

Funktionen in Tiger Jython

Alexandra Maximova

3. November 2020

1 Aus kleinen Bausteinen grössere bauen

In Tiger Jython haben wir schon viele Befehle gesehen. Mit `forward(100)`, zum Beispiel, bewegen wir die Schildkröte um 100 Schritte nach vorne, mit `dot(25)` zeichnen wir einen Punkt vom Diameter 25. Bei anderen Wörtern, wie zum Beispiel `square(100)` oder `circle(20)`, kriegen wir die Fehlermeldung, dass sie undefiniert sind. Heisst das, dass wir sie selber definieren können?

Ja! In Tiger Jython können wir Codestücke benennen und im Programm immer wieder verwenden. Das nennen wir eine **Funktion**. Im folgendem Codeabschnitt definieren wir eine Funktion `quadrat100()`, die ein Quadrat mit Seitenlänge 100 zeichnet, und **rufen** diese zwei Mal **auf**.

```
1 from turtle import *
2
3 def quadrat100():
4     repeat 4:
5         forward(100)
6         right(90)
7
8 makeTurtle()
9
10 setPenColor("teal")
11 quadrat100()
12 left(180)
13 setPenColor("deep pink")
14 quadrat100()
15
16
```

So definieren wir einen neuen Befehl: **def** ist selber ein Befehl **quadrat100()** ist der neue Befehl, der wir gerade definieren.

Körper des neuen Befehls

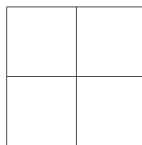
Erst hier tragen wir den Befehl **makeTurtle** ein!

Hier verwenden wir den neuen Befehl **quadrat100()**.

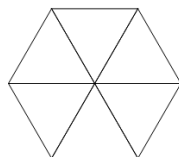
Abbildung 1: Definition und Aufruf der Funktion `quadrat100()`.

Achtung: Eine Funktion wird erst dann ausgeführt, wenn sie aufgerufen wird.

Aufgabe 1.1. Definiere eine Funktion `square()`, welche ein Quadrat zeichnet, und verwende sie, um folgendes Bild zu zeichnen.



Aufgabe 1.2. Definiere eine Funktion `triangle()`, welche ein gleichseitiges Dreieck zeichnet, und verwende sie, um folgendes Bild zu zeichnen.



Aufgabe 1.3. Fred und Gregory haben zwei Programme geschrieben, die das gleiche Bild zeichnen.

Kannst du erraten, was gezeichnet wird, ohne das Programm auszuführen? Bei welchem Programm findest du es einfacher?

```

1 from turtle import *
2
3 makeTurtle()
4 hideTurtle()
5
6
7 setPenColor("dark green")
8 repeat 4:
9     forward(100)
10    right(90)
11 right(90)
12 forward(35)
13 left(90)
14
15 setPenColor("brown")
16 repeat 2:
17     forward(50)
18     right(90)
19     forward(30)
20     right(90)
21
22 penUp()
23 forward(60)
24 right(90)
25 forward(5)
26 left(180)
27 penDown()
28
29 setPenColor("light blue")
30 repeat 4:
31     forward(30)
32     right(90)
33
34 right(180)
35 penUp()
36 forward(20)
37 left(90)
38 penDown()
39 repeat 4:
40     forward(30)
41     right(90)
42
43 penUp()
44 forward(40)
45 left(90)
46 forward(60)
47 right(120)
48 penDown()
49
50 setPenColor("red")
51 repeat 3:
52     forward(100)
53     right(120)

```

Listing 1: 'Fred's Programm'

```

1 from turtle import *
2
3 makeTurtle()
4 hideTurtle()
5
6 def square():
7     repeat 4:
8         forward(100)
9         right(90)
10
11 def triangle():
12     repeat 3:
13         forward(100)
14         right(120)
15
16 def window():
17     repeat 4:
18         forward(30)
19         right(90)
20
21 def door():
22     repeat 2:
23         forward(50)
24         right(90)
25         forward(30)
26         right(90)
27
28 def drawDoor():
29     penUp()
30     right(90)
31     forward(35)
32     left(90)
33     penDown()
34
35     door()
36
37     penUp()
38     left(90)
39     forward(35)
40     right(90)
41     penDown()
42
43 def drawWindows():
44     repeat 2:
45         penUp()
46         right(90)
47         forward(10)
48         left(90)
49         penDown()
50
51         window()
52
53         penUp()
54         right(90)
55         forward(40)
56         left(90)
57         penDown()
58
59         penUp()
60         left(90)
61         forward(100)
62         right(90)
63         penDown()
64
65 def drawRoof():
66     right(30)
67     triangle()
68     left(30)
69
70 setPenColor("dark green")
71 square()
72
73 setPenColor("brown")
74 drawDoor()
75
76 setPenColor("light blue")

```

```

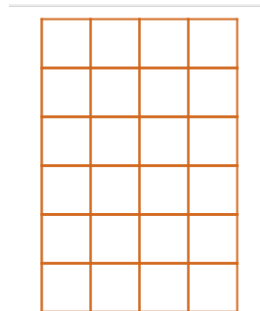
77 penUp()
78 forward(60)
79 penDown()
80 drawWindows()
81
82 setPenColor("red")
83 penUp()
84 forward(40)
85 penDown()
86 drawRoof()

```

Listing 1: 'Gregorys Programm'

Ändere eins der Programme so, dass die Fenster keine einfachen Quadrate sind, sondern wie in Aufgabe 1.1 aussehen.

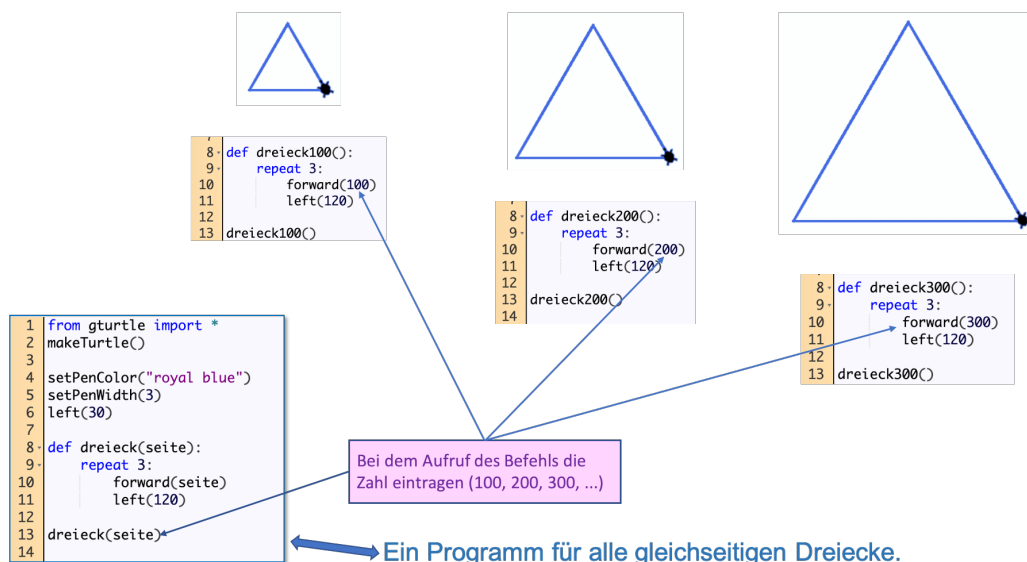
Aufgabe 1.4. Zeichne diese Schokoladentafel. Überlege, welche Funktionen du definieren möchtest.



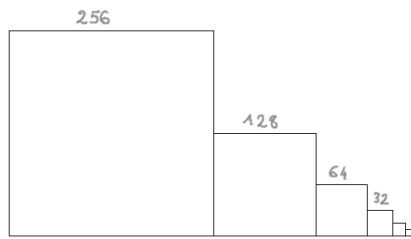
2 Befehle parametrisieren

Bei den Befehlen `forward(.)` und `dot(.)` können wir angeben, um wie viel Schritte die Schildkröte nach vorne gehen soll oder wie gross der Punkt sein soll. Unsere selbstdefinierte Funktionen können das auch.

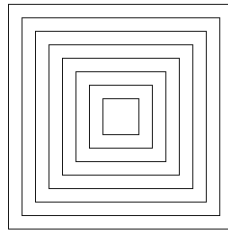
Zum Beispiel, müssen wir nicht für jede Dreiecksgrösse N eine eigene Funktion `dreieckN()` definieren. Wir können die Funktion so definieren, dass sie einen Parameter `seite` erwartet, welches die Länge einer Seite angibt.



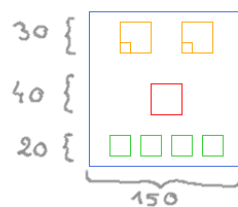
Aufgabe 2.1. Definiere die Funktion `square(side)`. Verwende sie, um folgendes Bild zu zeichnen.



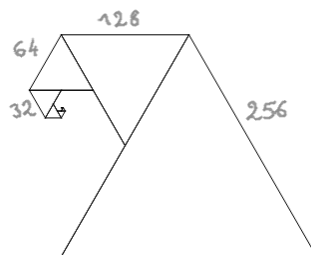
Aufgabe 2.2. Zeichne folgendes Bild.



Aufgabe 2.3. Zeichne folgendes Robotergesicht aus Quadrate.



Aufgabe 2.4. Zeichne folgendes Bild. Die Seiten der Dreiecke halbieren sich jedes Mal. Überlege, welche Funktionen möchtest du definieren.



Aufgabe 2.5. Zeichne folgendes Bild. Wie viele Funktionen musst du definieren? Reicht es, nur eine zu definieren?



Beispiellösungen

Lösungen zu Abschnitt 1

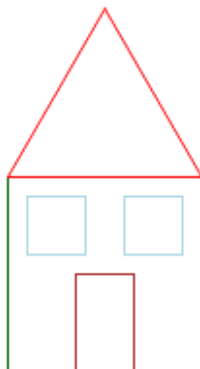
Aufgabe 1.1 Dieses Programm definiert die Funktion `square()` und zeichnet ein Fenster.

```
1 from gturtle import *
2
3 makeTurtle()
4 hideTurtle()
5
6 def square():
7     repeat 4:
8         forward(100)
9         right(90)
10
11 repeat 4:
12     square()
13     right(90)
```

Aufgabe 1.2 Dieses Programm definiert die Funktion `triangle()` und zeichnet ein Sechseck.

```
1 from gturtle import *
2
3 makeTurtle()
4 hideTurtle()
5
6 def triangle():
7     repeat 3:
8         forward(100)
9         right(120)
10
11 right(30)
12 repeat 6:
13     triangle()
14     right(60)
```

Aufgabe 1.3 Beide Programme zeichnen ein Haus.



Gregorys Programm zu verstehen und zu verändern sollte einfacher sein, weil sein Programm strukturierter ist und Funktionen verwendet. Insbesondere, es reicht, die Funktion `window()` zu ändern, um andere Fenster zu zeichnen.

```
21 def window():
22     repeat 4:
23         square15()
24         penUp()
25         forward(30)
26         penDown()
27         right(90)
```

Aufgabe 1.4 Dieses Programm definiert die Funktion `square()`, die Funktion `row()` und zeichnet eine Tafel Schokolade.

```
1 from turtle import *
2
3 makeTurtle()
4 hideTurtle()
5
6 def square():
7     repeat 4:
8         forward(50)
9         right(90)
10
11 def row():
12     repeat 4:
13         square()
14         right(90)
15         forward(50)
16         left(90)
17     left(90)
18     forward(4*50)
19     right(90)
20     forward(50)
21
22 setPenWidth(2)
23 setPenColor("chocolate")
24
25 repeat 6:
26     row()
```

Lösungen zu Abschnitt 2

Aufgabe 2.1 Dieses Programm definiert die Funktion `square(side)` und zeichnet immer kleiner werdende Quadrate.

```
1 from turtle import *
2
3 makeTurtle()
4 hideTurtle()
5
6 def square(side):
7     repeat 4:
8         forward(side)
9         right(90)
10
11 size = 256
12 repeat 8:
13     square(size)
14     right(90)
15     forward(size)
16     left(90)
17     size /= 2
```

Aufgabe 2.2 Dieses Programm definiert die Funktion `square(side)` und zeichnet verschachtelte immer kleiner werdende Quadrate.

```
1 from turtle import *
2
3 makeTurtle()
4 hideTurtle()
5
6 def square(side):
7     repeat 4:
8         forward(side)
9         right(90)
10
11 size = 250
12 decrement = 30
13 repeat 8:
14     square(size)
15     size -= decrement
16
17     penUp()
18     forward(decrement/2)
```

```

19     right(90)
20     forward(decrement/2)
21     left(90)
22     penDown()

```

Aufgabe 2.3 Dieses Programm definiert die Funktionen `square(side)`, die ein Quadrat zeichnet, `eye()`, `nose()` und `mouth()`, die jeweils Auge, Nase und Mund definieren und `drawEyes()`, `drawNose()` und `drawMouth()`, welche jeweils Augen, Nase und Mund zentriert im Gesicht zeichnen und wieder ans linke Gesichtsrand zurückkehren. Diese Funktionen ermöglichen es uns, das eigentliche Zeichnen einfach und strukturiert zu halten.

```

1  from turtle import *
2
3  makeTurtle()
4  hideTurtle()
5
6  def square(side):
7      repeat 4:
8          forward(side)
9          right(90)
10
11 def eye():
12     square(30)
13     square(10)
14
15 def mouth():
16     repeat 4:
17         square(20)
18         penUp()
19         right(90)
20         forward(30)
21         left(90)
22         penDown()
23     penUp()
24     left(90)
25     forward(4*30)
26     right(90)
27     penDown()
28
29 def nose():
30     square(30)
31
32 def drawEyes():
33     repeat 2:
34         penUp()
35         right(90)
36         forward(30)
37         left(90)
38         penDown()
39
40         eye()
41
42         penUp()
43         right(90)
44         forward(30)
45         left(90)
46         penDown()
47     penUp()
48     left(90)
49     forward(4*30)
50     right(90)
51     penDown()
52
53 def drawMouth():
54     penUp()
55     right(90)
56     forward(20)
57     left(90)
58     penDown()
59
60     mouth()
61
62     penUp()
63     left(90)
64     forward(20)
65     right(90)

```

```

66     penDown()
67
68 def drawNose():
69     penUp()
70     right(90)
71     forward(60)
72     left(90)
73     penDown()
74
75     nose()
76
77     penUp()
78     left(90)
79     forward(60)
80     right(90)
81     penDown()
82
83 setPenColor("royal blue")
84 square(150)
85
86 setPenColor("lime green")
87 penUp()
88 forward(10)
89 penDown()
90 drawMouth()
91
92 setPenColor("red")
93 penUp()
94 forward(40)
95 penDown()
96 drawNose()
97
98 setPenColor("orange")
99 penUp()
100 forward(60)
101 penDown()
102 drawEyes()

```

Aufgabe 2.4 Dieses Programm definiert die Funktion `triangle(side)` und zeichnet immer kleiner werdende Dreiecke.

```

1 from turtle import *
2
3 makeTurtle()
4 hideTurtle()
5
6 def triangle(side):
7     repeat 3:
8         forward(side)
9         right(120)
10
11 right(30)
12 size = 256
13 repeat 8:
14     triangle(size)
15     size /= 2
16     forward(size)
17     left(60)

```

Aufgabe 2.5 Dieses Programm definiert die Funktion `polygon(n)`, welches ein beliebiges n-Eck zeichnen kann, und verwendet sie, um Drei- bis Siebenecken zu zeichnen.

```

1 from turtle import *
2
3 makeTurtle()
4 hideTurtle()
5
6 def polygon(n):
7     repeat n:
8         forward(100)
9         right(360/n)
10
11 left(90)
12 setPenWidth(3)

```



```
13 setPenColor("purple")
14
15 nofSides = 3
16 repeat 5:
17     polygon(nofSides)
18     nofSides += 1
```