**Erforderliche Befehle für TigerJython**

Im Folgenden sind Beispiele für einige grundlegenden Befehle gegeben. Die Programmtexte können über copy-paste in den Editor von Tigerjython übertragen werden (Bei copy-paste aus einer pdf-Datei, werden die Einrückungen nicht übernommen, deshalb sollte die Word-Datei genutzt werden).

**1)** Die einfachste Schleife ist die **repeat-Schleife**, bei der der eingerückte Programmteil n mal wiederholt wird. (Im TigerJython-Editor wird mit der Tab-Taste eingerückt)

# vor Kommentaren muss ein # stehen

# repeat Schleife

#

n = 10

i = 0

repeat n:

 i += 1

 print i

**2)** Die **for-Schleife** zählt von einem Anfangswert bis zu einem Endwert durch. Sofern nicht anders angegeben zählt die Schleife von 0 bis n-1. Man kann aber auch Anfangs- und Endwert angeben, sowie eine Schrittweite.

#

# for Schleife

#

n = 10

for i in range(n): # von 0 bis n-1

 print i

print "---"

for i in range(1,n+1,1): # (start,stop,step)

 print i

**3)** Die **while-Schleife** wiederholt den eingerückten Programmteil solange, bis die angegebene Bedingung erfüllt ist.

#

# while Schleife

#

n = 10

i = 0 # definiert die Variable i, weist ihr den Anfangswert 0 zu

while i<n:

 i += 1

 print i

**4)** Für stochastische Simulationen werden **Zufallszahlen** benötigt. Die Bibliothek random stellt die Befehle random() und randint(a,b) zur Verfügung.

#

# Zufallszahlen

#

from random import \*

n = 10

repeat n:

 x = random() # von 0 bis 1

 k = randint(1,6) # ganze Zahlen von 1 bis 6

 print x,k

**5)** Zur Buchhaltung bei Simulationen müssen Daten gespeichert werden. Hierzu eigenen sich **Variablenfelder** (array). Arrays müssen zunächst definiert werden und den Elementen des Arrays einen Anfangswert zugeordnet werden. Im Beispiel hier werden einfach Zufallszahlen in dem Array gespeichert. Hierbei ist $A\_{i}=A\left[i\right]$

#

# array

#

from random import \*

n = 10

A = [0 for i in range(n)] # Definition, Wert 0 zugeordnet

for i in range(n):

 A[i] = random()

for i in range(n):

 print i,A[i]

**6)** Man kann auch **mehrdimensionale Felder** definieren, zum Beispiel um ein Spielbrett zu definieren.

Hierbei ist $A\_{i,j}=A\left[i\right][j]$

#

# array mit zwei Indices

#

from random import \*

n = 4

A = [[0 for i in range(n)] for i in range(n)] # Definition, Wert 0

for i in range(n):

 for j in range(n): # verschachtelte Schleife

 A[i][j] = random()

for i in range(n):

 for j in range(n):

 print i,j,A[i][j]

**7)** Bei einer **bedingten Abweisung** wird der eingerückte Programmteil nur ausgeführt, wenn die gegebene Bedingung erfüllt ist.

#

# bedingte Anweisung

#

from random import \*

n = 10

for i in range(n):

 a = random()

 if a > 0.5: # >, <, <=, >=, == vergleichendes gleich

 print i,a

**8)** Um sich die bisweilen aufwändige Programmierung der Grafik zu ersparen, kann man die berechneten Daten **in eine Textdatei ausgeben** und diese Datei dann in eine Tabellenkalkulation einlesen und graphisch garstellen.

#

# Ausgabe in eine Textdatei

#

from random import \*

n = 10

A = [0 for i in range(n)]

datei = open("Ausgabedatei.txt","w") # Datei zum Schreiben öffnen

for i in range(n):

 x = random()

 datei.write(str(i)+" "+str(x)+"\n") # Daten in Datei schreiben

datei.close() # Datei schließen

In dem Ordner, in dem sich der Programmcode befindet, wird eine Textdatei erzeugt mit dem Namen

Ausgabedatei.txt

**9) Simulation der Kinetik 1. Ordnung**

from random import \*

n = 6 # Größe des Spielbretts n mal n

Pos = [[1 for i in range(n+1)] for i in range(n+1)]

f = open("Kinetik1.txt", "w") # Ausgabedatei öffnen

m = n\*\*2 # Zahl der Steine zu Beginn,

i = 0 # Schleifenvariable auf null setzen

f.write("k = "+str(1/m)+"\n") # Ausgabe in Datei

f.write(str(i)+" "+str(m)+"\n") # Ausgabe in Datei

print i,m # Bildschirmausgabe

while m > 0: # solange noch Steine Spielbrett

 ix = randint(1,n) # x-Koordinate per Zufall

 iy = randint(1,n) # y-Koordinate per Zufall

 i = i + 1 # Schleifenvariable erhöhen

 if Pos[ix][iy]==1: # wenn ein Stein auf dem Feld liegt

 m = m - 1 # Zahl Edukt-Teilchen reduzieren

 Pos[ix][iy]=0 # Stein vom Feld entfernen

 print i,m # Bildschirmausgabe

 f.write(str(i)+" "+str(m)+"\n")# Ausgabe in Datei

f.close() # Ausgabedatei schließen