Programmieren mit Jython – Teil 4: Selektion und Zufallszahlen

Lernziele

- Verwendung der `if` – Anweisung: gewisse Zeilen werden nur dann ausgeführt, wenn eine Bedingung erfüllt (oder nicht erfüllt) ist.
- Erzeugung von Zufallszahlen.

Selektion

Manchmal sollen gewisse Teile eines Programmes nur dann ausgeführt werden, wenn eine bestimmte Bedingung erfüllt ist. Dazu stehen in Jython die Befehle `if`, `elif` und `else` zur Verfügung. Die allgemeine Form der Selektion sieht in Jython wie folgt aus:

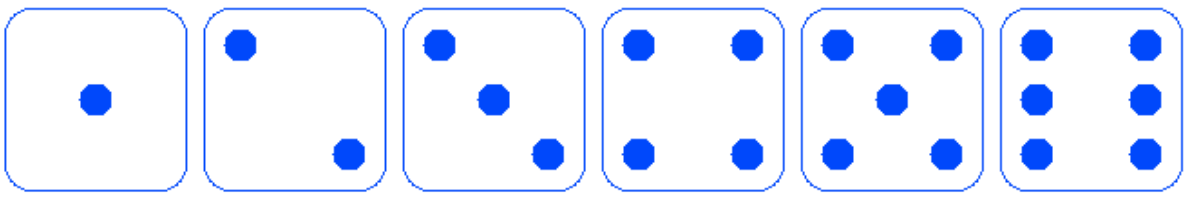
```
if bedingung1:
      anweisung1
elif bedingung2:
      anweisung2
else:
      anweisung3
```

Falls die Bedingung `bedingung1` wahr ist, wird `anweisung1` ausgeführt (sie kann auch aus mehreren Zeilen bestehen). Wenn nicht, wird, falls `bedingung2` wahr ist, `anweisung2` ausgeführt. Falls weder `bedingung1` noch `bedingung2` wahr ist, werden die Anweisungen nach `else` – also `anweisung3` – ausgeführt. Dies soll am folgenden Beispiel erläutert werden:

```
if stunden < 10:
      grussformel = "Guten Morgen"
      aktion = "Kaffee trinken"
elif stunden < 17:
      grussformel = "Guten Tag"
      aktion = "Arbeiten"
else:
      grussformel = "Guten Abend"
      aktion = "Ausruhen"
```

Alle Zeilen, die nach dem Doppelpunkt eingerückt sind, gehören zu der bedingten Anweisung.

Unser nächstes Ziel ist es, ein Programm `Wuerfel(Augenzahl)` zu schreiben, welches eine Seite eines Würfels zeichnet, abhängig vom Wert der Augenzahl:



Augenzahl=1 Augenzahl=2 Augenzahl=3 Augenzahl=4 Augenzahl=5 Augenzahl=6

So ist der mittlere Punkt nur dann zu zeichnen, wenn die Augenzahl ungerade ist. Dies kann man mit der Bedingung `augenzahl%2 == 1` testen (`augenzahl%2` liefert den Rest einer ganzzahligen Division von `augenzahl` geteilt durch `2`).

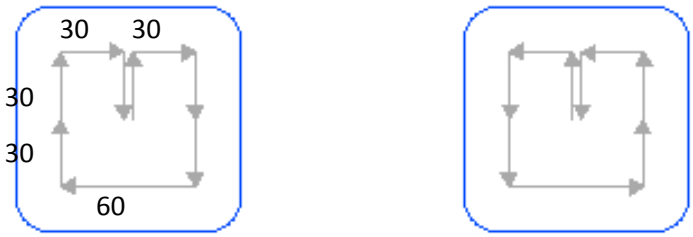
Wichtig: Das doppelte Gleichheitszeichen ist das vergleichende Gleichheitszeichen. Das einfache Gleichheitszeichen ist für die Zuweisung eines bestimmten Wertes an eine Variable reserviert.

Aufgaben zur Selektion

4.1) a) Öffne die Lösung zur Aufgabe 1.5 b) aus Teil 1 des Informatikkurses (1. Doppellektion). Führe das Programm aus: Es sollte ein Quadrat mit abgerundeten Ecken und der Seitenlänge 100 gezeichnet werden. Lösche die Zeile mit dem Befehl `dot(x)`, so dass kein Punkt mehr gezeichnet wird. Kontrolliere auch, ob sich sowohl zu Beginn als auch am Ende der Programmausführung die Schildkröte in der Mitte befindet und nach oben blickt.
Packe die Codezeilen zum Zeichnen des Quadrates in ein Unterprogramm mit dem Namen `RundEckQuadrat()`. Tippe anschliessend folgendes Unterprogramm ab und rufe es mit verschiedenen Werten von `Augenzahl` (1 bis 6) auf. Was fällt Dir auf?

```
def Wuerfel(Augenzahl):
      RundEckQuadrat()
      if Augenzahl%2 == 1:
            dot(18)
```

b) Nun soll das Unterprogramm `Wuerfel(Augenzahl)` so ergänzt werden, dass es für alle möglichen Werte von `Augenzahl` das richtige Bild zeichnet. Dazu soll die Turtle entlang eines vorgeschriebenen Weges einmal rundherum durch alle Positionen laufen, an denen möglicherweise ein Punkt gesetzt werden soll. Wähle dazu einen der folgenden Wege:



Zu Beginn des Weges soll der Stift nach oben gezogen werden, ganz am Ende soll er wieder nach unten gelassen werden. Wichtig ist auch, dass nach Ablaufen des Weges die Schildkröte um 180 Grad gedreht wird, damit sie wieder nach oben schaut. In jeder möglichen Position soll dann entschieden werden, ob ein Punkt gesetzt wird oder nicht. Das Unterprogramm wird also folgende Struktur aufweisen:

```
def Wuerfel(Augenzahl):
      RundEckQuadrat()
      if Augenzahl%2 == 1:
            dot(18)
      penUp()             #Hebt den Stift an

      ...                 #Hier muss der Weg abgelaufen werden.
      ...                 #An jeder Position muss entschieden werden,
      ...                 #ob der Punkt gesetzt wird oder nicht.

      right(180)          #Dreht die Schildkröte nach oben
      penDown()           #Lässt den Stift nach unten senken
```

c) Überprüfe das richtige Funktionieren Deines Unterprogrammes mittels folgender Codezeilen:

```
hideTurtle()
augen = 1;
repeat 6:
      setPos(-400 + augen * 110, 0)
      Wuerfel(augen)
      augen = augen + 1
```

Zufallszahlen

Jython stellt (eingebaute) Funktionen bereit, die Zufallszahlen liefern. Dazu muss allerdings zuerst die nötige Bibliothek „random“ importiert bzw. geladen werden (in den obersten Programmzeilen, z. Bsp. nach `from gturtle import *`):

```
from random import *
```

Anschliessend kann man folgende Funktionen benutzen:

`x = random()`	liefert eine zufällige *reelle* Zahl zwischen 0 und 1
`y = randint(a, b)`	liefert eine zufällige *ganze* Zahl zwischen a und b (diese einschliesslich)

Aufgaben zu Zufallszahlen

4.2) Schreibe ein Unterprogramm `Wurf()`, das eine zufällige natürliche Zahl zwischen 1 und 6 bestimmt und diese als Würfelbild zeichnet. Teste das Unterprogramm mit folgendem Hauptprogramm:

```
hideTurtle()
y = -200
repeat 5:
    x=-300
    repeat 6:
        setPos(x,y)
        Wurf()
        x=x+110
    y=y+110
```

4.3) Schreibe ein Unterprogramm `GezinkterWurf()`, das in der Hälfte aller Fälle eine 6 anzeigt und in den restlichen Fällen je eine der Zahlen 1 bis 5. Teste das Unterprogramm mit einem Hauptprogramm wie in Aufgabe 4.2.

Zusatzaufgaben

4.4) Schreibe ein Programm `WurfVar(anzahl)`, das den Wurf mehrerer Würfel anzeigt. Die Anzahl Würfel soll dabei als Variable gesetzt werden können. Zudem sollen die Würfel an einer zufälligen Position des Bildschirms angezeigt werden, eine mögliche Überlappung darf vorkommen.

4.5) Schreibe ein Programm `WuerfelVar(augenzahl,dimension)`, welches das „RundEckQuadrat“ bzw. den Würfelrahmen in variabler Grösse darstellt. Der Würfel soll dabei den Durchmesser `dimension` haben.

Hausaufgaben

Löse folgende Aufgaben mit **Papier und Bleistift** (ohne Computer):

H4.1) Es soll ein Programm geschrieben werden, das eine Ziffer wie folgt zeichnen kann.

Ein solches Programm ist unten angegeben. Jedoch fehlen die Bedingungen, wann ein Strich gezeichnet werden soll (bzw. wann der Stift unten oder oben sein soll). Am rechten Rand sieht man den Weg, der verfolgt wird.

Die Bedingung für die Zahlen 0, 1, 3, 5, 6, 8, 9 kann mit „or“ wie folgt bewerkstelligt werden:

```
zahl==0 or zahl==1 or zahl==3 or zahl==5 or zahl==6 or zahl==8 or zahl==9
```

Diese kann (vorausgesetzt die Variable `zahl` ist immer eine der Zahlen von 0 bis 9) als

```
zahl!=2 and zahl!=4 and zahl!=7
```

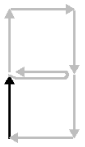
oder auch als

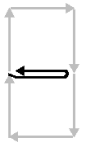
```
zahl in {0,1,3,5,6,8,9}
```

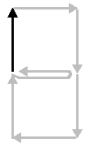
kürzer geschrieben werden.

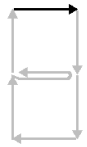
Verwende jede dieser drei Schreibarten **mindestens einmal und füge die fehlenden Bedingungen ein**:

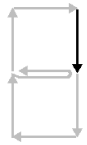
```
def digit(zahl,laenge):
    if [                                ]:
        penDown()
    else:
        penUp()
    forward(laenge)
    if [                                ]:
        right(90)
        penDown()
        forward(laenge)
        back(laenge)
        left(90)
    if [                                ]:
        penDown()
    else:
        penUp()
    forward(laenge)
    right(90)
    if [                                ]:
        penDown()
    else:
        penUp()
    forward(laenge)
    right(90)
    if [                                ]:
        penDown()
    else:
        penUp()
    forward(laenge)
```



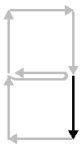


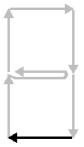






```
if                                                    :
    penDown()
else:
    penUp()
forward(laenge)
right(90)
if                                                    :
    penDown()
else:
    penUp()
forward(laenge)
right(90)
```





H4.2) Vervollständige folgende Tabelle:

Der Befehl	**liefert eine zufällige reelle Zahl zwischen:**
`random()`	0 und 1
`random()*2`	
`random()*2+1`	
`(random()+1)*3`	
	-1 und 1
	50 und 100
	a und b