Kurz-Referenz.

Allgemeines:

Gross-Kleinschreibung beachten
Die Programmstruktur wird über die Einrückung des Codes gesteuert
Kommentarzeilen beginnen mit #

Turtle-Umgebung initialisieren

`from gturtle import *`	Lädt alle Funktionen für die Graphik mit Turtle
`makeTurtle()`	Erstelle eine neue Turtle

Die wichtigsten Turtle-Befehle

`forward(314)`	`314` Pixel vorwärts bewegen
`back(44)`	`44` Pixel rückwärts bewegen
`left(31)`	Um den Winkel `31` Grad nach links drehen
`right(23)`	Um den Winkel `23` Grad nach rechts drehen
`penUp()`	Zeichenstift heben (die Turtle bewegt sich danach ohne zu zeichnen)
`penDown()`	Zeichenstift senken (die Turtle zeichnet danach wieder)
`setPenColor("red")`	Farbe des Stiftes wählen

Variablenzuweisung

`x=56`	Zuweisungen erfolgen immer von rechts nach links. Danach hat `x` den Wert `56`
`x=y`	Der Wert von `y` wird in der Variable `x` gespeichert.

Iteration

`repeat 9:`	Alle darunterstehenden eingerückten Zeilen werden `9` Mal wiederholt Die Anzahl der Wiederholungen ist von Beginn an festgelegt
`while x>10:`	Alle darunterstehenden eingerückten Zeilen werden so lange ausgeführt, wie die angegebene Bedingung (`x>10`) erfüllt ist. Die Anzahl der Wiederholungen ist an eine Bedingung geknüpft
`break`	Verlässt die Iterations-Struktur (Schlaufe) sofort

Selektion

`if x>0:`	Alle darunterstehenden eingerückten Zeilen werden nur dann ausgeführt, wenn die Bedingung (`x>0`) erfüllt ist.
`elif x==0:`	Alle darunterstehenden eingerückten Zeilen werden nur dann ausgeführt, wenn die obige Bedingung (`x>0`) nicht erfüllt, aber die neue Bedingung (`x==0`) erfüllt ist. Es können mehrere (oder auch gar keine) `elif` hintereinander verwendet werden.
`else`	Alle darunterstehenden eingerückten Zeilen werden ausgeführt, wenn keine der vorangehenden Bedingungen erfüllt ist.

Bedingungen

`a==3`	Ist `a` gleich `3`?
`a!=3`	Ist `a` ungleich `3`?
`a>3`	Ist `a` grösser als `3`?
`a<3`	Ist `a` kleiner als `3`?
`a>=3`	Ist `a` grösser gleich `3`?
`a<=3`	Ist `a` kleiner gleich `3`?
`and`	Verknüpft zwei Bedingungen (beide müssen richtig sein)
`or`	Verknüpft zwei Bedingungen (eine der beiden muss richtig sein)
`not`	Verneint eine Bedingung

Eigene Programme definieren

`def Name():`	Definiert das Programm `Name()` als alle nachsetehenden eingerückten Zeilen.
`def Recht(a,b):`	Definiert das Programm `Recht(a,b)` als alle nachsetehenden eingerückten Zeilen. Dabei werden zwei Argumente `a` und `b` übergeben

Weiteres zur Zuweisung von Variabeln

`x=154//60`	Quotient bei der ganzzahligen Division, hier wird x gleich 2 gesetzt
`y=154%60`	Rest bei der ganzzahligen Division, hier wird y gleich 34 gesetzt
`x+=1`	Erhöht den Wert von x um 1, Kurzscrhreibweise für `x=x+1`
`a*=3`	Verdreifacht den Wert von a, Kurzschreibweise für `a=3*a`
`from math import *`	lädt weitere mathematische Funktionen, z.B:
`h=3**8`	Potenzieren: `3` hoch `8`
`pi`	Die Kreiszahl π
`w=sqrt(8)`	Ziehen der Quadratwurzel
`x=round(pi,2)`	Rundet die Zahl `pi` auf zwei Nachkommastellen.
`from random import *`	lädt Funktionen, die Pseudozufallszahlen erzeugen
`z=randint(1,6)`	Liefert eine ganze Pseudozufallszahl zwischen `1` und `6` (einschliesslich `1` und `6`)
`z=random()`	Liefert eine reelle Pseudozufallszahl zwischen 0 und 1

Weitere Turtle-Befehle

`clean()`	Löscht alle Zeichnungen
`hideTurtle()`	Macht die Turtle unsichtbar. Dies beschleunigt das Programm erheblich
`showTurtle()`	Macht die Turtle wieder sichtbar.
`speed(60)`	Setzt die Turtlegeschwindigkeit. Verwende `speed(-1)` für maximale Geschwindigkeit.
`dot(13)`	Zeichnet einen Punkt mit Durchmesser `d`
`leftArc(60,90)`	Zeichnet einen Kreisbogen nach links mit Radius `60` und Sektorwinkel `90`
`rightArc(60,90)`	Zeichnet einen Kreisbogen nach rechts mit Radius `60` und Sektorwinkel `90`
`leftCircle(60)`	Zeichnet einen Kreis nach links mit Radius `60`
`rightCircle(60)`	Zeichnet einen Kreis nach rechts mit Radius `60`
`setFillColor("red")`	Farbe zur Flächenfüllung auf rot setzen
`fill()`	Füllt vom aktuellen Punkt die zusammenhängende Fläche mit Farbe aus
`fillToPoint()`	Alle weiteren Punkte werden mit dem aktuellen Punkt verbunden
`fillOff()`	Beendet den Verbindungsmodus
`setLineWidth(3)`	Setzt die Breite der zu zeichnenden Linien auf `3`
`penErase()`	Setzt die Stiftfarbe auf die Hintergrundfarbe
`heading(60)`	Setzt die Richtung der Turtle auf `60` Grad bezüglich der Senkrechten nach oben
`home()`	Setzt Turtle in die Mitte des Fensters, Richtung nach oben
`setPos(60,100)`	Setzt den Farbstift direkt an die Position (`60,100`); (`0,0`) ist die Mitte des Feldes Die erste Zahl gibt die horizontale Koordinate an, positiv nach rechts, die zweite Zahl gibt die vertikale Koordinate an, positiv nach oben.
`moveTo(60,100)`	Verbindet aktuelle Position mit der Position (`60,100`)
`getX()`	Liefert die aktuelle x-Koordinate. Verwende z.B. `ix=getX()`.
`getY()`	Liefert die aktuelle y-Koordinate. Verwende z.B. `iy=getY()`.

Interaktion

`input("Wähle x=")`	Liefert ein vom Benutzer definierten Wert. Verwende z.B. `x=input("Wähle x=")`
`msgDlg("Wert x:",x)`	Öffnet ein Fenster und gibt einen Text aus. Hier den Text `"Wert x:"` und danach den Wert der Variable `x`

Abkürzungen

`fd(100)`	`forward(100)`
`bk(100)`	`back(100)`
`rt(90)`	`right(90)`
`lt(90)`	`left(90)`
`ht()`	`hideTurtle()`
`st()`	`showTurtle()`
`pu()`	`penUp()`
`pd()`	`penDown()`
`pe()`	`penErase()`