While-Schleife

2.8 WHILE-SCHLEIFEN

EINFÜHRUNG

Du hast bereits den Befehl *repeat* kennengelernt, mit dem du einen Programmblock mehrmals wiederholen kannst. *repeat* kannst du so allerdings nur im TigerJython verwenden. Die *while*-Struktur ist aber überall einsetzbar.

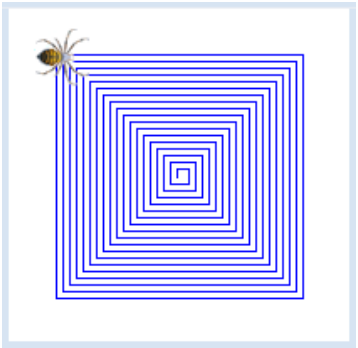
Die *while-Schleife* wird mit dem Schlüsselwort *while* eingeleitet, gefolgt von einer Schleifenbedingung. Die Anweisungen im Schleifenblock werden wiederholt, so lange die Bedingung erfüllt ist. Nach Ende der Wiederholungen wird das Programm mit der nächsten Anweisung nach dem Schleifenblock fortgesetzt.

PROGRAMMIERKONZEPTE: *Iteration, While-Struktur, Verknüpfte Bedingungen, Schleifenabbruch*

SPINNENNETZ

Mit Hilfe einer while-Schleife soll die Turtle eine rechteckige Spirale zeichnen. Dazu verwenden wir eine Variable a, die den **Startwert 5** erhält und bei jedem Schleifendurchlauf **um 2 vergrössert** wird. Solange die **Bedingung a < 200** wahr ist, werden die Anweisungen im Schleifenblock ausgeführt.

Zur Erhöhung des Spassfaktors nimmst du anstelle der Turtle eine Spinne.



```
from gturtle import *

makeTurtle("sprites/spider.png")

a = 5
while a < 200:
forward(a)
right(90)
a = a + 2
```

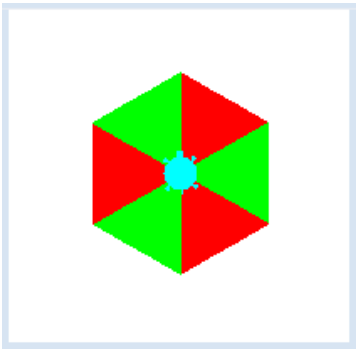
MEMO

Eine while-Schleife dient zur Wiederholung eines Programmblocks. Die Bedingung muss wahr sein, damit der Programmblock ausgeführt wird. Darum spricht man auch von einer *"Ausführungsbedingung"* oder *"Laufbedingung"*. Fehlt im Schleifenblock die Wertänderung, bleibt die Laufbedingung immer wahr und das Programm bleibt endlos in der Schleife "hängen".

In unserer Lernumgebung kannst du ein hängendes Programm mit dem Stoppknopf oder mit Schliessen des Turtlefensters abbrechen. Im allgemeinen sind endlose Schleifen ohne Abbruch-Möglichkeit aber gefährlich, da im Extremfall ein Neustart des Computers nötig ist.

BEDINGUNGEN MIT OR VERKNÜPFEN

Die Turtle soll mit einer **while-Schleife** die nebenstehende Figur zeichnen. Wie du siehst, zeichnet sie abwechslungsweise rote und grüne Dreiecke.



Für den Farbwechsel kannst du folgenden Trick verwenden: Du testest die Schleifenvariable darauf, ob sie 0, 2 oder 4 ist und wählst die Stiftfarbe rot.

Mit dem Befehl **fillToPoint(0. 0)** kannst du eine Figur während des Zeichnens füllen. Dabei wird am Punkt (0, 0) sozusagen ein Gummiband befestigt, dessen anderes Ende die Turtle mitzieht. Dabei werden alle Punkte, über die sich das Gummiband bewegt, fortlaufend eingefärbt.

```
from gturtle import *

def triangle():
repeat 3:
forward(100)
right(120)
makeTurtle()
i = 0
while i < 6:
if i == 0 or i == 2 or i == 4:
setPenColor("red")
else:
setPenColor("green")

fillToPoint(0, 0)
triangle()
right(60)
i = i + 1
```

MEMO

Bei Verwendung von mehreren Programmstrukturen musst du auf die korrekte Einrückung der einzelnen Schleifenblöcke achten.

Wie du siehst, kannst du zwei oder mehr Bedingungen mit *or* verknüpfen. Eine so verknüpfte Bedingung ist dann wahr, wenn die eine oder auch die andere Bedingung erfüllt ist (also auch, wenn beide Bedingungen erfüllt sind).

Mit dem Befehl *fillToPoint(x, y)* kannst du Figuren mit der Stiftfarbe während des Zeichnens füllen, im Gegensatz zum Befehl *fill()*, mit dem du bereits gezeichnete geschlossene Figuren füllen kannst.

BEDINGUNGEN MIT AND VERKNÜPFEN

Die Turtle soll mit einer while-Schleife 10 zusammengebaute Häuser zeichnen. Die Häuser sind von 1 bis 10 nummeriert. Die Häuser mit den Nummern 4 bis 7 sind gross, die übrigen klein.

In der *while*-Schleife wird die Hausnummer *nr* benutzt, um die Grösse der Häuser zu bestimmen. Wenn *nr* grösser als 3 und kleiner als 8 ist, sind die Häuser gross.

Zum Färben verwenden wir den Befehl **fillToHorizontal(0)**. Dadurch wird die Fläche zwischen der gezeichneten Figur und der waagrechten Linie y = 0 fortlaufend gefüllt.

```
from gturtle import *

makeTurtle()
setPos(-200, 30)
right(30)
fillToHorizontal(0)
setPenColor("sienna")

nr = 1
while nr <= 10:
if nr > 3 and nr < 8:
forward(60)
right(120)
forward(60)
left(120)
else:
forward(30)
right(120)
forward(30)
left(120)
nr += 1
```

MEMO

Zwei Bedingungen kannst du mit **and** verknüpfen. Eine solche Verknüpfung ist nur dann wahr, wenn beide Bedingungen erfüllt sind.

Mit dem Befehl **fillToHorizontal(y)** kannst du Figuren mit der Stiftfarbe während des Zeichnens füllen. Gefüllt wird die Fläche zwischen der gezeichneten Figur und der horizontalen Linie durch y.

nr += 1 kannst du lesen als: *nr* wird erhöht um 1. Es ist eine abgekürzte Scheibweise für die Zuweisung *nr = nr + 1*.

Mit einer **while True** Schleife kann ein Programmblock so lange wiederholt, bis das Programm durch Schliessen des Turtle-Fensters abgebrochen wird.

Die Schleife wird in Zweierschritten durchlaufen. Statt i = i + 2 verwendest du die abgekürzte Schreibweise **i += 2** (i wird erhöht um 2).

Mit dem **print**-Befehl kannst du etwas in die TigerJython-Konsole im unteren Bereich des Editors schreiben. Für Text verwendest du Anführungszeichen und trennst Zahlen mit einem Komma ab. Es wird automatisch ein Leerzeichen zwischen dem Text und der Zahl eingefügt. Verstehst du, warum i = 122 ausgegeben wird?

EINGABE-VALIDIERUNG

Gibst du dem Benutzer die Möglichkeit mit einem Eingabedialog einen Wert in einem bestimmten Bereich einzugeben, so kannst du dich nicht darauf verlassen, dass er sich an deine Vorgaben hält. Ein "robustes" Programm überprüft die Eingabe und fängt eine fehlerhafte Eingabe mit einer Rückmeldung ab. Diese Eingabeprüfung kannst du am besten mit einer while-Schleife durchführen, die solange durchlaufen wird, bis sich der Eingabewert an die Vorgaben hält. In deinem Programm wählt der Benutzer mit den Zahlen 1, 2, oder 3 die Farben rot, grün oder gelb des gezeichneten Kreises aus.

```
from gturtle import *

makeTurtle()

n = 0
while n < 1 or n > 3:
```

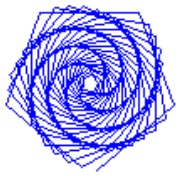
```
n = inputInt("Enter 1, 2 or 3")
if n == 1:
setPenColor("red")
elif n == 2:
setPenColor("green")
else:
setPenColor("yellow")
dot(200)
```

AUFGABEN

1.

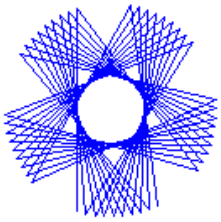
Die Turtle bewegt sich um eine Strecke 5 vorwärts, dreht sich um 70° nach rechts und vergrössert die Streckenlänge um 0.5. Diese Schritte wiederholt sie, solange die Streckenlänge kleiner als 150 ist.

Versuche es auch mit dem Drehwinkel 89°!



2.

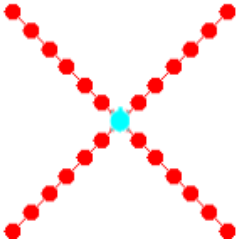
Wie du sicher gemerkt hast, beträgt die Drehung an der Spitze bei einem 5-er Stern 144°. Verändere diesen Drehwinkel ganz wenig, z. B. 143° und vergrössere die Anzahl der Wiederholungen. Dann erhältst du eine neue Figur.



3.

Die Turtle zeichnet ein Diagonalmuster mit gefüllten roten Kreisen. Alle Kreise liegen im Turtlefenster, d.h. dass ihre Distanz vom Mittelpunkt kleiner als 400 ist.

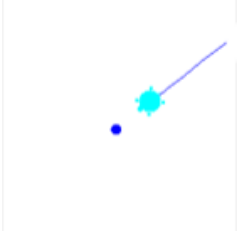
Verwende den Befehl *dot(25)*, um die Kreise zu zeichnen.



4.

Die Turtle befindet sich an der Position (250, 200). Mit Schritten der Länge 10 bewegt sie sich auf einer Geraden zur Homeposition bis der Abstand kleiner als 1 ist.

Verwende die Befehle *towards()* und *heading(degrees)* aus der **Dokumentation**.



5*.

Du möchtest die Turtle präziser bei Home positionieren und

wählst als Abstandskriterium einen 10 bis 100 Mal kleineren Wert. Es kann sein, dass die Turtle jetzt nicht mehr anhält. Begründe das Verhalten.