Lehrer-PDF - Learn Python with Turtle Graphics

In dieser Lehrer-PDF zum Szenario und Webanwendung "Learn Python with Turtle Graphics" wird die Installation und Inbetriebnahme der Webanwendung beschrieben. Außerdem wird eine Anleitung zum Generieren neuer Levels gegeben. Des Weiteren enthält diese das Passwort zum Anzeigen von Musterlösungen und wie dieses verändert werden kann. Außerdem werden Musterlösungen zu allen bestehenden Leveln gegeben wie auch typische Fehlermeldungen und deren Behebung erläutert.

Inhaltsverzeichnis

1 Installation und Inbetriebnahme **2**

2 Erstellung von neuen Levels **2**

2.1 Level Struktur 2

2.2 Tutorial erstellen 3

2.3 Code-Hinweise für Tutorial, initialen Code und Musterlösung erstellen 3

3 Passwort für Musterlösung **4**

4 Musterlösungen **5**

4.1 Level 0 5

4.2 Level 1 5

4.3 Level 2 6

4.4 Level 3 6

4.5 Level 4 7

4.6 Level 5 7

4.7 Level 6 8

4.8 Level 7 9

4.9 Level 8 9

5 Typische Fehlermeldungen und Quick-Help **10**

1 Installation und Inbetriebnahme

Um eine reibungslose Installation und Inbetriebnahme der Webanwendung zu gewährleisten, wird das Bash-Script `brython_turtle_init.sh` zur Verfügung gestellt. In Listing 1 ist die Ausgabe der Hilfsfunktion mit Anweisungen zum Einsatz des Scripts dargestellt.

Listing 1: Hilfsfunktion vom Script brython_turlte_init.sh

```
Usage: ./brython_turtle_init.sh [options]
Initializes the Learn-Python-with-Turtle-Graphics repository
   ,
starts the Python HTTP-server and opens webapp.

Options:
  -dir, --directory DIRECTORY Specify the target directory.
     Default is 'Learn-Python-with-Turtle-Graphics'.
  -h, --help Display this help message.
```

Dieses cloned das GitHub-Repository in einen angegebenes directory oder ein default-directory, startet den Python HTTP-Server und öffnet die Webanwendung im Standard-Browser. Für den Fall das bereits gecloned wurde und dieses lokal nicht auf dem neusten Stand ist wird gepulled.

2 Erstellung von neuen Levels

Das Hinzufügen neuer Level sowie das Bearbeiten bestehender Level ist unkompliziert. Im folgenden wird die Struktur eines Levels wie auch Skripte zum Erstellen dieser beschrieben.

2.1 Level Struktur

Zum Hinzufügen eines neuen Levels muss dieses lediglich als JSON-Datei im Ordner ”levels” abgelegt werden. Zum Bearbeiten eines Levels wird die jeweilige JSON-Datei bearbeitet. Die Bezeichnung eines Levels sollte ”level_n” lauten, wobei ”n” für die Nummer des jeweiligen Levels steht. Die Struktur einer solchen JSON-Datei ist in Listing 2 dargestellt.

Listing 2: level_n JSON-Datei

```
{
   "tutorial": "",
   "init_code": "t = init_turtle(turtle) # Diese Zeile bitte
       nicht entfernen oder ändern\n\n",
   "solution_code": "t = init_turtle(turtle) # Diese Zeile
      bitte nicht entfernen oder ändern\n\n"
}
```

Ein Level besteht aus einem Tutorial, einem initialen Code und einer Musterlösung. Diese sind als Strings angeben. Der String `tutorial` enthält HTML-Code. `init_code` und `solution_code` enthalten Python-Code. Diese müssen zunächst in das Format eines einzeiligen Strings gebracht werden. Das Konvertieren sowie das Maskieren von Sonderzeichen geschieht in zwei Bash-Scripts, die im Folgenden erläutert werden.

2.2 Tutorial erstellen

Der String `tutorial` ist ein beliebiger HTML-Code, der mithilfe des Bash-Scripts `html_2_string.sh` generiert wird. In Listing 3 ist die Ausgabe der Hilfsfunktion mit Anweisungen zum Einsatz des Scripts dargestellt.

Listing 3: Hilfsfunktion vom Script html_2_string.sh

```
Usage: ./html_2_string.sh [options] HTML_FILE
Converts an HTML file to a single-line string.

Options:
  -out, --output_file FILENAME Specify the output file.
  -help, --help Display this help message.

Example:
  ./html_2_string.sh input.html
  ./html_2_string.sh input.html -out output.txt
```

Die übergebene HTML-Datei kann beliebige HTML-Elemente enthalten. Dennoch existieren für ein konsistentes Level Design einige empfohlene Styleguidelines. Diese sind in Tabelle 1 dargestellt und nutzen vordefinierte CSS-Klassen. Für zusätzliches Styling können direkt in HTML-Elementen weitere Stilregeln ohne Klassendefinitionen angegeben werden.

Aufgabenstellungen und Text-Hinweise werden als String ohne Anführungsstriche, anstelle der Platzhalter, angegeben. Nach dem Konvertieren der HTML-Datei in den generierten String, werden Code-Hinweise anstelle des Platzhalters eingesetzt. Zunächst müssen auch die Code-Hinweise in einen String gewandelt werden. Dies erfordert ein weiteres Script, das im folgenden Abschnitt erläutert wird.

2.3 Code-Hinweise für Tutorial, initialen Code und Musterlösung erstellen

Das Bash-Script `python_2_string.sh` wird verwendet, um Python Dateien in einen String zu konvertieren. Dieses wird auf Code-Hinweise sowie auf `init_code` und `solution_code` für die JSON-Datei angewandt. In Listing 4 ist die Ausgabe der Hilfsfunktion mit Anweisungen zum Einsatz des Scripts dargestellt.

Tabelle 1: Tutorial Styleguidelines

Typ	HTML Code
Aufgabenstellung	`<p class="p-font"><b>AUFGABENSTELLUNG</b></p>`
Text-Hinweis	`<p class="p-font">TEXT-HINWEIS</p>`
Code-Hinweis	`<pre class="code-hint"><code><span class="code-hint-font">CODE-HINWEIS</span></code></pre>`

Listing 4: Hilfsfunktion vom Script python_2_string.sh

```
Usage: ./python_2_string.sh [options] PYTHON_FILE
Converts a Python file to a single-line string.

Options:
  -out, --output_file FILENAME Specify the output file.
  -h, --help Display this help message.

Example:
  ./python_2_string.sh input.py
  ./python_2_string.sh input.py -out output.txt
```

Die als Input übergebene Python Datei kann hierbei beliebigen Code enthalten. Beim Importieren von Python Modulen mittels `import <module>` sollte überprüft werden, ob diese möglicherweise von Brython nicht unterstützt werden. Die erste Zeile der Strings `init_code` und `solution_code` sollte folgendermaßen lauten:

```
t = init_turtle(turtle) # Diese Zeile bitte nicht entfernen oder
 ändern
```

Diese Zeile initialisiert Turtle Graphics. Ohne diese ist Turtle Code nicht ausführbar und wirft folgenden Error:

```
line 7, in Code-Editor NameError: name 't'is not defined. Did
 you mean '[object Object]'?
```

3 Passwort für Musterlösung

Lösungen können in zwei Schritten angezeigt werden. Schüler haben immer Zugriff auf die Kommentare (ohne Code) einer Musterlösung. Für die Musterlösung mit Code brauchen Schüler ein Passwort. Diese Passwort lautet aktuell "Passwort". Dieses lässt sich konfigurieren indem der String `password` in Zeile 6 der Datei `/modules/solution.py` angepasst wird.

4 Musterlösungen

Im diesem Abschnitt werden zu jedem Level Musterlösungen mit Code und Kommentaren dargestellt.

4.1 Level 0

Listing 5: Level 0

```
t = init_turtle(turtle) # Diese Zeile bitte nicht entfernen oder ändern

t.color("red")  # Zeichenfarbe wird auf Rot gesetzt
t.forward(100)  # Bewegt die Turtle 100 Einheiten nach vorne
t.right(90) # Dreht die Turtle um 90 Grad nach Rechts
t.color("blue") # Zeichenfarbe wird auf Blau gesetzt
t.backward(100) # Bewegt die Turtle 100 Einheiten nach hinten
t.left(45)  # Dreht die Turtle um 45 Grad nach Links
```

4.2 Level 1

Listing 6: Level 1

```
t = init_turtle(turtle) # Diese Zeile bitte nicht entfernen oder ändern

length = 100    # Kantenlänge
angle = 90      # Winkel
color = 'red'   # Farbe

t.color(color)    # ändert Farbe des Stiftes und der Schildkröte auf die Farbe in der Variable color

t.begin_fill()

t.forward(length) # 100 Schritte nmachvorne
t.right(angle)    # Schildkröte um 90 Grad nach rechts drehen

# das gleiche wie oben noch 3 mal
t.forward(length)
t.right(angle)

t.forward(length)
t.right(angle)

```

```
t.forward(length)
t.right(angle)

t.end_fill()
```

4.3 Level 2

Listing 7: Level 2

```
t = init_turtle(turtle) # Diese Zeile bitte nicht entfernen oder ändern

length = 100    # Kantenlänge
angle = 90    # Winkel
text_style=('Arial', 20, 'normal')  # Variable mit Text-Style

for i in range(4):  # darunter eingerückter Code wird viermal wiederholt
  t.color('red')    # ändert Farbe des Stiftes und der Schildkröte auf rot
  t.write(i, font=text_style) # schreibt die Zahl "i" in die Ecke mit dem Style aus einer Variable
  t.color('blue')   # ändert Farbe des Stiftes und der Schildkröte auf blau

  t.forward(length) # 100 Schritte nachvorne
  t.right(angle)    # Schildkröte um 90 Grad nach rechts drehen

t.hideturtle()    # versteckt die Schildkröte
```

4.4 Level 3

Listing 8: Level 3

```
t = init_turtle(turtle) # Diese Zeile bitte nicht entfernen oder ändern

length = 100    # Kantenlänge
corner_num = 6    # Anzahl der Ecken
angle = 360 / corner_num  # Winkel berechnet mit Anzahl der Ecken

for i in range(corner_num): # for-Schleifenkopf, darunter eingerückter Code wird so oft wiederholt wie in der Variable mit Anzahl der Ecken angegeben
  if i % 2 == 0:    # Wenn Zahl "i" durch 2 teilbar ist ...
```

```
9       t.color('red')     # ist die Schildkröte rot
10    else:         # Sonst ...
11      t.color('blue')    # ist die Schildkröte blau
12
13    t.forward(length) # Kantenlänge nachvorne
14    t.right(angle)     # Schildkröte um den Wert Winkel
       Variable nach rechts drehen
```

4.5 Level 4

Listing 9: Level 4

```
1 t = init_turtle(turtle) # Diese Zeile bitte nicht entfernen
     oder ändern
2
3 color_array=['red','green','blue','turquoise','yellow','#3
     c79b8']   # Farb-Array mit verschiedenen Farben
4 for i in range(6):         # for-Schleife mit einer 6-fachen
     Wiederholung erstellen. Beginnned mit 0
5   if(i%2==1):          # Wenn die Zahl ungerade ist, dann
     wird die Turtle versteckt
6     t.hideturtle()
7   else:            # Wenn die Zahl gerade ist, dann wird die
     Turtle wieder gezeigt
8     t.showturtle()
9   t.speed(i+1)         # Turtle-Speed abhängig des Index erhö
     hen
10  t.color(color_array[i])    # Setzt die Farbe abhängig des
     Index durch Auswahl im Array
11  t.forward(50)        # Bewegt die Turtle 50 Einheiten nach
     vorne
12  t.right(60)          # Dreht die Turtle um 60 Grad nach
     Rechts
```

4.6 Level 5

Listing 10: Level 5

```
1 t = init_turtle(turtle) # Diese Zeile bitte nicht entfernen
     oder ändern
2
3 import random
4
5 def draw_form(length=100, angle=90, lines=4): # Funktion
     definiert mit 3 vorgelegten Variablen.
6   color_array=['red','green','blue','turquoise','yellow','#3
     c79b8']   # Farb-Array mit verschiedenen Farben
7   for x in range(lines):
```

```
    t.speed(x+1)  # Turtle-Speed
    if(random.randint(1,2)==1): # Turtle wird zufällig Sichtbar oder nicht mit einer 50/50-Chance
      t.hideturtle()  # Turtle wird versteckt
    else:
      t.showturtle()  # Turtle wird Sichtbar
    color = color_array[random.randint(0,5)]
    t.color(color)  # Setzt die Farbe der Turtle zufällig aus dem Array

    t.forward(length) # Bewegt die Turtle um length Einheiten nach vorne
    t.right(angle)  # Dreht die Turtle um angle Grad nach Rechts
  return color

print(draw_form(angle=60, lines=6))
```

4.7 Level 6

Listing 11: Level 6

```
t = init_turtle(turtle) # Diese Zeile bitte nicht entfernen oder ändern

def form(corner_num=30, length=10, color='blue'): # Funktionskopf, darunter eingerückter Code wird beim Aufruf der Funktion ausgeführt
  angle = 360 / corner_num  # Winkel berechnet mit Anzahl der Ecken
  t.color(color)        # färbt Schildkröte mit übergebener Farbe
  for _ in range(corner_num): # for-Schleifenkopf, darunter eingerückter Code wird so oft wiederholt wie in der Variable mit Anzahl der Ecken angegeben
    t.forward(length)     # Kantenlänge nachvorne
    t.left(angle)       # Schildkröte um den Wert Winkel Variable nach links drehen

for x in range(3):  # for-Schleifenkopf, darunter eingerückter Code wird 3 mal wiederholt
  for y in range(3):  # for-Schleifenkopf, darunter eingerückter Code wird 3 mal wiederholt
    t.goto(100 * x, 100 * y)  # zu Koordinaten gehen
    form()            # Aufruf der Funktion zum Form zeichnen
```

4.8 Level 7

Listing 12: Level 7

```
t = init_turtle(turtle) # Diese Zeile bitte nicht entfernen oder ändern

import random

length = 3
last = 0

def right():
    t.color("red")
    for _ in range(10):
        t.forward(length)
        t.right(9)

def left():
    t.color("yellow")
    for _ in range(10):
        t.forward(length)
        t.left(9)

def forward():
    t.color("green")
    t.forward(20)

def teleport(x=0,y=0):
  t.penup()
  t.goto(x,y)
  t.pendown()

for _ in range(40):
    x = random.randint(0,6)
    if x == 0:
        right()
    elif x == 1 or x == 2:
        left()
    elif x == 3 or x == 4 or x == 5:
        forward()
    elif x == 6:
        teleport(0,0)
```

4.9 Level 8

Keine Lösung, da es sich um eine Sand-Box handelt.

5 Typische Fehlermeldungen und Quick-Help

Im Folgenden sind typische Fehlermeldung wie auch Hilfe beim Auftritt dieser dargestellt.

1. **Error:**
   ```
   line 7, in Code-Editor NameError: name 't'is not defined.
    Did you mean '[object Object]'?
   ```
   Quick-Help:
   Zeile 1 wurde entfernt oder verändert. Zeile 1 sollte folgendermaßen sein:
   ```
   t = init_turtle(turtle) # Diese Zeile bitte nicht entfernen
    oder ändern
   ```

2. **Error:**
   ```
   line 16 print(1)IndentationError: unexpected indent
   ```
   Quick-Help:
   Entweder ist die Einrückung der Zeile tatsächlich falsch oder es wurden Tabulator Einrückungen mit Einrückungen mit 4 Leerzeichen kombiniert. Dies wird zurzeit von der Webanwendung nicht unterstützt. In dem initialer Code aller Level wird Tabulator für Einrückungen verwendet.

Listings

1 Hilfsfunktion vom Script brython_turlte_init.sh 2
2 level_n JSON-Datei 2
3 Hilfsfunktion vom Script html_2_string.sh 3
4 Hilfsfunktion vom Script python_2_string.sh 4
5 Level 0 5
6 Level 1 5
7 Level 2 6
8 Level 3 6
9 Level 4 7
10 Level 5 7
11 Level 6 8
12 Level 7 9