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 Wörter, 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**.

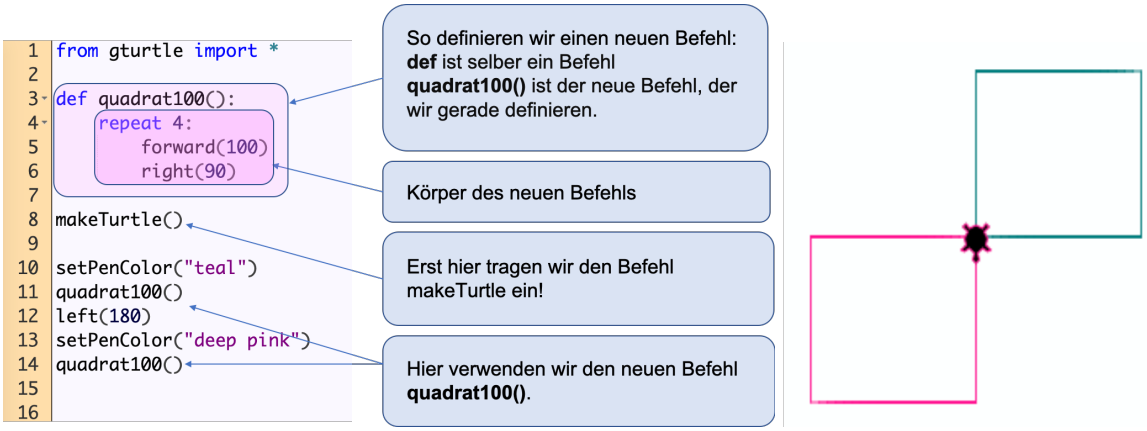
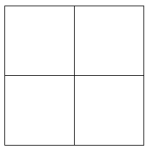


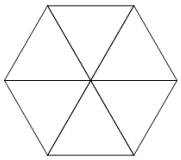
Abbildung 1: Definition und Aufruf der Funktion `quadrat100()`.

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?

```
from gturtle import *

makeTurtle()
hideTurtle()


setPenColor("dark green")
repeat 4:
    forward(100)
    right(90)
right(90)
forward(35)
left(90)

setPenColor("brown")
repeat 2:
    forward(50)
    right(90)
    forward(30)
    right(90)

penUp()
forward(60)
right(90)
forward(5)
left(180)
penDown()

setPenColor("light blue")
repeat 4:
    forward(30)
    right(90)

right(180)
penUp()
forward(20)
left(90)
penDown()
repeat 4:
    forward(30)
    right(90)

penUp()
forward(40)
left(90)
forward(60)
right(120)
penDown()

setPenColor("red")
repeat 3:
    forward(100)
    right(120)
```

Listing 1: 'Freds Programm'

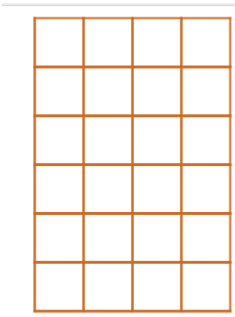
```
from gturtle import *

makeTurtle()
hideTurtle()

def square():
    repeat 4:
        forward(100)
        right(90)

def triangle():
    repeat 3:
        forward(100)
        right(120)

def window():
    repeat 4:
        forward(30)
        right(90)

def door():
    repeat 2:
        forward(50)
        right(90)
        forward(30)
        right(90)

def drawDoor():
    penUp()
    right(90)
    forward(35)
    left(90)
    penDown()

    door()

    penUp()
    left(90)
    forward(35)
    right(90)
    penDown()

def drawWindows():
    repeat 2:
        penUp()
        right(90)
        forward(10)
        left(90)
        penDown()

        window()

        penUp()
        right(90)
        forward(40)
        left(90)
        penDown()
    penUp()
    left(90)
    forward(100)
    right(90)
    penDown()

def drawRoof():
    right(30)
    triangle()
    left(30)


setPenColor("dark green")
square()

setPenColor("brown")
drawDoor()

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.

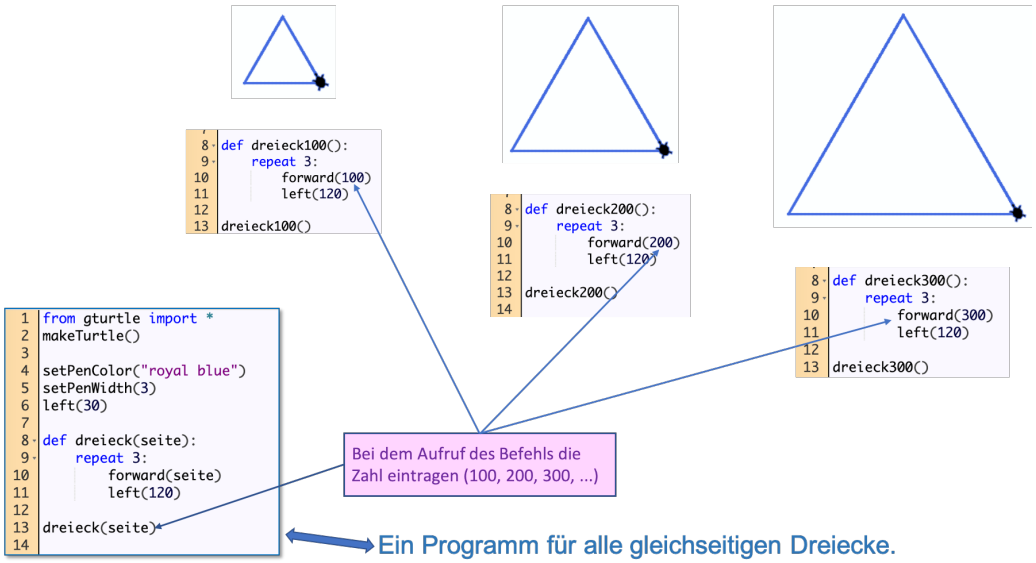
Aufgabe 1.4. *Zeichne diese Schokoladentafel. Überlege, welche Funktionen du definieren möchtest.*



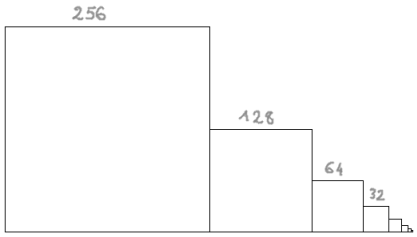
2 Befehle parametrisieren

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.

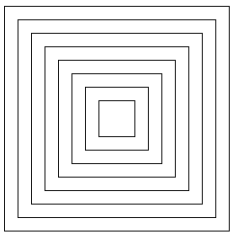
Zum Beispiel, müssen wir nicht für jede Dreiecksgrösse N eine eigene Funktion `dreieckN()` definieren. Wir können die Funktion so definieren, dass sie einen Parameter `seite` erwartet, welches die Länge einer Seite angibt.



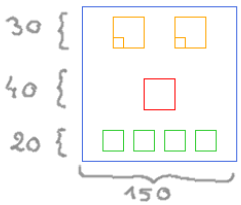
Aufgabe 2.1. *Definiere die Funktion `square(side)`. Verwende sie, um folgendes Bild zu zeichnen.*



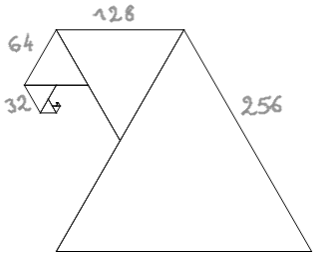
Aufgabe 2.2. *Zeichne folgendes Bild.*



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

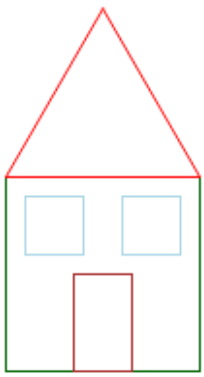
Aufgabe 1.1 Dieses Programm definiert die Funktion `square()` und zeichnet ein Fenster.

```
from gturtle import *

makeTurtle()
hideTurtle()

def square():
    repeat 4:
        forward(100)
        right(90)

repeat 4:
    square()
    right(90)
```

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

```
from gturtle import *

makeTurtle()
hideTurtle()

def triangle():
    repeat 3:
        forward(100)
        right(120)

right(30)
repeat 6:
    triangle()
    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.

```
def window():
    repeat 4:
        square15()
        penUp()
        forward(30)
        penDown()
        right(90)
```

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

```
from gturtle import *

makeTurtle()
hideTurtle()

def square():
    repeat 4:
        forward(50)
        right(90)

def row():
    repeat 4:
        square()
        right(90)
        forward(50)
        left(90)
    left(90)
    forward(4*50)
    right(90)
    forward(50)

setPenWidth(2)
setPenColor("chocolate")

repeat 6:
    row()
```

Lösungen zu Abschnitt 2

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

```
from gturtle import *

makeTurtle()
hideTurtle()

def square(side):
    repeat 4:
        forward(side)
        right(90)

size = 256
repeat 8:
    square(size)
    right(90)
    forward(size)
    left(90)
    size /= 2
```

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

```
from gturtle import *

makeTurtle()
hideTurtle()

def square(side):
    repeat 4:
        forward(side)
        right(90)

size = 250
decrement = 30
repeat 8:
    square(size)
    size -= decrement

    penUp()
    forward(decrement/2)
```

```
    right(90)
    forward(decrement/2)
    left(90)
    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 zeichen und wieder ans linke Gesichtsrand zurückkehren. Diese Funktionen ermöglichen es uns, das eigentliche Zeichnen einfach und strukturiert zu halten.

```
from gturtle import *

makeTurtle()
hideTurtle()

def square(side):
    repeat 4:
        forward(side)
        right(90)

def eye():
    square(30)
    square(10)

def mouth():
    repeat 4:
        square(20)
        penUp()
        right(90)
        forward(30)
        left(90)
        penDown()
    penUp()
    left(90)
    forward(4*30)
    right(90)
    penDown()

def nose():
    square(30)

def drawEyes():
    repeat 2:
        penUp()
        right(90)
        forward(30)
        left(90)
        penDown()

        eye()

        penUp()
        right(90)
        forward(30)
        left(90)
        penDown()
    penUp()
    left(90)
    forward(4*30)
    right(90)
    penDown()

def drawMouth():
    penUp()
    right(90)
    forward(20)
    left(90)
    penDown()

    mouth()

    penUp()
    left(90)
    forward(20)
    right(90)
```

```
    penDown()

def drawNose():
    penUp()
    right(90)
    forward(60)
    left(90)
    penDown()

    nose()

    penUp()
    left(90)
    forward(60)
    right(90)
    penDown()

setPenColor("royal blue")
square(150)

setPenColor("lime green")
penUp()
forward(10)
penDown()
drawMouth()

setPenColor("red")
penUp()
forward(40)
penDown()
drawNose()

setPenColor("orange")
penUp()
forward(60)
penDown()
drawEyes()
```

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

```
from gturtle import *

makeTurtle()
hideTurtle()

def triangle(side):
    repeat 3:
        forward(side)
        right(120)

right(30)
size = 256
repeat 8:
    triangle(size)
    size /= 2
    forward(size)
    left(60)
```

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

```
from gturtle import *

makeTurtle()
hideTurtle()

def polygon(n):
    repeat n:
        forward(100)
        right(360/n)

left(90)
setPenWidth(3)
```

```
setPenColor("purple")

nofSides = 3
repeat 5:
    polygon(nofSides)
    nofSides += 1
```