Befehlsübersicht Turtle-Modul

In dieser Übersicht sind die Funktionen aus dem Turtle-Modul beschrieben, die im Buch und in unseren Programmen verwendet werden. Es gibt aber noch viele weitere Befehle im Turtle-Modul. Die vollständige Referenz gibt es in PYTHON IDLE unter dem Menü HELP | PYTHON DOCS (allerdings nur auf Englisch). Eine ausführliche deutsche Referenz der Befehle im Turtle-Modul findest du unter folgendem Link im Internet: http://python4kids.net/downloads/py4k_cda4/turtlemodul-referenz/turtlemodul_referenz.pdf

Das Turtle-Grafik-Modul *turtle.py* stellt alle notwendigen Funktionen zur Verfügung, mit denen du die Schildkröte steuerst, um vielfältigste Bilder zu gestallten.

Du importierst alle Turtle-Funktionen mit:

```
from turtle import *
```

Gestalt der Schildkröte

`shape(Name)`
setzt das Aussehen der Schildkröte auf die Gestalt mit dem angegebenen Namen.

> *Name*: bezeichnet eine mögliche Gestalt der Schildkröte. Folgende Gestalten sind möglich: “turtle“ (Schildkröte), “arrow“ (schmale Pfeilspitze), “circle” (Kreis), “square” (Quadrat), “triangle” (Dreieck), “classic” (ausgebeulte Pfeilspitze)

`shape()`
gibt den Namen der aktuellen Gestalt für die Schildkröte aus.

`shapesize(Streckfaktor)`
streckt die Länge und Breite der Schildkröte um den Faktor *Streckfaktor*. Die Breite der Konturlinie bleibt unverändert.

> *Streckfaktor*: eine positive ganze Zahl

`shapesize(StreckBreit, StreckLang)`
streckt die Breite um den Faktor *StreckBreit* und die Länge um den Faktor *StreckLang*. Die Breite der Konturlinie bleibt unverändert.

> *StreckBreit, StreckLang*: zwei positive ganze Zahlen

`shapesize(StreckBreit, StreckLang, Kontur)`
streckt die Breite der Schildkröte um den Faktor *StreckBreit* und die Länge um den Faktor *StreckLang*. Setzt die Breite der Konturlinie auf *Kontur*.

> *StreckBreit, StreckLang, Breite*: drei positive ganze Zahlen

`shapesize()`
gibt die aktuelle Größenwerte für die Gestalt der Schildkröte aus.

Bewegen und zeichnen

`forward(Schritte) | fd(Schritte)`
bewegt die Schildkröte um die *Schritte* (in Pixel) vorwärts in die Richtung, in die sie zeigt.

Schritte: eine Zahl

`backward(Schritte) | bk(Schritte) | back(Schritte)`
bewegt die Schildkröte um die *Schritte* (in Pixel) rückwärts, ohne sie zu drehen.

Schritte: eine Zahl

`left(Winkel) | lt(Winkel)`
dreht die Schildkröte um den *Winkel* nach links (gegen den Uhrzeigersinn).

Winkel: eine Zahl (in Grad)

`right(Winkel) | rt(Winkel)`
dreht die Schildkröte um den *Winkel* nach rechts (in die Uhrzeigerrichtung).

Winkel: eine Zahl (in Grad)

`setx(x)`
setzt die x-Koordinate der Schildkröte auf die Zahl *x*. Die Schildkröte wird zu der x-Koordinate bewegt. Die y-Koordinate bleibt unverändert. Wenn der Pinsel unten ist, bleibt eine Spur sichtbar.

x: eine Zahl

`xcor()`
gibt die aktuelle x-Koordinate der Schildkröte als Zahl zurück.

`sety(y)`
setzt die y-Koordinate der Schildkröte auf die Zahl *y*. Die Schildkröte wird zu der y-Koordinate bewegt. Die x-Koordinate bleibt unverändert. Wenn der Pinsel unten ist, bleibt eine Spur sichtbar.

y: eine Zahl

`ycor()`
gibt die aktuelle y-Koordinate der Schildkröte als Zahl zurück.

`setpos(x, y) | setposition(x, y) | goto(x, y)`
setzt die Position auf den Punkt (*x*,*y*). Die Orientierung der Schildkröte bleibt unverändert. Wenn der Pinsel unten ist, bleibt eine Spur sichtbar.

x, *y*: zwei Zahlen

`position() | pos()`
gibt die aktuelle Position der Schildkröte zurück in der Form: (x, y).

`setheading(Grad) | seth(Grad)`
richtet die Schildkröte auf den *Grad* aus. Die Standardrichtungen in Grad sind:
0 – Osten, 90 – Norden, 180 – Westen, 270 – Süden (gezählt im Uhrzeigersinn)

Grad: eine Zahl (in Grad)

`heading()`
gibt die aktuelle Ausrichtung (in Grad) der Schildkröte als Gradzahl zurück.

`home()`
bewegt die Schildkröte zu dem Ausgangspunkt (0,0) und richtet sie nach rechts auf 0 Grad aus.

`speed(Geschw)`
setzt die Geschwindigkeit der Schildkröte auf *Geschw*. Die Zahlen 1 bis 10 bedeuten von langsam zu schnell aufsteigende Geschwindigkeiten. Die Zahl 0 ist die schnellste Geschwindigkeit.

> *Geschw*: ganze Zahl von 0 bis 10

`speed()`
gibt die aktuelle Geschwindigkeit der Schildkröte aus.

`circle(Radius)`
zeichnet einen Kreis mit dem angegebenen *Radius* (Länge der Strecke vom Kreismittelpunkt bis zum Kreisrand). Die Kreismitte liegt eine Radiuslänge oberhalb der Schildkröte, wenn Radius > 0 ist, andernfalls unterhalb der Schildkröte.

> *Radius*: eine Zahl

`circle(Radius, Winkel)`
zeichnet den Teil eines Kreises mit dem angegebenen *Radius*. Der *Winkel* in Grad gibt an, welcher Teil des Kreises gezeichnet wird. So bewirkt die Zahl 90, dass die Schildkröte nur einen Viertelkreis zeichnet.

> *Radius, Winkel*: zwei Zahlen

`circle(Radius, Winkel, Ecken)`
zeichnet den Teil eines Kreises mit dem angegebenen *Radius*. Der *Winkel* in Grad gibt an, welcher Teil des Kreises gezeichnet wird. Die *Ecken* geben an, mit wie vielen Ecken der Kreis (als Vieleck) berechnet wird.

> *Radius, Winkel, Ecken*: drei Zahlen

`dot()`
zeichnet einen dicken Punkt.

`dot(Dicke)`
zeichnet einen Punkt mit der angegebenen *Dicke*.

> *Dicke*: eine ganze Zahl größer oder gleich 1

`dot(Dicke, Farbe)`
zeichnet einen Punkt mit der angegebenen *Dicke* und *Farbe.*

> *Dicke*: eine ganze Zahl größer oder gleich 1
> *Farbe*: ein Farbstring (z.B. „red“) oder eine RGB-Farbe (z.B. color(1,1,0))

`stamp()`
erzeugt einen »Stempelabdruck« der Schildkröte an der aktuellen Position. Gibt eine eindeutige Abdrucknummer zurück.

`clearstamp(Nummer)`
löscht den »Stempelabdruck« der Schildkröte mit der angegebenen Abdrucknummer.

> *Nummer*: Zahl (Rückgabewert eines `stamp()`-Aufrufs)

Sichtbarkeit

`hideturtle() | ht()`
macht die Schildkröte unsichtbar.

`showturtle() | st()`
macht die Schildkröte sichtbar.

`isvisible()`
gibt *True* zurück, wenn die Schildkröte sichtbar ist, andernfalls *False*.

Steuerung des Pinsels

`penup() | pu() | up()`
hebt den Pinsel hoch.

`pendown() | pd() | down()`
senkt den Pinsel nach unten auf das »Papier«.

`isdown()`
gibt *True* zurück, wenn der Pinsel unten ist, andernfalls *False*.

`pensize(Breite) | width(Breite)`
setzt die Pinselbreite auf die *Breite*.

> *Breite*: eine Zahl

Farben und färben

`pencolor(Farbe)`
setzt die Pinselfarbe auf *Farbe*.

> *Farbe*: ein Farbstring (z.B. „red“) oder eine RGB-Farbe (z.B. color(1,1,0))

`pencolor()`
gibt die aktuelle Pinselfarbe als Farbstring oder RGB-Farbe zurück.

`fillcolor(Farbe)`
setzt die Füllfarbe auf *Farbe*.

> *Farbe*: ein Farbstring (z.B. „red“) oder eine RGB-Farbe (z.B. color(1,1,0))

`fillcolor()`
gibt die aktuelle Füllfarbe als Farbstring oder RGB-Farbe zurück.

`color(Farbe)`
setzt die beiden Farben Pinselfarbe und Füllfarbe auf *Farbe*.

> *Farbe*: ein Farbstring (z.B. „red“) oder eine RGB-Farbe (z.B. color(1,1,0))

`color()`
gibt die aktuellen Farben Pinselfarbe und Füllfarbe zurück.

`begin_fill()`
beginnt das Zeichnen einer Figur, die gefüllt werden soll.

`end_fill()`
füllt die Figur, die seit dem letzten `begin_fill()` gemalt wurde, mit der Füllfarbe.

`filling()`
gibt *True* zurück, wenn das Füllen mit `begin_fill()` begonnen, aber noch nicht abgeschlossen wurde; gibt sonst *False* zurück

Eigenschaften des Grafik-Fensters

`window_height()`
gibt die Höhe des Grafik-Fensters zurück.

`window_width()`
gibt die Breite des Grafik-Fensters zurück.

`bgcolor()`
gibt die Hintergrundfarbe des Grafik-Fensters zurück.

`bgcolor(Farbe)`
setzt die Hintergrundfarbe des Grafik-Fensters auf *Farbe*.

> *Farbe*: ein Farbstring (z.B. „red“) oder eine RGB-Farbe (z.B. color(1,1,0))

`clear()`
löscht alle Zeichnungen aus dem Grafik-Fenster. Die Schildkröte bleibt unverändert.

`reset()`
löscht alle Zeichnungen aus dem Grafik-Fenster. Die Schildkröte wird an ihre home-Position versetzt.

`exitonclick()`
schließt das Grafik-Fenster beim Klicken auf die Fensterfläche.