

Grundlagen	Diese Befehle brauchst du für jedes neue Programm.
<code>F5</code>	Drücken der Funktionstaste F5 führt das Skript aus.
<code># Kommentar</code>	Alles, was hinter einer Raute eingegeben wird, ist nur als Kommentar sichtbar und wird nicht in die Befehlskette aufgenommen.
<code>from turtle import*</code>	Importiere die Turtle-Bibliothek.
<code>Shape („turtle“)</code>	Bildet als Form eine Schildkröte <i>turtle</i> ab.

Aussehen	Mit diesen Befehlen kannst du das Aussehen der Turtle und des Zeichenstifts verändern.																																			
<p><code>pencolor(„farbe“)</code></p> <p><i>farbe</i> = ein Wort (eine Farbe)</p>	<p>Setzt die Farbe des Stifts auf die eingegebene Farbe.</p> <p>z. B. <code>pencolor(„green“)</code> = setzt die Stiftfarbe auf grün</p> <p>Es gibt eine Liste verschiedener Farben, die jeweils in vier Abstufungen existieren: green oder green1, green2, green3, green4. Hier eine Auswahl:</p> <table border="1"> <tbody> <tr> <td>Snow</td> <td>tan</td> <td>chocolate</td> <td>cyan</td> <td>salmon</td> </tr> <tr> <td>black</td> <td>aquamarine</td> <td>green</td> <td>red</td> <td>blue</td> </tr> <tr> <td>azure</td> <td>chartreuse</td> <td>khaki</td> <td>pink</td> <td>LightPink</td> </tr> <tr> <td>yellow</td> <td>gold</td> <td>maroon</td> <td>magenta</td> <td>sienna</td> </tr> <tr> <td>white</td> <td>wheat</td> <td>burlywood</td> <td>Thistle</td> <td>orchid</td> </tr> <tr> <td>firebrick</td> <td>orange</td> <td>cornsilk</td> <td>ivory</td> <td>coral</td> </tr> <tr> <td>OliveDrab</td> <td>SkyBlue</td> <td>purple</td> <td>plum</td> <td>SlateGray</td> </tr> </tbody> </table>	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
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																																
<p><code>pencolor(r,g,b)</code></p> <p><i>r, g, b</i> = für drei Zahlen zur numerischen Beschreibung des Rot-, Grün- und Blauanteils von RGB-Farben</p>	<p>Setzt die Farbe des Stifts auf die eingegebenen RGB-Farbwerte. <i>r, g</i> und <i>b</i> müssen im Bereich 0 - colormode liegen.</p> <p>z. B. Die Farbe Lachsrosa etwa wird nach <code>colormode(1)</code> durch das Tripel (1.0, 0.5, 0.25) angegeben, nach <code>colormode(255)</code> durch das Tripel (128, 64, 32).</p>																																			
<p><code>fillcolor(„farbe“)</code></p> <p><i>farbe</i> = ein Wort (eine Farbe)</p>	<p>Stellt die Füllfarbe der Turtle ein.</p>																																			

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

Bewegung	Mit diesen Befehlen bestimmst du, wie sich die Turtle bewegt.
<code>forward(distance)</code> <code>fd(distance)</code> <i>distance</i> = eine Zahl	Gehe um die Strecke <i>distance</i> (in Pixel) vorwärts, und zwar in die Richtung, in die die Turtle zeigt.
<code>backward(distance)</code> <code>bk(distance)</code> <i>distance</i> = eine Zahl	Gehe um die Strecke <i>distance</i> (in Pixel) rückwärts.
<code>right(angle)</code> <code>rt(angle)</code> <i>angle</i> = Winkel, eine Zahl	Drehe nach rechts um <i>angle</i> Winkleinheiten in Grad. z. B. <code>right(90)</code> =Drehe die Turtle um 90 Grad nach rechts.
<code>left(angle)</code> <code>lt(angle)</code> <i>angle</i> = Winkel, eine Zahl	Drehe nach links um <i>angle</i> Winkleinheiten in Grad. z. B. <code>left(180)</code> =Drehe die Turtle um 180 Grad nach links.
<code>goto(x,y)</code> <code>setpos(x,y)</code> <code>setposition(x,y)</code>	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. <code>goto(-110,90)</code> =bewegt die Turtle zum Punkt x=-110, y=90
<code>home()</code>	Bewegt die Turtle in die Ausgangslage und die Ausgangsorientierung zurück.

<p><code>speed(speed)</code></p> <p><i>speed</i> = ganze Zahl von 0 – 10</p> <p>0 = keine Animation, Turtle ‚springt‘</p> <p>1...10 = langsam ... schnell</p>	<p>Bestimmt die Geschwindigkeit der Turtle.</p>										
<p><code>circle(radius, extent)</code></p> <p><i>radius</i> = eine Zahl</p> <p><i>extent</i> (optional) = eine Zahl (Winkel)</p>	<p>Zeichne einen Kreis(bogen) mit gegebenem <i>radius</i>. Der Kreismittelpunkt ist <i>radius</i> Einheiten links von der Turtle, wenn <i>radius</i> > 0 ist, sonst rechts von der Turtle. Der Winkel <i>extent</i> gibt an, welcher Teil des Kreises gezeichnet wird. Ohne Angabe von <i>extent</i> wird ein voller Kreis gezeichnet.</p> <p>z. B. <code>circle(100)</code> = zeichnet einen Kreis mit Durchmesser 100 Pixel</p>										
<p><code>setheading(to_angle)</code></p> <p><i>to_angle</i> = eine Zahl (ein Winkel)</p>	<p>Dreht die Turtle so, dass sie in Richtung des angegebenen Winkels <i>to_angle</i> orientiert ist. Hier sind einige gebräuchliche Richtungen in Grad:</p> <table border="1"> <thead> <tr> <th>‚standard‘-mode:</th> <th>‚logo‘-mode:</th> </tr> </thead> <tbody> <tr> <td>0 – ost</td> <td>0 – nord</td> </tr> <tr> <td>90 – nord</td> <td>90 – ost</td> </tr> <tr> <td>180 – west</td> <td>180 – süd</td> </tr> <tr> <td>270 – süd</td> <td>270 – west</td> </tr> </tbody> </table>	‚standard‘-mode:	‚logo‘-mode:	0 – ost	0 – nord	90 – nord	90 – ost	180 – west	180 – süd	270 – süd	270 – west
‚standard‘-mode:	‚logo‘-mode:										
0 – ost	0 – nord										
90 – nord	90 – ost										
180 – west	180 – süd										
270 – süd	270 – west										

Steuerung und Funktionen

Funktionen und Schleifen erleichtern dir das Coden.

`def funktion():`

funktion = Name deiner

Funktion

Definiert eine **Funktion**, die aus allen Befehlen besteht, die darunter **eingerrückt** 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.

<p><code>for i in range(number):</code></p> <p><i>number</i> = eine Zahl, Anzahl der Wiederholungen</p>	<p>Definiert eine for Schleife, die den Code, der darunter eingerückt ist, eine bestimmte Anzahl <i>number</i> von Malen wiederholt. Ein Beispiel:</p> <p><code>for i in range(30): quadrat()</code> = Die Turtle zeichnet 30 Mal ein Quadrat, wie wir es vorher definiert haben.</p>
<p><code>name = wert</code></p> <p><i>name</i> = ein beliebiges Wort oder ein Buchstabe</p> <p><i>wert</i> = eine beliebige Zahl</p>	<p>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.</p> <p><code>seitenlänge=100</code></p> <p><code>forward(seitenlänge)</code> → Die Turtle läuft 100 Pixel nach vorne.</p>
<p><code>onclick(fun)</code></p> <p><i>fun</i> = eine Funktion mit 2 Argumenten (muss vorher definiert werden)</p>	<p>Führt die Funktion <i>fun</i> aus, wenn auf die Turtle geklickt wird.</p> <p>z.B. <code>onclick(getroffen)</code> = führt die Funktion ‚getroffen‘ aus, wenn auf die Turtle geklickt wird. Die Koordinaten der Turtle werden automatisch als Argumente übergeben.</p>

Turtle-Grafik- Fenster	Mit diesen Befehlen kannst du das Aussehen des Grafik-Fensters verändern.									
<p><code>bgcolor(„farbe“)</code></p>	<p>Füllt den Hintergrund mit der angegebenen <i>farbe</i>.</p>									
<p><code>bgpic(„name“)</code></p>	<p>Nimmt das Bild mit dem Dateinamen <i>name</i> als Hintergrund. Die Datei muss im selben Ordner abgespeichert sein wie das Skript.</p> <p>z. B. <code>bgpic(„maze.gif“)</code> = setzt die Datei maze.gif als Hintergrund ein</p>									
<p><code>screensize(canwidth, canvheight)</code></p> <p><i>canwidth, canheight</i> = zwei positive Zahlen</p>	<p>Stellt die Größe der mittels Scrollbars erreichbaren Zeichenfläche ein, auf der die Turtle zeichnet. Die Größe des Fensters wird nicht verändert.</p>									
<p><code>setup(width, height)</code></p> <p><i>width, height</i> = zwei positive Zahlen</p>	<p>Stellt die Größe des Fensters ein (in Pixel).</p>									
<p><code>mode(„mode“)</code></p> <p><i>mode</i> = die Zeichenkette „standard“ oder „logo“</p>	<p>Setzt den Turtle-Modus auf ‚standard‘ oder ‚logo‘ und führt ein <code>reset()</code> aus. Modus ‚standard‘ ist kompatibel mit turtle.py. Modus ‚logo‘ ist kompatibel mit den meisten Turtle-Grafik-Implementierungen in Logo.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Modus</th> <th>Anfängliche Orientierung der Turtle</th> <th>positive Winkel</th> </tr> </thead> <tbody> <tr> <td>‚standard‘</td> <td>nach rechts (Osten)</td> <td>im Gegenuhrzeigersinn</td> </tr> <tr> <td>‚logo‘</td> <td>nach oben (Norden)</td> <td>im Uhrzeigersinn</td> </tr> </tbody> </table>	Modus	Anfängliche Orientierung der Turtle	positive Winkel	‚standard‘	nach rechts (Osten)	im Gegenuhrzeigersinn	‚logo‘	nach oben (Norden)	im Uhrzeigersinn
Modus	Anfängliche Orientierung der Turtle	positive Winkel								
‚standard‘	nach rechts (Osten)	im Gegenuhrzeigersinn								
‚logo‘	nach oben (Norden)	im Uhrzeigersinn								

`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	Python versteht ein Wort nicht.
z. B. name 'pront' is not defined	
<pre>Traceback (most recent call last): File "C:/Python33/Test.py", line 10, in <module> pront("lklkj") NameError: name 'pront' is not defined >>> </pre>	

TIPPI!

- ➔ 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.

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.