Funktionen in Tiger Jython - Konzeption

Alexandra Maximova

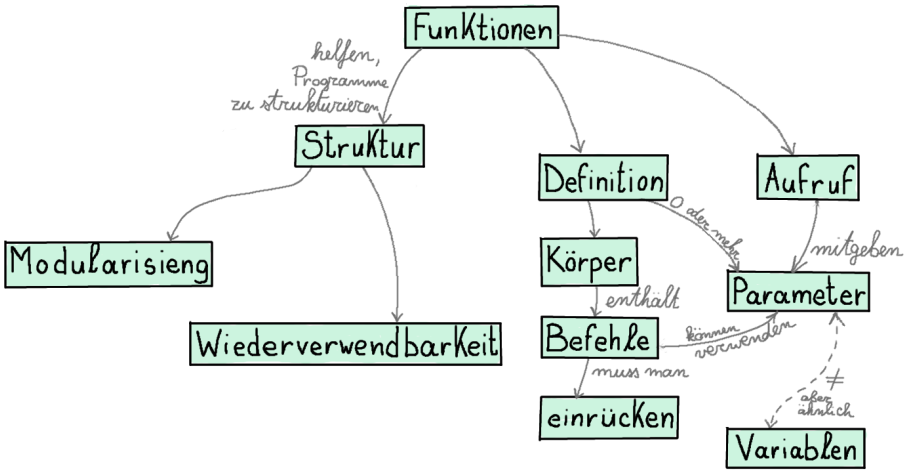
3. November 2020

1 Rahmenbedingungen und Vorwissen

Diese Lektion wird im Rahmen vom Einführungspraktikum an der Kantonsschule Menzingen zwei Mal in unterschiedlichen Halbklassen gehalten. Die SuS absolvieren das erste Jahr am Kurzzeitgymnasium und besuchen das OFI.

In den vorherigen Wochen haben die SuS angefangen, in Tyger Jython zu programmieren, und kennen schon viele Turtle-Befehle, Variablen und Schleifen. In dieser Lektion geht es darum, Funktionen einzuführen.

2 Stoffanalyse



Funktionen dienen dazu, den Code zu **strukturieren**. Sie fördern die **Modularisierung**, d.h. die Aufteilung in logischen **wiederverwendbaren** Bausteinen.

Funktionen muss man **definieren** und dann kann man sie beliebig oft **aufrufen**. In der **Definition** gibt man die Signatur und den **Körper** an. Zu der Signatur gehören der Name der Funktion und die **Parameter**. Im Körper stehen die **Befehle**, die ausgeführt werden, wenn die Funktion aufgerufen wird. Alle Befehle, die zu einer Funktion gehören, müssen in Tiger Jython **eingerückt** sein.

Beim **Funktionsaufruf** muss man den Namen der Funktion kennen und alle Parameter instanzieren.

Da der Begriff von **Parameter** und die Unterschiede zu den **Variablen** nicht selbstverständlich sind, empfiehlt es sich, zuerst **Funktionen ohne Parameter** einzuführen, und erst in der zweiten Stunde mit **Funktionen mit Parameter** fortzufahren.

3 Lernziele

Leitidee Digitale Medien und Technologien sind zum Alltag geworden. Damit die Erwachsenen von morgen diese neue Welt nicht nur konsumieren, sondern auch mitgestalten können, müssen die SuS von heute die Werkzeuge und die grundlegenden Konzepte kennenlernen. Programmieren und vor allem Modularisierung gehören dazu.

Das modulare Vorgehen, welches mit der Einführung von Funktionen trainiert wird, kann auch in anderen Situationen angewendet werden. Die SuS sollten lernen bei Projekten und Aufgaben Teile zu erkennen, die sie als Bausteine wiederverwenden können, um etwas Komplexeres zusammenzusetzen.

Dispositionsziele

- Die SuS bevorzugen strukturierten Code.
- Die SuS zerlegen Aufgaben in kleinere Teile, die als Module verwendet werden können.

Operationalisierte Lernziele

- Die SuS definieren Funktionen ohne Parameter, die etwas Bestimmtes machen.
- Die SuS rufen vor- oder selbstdefinierte Funktionen ohne Parameter auf.
- Beim Zeichnen mit der Turtle, die SuS entscheiden selbstständig, welche Befehle sie in eine Funktion auslagern wollen, und rufen diese an den richtigen Stellen auf.
- Die SuS definieren Funktionen mit Parameter, die etwas Bestimmtes machen.
- Die SuS rufen vor- oder selbstdefinierte Funktionen mit Parameter auf, wobei sie für Parameter Zahlen einsetzen.

- Die SuS rufen vor- oder selbstdefinierte Funktionen mit Parameter auf, wobei sie für Parameter Variablen einsetzen.

4 Informativer Unterrichtseinstieg

Wenn wir `forward(25)` schreiben, geht die Schildkröte um 25 Schritte geradeaus. Wenn wir `dot(25)` schreiben, zeichnet die Schildkröte einen Kreis mit Diameter 25. Wenn wir aber `square()` schreiben, kommt eine Fehlermeldung.

Warum versteht die Schildkröte manche Befehle, und andere nicht? Können wir es ihr beibringen? Offensichtlich kann sie ein Quadrat zeichenen:

```
repeat 4:
  forward(100)
  right(90)
```

Sie muss einfach verstehen, dass sie diese Befehle ausführen soll, wenn wir `square()` schreiben.

Heute werden wir lernen, wie man in Tyger Jython dem Computer neue Befehle beibringt. Das nennen wir **Funktionen**. In der ersten Stunde befassen wir uns mit einfachen Befehlen. Ihr werdet Zeit haben, selbstständig 4 Aufgaben zu lösen. Nach der Pause werden wir sehen, wie wir mehrere leicht unterschiedliche Befehle auf einmal definieren können.

Nach der Stunde wird jeder von euch der Schildkröte neue Befehle beibringen können und den Code besser strukturieren.

Warum wollen wir überhaupt neue Wörter der Schildkröte beibringen? Wir können doch alles mit den Grundbefehlen penUp, penDown, forward, right zeichnen. Wenn die Formen aber komplexer werden, sehnen wir uns nach Struktur. Wenn wir ein Haus zeichnen, zeichnen wir das Haus, das Dach, die Tür, die Fenster usw. Wir können aus einfachen Formen kompliziertere bauen, und diese wiederum wiederverwenden, um noch etwas grösseres und komplexeres zu bauen. (Das gilt, übrigens, nicht nur für Zeichnungen mit der Schildkröte, sondern auch für Berechnungen und Computerspiele).

Angenommen, wir wollen der Schildkröte beibringen, was ein Quadrat ist. Dafür definieren wir in Tiger Jython eine Funktion `square()`. Wir wissen, was sie machen muss.

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

Das Keyword `def` macht die Schildkröte aufmerksam, dass sie die Definition eines neuen Befehls sieht. Nach dem Keyword kommt der **Name** der Funktion. Es ist frei wählbar, wie der Name einer Variable, und es empfiehlt sich, einen Namen zu wählen, welcher beschreibt, was die Funktion machen soll. In diesem Fall haben wir die Funktion `square` genannt, weil sie ein Quadrat zeichnen soll.

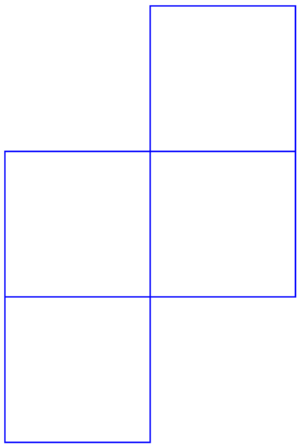
Diese Funktion, die wir gerade definiert haben, können wir **aufrufen**, d.h. verwenden, um andere Figuren zu zeichnen. Zum Beispiel, können wir aus Quadrate Tetris-Figuren zeichnen:

```
from gturtle import *

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


makeTurtle()

square()
forward(100)
square()
right(90)
forward(100)
left(90)
square()
forward(100)
square()
```



5 Literatur

Für die Erstellung von Skript und Aufgaben wurden das Skript von Tobias Kohn (zu finden auf https://tobiaskohn.ch/index.php/teaching/teaching-python/) und der Kurs 2 der Begabtenförderung (verfügbar auf Moodle) verwendet.