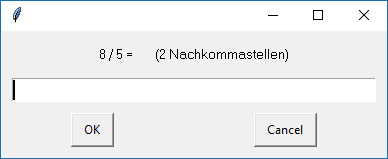
am Gymnasium Ernestinum Coburg bei Dr. Bernd Jakob

**Features**

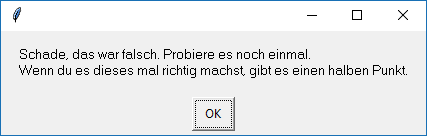
* Startseite mit Titelbild, Start- & Verlassenschaltfläche
* Automatisches Prüfen jeder Eingabe auf ungültige Eingaben (bspw. Text statt Zahl)
* 5 Rechenarten in beliebiger Kombination wählbar
  + (Additieren, Subtrahieren, Multiplizieren, Dividieren, Potenzieren)
  + Warnung vor zu großen Zahlen beim Potenzieren

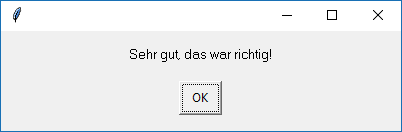
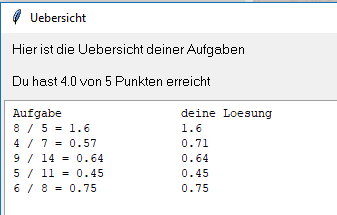
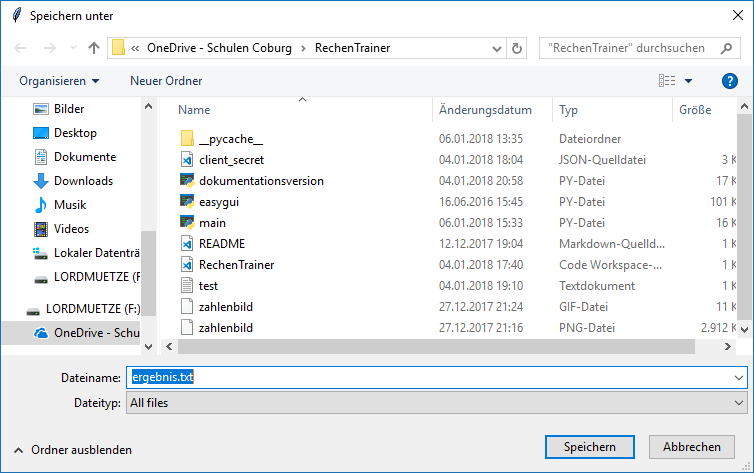


* Freie Wahl beider Komponenten der Rechnungen (natürliche Zahlen)
* Wahl der Anzahl der zu rechnenden Aufgaben (ausreichend hohes Limit)



* Angabe aller Ergebnisse auf 2 Nachkommastellen

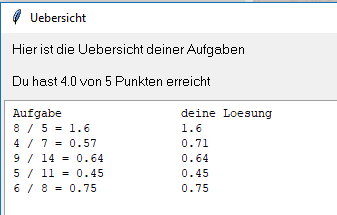
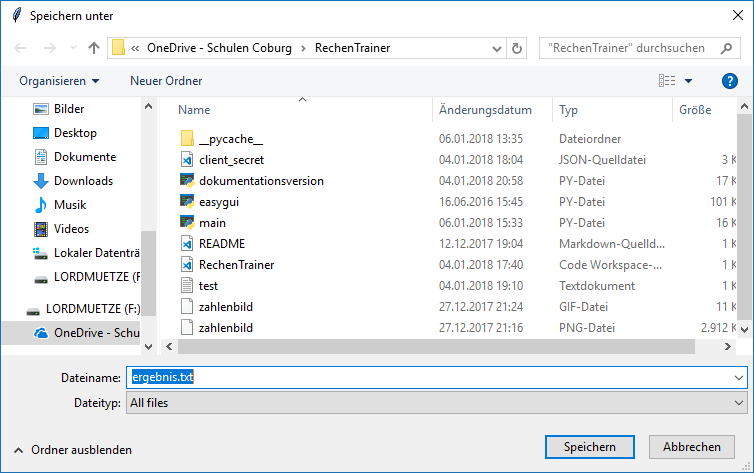
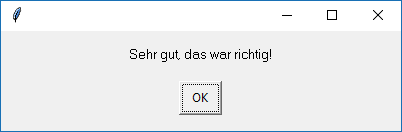
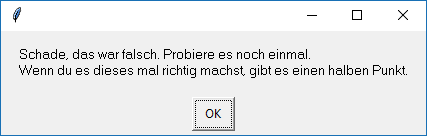
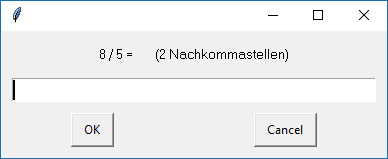
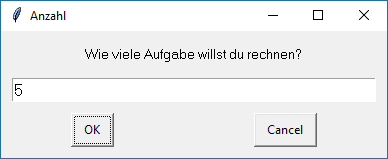
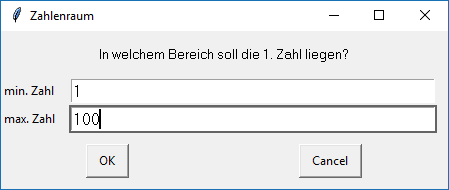
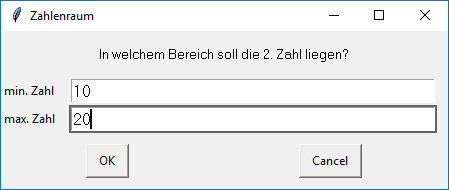
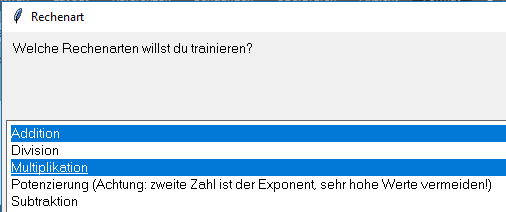
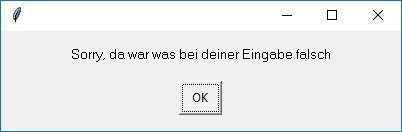
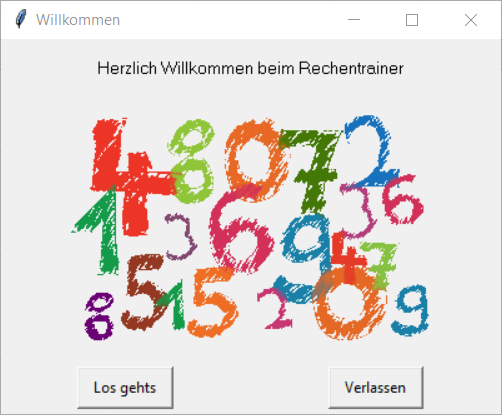
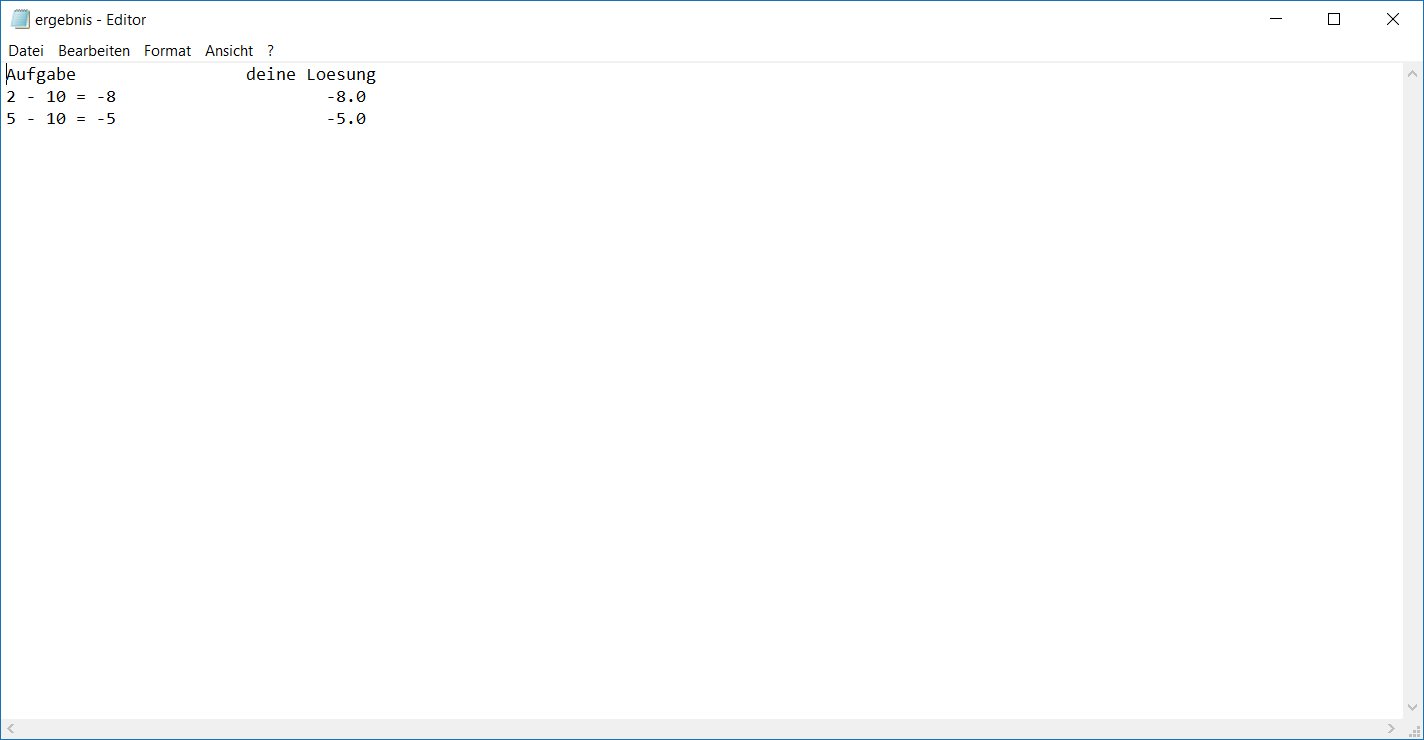


* 2 Versuche pro Aufgabe
  + Ein Punkt im ersten Versuch
  + Ein halber Punkt im zweiten Versuch
* Übersicht aller Aufgaben mit Nutzerergebnis am Ende
* Speichern der Übersicht möglich
* Direkte Rückkehr zur Startseite

von überall mittels Cancel-Schaltfläche

* Verlassen ausschließlich durch Verlassen-Schaltfäche auf der Startseite möglich

Anwendungsbeispiele:



Im folgenden dunkel unterlegten Teil befindet sich der Programmcode des Math-Trainers. Im direkten Vergleich zum Originalcode (‘main.py‘) enthält er einige formatbedingte Änderungen (Entfernen/Hinzufügen von Leerzeilen, verschieben von Kommentare, usw.), die sich aber nicht auf den Programmablauf auswirken.

Neben den einzelnen Funktionen finden sich zudem Struktogramme, welche die Struktur dieser Funktionen erläutern. Wo nötig wurden außerdem ergänzende Verbindungen zwischen Struktogramm und Code eingezeichnet. Außerdem werden try-except-Statements in den Struktogrammen nicht beachtet, da diese dann den Rahmen sprengen würden.

#--------------------Modulimport--------------------

from easygui import \* #Grafische Oberfläche, Dokumentation: http://easygui.sourceforge.net

from random import \* #Zufallsgenerator

import sys

# wird bei einer Box (Element der GUI) der Cancel-Button gedrückt, wird ein NoneType ausgegeben,

# was später zur Überprüfung genutzt wird, ob dieser Button gedrückt wurde

#--------------------initialisieren von globalen Variablen, die später im Programm verwendet werden--------------------

#----------benötigte globale Variablen werden mit "global Variablenname" in die einzelnen Funktionen importiert----------

richtige\_loesungen = 0.0

aufgabenliste = []

nutzer\_loesungsliste = []

rechenart = []

min\_zahl1 = None

max\_zahl1 = None

min\_zahl2 = None

max\_zahl2 = None

anzahl\_aufgaben = 0

#----------------------------------------------------------------------------------------------------



startseite()-Funktion

#--------------------Definition der Startseite--------------------

def startseite():

global richtige\_loesungen

global aufgabenliste

global nutzer\_loesungsliste

global rechenart

global min\_zahl1

global max\_zahl1

global min\_zahl2

global max\_zahl2

global anzahl\_aufgaben

#Reset der wichtigsten globalen Variablen & Listen mit jedem neuen Durchlauf des Rechen Trainers

richtige\_loesungen = 0.0

aufgabenliste = []

nutzer\_loesungsliste = []

rechenart = []

min\_zahl1 = None

max\_zahl1 = None

min\_zahl2 = None

max\_zahl2 = None

anzahl\_aufgaben = 0

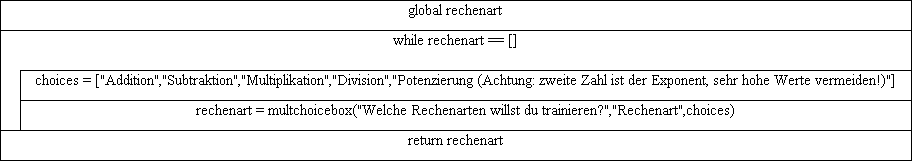
choices = ["Los gehts", "Verlassen"] #Festlegen der Buttons

start = buttonbox("Herzlich Willkommen beim Rechentrainer", "Willkommen", choices, "zahlenbild.gif") #angeklickten Button an Variable binden

return start #angeklickten Button als Funktionswert ausgeben

#----------------------------------------------------------------------------------------------------

#--------------------Definition, welche Rechenart trainiert werden soll--------------------



rechenart\_eingeben()-Funktion

def rechenart\_eingeben():

global rechenart

while rechenart == []: #Schleife läuft, solange keine Rechenart ausgewählt wurde

#wählbare Rechenarten festlegen

choices = ["Addition","Subtraktion","Multiplikation","Division",

"Potenzierung (Achtung: zweite Zahl ist der Exponent, sehr hohe Werte vermeiden!)"]

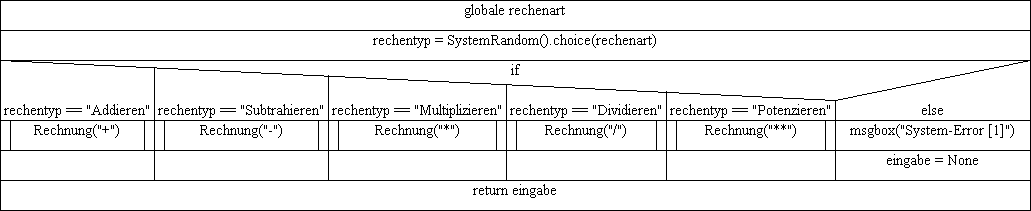
rechenart = multchoicebox("Welche Rechenarten willst du trainieren?","Rechenart",choices) #Liste aus ausgewählte(n) Rechenart(en) erstellen

return rechenart #ausgewählte Rechenarten [Liste] als Funktionswert zurückgeben

#----------------------------------------------------------------------------------------------------

#--------------------Definition, welche Rechenart trainiert werden soll--------------------

def rechenart\_abrufen():



rechenart\_abrufen()-Funktion

global rechenart

rechentyp = SystemRandom().choice(rechenart) #zufälliges Rechenart von Liste auswählen

#abhängig von der gewählten Rechenart "Rechunung(operator)" mit Operator

#der jeweiligen Rechenart als Parameter aufrufen

if rechentyp == "Addition":

eingabe = Rechnung("+")

elif rechentyp == "Subtraktion":

eingabe = Rechnung("-")

elif rechentyp == "Multiplikation":

eingabe = Rechnung("\*")

elif rechentyp == "Division":

eingabe = Rechnung("/")

elif rechentyp == "Potenzierung (Achtung: zweite Zahl ist der Exponent, sehr hohe Werte vermeiden!)":

eingabe = Rechnung("\*\*")

#wird keine möglichen Rechenart ausgewählt, wird ein SystemError ausgegeben und "eingabe" als NoneType gesetzt

else:

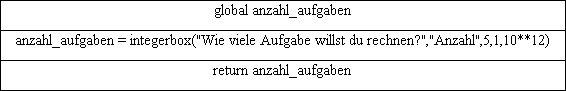
msgbox("System-Error [1]")

eingabe = None

return eingabe #"eingabe" als Funktionswert ausgeben

#----------------------------------------------------------------------------------------------------

#--------------------Anzahl der Aufgaben--------------------

def anzahl\_aufgaben\_eingeben():

global anzahl\_aufgaben

#Anzahl der zu rechnenden Aufgaben an "anzahl\_aufgaben" binden, automatische Prüfung auf natürliche Zahl, maximal: 10\*\*12

anzahl\_aufgaben = integerbox("Wie viele Aufgabe willst du rechnen?","Anzahl",5,1,10\*\*12)

anzahl\_aufgaben\_eingeben()-Funktion

return anzahl\_aufgaben #Anzahl als Funktionswert ausgeben

#----------------------------------------------------------------------------------------------------

#--------------------Festlegung des Zahlenraums--------------------

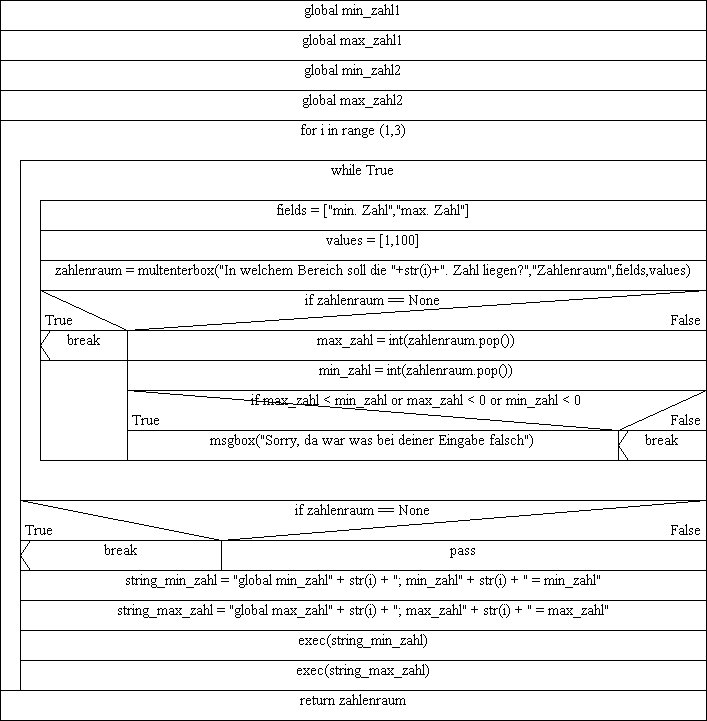
def zahlenraum():

global min\_zahl1

global max\_zahl1

global min\_zahl2

global max\_zahl2



zahlenraum()-Funktion

for i in range (1,3): #for-Schleife mit zwei Durchläufen: i=1 & i=2

while True: #Endlosschleife

try: #try-Teil wird ausgeführt, bis ein Fehler auftritt

fields = ["min. Zahl","max. Zahl"] #Namen der Felder festlegen

values = [1,100] #Standardwerte für die Felder festlegen

zahlenraum = multenterbox("In welchem Bereich soll die "+str(i)+". Zahl liegen?",

"Zahlenraum",fields,values)

if zahlenraum == None: #Cancel-Button wurde gedrückt

break #Endlosschleife abbrechen

else:

max\_zahl = int(zahlenraum.pop())

min\_zahl = int(zahlenraum.pop())

#letzten Listenwert als Minimal-/Maximalwert festlegen & von Liste entfernen

#wird ein nicht-Integer (Float,Text,...) eingegeben, wird ein Fehler auftreten

#prüfen, ob der Maximalwert größer als der Minimalwert ist

#oder ob eine der beiden Zahlen kleiner als 0 ist (keine natürliche Zahl)

if max\_zahl < min\_zahl or max\_zahl < 0 or min\_zahl < 0:

msgbox("Sorry, da war was bei deiner Eingabe falsch")

else: #Zahlen sind ok

break #Endlosschleife abbrechen

except: #tritt ein Fehler auf, so wird der except-Teil ausgeführt

msgbox("Sorry, da war was bei deiner Eingabe falsch")

if zahlenraum == None: #Cancel-Button wurde gedrückt

break #for-Schleife abbrechen

else:

pass #nichts tun

#Strings zum späteren Ausführen erstellen, Nummer von min\_zahl & max\_zahl abhängig von

#Schleifendurchlauf

string\_min\_zahl = "global min\_zahl" + str(i) + "; min\_zahl" + str(i) + " = min\_zahl"

string\_max\_zahl = "global max\_zahl" + str(i) + "; max\_zahl" + str(i) + " = max\_zahl"

#min\_zahl & max\_zahl, min\_zahl1 & max\_zahl1 bzw min\_zahl2 & max\_zahl2 zuweisen

exec(string\_min\_zahl)

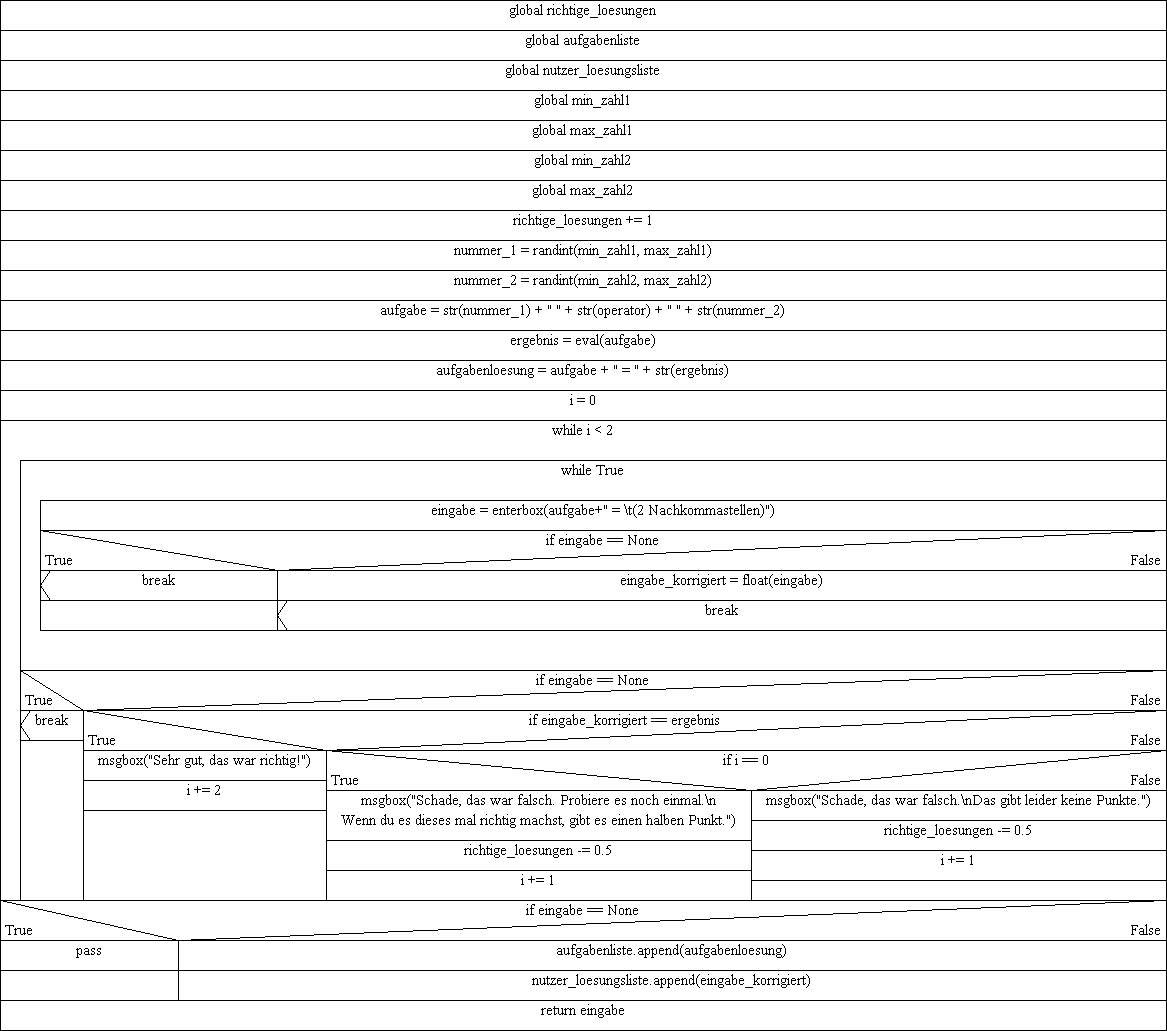
exec(string\_max\_zahl)

return zahlenraum #die Liste "zahlenraum" wird als Funktionswert ausgegeben

#----------------------------------------------------------------------------------------------------

#--------------------Aufgabenstellung mit Eingabe für die Lösung--------------------

def Rechnung(operator):



Ausschnitt 1

rechnung()-Funktion

global richtige\_loesungen

global aufgabenliste

global nutzer\_loesungsliste

global min\_zahl1

global max\_zahl1

global min\_zahl2

global max\_zahl2

richtige\_loesungen += 1 #Punktzahl um 1 erhöhen

#erste Zahl für die Rechnung mit kleinster & größter Zahl zufällig festlegen

#zweite Zahl für die Rechnung mit kleinster & größter Zahl zufällig festlegen

nummer\_1 = randint(min\_zahl1, max\_zahl1) nummer\_2 = randint(min\_zahl2, max\_zahl2)

aufgabe = str(nummer\_1) + " " + str(operator) + " " + str(nummer\_2) #Aufgabe in der Form "1 + 1" erstellen

ergebnis = round(eval(aufgabe),2) #zuvor erstellte Aufgabe auf 2 Nachkommastellen berechnen

aufgabenloesung = aufgabe + " = " + str(ergebnis) #Aufgabe mit Lösung in der Form "1 + 1 = 2" zusammensetzen

i = 0 #Zählervariable für Versuchs-Schleife initialisieren

while i < 2: #Schleife mit Maximalwert i<2 für Versuche

while True: #Endlosschleife

try: #try-Teil wird ausgeführt, bis ein Fehler auftritt

#Aufgabenstellung mit Eingabefeld wird angezeigt, Form "1 + 1 = (2 Nachkommastellen)"

eingabe = enterbox(aufgabe+" = \t(2 Nachkommastellen)")

if eingabe == None: #Cancel-Button wurde gedrückt

break #Endlosschleife wird unterbrochen

else:

#Fehler wird ausgegeben, wenn sich die Eingabe nicht

#in einen Float mit 2 Nachkommastellen umwandeln lässt (zB Text)

eingabe\_korrigiert = round(float(eingabe),2)

break #Endlosschleife wird unterbrochen

#tritt ein Fehler auf, wird der except-Teil ausgeführt

except:

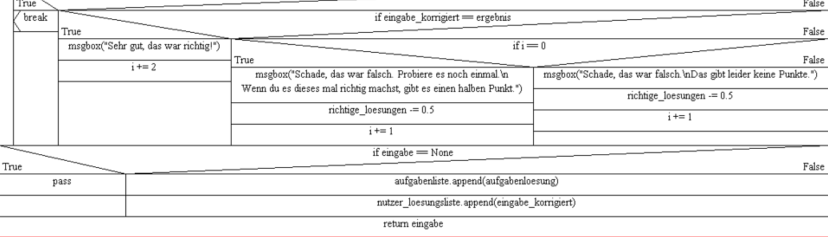
msgbox("""Sorry da was bei deiner Eingabe falsch\n

(z.B. falsche Zahlenart oder Komma statt Punkt bei Kommazahlen)""")

#Zurückspringen zum Schleifenanfang

if eingabe == None: #Cancel-Button wurde gdedrückt

break #Abbruch der Schleife, welche die Anzahl der Versuche zählt und



rechnung()-Funktion (Ausschnitt 1)

#in der die Aufgabenstellung angezeigt wird

else:

if eingabe\_korrigiert == ergebnis: #Eingabe ist richtig

msgbox("Sehr gut, das war richtig!")

i += 2 #Zähler der Schleife auf 2 erhöhen und dadurch die Schleife beenden

else: #Eingabe ist falsch

if i == 0: #prüfen, ob es der erste Versuch ist

msgbox("""Schade, das war falsch. Probiere es noch einmal.\n

Wenn du es dieses mal richtig machst,

gibt es einen halben Punkt.""")

richtige\_loesungen -= 0.5 #halben Punkt abziehen =>0,5 Punkte

i += 1 #Zähler der Schleife um 1 (auf 1) erhöhen

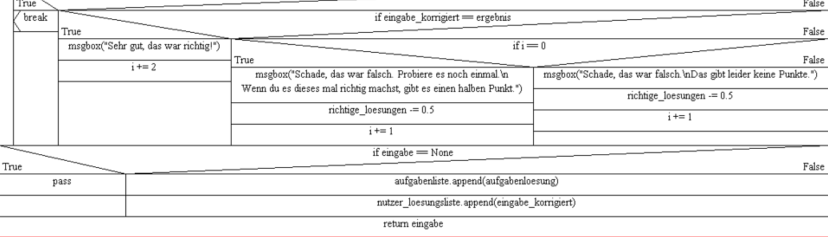
else: #zweiter (letzter) Versuch

msgbox("Schade, das war falsch.\nDas gibt leider keine Punkte.")

richtige\_loesungen -= 0.5 #einen weiteren halben Punkt abziehen

#--> 0 Punkte

i += 1 #Zähler der Schleife um 1 (auf 2) erhöhen



rechnung()-Funktion (Ausschnitt 1)

if eingabe == None: #Cancel-Button wurde gdedrückt

pass #nichts tun

else:

aufgabenliste.append(aufgabenloesung) #Aufgabe mit Lösung zur Liste für die Übersicht hinzufügen

nutzer\_loesungsliste.append(eingabe\_korrigiert) #Lösung des Nutzers zur Liste für die Übersicht hinzufügen

return eingabe #"eingabe" als Funktionswert ausgegeben

#----------------------------------------------------------------------------------------------------

#--------------------erstellen der Abschlussuebersicht--------------------

def uebersicht():

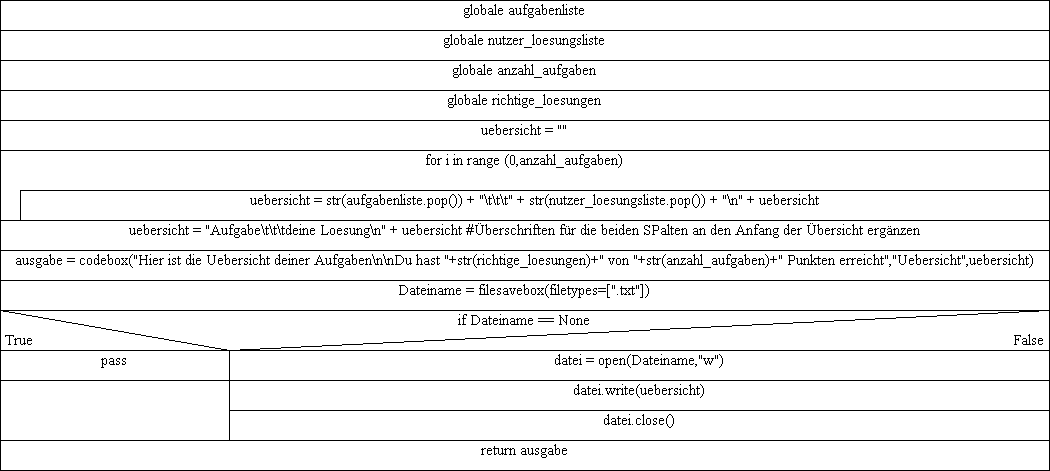
global aufgabenliste

global nutzer\_loesungsliste

global anzahl\_aufgaben

global richtige\_loesungen

uebersicht = "" #"uebersicht" als leeren String initialisieren



for i in range (0,anzahl\_aufgaben): #Schleife mit so vielen Durchläufen wie Aufgaben

#bei jedem Durchlauf werden das letzte Element der Aufgabenliste & das letzte Element der Nutzerlösungsliste

#mit drei Tabulatoren Abstand und einer neuen Zeile am Ende an den Anfang der Übersicht gesetzt

uebersicht = str(aufgabenliste.pop()) + "\t\t\t" + str(nutzer\_loesungsliste.pop()) + "\n" + uebersicht

#Überschriften für die beiden Spalten an den Anfang der Übersicht ergänzen

uebersicht = "Aufgabe\t\t\tdeine Loesung\n" + uebersicht

#Übersicht in einer Scrollbaren Codebox ausgeben

ausgabe = codebox("Hier ist die Uebersicht deiner Aufgaben\n\nDu hast "+str(richtige\_loesungen)+" von "+

str(anzahl\_aufgaben)+" Punkten erreicht","Uebersicht",uebersicht)

#Übersicht als Datei speichern

#Speicherort und Dateiname festlegen

Dateiname = filesavebox(filetypes=[".txt"])

if Dateiname == None: #Cancel-Button gedrückt

pass #nichts tun & nicht speichern

uebersicht()-Funktion

else:

#Datei mit obigem Namen erstellen/öffnen und leeren

datei = open(Dateiname,"w")

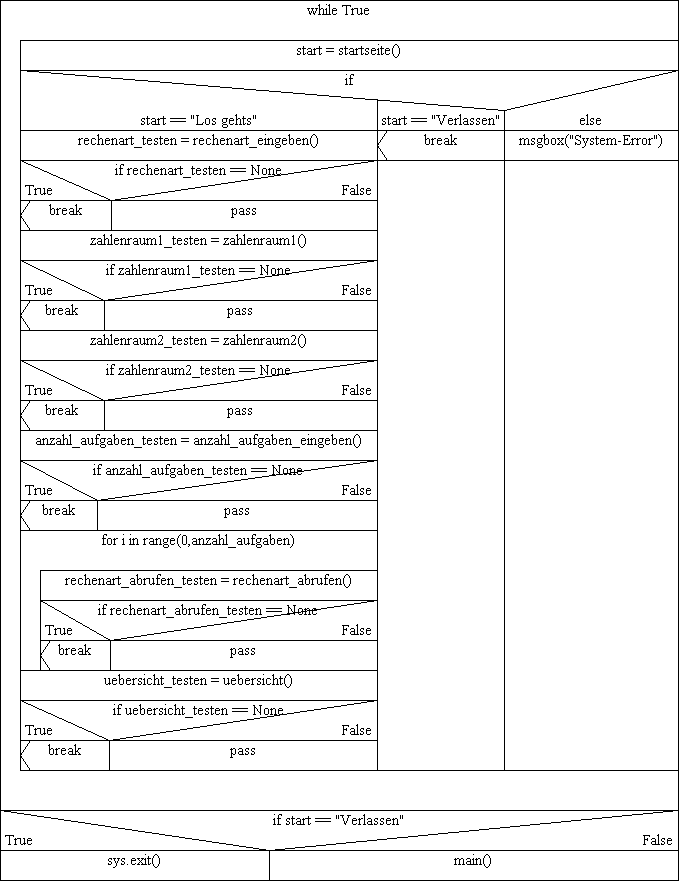
datei.write(uebersicht) #"uebersicht" in Datei einfügen

datei.close() #Datei schließen

return ausgabe #"ausgabe" als Funktionswert ausgeben

#----------------------------------------------------------------------------------------------------

#--------------------Erstellen der main-Funktion--------------------



Ausschnitt 1

Ausschnitt 2

#--------------------Gesamtablauf und Zusammenfuehren vorheriger Funktionen--------------------

def main():

while True: #Endlosschleife

start = startseite() #Startseite aufrufen

if start == "Los gehts": #"Los gehts"-Button gedrückt

rechenart\_testen = rechenart\_eingeben()

if rechenart\_testen == None: #Cancel-Button wurde in dieser Funktion gedrückt

break #Endlosschleife und damit weiteren Programmablauf abbrechen

else:

pass #nichts tun

zahlenraum\_testen = zahlenraum()

if zahlenraum\_testen == None: #Cancel-Button wurde in dieser Funktion gedrückt

break #Endlosschleife und damit weiteren Programmablauf abbrechen

else:

pass #nichts tun

anzahl\_aufgaben\_testen = anzahl\_aufgaben\_eingeben()

if anzahl\_aufgaben\_testen == None: #Cancel-Button wurde in dieser Funktion gedrückt

break #Endlosschleife und damit weiteren Programmablauf abbrechen

else:

pass #nichts tun

for i in range(0,anzahl\_aufgaben): #untere Funktion so oft aufrufen, wie Aufgaben gerechnet werden sollen

main()-Funktion

rechenart\_abrufen\_testen = rechenart\_abrufen()

if rechenart\_abrufen\_testen == None: #Cancel-Button wurde in dieser Funktion gedrückt

break #Endlosschleife und damit weiteren Programmablauf abbrechen

else:

main()-Funktion (gesamt)

pass #nichts tun

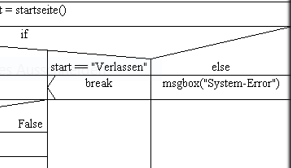
try: #try-Teil wird solange ausgeführt bis ein Fehler auftritt

uebersicht\_testen = uebersicht()

if uebersicht\_testen == None: #Cancel-Button wurde in dieser Funktion gedrückt

main()-Funktion

break #Endlosschleife und damit weiteren Programmablauf abbrechen



main()-Funktion (Ausschnitt 1)

else:

pass #nichts tun

except: #tritt ein Fehler auf, wird der except-Teil ausgeführt

break #Endlosschleife und damit weiteren Programmablauf abbrechen

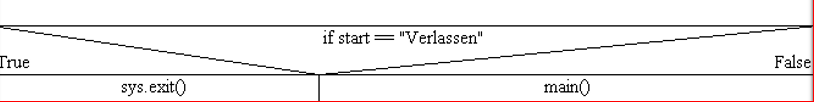
elif start == "Verlassen": #"Verlassen"-Button wurde gedrückt

break #Endlosschleife und damit weiteren Programmablauf abbrechen

else: #es wurde keiner der beiden Buttons gedrückt aber es entsteht trotzdem eine Eingabe

msgbox("System-Error")

if start == "Verlassen": #"Verlassen"-Button wurde gedrückt



main()-Funktion (Ausschnitt 2)

sys.exit() #Programm schließen

else:

main() #Programm von Anfang an starten

#----------------------------------------------------------------------------------------------------

main() #Aufrufen der main()-Funktion  Starten des Programms

