Int() / float() #auch typecasting  
type() #gibt Typ zurück

**Var**

*[var]* += 1 #erhöht *[var]* um 1

str()  
 Zeichenkette[0] #nulltes Element der Zeichenkette

[1:3] #Elemente von 1 bis 2  
 [:7] #Elemente bis 6  
 [4:] #von Element 4 bis Ende  
 ‚‘‘*[Text]*‚‘‘ #Zeilensprünge mit gespeichert

*[var]*.lower() #alles klein  
 *[var]*.upper() #alles groß  
 *[var]*.strip() #Anfang/Ende Leeraum weg  
 *[var]*.replace(*[alter Wert]*,*[neuer Wert]*) #alles alte durch neu  
 *[var]*.startswith(*[var]*) #auch end; gleicht ab

*[var]*.format(*[var1]*, *[var2]*, …)

Text=‘‘‘Am Morgen ging {sie} im {ort] spazieren.‘‘‘  
print(text.format(sie=‘Clara‘, ort=‘Wald‘))

**Zeichenketten**

#\n New Line

*[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

*[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,))

**Listen**

+. -, \*, /, %, \*\* oder auch pow([Basis], [Potenz]

In / not in #enthalten in

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

**Operatoren**

And  
Or

Print(„Ergebnis: „, *[var]*, „Gut gemacht!“)

Input(„Wert eintragen:“)

Len([Zeichenkette, Liste])

set(‚abaac‘) #gibt Menge wieder, entfernt doppeltes  
-->{‚a‘, ‚b‘, ‚c‘} #auch Listen so

range(*[zahl]*) #gibt alle Zahlen bis zahl-1 wieder

**Methoden**

round(*[zahl]*, *[Nachkomma]*) #rundet

If *[Bedingung]*:  
 *[Funktionsblock]*  
elif *[zweite zu prüfende Bedingung]*:  
 *[Funktionsblock]*  
else:  
 *[Funktionsblock]*

**If**

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]*

**Schleifen**

Def *[Fkt.]*(*[var1]*, *[zweiteübergebenevar]*):  
 *[Funktionsblock]*  
 return *[Rückgabewert]* #wenn Wiedergabe erfolgen soll

[var*]=[Fkt.*](50,20,*[var2]*, ...) #var nur nötig, wenn return

Global *[var]* #in Fkt. auf globale var zugreifen

**Funktionen/Unterprogramme**

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.randint(*[Untergrenze]*, *[Obergrenze]*)  
random.choice() #wählt zufälliges Element aus  
random.shuffle() #mischt liste

Math:  
 sqrt() degrees(*[Radius]*) #wandelt um  
 sin/cos/tan radians(*[Grad]*)  
 pi log(*[zahl]*)

**Bibliotheken/Module**

*[var]*=Time.localtime()  
*[var]*.tm\_year #auch month, hour, …

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

Bsp.:

F=open(Ordner/Renndaten.dat, mode=‘wb‘) #speichern  
pickle.dump(Liste, f)  
f.close()

F=open(pfad, ‚rb‘) #lesen  
*[var]*=pickle.load(f)  
f.close()

From tkinter import filedialog  
*[pfad]*=filedialog.askopenfilename() #pfad muss noch geöffnet #werden --> für Codierung  
*[Stream]*=filedialog.askopenfile()

**Speichern/Lesen**

*[Stream]*=filedialog.asksaveasfile()

Try:  
 *[Funktionsblock]*  
except:  
 *[Funktionsblock]*

**Fehler abfangen**

From tkinter import \*  
fenster=Tk()  
fenster.title(‘’)  
fenster.geometry("1000x800")  
fenster.mainloop()

**Fenster**

*[var]*=Label(master=*[wo es drin ist]*, text=‘‘,…)  
#rechteckige Fläche

**Knopf**

**Label**

*[var]*=Button(master=[], text’’, command=*[Fkt]*)  
#Button

*[var]*=Frame(master=[])  
#Container (für Layout und Anordnung praktisch)

**Frames**

*[var]*=Entry(master=[])  
#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

**Eingabefelder**

*[varKontroll]*=Stringvar()  
*[varButton1]*=Radiobutton(master=[], text=‘‘, value=*[varKontroll]*, variable=*[Wertvariable]*)  
*[varButton2]*=Radiobutton(master=[], text=‘‘, value=*[varKontroll]*, variable=*[Wertvariable2]*)  
*[varAusgabe]*=*[varKontroll]*.get()  
#1 aus n Auswahl; Kontrollvariable für alle nötig, nimmt Wert für ausgewählten an (deswegen get zum Wert bekommen!)

**Radiobuttons**

Checkbutton  
#m aus n Auswahl

**Checkboxen**

*[var]*=PhotoImage(master=[], file=*[pfad]*)  
#Bild, nur png!

**Bilder**

*[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

*[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

**Widgetmethoden**

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

**Widget anordnen**

Für meisten Widgets:

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

**Widgetattribute**

*[varMenu]*=Menu(master=[])  
*[master]*.config(menu=*[varMenu]*) #Fenster hat Menü

*[varerstes]*=Menu(*[varMenu]*)  
*[varMenu]*.add\_cascade(label=’erstes’, menu=*[varerstes]*)  
 #Hauptauswahl  
*[varerstes]*.add\_command(label="Hi!", command=*[Fkt.]*)  
 #Untermenüpunkt  
*[varerstes]*.add\_separator() #Trennstrich

**Menübar**

Man muss nicht pack()

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**

**Threads**

Assert *[Bedingung]* #bricht Programm ab, wenn falsch

Class *[Klassenname]*:  
 *[Docstring (Erklärung als str)]*  
 def \_\_init\_\_(self, …):  
 *[Anweisung zur Initialisierung, Attribute]*  
 #self.inhalt=0  
 def \_\_*[Methode1]*(self, …):  
 *[Anweisungen; definiert diese Methode]*  
 #self.inhalt+=500

A=*[Klassenname]*()  
a.*[Methode1]*()

**Objektorientierung**

Import 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

Position() #gibt Position wieder  
xcor/ycor #gibt Positionskoordinaten wieder  
#Bsp.: if b.xcor>30: […]

**Turtle**