# Variablen ------------------------------------------------------------------

var = 3

int() / float() #auch typecasting

type()          #gibt Typ zurück

var2 += 1       #erhöht var um 1

# Zeichenketten --------------------------------------------------------------

str()

Zeichenkette[0]         #nulltes Element der Zeichenkette

Zeichenkette[1:3]       #Elemente von 1 bis 2

Zeichenkette[:7]        #Elemente bis 6

Zeichenkette[4:]        #von Element 4 bis Ende

'''[Text]'''            #Zeilensprünge mit gespeichert

Zeichenkette.lower()    #alles klein

Zeichenkette.upper()    #alles groß

Zeichenkette.strip()    #Anfang/Ende Leeraum weg

Zeichenkette.replace(alterWert, neuerWert)      #alles alte durch neu

Zeichenkette.startswith('s')                    #auch end; gleicht ab

var.format(var1, var2, ...)

Text='''Am Morgen ging {sie} im {ort} spazieren.'''

print(text.format(sie='Clara', ort='Wald'))

#\n New Line

# Listen ---------------------------------------------------------------------

var = 3 \* liste[:3] + ['Buh']

var=[ausdruck] for i in [range/liste] if [Bedingung]:

#speichert Liste, zum Beispiel gerade Quadratzahlen

liste[0]=3                  #nulltes Element == 3

liste.append()              #auch liste=liste+[Element]

liste.extend([6, 3, 9])     #erweitert Liste um Elemente

liste.insert(10, L)         #fügt an Stelle 10 ein L ein

liste.remove(90)            #entfernt erste 90

liste.index(L)              #gibt Index von ersten L wieder

liste.count(L)              #zählt, wie oft L

liste.pop(L)                #gibt L wieder und löscht aus Liste

liste.sort()                #wird aufsteigend sortiert

liste.reverse()             #Reihenfolge wird umgekehrt

# Dictionary ----------------------------------------------------------------

var = {'schlüssel':'Wert', ...} #wenn Schlüssel aufgerufen, wert wird wiedergegeben

var[index]                      #Aufruf Dictionary

var.keys()                      #liefert alle Schlüssel

var.values()                    #liefert alle Werte

var=(1,2)                       #unveränderliche Tupel (nur ein Wert: #var=(1,))

# Operatoren -----------------------------------------------------------------

+, -, \*, /, %, \*\* oder auch pow(basis, potenz)

in / not in             #enthalten in

<, >, <=, >=, ==, !=

and

or

# Methoden -------------------------------------------------------------------

print('Ergebnis: ', var, 'Gut gemacht!')

input('Wert eintragen:')

len([Zeichenkette/Liste])

set('abaac')                    #gibt Menge wieder, entfernt doppeltes

liste = {'a', 'b', 'c'}         #auch ein Set

range(zahl, zahl2)              #gibt alle Zahlen bis zahl-1 wieder

round(zahl, nachkomma)          #rundet

# If -------------------------------------------------------------------------

if [Bedingung]:

    [Funktionsblock]

elif [zweite zu prüfende Bedingung]:

    [Funktionsblock]

else:

    [Funktionsblock]

#Schleifen -------------------------------------------------------------------

while [Bedingung]:

    [Funktionsblock]

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

for i in range(0,3):

    #i nimmt alle Werte von 0 bis 2 an

for i in [0,1,2]:

    #selbe wie oben

# Funktionen/Unterprogramme --------------------------------------------------

def fkt (var, zweiteübergebenevar):

    global var                      #in Fkt. auf globale var zugreifen

    [Funktionsblock]

    return rückgabewert             #wenn Wiedergabe erfolgen soll

var=fkt (50,20,var1, ...)           #var nur nötig, wenn return

# Bibliotheken/Module --------------------------------------------------------

from Bibliothek import \*

from Bibliothek import methode

import Bibliothek #--> Bibliothek.methode

import Bibliothek as kurzname

#Eigene Module auch möglich, wenn im selben Ordner wie Projekt.

# random

import random

random.randint(Untergrenze, Obergrenze)

random.choice()         #wählt zufälliges Element aus

random.shuffle()        #mischt liste

# Math

from math import \*

sqrt()

sin/cos/tan

pi

degrees(Radius) #wandelt um

radians(Grad)

log(zahl)

#Time

import time

var=time.localtime()

var.tm\_year()           #auch month, hour, …

# Speichern/Lesen ------------------------------------------------------------

import pickle

f = open(datei, mode='r' [, encoding='utf-8'])

#r…Lesen, w…Schreiben, a…neuer Text wird angehangen, rb…Binär lesen, wb…Binär schreiben

f.close()               #speichert und schließt

f.read()                #Inhalt gelesen und als str wiedergeg. (txt)

f.write(var)            #var wird reingeschrieben (txt)

f.flush()               #gespeichert, aber nicht geschlossen (txt)

pickle.dump()           #für Binärdateien: speichern

pickle.load()           #für Binärdateien: laden

#speichern

f=open('Ordner/Renndaten.dat', mode='wb')

pickle.dump(liste, f)

f.close()

#lesen

f=open(pfad, 'rb')

var=pickle.load(f)

f.close()

#Filedialog

from tkinter import filedialog

pfad=filedialog.askopenfilename() #gibt nur Pfad zurück

stream=filedialog.askopenfile()

stream=filedialog.asksaveasfile()

# Fehler abfangen ------------------------------------------------------------

try:

    [Funktionsblock]

except:

    [Funktionsblock]

# TK -------------------------------------------------------------------------

# Fenster --------------------------------------------------------------------

from tkinter import \*

fenster=Tk()

fenster.title('')

fenster.geometry("1000x800")

fenster.mainloop()

# Label ----------------------------------------------------------------------

var=Label(master=fenster, text='')

#rechteckige Fläche

# Button ---------------------------------------------------------------------

var=Button(master=fenster, text = '', command=fkt)

#mit () dahinter wird fkt sofort ausgeführt

# Frame ----------------------------------------------------------------------

var=Frame(master=fenster)

#Container (für Layout und Anordnung praktisch) (statt Fenster frame als master)

# Entry ----------------------------------------------------------------------

var=Entry(master=fenster)

#einzeilige Eingabe; Text für mehrzeilig

var.get()                           #Rückgabe (Auch Indexe wie Delete möglich)

var.delete(index[, index2])         #löscht Zeichen, oder von 1 bis 2

var.insert(index, text)             #fügt an index text ein

# Radiobuttons ---------------------------------------------------------------

varKontroll=StringVar()

varButton1=Radiobutton(master=fenster, text='', value=varKontroll, variable=Wertvariable)

varButton2=Radiobutton(master=fenster, text='', value=varKontroll, variable=Wertvariable2)

varAusgabe=varKontroll.get()

#Kontrollvariable für alle nötig, nimmt Wert für ausgewählten an (deswegen get zum Wert bekommen!)

varButton1.select()     #Vorauswahl

#1 aus n Auswahl

# Checkbutton ----------------------------------------------------------------

#m aus n Auswahl

# Bilder ---------------------------------------------------------------------

photoimage=PhotoImage(master=fenster, file=pfad)        #Bild, nur png!

photoimage.get(x,y)     #Gibt Farbwert als Tupel des Pixels zurück

photoimage.height()     #gibt Höhe Foto zurück /width

photoimage.put(farbe,position)

                        #farbe rgb,POS als Rechteck oder Pixel

photoimage.write(pfad)  #speichert Bild in Pfad

#für andere Dateien:

from PIL import \*

varPIL=Image.open(pfad)

varTK=ImageTK.PhotoImage(varPIL)    #notwendig für Darstellung

varPIL.getpixel((x,y))              #entspricht get

varPIL.putpixel((x,y),(r,g,b))      #entspricht put

varPIL.thumbnail(size=(w,h))        #verkleinert auf width und height

varPIL.size()                       #gibt Größe in w und h zurück

# Widgetmethoden -------------------------------------------------------------

Widget.pack()               #erzeugt Widget

Widget.destroy()            #löscht W

Widget.cget(text)           #gibt Zustand des W wieder, Bsp: text

Widget.config(text='b')     #ändert Wert, Bsp: text='b'

Widget.bell()               #erzeugt Glockenschlag

Widget.after(zeit, fkt)     #nach Millisekunden wird Fkt. ausgeführt

#Widgetattribute (für die meisten)

text

image           #Bild auf Widget zu sehen

height

width

font=(Schriftart, Schriftgröße)

justify         #Ausrichtung Text des Widgets

fg/bg           #Hintergrund/Vordergrundfarbe, mit String oder Hexadezimal

bd/borderwith

relief          #Form Rahmen; SUNKEN,RAISED,GROOVE,RIDGE,FLAT

# Widget anordnen ------------------------------------------------------------

Für pack():

anchor          #CENTRE, E, N, NE, NW, S, SE, SW, W

expand          #1-->ändert Größe mit Großziehen von fenster (0 nicht)

padx/pady       #leerer Raum drumherum

side            #LEFT,RIGHT,TOP,BUTTOM ; an dessen Rand gesetzt

fill            #passt sich an Master an (Widget füllt mit leeren Raum): X,Y,BOTH,NONE

# Menübar --------------------------------------------------------------------

varMenu=Menu(master=fenster)

fenster.config(menu=varMenu)    #Fenster hat Menü

varerstes=Menu(varMenu)

varMenu.add\_cascade(label='erstes', menu=varerstes)

                                #Hauptauswahl (wie Datei, Edit, ...)

varerstes.add\_command(label="Hi!", command=fkt)

                                #Untermenüpunkt (wie speichern, öffnen, ...)

varerstes.add\_separator()       #Trennstrich

#Man muss nicht pack()

# Threads --------------------------------------------------------------------

from \_thread import start\_new\_thread

start\_new\_thread(fkt, (...))    #erst Fkt., dann übergebene Var, für leer: ()

#neuer Ablauf des Programms --> mehrere Abläufe parallel (Widgets #können sich möglicherweise nicht aktualisieren, dann mit neuen Thread probieren)

# Assert ---------------------------------------------------------------------

assert [Bedingung]      #bricht Programm ab, wenn falsch

# Objektorientierung ---------------------------------------------------------

class klassenname:

    #[Docstring (Erklärung als str)]

    def \_\_init\_\_(self, weiteresAttribut):

        #Anweisung zur Initialisierung, Attribute

        self.inhalt=0

    def Methode1(self, weiteresAttribut):

        #Anweisungen; definiert diese Methode

        self.inhalt+=500

a=klassenname()

a.Methode1()

# Turtle ---------------------------------------------------------------------

import turtle

var = turtle.Turtle()

var.pendown()

var.penup()

var.pencolor()

var.pensize()

var.forward(px)/fd

var.backward(px)/bk/back

var.right()/rt

var.left()/lt

var.goto()

var.write()

var.circle(radius)

turtle.tracer(0,0)          #zeichnet sofort

turtle.update()

turtle.exitonclick()        #wichtig am Ende, sonst zeichnet nicht

var.position()              #gibt Position wieder

var.xcor/ycor               #gibt Positionskoordinaten wieder

#Bsp.: if b.xcor > 30: […]