

Einführung in die objektorientierte Programmiersprache Python

1. Erste einfache Programme

```
#hallo1
a = "Herr"
b = "Albert"
c = "Hallo" + a + b
print c
```

```
#hallo2
nachname = raw_input("Gib deinen Nachnamen ein: ")
vorname = raw_input("Jetzt deinen Vornamen: ")
name = "Hallo " + vorname + " " + nachname
print name
```

```
#rechnen1
a = 4
b = 5
Umfang = 2*(a + b)
print Umfang           # python unterscheidet zwischen gross und kleinschreibung
```

```
#rechnen2
erste = raw_input("Erste Zahl eingeben: ")
zweite = raw_input("Zweite Zahl eingeben: ")
erste = float(erste)
zweite = float(zweite)
ergebnis = erste + zweite
print ergebnis
```

```
#rechnen3
erste = input("Erste Zahl eingeben: ")
zweite = input("Zweite Zahl eingeben: ")
ergebnis = erste + zweite
print ergebnis
```

```
#quader1
laenge = 3; breite = 4; hoehe = 5
volumen = laenge*breite*hoehe
oberflaeche = 2*( laenge * breite
                  + laenge * hoehe
                  + breite * hoehe)
print "Volumen: ", volumen
print "Oberflaeche: ", oberflaeche
```

```
#geld1
geld = input("Geldbetrag eingeben: ")
geld = int(geld)
zwanziger = geld/20;   geld = geld%20
zehner = geld/10;     geld = geld%10
fuenfer = geld/5;     geld = geld%5
zweier = geld/2;     geld = geld%2
einer = geld
print "Der Betrag setzt sich wie folgt zusammen: "
print zwanziger, "mal 20 Euro"
print zehner, "mal 10 Euro"
print fuenfer, "mal 5 Euro"
print zweier, "mal 2 Euro"
print einer, "mal 1 Euro"
```

```

#turtle0
from turtle import *
forward(100)
left(90)
forward(60)
circle(40)
raw_input()

```

```

#rtrainer1.py
import random
import time
print 'Multiplikationstrainer'
print
startzeit = time.time()
for i in range(1):
    a = random.randint(1, 20)
    b = random.randint(1, 10)
    ergebnis = -1
    while ergebnis != a*b:
        ergebnis = input(str(a) + "*" + str(b) + "=")
        if ergebnis == a*b:
            print "Richtig"
        else:
            print "Falsch! Versuchen sie es noch einmal."
zeit = int(time.time() - startzeit)
print u"F374r die Aufgaben haben sie", zeit, "Sekunden beoetigt"

```

2. Fehlersuche

```

#fehler1
a = "Herr"
b = "Albert"
c = "Hallo" + a + x
print c

```

```

#fehler2
a = 4
b = 5
Umfang = 2 ( a + b )
print umfang

```

```

#fehler3
erste = raw_input("Erste Zahl eingeben: ")
zweite = raw_input("Zweite Zahl eingeben: ")
ergebnis = erste + zweite
print ergebnis

```

```

#fehler4
summe = 0
for i in [1, 2, 3, 4, 5]
summe = summe + i
print "Summe von 1 bis ", i, Summe
print "Ende der Rechnung"

```

3. Was erscheint auf dem Bildschirm

```
#bildsch1
from math import *

r_zahl = 5.6789E-3; i_zahl = 123; zeichen = "Python"

print i_zahl
print "%6i" % i_zahl
print "%6.4i" % i_zahl

print r_zahl
print "%e" % r_zahl
print "%10.3f" %r_zahl

print zeichen
print "%10s" % zeichen
print "%10.3s" % zeichen

for x in range(5):
    print x, "%10.3f" %sqrt(x)
```

4. Schleifen

```
#schleifen1
x=0
while x<=10:
    x=x+1
    print(x),

#schleifen2
for i in range(-6, 6, 2):
    print i
```

5. Verzweigungen

```
#verzw1; Verzweigung mit if, elif und else
#zahl = raw_input("Zahl eingeben: ")
#zahl = float(zahl)
zahl = input("Zahl(>100, =50, =10 oder andere) eingeben: ")
if zahl > 100:
    print"groesser 100"
elif zahl == 50:
    print"gleich 50"
elif zahl == 10:
    print"gleich 10"
else:
    print"hallo"
```

6. Funktionen

#fkt1; lokale und globale Variable

```
from turtle import *
```

```
seite = 10
```

```
def dreieck():
```

```
    seite = 100
```

```
    forward(seite)
```

```
dreieck()
```

```
raw_input()
```

#fkt2; globale und lokale Variable

```
from turtle import *
```

```
seite = 10
```

```
def dreieck():
```

```
    global seite
```

```
    seite = seite + 80
```

```
    forward(seite)
```

```
dreieck()
```

```
raw_input()
```

#fkt3; Funktion mit Rueckgabewert

```
def quadrat(x):
```

```
    global qu
```

```
    qu = x*x
```

```
    return qu
```

```
quadrat(2)
```

```
print qu
```

```
quadrat(4)
```

```
print qu
```

```
summe = quadrat(3) + quadrat(4)
```

```
print summe
```

7. Die Funktionen min, max, len, reverse

#min1; min; max; len; reverse

```
a = (1, 2, 3); b = 'hallo'
```

```
print min(a)
```

```
print max(a)
```

```
print len(b)
```

```
s = ['rot', 'gelb', 'gruen']
```

```
print s
```

```
s.reverse()
```

```
print s
```

8. Turtle

```
#turtle1
from turtle import *
from math import *
from time import *
x=100
a=hypot(x, x)
width(4)
fill(1)
right(90); forward(x)
left(90); forward(100)
left(135); forward(a)
right(135); forward(100)
right(90); forward(100)
fill(0)
color('red')
tracer(0)
up()
#goto(-100,-50)
down()
tracer(1)
right(135)
forward(141.4)
tracer(0)
circle(50)
tracer(1)
for i in range(10):
    circle(i)
    sleep(0.5)
raw_input()
```

```
#turtle2
from turtle import *
b=100
up()
goto(-80, -80)
down()
forward(b); left(135)
forward(150); right(135)
forward(100)
raw_input()
```

```
#turtle3
from turtle import *

def sprung(laenge, winkel):
    up()
    left(winkel); forward(laenge); right(winkel)
    down()

forward(20)
sprung(40, 0)
forward(40)
sprung(40, 135)
forward(40)
sprung(160, 180)
forward(40)
raw_input()
```

9. Grafik

```
#grafik1
from Tkinter import *
fenster = Tk()
c = Canvas(fenster, width=800, height=600, bg="yellow")
c.pack()

def putpixel(x, y, farbe):
    l = c.create_line(x,y,x+1,y, width=4, fill=farbe)

r = c.create_rectangle(40,20,160,80,fill="red")
o = c.create_oval(140,120,260,240,fill="PeachPuff")
c.create_text(100, 120, text="hallo")

for i in range(20):
    putpixel(180+i, 100, "blue")

fenster.mainloop()
```

10.Objekte

```
#objekte1
from turtle import *

anna=Pen(); bea=Pen(); carla=Pen(); doris=Pen()
skroeten=[anna, bea, carla, doris]
anna.left(0); anna.forward(50)
bea.left(90); bea.forward(50)
carla.left(180); carla.forward(50)
doris.left(270); doris.forward(50)

for s in skroeten:
    s.width(4)
    s.right(30)
    s.color('red')

for i in range(3):
    for s in skroeten:
        s.forward(50)
        s.left(120)
```

11. Sequenzen: Tupel; Listen

```
#sequenzen1  
s=[1]; s=s*10  
print s[2]
```

```
for i in range(3):  
    s[i]=i  
    print s[i],
```

```
#sequenzen2  
a = [1, 2, 3]  
s = [a]; s=s*2  
b = [[10, 11, 12],[20, 21, 22]]  
print a  
print s  
print s[0][2]  
print b  
print b[1][1]
```

```
#liste1; listen erzeugen  
from random import *  
liste1 = [1, 2, 3, 4, 5]  
print liste1  
print liste1[1]  
print liste1[-1]  
shuffle(liste1)  
print liste1  
x = choice(liste1)  
print x  
liste2 = ["Hallo", 3, (4, 5)]  
print liste2  
liste3 = [i**2 for i in [0, 1, 2, 3]]  
print liste3  
liste4 = [i*2 for i in range(4)]  
print liste4  
liste5 = [ i for i in range(50) if i%7 ==0]  
print liste5  
liste6 = [i for i in range(10, 20, 3)]  
print liste6
```

12. Weitere Programme

```
#grafikfkt1
from Tkinter import *
fenster = Tk()
c = Canvas(fenster, width=600, height=600, bg="green")
c.pack()

def achsen():
    c.create_line(0, 300, 600, 300, arrow="last", fill="black")
    c.create_line(300, 0, 300, 600, arrow="first", fill="black")
    c.create_text(350, 310, text='1', font=('Arial', 14))
    c.create_text(310, 250, text='1', font=('Arial', 14))

def graf1(x1, y1, x2, y2):
    c.create_line(x1, y1, x2, y2, fill="blue", width=2)

def graf2(x1, y1, x2, y2):
    c.create_line(x1, y1, x2, y2, fill="red", width=2)

def gleichung1(x1_1, x1_2, y1_1, y1_2):
    global m1, m1r, n1
    m1=(float(y1_1) - float(y1_2))/(float(x1_1) - float(x1_2))
    n1=float(y1_1) - m1*float(x1_1)
    m1r=round(m1, 2)
    return m1r

def gleichung2(x2_1, x2_2, y2_1, y2_2):
    global m2, m2r, n2
    m2=(float(y2_1) - float(y2_2))/(float(x2_1) - float(x2_2))
    n2=float(y2_1) - m2*float(x2_1)
    m2r=round(m2, 2)
    return m2r

def schnittpunkt():
    global xs, ys
    xs=(float(n2)-float(n1))/(float(m1)-float(m2))
    ys=m1*xs + float(n1)
    xs=round(xs/50,2); ys=round(ys/50, 2)

def loesung():
    c.create_text(80, 10, text='y1 = ' + str(m1r) + ' x + ' + str(n1/50))
    c.create_text(80, 30, text='y2 = ' + str(m2r) + ' x + ' + str(n2/50))
    c.create_text(80, 50, text='S ( ' + str(xs) + ' ; ' + str(ys) + ' ) ')

x1_1=-300; x1_2=300; y1_1=-200; y1_2=260
x2_1=-300; x2_2=300; y2_1=200; y2_2=-400

achsen()
graf1(x1_1+300, -y1_1+300, x1_2+300, -y1_2+300)
graf2(x2_1+300, -y2_1+300, x2_2+300, -y2_2+300)
gleichung1(x1_1, x1_2, y1_1, y1_2)
gleichung2(x2_1, x2_2, y2_1, y2_2)
schnittpunkt()
loesung()

fenster.mainloop()
```

```

#grafikfkt2
from Tkinter import *
from math import *
fenster = Tk()
c = Canvas(fenster, width=600, height=600, bg="green")
c.pack()

def achsen():
    c.create_line(0, 300, 600, 300, arrow="last", fill="black")
    c.create_line(10, 0, 10, 600, arrow="first", fill="black")

def putpixel(x, y, farbe):
    c.create_line(x, y, x+1, y, width=4, fill=farbe)

def fkt1():
    for i in range(600):
        putpixel(i+10, -(sin(float(i)/30))*100+300, 'red')

def fkt2():
    for i in range(600):
        putpixel(i+10, -(cos(float(i)/60))*100+300, 'blue')

def anzeigen():
    c.create_text(40, 20, text='Sinus', font=('Arial', 18), fill='red')
    c.create_text(50, 60, text='Cosinus', font=('Arial', 18), fill='blue')

achsen()
fkt1()
fkt2()
anzeigen()
fenster.mainloop()

#r_beute1
from Tkinter import *
#from math import *
fenster = Tk()
c = Canvas(fenster, width=800, height=600, bg="green")
c.pack()
def achsen():
    c.create_line(0, 580, 800, 580, arrow="last", fill="black")
    c.create_line(20, 0, 20, 800, arrow="first", fill="black")
    c.create_text(60, 20, text='Hechte', fill='red', font=('Arial', 18))
    c.create_text(62, 40, text='Karpfen', fill='yellow', font=('Arial', 18))
def putpixel(x, y, farbe):
    c.create_line(x, y, x+1, y, width=4, fill=farbe)
def rechnen():
    q=0.05; v=0.05; c=0.01; d=0.00034;
    r0=6.0; b0=167.0 #r0=5.0 b0=167.0
    r=r0; b=b0
    for i in range(800):
        hilf=((1+q)*b-c*r*b)
        r=((1-v)*r+d*r*b)
        b=hilf
        putpixel(i+20, float(b)+300, 'yellow')
        putpixel(i+20, float(r*10)+300, 'red')

achsen()
rechnen()
fenster.mainloop()

```

```

#laby1s; version fuer schueler
from Tkinter import *
fenster = Tk()
c = Canvas(fenster, width=800, height=600, bg="green"); c.pack()

```

```

x1a=300; y1a=200; x1e=400; y1e=400
x2a=400; y2a=200; x2e=550; y2e=250
c.create_rectangle(x1a,y1a,x1e,y1e, fill="red")
c.create_rectangle(x2a,y2a,x2e,y2e, fill="blue")

```

```

def hindernisse_loeschen():
    global h
    hy = [0 for i in range(600)];
    h = [hy[:] for i in range(800)]
def hindernis_aufstellen():
    for i in range(y1a, y1e):
        for j in range(x1a, x1e):
            h[j][i] = 1
    for i in range(y2a, y2e):
        for j in range(x2a, x2e):
            h[j][i] = 1

def ausschau_halten():
    global vornefrei, rechtsfrei
    if richtung == 1:
        if h[x+a][y] == 1:
            vornefrei = False
        else:
            vornefrei = True
        if h[x][y+a] == 1:
            rechtsfrei = False
        else:
            rechtsfrei = True
    if richtung == 2:
        if h[x][y-a] == 1:
            vornefrei = False
        else:
            vornefrei = True
        if h[x+a][y] == 1:
            rechtsfrei = False
        else:
            rechtsfrei = True
    if richtung == 3:
        if h[x-a][y] == 1:
            vornefrei = False
        else:
            vornefrei = True
        if h[x][y-a] == 1:
            rechtsfrei = False
        else:
            rechtsfrei = True
    if richtung == 4:
        if h[x][y+a] == 1:
            vornefrei = False
        else:
            vornefrei = True
        if h[x-a][y] == 1:
            rechtsfrei = False
        else:
            rechtsfrei = True

```

```

def drehen_l():
    global richtung
    richtung = richtung + 1
    if richtung > 4:
        richtung = 1
    else:
        richtung = richtung

def vor():
    global x, y
    if richtung == 1:
        x=x+1
    if richtung == 2:
        y=y-1
    if richtung == 3:
        x=x-1
    if richtung == 4:
        y=y+1

def zeigen():
    c.create_line(x,y,x+1, y+1, width=4, fill="yellow")

x=0; y=300
a = 5
richtung = 1
hindernisse_loeschen()
hindernis_aufstellen()

for i in range(20):
    ausschau_halten()
    vor(); zeigen()

ausschau_halten()
while vornefrei == True:
    ausschau_halten()
    vor(); zeigen()

drehen_l()
for i in range(20):
    ausschau_halten()
    vor(); zeigen()

while rechtsfrei == False:
    ausschau_halten()
    vor(); zeigen()

c.create_text(100, 100, text=("richtung=",richtung))
c.create_text(100, 120, text=("x=",x))
c.create_text(100, 140, text=("y=",y))
c.create_text(100, 160, text=("vornefrei=",vornefrei))
c.create_text(100, 180, text=("rechtsfrei=",rechtsfrei))
c.create_text(100, 220, text=("h(x,y)=",h[x][y]))

fenster.mainloop() # entfällt bei windows

```

#alkohol1; Beispiel fuer Simulationen

```
def information():
    print "Alkoholabbau"

def dateneingabe():
    global a0
    a0 = input("Anfangsalkoholgehalt in Promill eingeben: ")

def berechnung():
    global a, z
    a = a - a * 0.2
    z = z + 1

def datenausgabe():
    ap = round(a, 2)
    print ap,

def abfrage():
    global antwort
    antwort = raw_input(" Weitere Werte j/n ")

information()
dateneingabe()
a = a0; z = 0; antwort = "j"
while antwort != "n":
    berechnung()
    datenausgabe()
    abfrage()
```

13. Weitere Informationen

Zuweisungen S. 46

`x = 'Wort'` `x = y = 1` `x, y = 1, 2`

Rechenoperationen und Funktionen S. 51

`x = y/2` ganzzahlige Division; liefert 0 für x wenn y = 1 ist

`x = y/2.0` liefert 0.5 für x wenn y = 1 ist

`y = min (1, 2, 3)` liefert 1 für y
`y = len ('hallo')` liefert 5 für y

`s = ['rot', 'gelb', 'gruen']`
`s.reverse()`
`print s` liefert gruen gelb rot

Verbinden von Zeilen S. 58

`summe = 1 \`
`+2` verbindet die Zeilen

`summe = 1 \ #Kommentar` ist nicht erlaubt

`summe = (1 + 2 + 3 + #Kommentar`
`4 + 5)` Klammern verbinden Zeilen ebenfalls

Eingaben S. 62

`a = raw_input("Name: ")` a ist String

`x = input("Name: ")` x ist Zahl

Sequenzen S. 177

Datentypen; Typenumwandlungen S. 73; 565

`int; long` ganze Zahlen
`float` Kommazahlen

`abs(x)` liefert den absoluten Wert einer Zahl
`chr(i)` liefert das Zeichen was den ASCII-Code i besitzt
`cmp(x, y)` vergleicht x und y; negativ falls $x < y$; 0 falls $x == y$; positiv falls $x > y$
`float(x)` liefert zu einen String oder einer Zahl eine Gleitkommazahl
`int(x)` konvertiert String oder Zahl in ganze Zahl
`len(s)` liefert die Länge einer Sequenz
`ord(c)` liefert zu einem ASCII-Zeichen dessen ASCII-Code
`pow(x, y)` berechnet die Potenz x hoch y
`range(a, b)` liefert eine Liste von a bis b-1
`round(x, n)` rundet auf n Stellen hinter dem Komma
`str(x)` konvertiert Zahl in Zeichenkette

Funktionen der Module math und random S. 567; 577

| | | | |
|----------------------------|------------------------------|---------------------------|---------------------------|
| <code>cos(x)</code> | Kosinus | <code>e</code> | Eulersche Zahl |
| <code>exp(x)</code> | e hoch x | <code>hypot(x, y)</code> | Euklidischer Abstand |
| <code>log(x)</code> | ln x | <code>log10(x)</code> | Basis 10 |
| <code>pow(x,y)</code> | x hoch y | <code>pi</code> | die Zahl 3.1415 ... |
| <code>sin(x)</code> | Sinus | <code>tan(x)</code> | Tangens |
| <code>sqrt(x)</code> | Quadratwurzel | | |
| <code>randint(a, b)</code> | ganze Zahl aus dem Intervall | <code>random()</code> | Kommazahl aus 0.0 bis 1.0 |
| <code>shuffle(x)</code> | mischt die Liste x | <code>uniform(a,b)</code> | Gleitkommazahl aus [a,b] |
| <code>choice(x)</code> | zufälliges Element aus x | | |