erlebe IT
by bitkom

Glossar
Python Turtle

Grundlagen

Diese Befehle brauchst du für jedes neue Programm.

F5	Drücken der Funktionstaste F5 führt das Skript aus.
# Kommentar	Alles, was hinter einer Raute eingegeben wird, ist nur als Kommentar sichtbar und wird nicht in die Befehlskette aufgenommen.
from turtle import*	Importiere die Turtle-Bibliothek
shape(»*turtle*«)	Bildet als Form eine Schildkröte turtle ab.

Aussehen

Mit diesen Befehlen kannst du das Aussehen der Turtle und des Zeichenstifts verändern.

pencolor(»*farbe*«)
farbe = ein Wort (eine Farbe)

Setzt die Farbe des Stifts auf die eingegebene Farbe.
z. B. pencolor(»green«) = setzt die Stiftfarbe auf grün

Es gibt eine Liste verschiedener Farben, die jeweils in vier Abstufungen existieren: green oder green1, green2, green3, green4. Hier eine Auswahl:

Snow	tan	chocolate	cyan	salmon
black	aquamarine	green	red	blue
azure	chartreuse	khaki	pink	LightPink
yellow	gold	maroon	magenta	sienna
white	wheat	burlywood	Thistle	orchid
firebrick	orange	cornsilk	ivory	coral
OliveDrab	SkyBlue	purple	plum	SlateGray

pencolor(*r,g,b*)
r, g, b = für drei Zahlen zur numerischen Beschreibung des Rot-, Grün- und Blau-anteils von RGB-Farben

Setzt die Farbe des Stifts auf die eingegebenen RGB-Farbwerte. r, g und b müssen im Bereich 0 – colormode liegen.
z. B. Die Farbe Lachsrosa etwa wird nach colormode(1) durch das Tripel (1.0, 0.5, 0.25) angegeben, nach colormode(255) durch das Tripel (128, 64, 32).

erlebe IT
by bitkom

fillcolor(»*farbe*«) *farbe* = ein Wort (eine Farbe)	Stellt die Füllfarbe der Turtle ein.
begin_fill()	Schaltet den Füll-Modus der Turtle ein. Wird aufgerufen, wenn das Zeichnen einer Figur beginnt, die gefüllt werden soll.
end_fill()	Schaltet den Füll-Modus der Turtle aus. Wird aufgerufen, wenn die Figur fertig gezeichnet ist, um sie zu füllen.
pensize(*width*) *width* = positive Zahl	Setzt die Strichdicke auf width. z. B. pensize(3) = setzt die Strichbreite auf 3 Pixel
penup() | up	Hebt den Zeichenstift an, zeichnet nicht bei nachfolgenden Bewegungen.
pendown() | down	Setzt den Zeichenstift nach unten, zeichnet bei nachfolgenden Bewegungen.
reset()	Löscht alle Zeichnungen im Grafik-Fenster und setzt alle Werte (Farbe der Turtle, Strichdicke etc.) auf die Ausgangswerte zurück.
clear()	Löscht alle Zeichnungen im Grafik-Fenster, alle Werte (Farbe der Turtle, Strichdicke etc.) bleiben erhalten.
undo()	Macht die letzte Anweisung rückgängig. Kann mehrfach angewendet werden.
showturtle()	Macht die Turtle sichtbar.
hideturtle()	Macht die Turtle unsichtbar, ohne ihre sonstigen Eigenschaften zu verändern.

Bewegung	**Mit diesen Befehlen bestimmst du, wie sich die Turtle bewegt.**
forward(*distance*) | fd(*distance*) *distance* = eine Zahl	Gehe um die Strecke *distance* (in Pixel) vorwärts, und zwar in die Richtung, in die Turtle zeigt.
backward(*distance*) | bk(*distance*) *distance* = eine Zahl	Gehe um die Strecke *distance* (in Pixel) rückwärts.
right(*angle*) | rt(*angle*) *angle* = Winkel, eine Zahl	Drehe nach rechts um *angle* Winkeleinheiten in Grad. z. B. right(90) = Drehe die Turtle um 90 Grad nach rechts.
left(*angle*) | lt(*angle*) *angle* = Winkel, eine Zahl	Drehe nach links um angle Winkeleinheiten in Grad. z. B. left(180) =Drehe die Turtle um 180 Grad nach links.
goto(*x,y*) | setpos(*x,y*) | setposition (*x,y*)	Bewegt die Turtle zum Punkt mit den absoluten Koordinaten (*x,y*). Wenn der Zeichenstift unten ist, wird die Strecke gezeichnet. Die Orientierung der Turtle wird nicht geändert. z. B. goto(-110,90) =bewegt die Turtle zum Punkt x=-110, y=90

erlebe IT
by bitkom

home()	Bewegt die Turtle in die Ausgangslage und die Ausgangsorientierung zurück.
speed(*speed*) *speed* = ganze Zahl von 0–10 0 = keine Animation, Turtle ›springt‹ 1...10 = langsam ... schnell	Bestimmt die Geschwindigkeit der Turtle.
circle(*radius, extent*) *radius* = eine Zahl *extent* (optional) = eine Zahl (Winkel)	Zeichne einen Kreis(bogen) mit gegebenem *radius*. Der Kreismittelpunkt ist *radius* Einheiten links von der Turtle, wenn *radius* > 0 ist, sonst rechts von der Turtle. Der Winkel *extent* gibt an, welcher Teil des Kreises gezeichnet wird. Ohne Angabe von *extent* wird ein voller Kreis gezeichnet. z. B. circle(100) = zeichnet einen Kreis mit Durchmesser 100 Pixel
setheading(to_angle) *to_angle* = eine Zahl (ein Winkel)	Dreht die Turtle so, dass sie in Richtung des angegebenen Winkels to_angle orientiert ist. Hier sind einige gebräuchliche Richtungen in Grad:

›standard‹-mode:	**›logo‹-mode:**
0 – ost	0 – nord
90 – nord	90 – ost
180 – west	180 – süd
270 – süd	270 – west

Steuerung und Funktion

Mit diesen Befehlen bestimmst du, wie sich die Turtle bewegt.

def *funktion*():
funktion = Name deiner Funktion

Definiert eine **Funktion**, die aus allen Befehlen besteht, die darunter **eingerückt** sind.
Ein Beispiel für ein Quadrat:

```
def quadrat():
  forward(100)
  right(90)
  forward(100)
  right(90)
  forward(100)
  right(90)
  forward(100)
```

Die Funktion. wird nicht automatisch ausgeführt. Du kannst die **Funktion aufrufen**, indem du quadrat() in das Shell-Fenster eingibst.

erlebe IT
by bitkom

for i in range(*number*): *number* = eine Zahl, Anzahl der Wiederholungen	Definiert eine **for Schleife**, die den Code, der darunter eingerückt ist, eine bestimmte Anzahl *number* von Malen wiederholt. Ein Beispiel: for i in range(30): quadrat() = Die Turtle zeichnet 30 Mal ein Quadrat, wie wir es vorher definiert haben.
name = *wert* *name* = ein beliebiges Wort oder ein Buchstabe *wert* = eine beliebige Zahl	Definiert einen Namen/**Parameter**: Das heißt, man teilt einem Wort oder Buchstaben einen beliebigen Wert zu. Der Name kann beim Zeichnen verwendet werden. Der Wert des Parameters kann im Laufe des Programms geändert werden. seitenlänge = 100 forward(seitenlänge) > Die Turtle läuft 100 Pixel nach vorne.
onclick(*fun*) *fun* = eine Funktion mit 2 Argumenten (muss vorher definiert werden)	Führt die Funktion fun aus, wenn auf die Turtle geklickt wird. z. B. onclick(getroffen) = führt die Funktion ›getroffen‹ aus, wenn auf die Turtle geklickt wird. Die Koordinaten der Turtle werden automatisch als Argumente übergeben.

Turtle-Grafik-Fenster

Mit diesen Befehlen kannst du das Aussehen des Grafik-Fensters verändern.

bgcolor(»*farbe*«)	Füllt den Hintergrund mit der angegebenen *farbe*.
bgpic(»*name*«)	Nimmt das Bild mit dem Dateinamen *name* als Hintergrund. Die Datei muss im selben Ordner abgespeichert sein wie das Skript. z. B. bgpic(»maze.gif«) = setzt die Datei maze.gif als Hintergrund ein
screensize(*canwidth, canvheight*) *canwidth, canheight* = zwei positive Zahlen	Stellt die Größe der mittels Scrollbars erreichbaren Zeichenfläche ein, auf der die Turtle zeichnet. Die Größe des Fensters wird nicht verändert.
setup(*width, height*) *width, height* = zwei positive Zahlen	Stellt die Größe des Fensters ein (in Pixel).
mode(»*mode*«) *mode* = die Zeichenkette »standard« oder »logo«	Dreht die Turtle so, dass sie in Richtung des angegebenen Winkels to_angle orientiert ist. Hier sind einige gebräuchliche Richtungen in Grad:

Modus	**Anfängliche Orientierung der Turtle**	**positive Winkel**
›standard‹	nach rechts (Osten)	im Gegenuhrzeigersinn
›logo‹	nach oben (Norden)	im Uhrzeigersinn

erlebe IT
by bitkom

colormode(*max_Farbwert*) *max_Farbwert* = eine Zahl	Legt den maximalen Farbwert im RGB-Farbraum fest (entweder 1 oder 255).

Extra Befehle	**Mit diesen Befehlen kannst du das Aussehen des Grafik-Fensters verändern.**
write()	Füllt den Hintergrund mit der angegebenen *farbe*.
bgpic(»*name*«)	Nimmt das Bild mit dem Dateinamen *name* als Hintergrund. Die Datei muss im selben Ordner abgespeichert sein wie das Skript. z. B. bgpic(»maze.gif«) = setzt die Datei maze.gif als Hintergrund ein.
screensize(*canwidth, canvheight*) *canwidth, canheight* = zwei positive Zahlen	Stellt die Größe der mittels Scrollbars erreichbaren Zeichenfläche ein, auf der die Turtle zeichnet. Die Größe des Fensters wird nicht verändert.

Häufige Fehler

Fehler im Code-Fenster

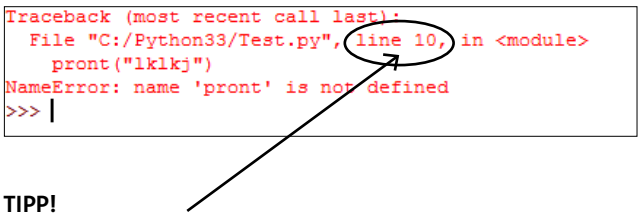
Manchmal erscheinen Fehler im Code-Fenster, so dass das Programm nicht ausgeführt werden kann.

invalid syntax	Im Code ist irgendwo ein Tippfehler.
unexpected indent	Es sind falsche Leerzeichen im Code, so dass das Programm nicht ausgeführt werden kann.

Fehler im Shell-Fenster

Manchmal erscheinen rot geschriebene Fehler im Shell-Fenster, so dass das Programm in dieser Form nicht funktioniert.

NameError z. B. name ›pront‹ is not defined	Python versteht ein Wort nicht.



TIPP!

Klicke mit der **rechten Maustaste** auf die Fehlermeldung im Shell-Fenster und dann auf »Go to file/line« (Gehe zu Datei/Zeile). Die Maus springt in die richtige Zeile – jetzt kannst du den Fehler beheben.

erlebe IT
by bitkom

Keine Chance für Fehler!
– Checkliste –

- [ ] Hast du alles genauso geschrieben wie in der Vorlage?
- [ ] Stimmt die Rechtschreibung?
- [ ] Hast du Wörter in einfache oder doppelte Anführungszeichen gesetzt?
- [ ] Hast du am Anfang der Zeile aus Versehen Leerzeichen eingegeben?
- [ ] Hast du auch die Zeile über und unter der markierten Zeile geprüft? Manchmal steht der Fehler dort!
- [ ] Suche dir Hilfe: Lass jemand anderes den Code überprüfen. Vielleicht hast du etwas übersehen.