Dokumentation Turtlegrafik[1]
Module import:
from gturtle import *
Sprite-Bibliothek:
http://www.jython.ch/index.php?inhalt_links=navigation.inc.php&inhalt_mitte=sprites.html

Funktion	Aktion
makeTurtle()	erzeugt eine (globale) Turtle im neuen Grafikfenster
makeTurtle(color)	erzeugt eine Turtle mit angegebener Farbe
makeTurtle("sprites/turtle.gif")	erzeugt Turtle mit einem eigenen Turtle-Bild turtle.gif
t = Turtle()	erzeugt ein Turtleobjekt t
tf = TurtleFrame()	erzeugt ein Bildschirmfenster, in dem mehrere Turtles leben
t = Turtle(tf)	erzeugt ein Turtleobjekt t im TurtleFrame tf
clone()	erzeugt ein Turtleklon (gleiche Farbe, Position, Blickrichtung)
isDisposed()	gibt True zurück, falls das Turtlefenster geschlossen ist
putSleep()	hält den Programmablauf an, bis wakeUp() aufgerufen wird
wakeUp()	führt angehaltenen Programmablauf weiter
enableRepaint(False)	schaltet das automatische Bildschirmrendering aus
repaint()	rendert den Bildschirm (nach dem Ausschalten des automatischen Rendering)
savePlayground()	speichert den Playground in einem internen Bildbuffer (zurückholen mit clear())
savePlayground(fileName, format)	speichert den Playground als Bilddatei (format: "png" oder "gif"). Im Fehlerfall wird False zur□ckgegeben
setPlaygroundSize(width, height)	setzt die Grösse des Turtlefensters unabhängig von den Einstellungen in TigerJython (muss vor makeTurtle() aufgerufen werden)
setFramePosition(x, y)	setzt die obere linke Ecke des Turtlefensters an die gegebene Bildschirmposition
setFramePositionCenter()	setzt das Turtlefensters in Bildschirmmitte

Bewegen

back(distance), bk(distance)	bewegt Turtle rückwärts
forward(distance), fd(distance)	bewegt Turtle vorwärts
hideTurtle(), ht()	macht Turtle unsichtbar (Turtle zeichnet schneller)
home()	setzt Turtle in die Mitte des Fensters mit Richtung nach oben
left(angle), lt(angle)	dreht Turtle nach links
penDown(), pd()	setzt Zeichenstift ab (Spur sichtbar)
penErase(), pe()	setzt die Stiftfarbe auf die Hintergrundfarbe
leftArc(radius, angle)	bewegt Turtle auf einem Bogen mit dem Sektor-Winkel *angle* nach links

[1] Auszug aus http://www.jython.ch/index.php?inhalt_links=navigation.inc.php&inhalt_mitte=turtle/turtledoc.html

leftCircle(radius)	bewegt Turtle auf einem Kreis nach links
penUp(), pu()	hebt den Zeichenstift (Spur unsichtbar)
penWidth(width)	setzt die Dicke des Stifts in Pixel
right(angle), rt(angle)	dreht Turtle nach rechts
rightArc(radius, angle)	bewegt Turtle auf einem Bogen mit dem Sektor-Winkel *angle* nach rechts
rightCircle(radius)	bewegt Turtle auf einem Kreis nach rechts
setCustomCursor(cursorImage)	wählt die Bilddatei des Mauscursors
setCustomCursor(cursorImage, Point(x, y))	wählt die Bilddatei des Mauscursors unter Angabe der Mausposition innerhalb des Bildes
setLineWidth(width)	setzt die Dicke des Stifts in Pixel
showTurtle(), st()	zeigt Turtle
speed(speed)	setzt Turtlegeschwindigkeit
delay(time)	hält das Programm während der Zeit time (in Millisekunden) an
wrap()	setzt Turtlepositionen ausserhalb des Fensters ins Fenster zurück
clip()	turtles ausserhalb des Fensters sind nicht sichtbar
getPlaygroundWidth()	gibt die Breite m des Turtlefensters zurück (Turtlekoordinaten -m/2... m/2)
getPlaygroundHeight()	gibt die Höhe n des Turtlefensters zurück (Turtlekoordinaten -n/2... n/2)

Positionieren

direction(x, y)	gibt den Winkel (in Grad) für die Drehung zur Position (x, y) zurück
direction(coords)	dasselbe, aber Koordinatenangabe als Liste, Tupel oder komplexe Zahl
direction(turtle)	gibt den Winkel (in Grad) für die Drehung zu einer anderen Turtle zurück
distance(x, y)	gibt die Entfernung der Turtle zum Punkt(x, y) zurück
distance(coords)	dasselbe, aber Koordinatenangabe als Liste, Tupel oder komplexe Zahl
distance(turtle)	gibt die Entfernung der Turtle zu einer anderen Turtle zurück
getPos()	gibt die Turtleposition zurück als Punkt
getX()	gibt die aktuelle x-Koordinate der Turtle zurück
getY()	gibt die aktuelle y-Koordinate der Turtle zurück
heading()	gibt die Richtung der Turtle zurück
heading(degrees)	setzt die Richtung der Turtle (0 ist gegen oben, im Uhrzeigersinn)
moveTo(x, y)	bewegt Turtle auf die Position (x, y)
moveTo(coords)	dasselbe, aber Koordinatenangabe als Liste, Tupel oder komplexe Zahl
setHeading(degrees), setH(degrees)	setzt die Richtung der Turtle (0 gegen oben, im Uhrzeigersinn)
setRandomHeading()	setzt die Richtung zufällig zwischen 0 und 360°
setPos(x, y)	setzt Turtle auf die Position (x, y)

setPos(coords)	dasselbe, aber Koordinatenangabe als Liste, Tupel oder komplexe Zahl
setX(x)	setzt Turtle auf x-Koordinate
setY(y)	setzt Turtle auf y-Koordinate
setRandomPos(width, height)	setzt Turtle auf einen zufälligen Punkt im Bereich 0..width, 0..height
setScreenPos(x, y)	setzt Turtle auf gegebene Pixelkoordinaten (x, y)
setScreenPos(Point(x, y))	setzt Turtle auf gegebene Pixelkoordinaten
towards(x, y)	gibt die Richtung (in Grad) zur Position (x, y) (auch list, tuple, complex)
towards(turtle2)	gibt die Richtung (in Grad) zu einer anderen Turtle turtle2 zurück
toTurtlePos(x, y)	gibt die Turtlekoordinaten zu den gegebenen Pixelkoordinaten (x, y) als Liste zurück
toTurtlePos(Point(x, y))	gibt die Turtlekoordinaten zu den gegebenen Pixelkoordinaten als Liste zurück
pushState()	speichert den Turtlezustand in einem Stapelspeicher
popState()	holt den zuletzt gespeicherten Zustand vom Stapelspeicher
clearStates()	löscht den Stapelspeicher

Farben

askColor(title, defaultColor)	zeigt ein Fenster zur Farbwahl und gibt die gewählte Farbe zurück, (None, wenn Abbrechen gedrückt wurde)title: Fenstertitel an, defaultColor: Farbvorschlag
clear()	löscht die Zeichnung und versteckt alle Turtles (sie bleiben am Ort). Falls der mit savePlayground() erzeugte Bildbuffer nicht leer ist, wird dieser angezeigt
clear(color)	löscht die Zeichnung, versteckt alle Turtles (sie bleiben am Ort) und färbt den Hintergrund
clean()	löscht die Turtlespuren (Turtles bleiben am Ort sichtbar). Der mit savePlayground() erzeugte Bildbuffer wird gelöscht
clean(color)	löscht die Zeichnung und färbt den Hintergrund (Turtles bleiben am Ort sichtbar).
clearScreen(), cs()	löscht die Turtlespuren und setzt die Turtle an Homeposition
dot(diameter)	zeichnet einen mit Stiftfarbe gefüllten Kreis
openDot(diameter)	zeichnet einen nicht gefüllten Kreis
spray(density, spread, size)	zeichnet eine zufällige Punktwolke an der Turtleposition mit geg. Anzahl Punkten, Ausdehnung und Punktgrösse (ohne size: size = 1)
fill()	füllt die geschlossene Figur, in der sich die Turtle befindet mit der Füllfarbe
fill(x , y)	füllt eine geschlossene Figur um den inneren Punkt (x, y) mit der Füllfarbe
fill(coords)	dasselbe, aber Koordinatenangabe als Liste, Tupel oder komplexe Zahl
fillToPoint()	füllt fortlaufend die gezeichnete Figur von der aktuellen Turtleposition mit der Stiftfarbe
fillToPoint(x , y)	füllt fortlaufend die gezeichnete Figur vom Punkt (x, y) mit der Stiftfarbe

fillToPoint(coords	dasselbe, aber Koordinatenangabe als Liste, Tupel oder komplexe Zahl
fillToHorizontal(y)	füllt fortlaufend die Fläche zwischen der Figur und der horizontalen Linie in Höhe y mit der Stiftfarbe
fillToVertical(x)	füllt fortlaufend die Fläche zwischen der Figur und der vertikalen Linie am Wert xmit der Stiftfarbe
fillOff()	beendet den fortlaufenden Füllmodus
getColor()	gibt die Turtlefarbe zurück
getColorStr()	gibt die X11-Turtlefarbe zurück
getFillColor()	gibt die Füllfarbe zurück
getFillColorStr()	gibt die X11-Füllfarbe zurück
getPixelColor()	gibt die Farbe des Pixels an der Turtlekoordinate zurück (None, falls ausserhalb des Turtlefensters)
getPixelColorStr()	gibt die Farbe des Pixels an der Turtlekoordinate als X11-Farbe zurück (leerer String, falls kein X11-Farbname existriert; None, falls ausserhalb des Turtlefensters)
makeColor()	gibt eine Farbreferenz von value zurück. Werte-Beispiele: ("7FFED4"), ("Aqua-Marine"), (0x7FFED4), (8388564), (0.5, 1.0, 0.83), (128, 255, 212), ("rainbow", n) mit n = 0..1, Lichtspektrum
getRandomX11Color()	gibt eine zufällige X11-Farbe zurück
setColor(color)	legt Turtlefarbe fest
setPenColor(color)	legt Stiftfarbe fest
setPenWidth(width)	setzt die Dicke des Stiftes in Pixel
setFillColor(color)	legt Füllfarbe fest
startPath()	startet die Aufzeichnung der Turtlebewegung zum nachträglichen Füllen
fillPath()	verbindet die aktuelle Turtleposition mit dem Startpunkt und füllt die geschlossene Figur mit der Füllfarbe
stampTurtle()	erzeugt ein Turtlebild an der aktuellen Turtleposition
stampTurtle(color)	erzeugt ein Turtlebild mit angegebener Farbe an der aktuellen Turtleposition

Callbacks

makeTurtle(mouseNNN = onMouseNNN) auch mehrere, durch Komma getrennt	registriert die Callbackfunktion onMouseNNN(x, y), die beim Mausevent aufgerufen wird. Werte für NNN: Pressed, Released, Clicked, Dragged, Moved, Entered, Exited, SingleClicked, DoubleClicked, Hit: Aufruf im eigenen Thread, HitX: Dasselbe, aber nachfolgende Events ignoriert, bis Callback zurückkehrt
isLeftMouseButton(), isRightMouseButton()	gibt True zurück, falls beim Event die linke bzw. rechte Maustaste verwendet wurde
makeTurtle(keyNNN = onKeyNNN)	registriert die Callbackfunktion onKeyNNN(keyCode), die beim Drücken einer Tastaturtaste aufgerufen wird. Werte für NNN: Pressed, Hit: Aufruf im eigenen Thread, HitX: Dasselbe, aber nachfolgende Events ignoriert, bis Callback zurückkehrt. keyCode ist ein für die Taste eindeutiger integer Code

getKeyModifiers()	liefert nach einem Tastaturevent einen Code für Spezialtasten (Shift, Ctrl, usw., auch kombiniert)
makeTurtle(closeClicked = onCloseClicked)	registriert die Callbackfunktion onCloseClicked(), die beim Klick des Close-Buttons des Turtlefensters aufgerufen wird. Das Fenster muss mit dispose() geschlossen werden
makeTurtle(turtleHit=onTurtleHit)	registriert die Callbackfunktion onTurtleHit(x, y), die aufgerufen wird, wenn auf das Turtlebild geklickt wird
t = Turtle(turtleHit = onTurtleHit)	registriert die Callbackfunktion onTurtleHit(t, x, y), die aufgerufen wird, wenn auf das Bild der Turtle t geklickt wird
showSimulationBar(NNN = onNNN)	zeigt ein Dialogfenster mit den Button 'Step', 'Run'/'Pause', 'Reset' und einem Slider zum Einstellen der Simulationsperiode. Folgende Callbacks NNN können registiert werden: start, pause, step, reset, change(Parameter: simulationPeriod), loop, exit (close button gedrückt). loop wird in jedem Simulationszyklus vom Simulationsthread aufgerufen
showSimulationBar(ulx, uly, initPeriod, NNN = onNNN)	wie oben, aber mit Positionierung des Dialogfensters (obere linke Ecke) und Anfangswert der Simulationsperiode (default: 100)
hideSimulationBar()	schliesst das Dialogfenster und gibt alle Ressourcen frei

Tastatur

getKey()	holt den letzten Tastendruck ab und liefert String zurück (leer, falls illegale Taste)
getKeyCode()	holt den letzten Tastendruck ab und liefert Code zurück
getKeyWait()	wartet bis Taste gedrückt und liefert String zurück (leer, falls illegale Taste)
getKeyCodeWait()	wartet bis Taste gedrückt und liefert Code zurück
kbhit()	liefert True, falls ein Tastendruck noch nicht mit getKey() od. getKeyCode() abgeholt ist

Texte, Bilder und Sound

addStatusBar(20)	fügt eine Statusbar mit der Höhe 20 Pixel hinzu
label(param)	schreibt str(param) an der aktuellen Position aus (linksbündig)
label(param1, param2, ...)	konkateniert str(params) mit Leerstellen getrennt und schreibt den Text aus
label(param1, param2, ..., adjust = 'x')	dasselbe, aber mit x = 'l' linksbündig (default), 'c' zentriert, 'r' rechtsbündig
printerPlot(draw)	druckt die mit der Funktion draw erstellte Zeichnung
setFont(Font font)	legt Schriftart fest. font ist ein Objekt der Klasse Font Beispiel:. Font("Courier New", Font.BOLD, 12). Default: Font("SansSerif", Font.PLAIN, 24)
setFont(name)	legt neue Schriftart mit bestehenden Schriftstil und Schriftgrösse fest
setFont(name, style)	legt neue Schriftart und Schriftstil mit bestehender Schriftgrösse fest style = Font.PLAIN, Font.BOLD, Font.ITALIC
setFont(name, style, size)	legt neue Schriftart, Schriftstil und Schriftgrösse fest
setFontSize(size)	legt neue Schriftgrösse fest (Schriftart und Stil bleiben erhalten)

getTextHeight()	liefert die Höhe des Texts im aktuellen Font (in Pixels)
getTextAscent()	liefert die Höhe des Text-Ascenders im aktuellen Font (in Pixels)
getTextDescent()	liefert die Höhe des Text-Decenders im aktuellen Font (in Pixels)
getTextWidth(text)	liefert die Breite des gegebenen Texts im aktuellen Font (in Pixels)
setStatusText("Press any key!")	schreibt eine Mitteilung in die Statusbar
setTitel("Text")	schreibt den Text in die Titelzeile
img = getImage(path)	lädt ein Bild (im png- gif, jpg-Format) vom lokalen Filesystem oder von einer URL und gibt eine Referenz darauf zurück. Für path = sprites/nnn werden auch Bilder von der TigerJython-Distribution geladen (Help/Bilderbibliothek zeigt die vorhanden Bilder)
drawImage(img)	stelt das Bild img an der Position der Turtle mit ihrer Blickrichtung dar
drawImage(path)	lädt ein Bild (im png-, gif-, jpg-Format) vom lokalen Filesystem oder von einer URL und stellt es an der Position der Turtle mit ihrer Blickrichtung dar. Für path = sprites/nnn werden auch Bilder von der TigerJython-Distribution geladen

Dialoge [2]

msgDlg(message)	öffnet einen modalen Dialog mit einem OK-Button und gegebenem Mitteilungstext
msgDlg(message, title = title_text)	dasselbe mit Titelangabe
inputInt(prompt)	öffnet einen modalen Dialog mit OK/Abbrechen-Buttons. OK gibt den eingegebenen Integer zurück (falls kein Integer, wird Dialog neu angezeigt). Abbrechen od. Schliessen beendet das Programm
inputInt(prompt, False)	dasselbe, aber Abbrechen/Schliessen beendet das Programm nicht, sondern gibt None zurück
inputFloat(prompt)	öffnet einen modalen Dialog mit OK/Abbrechen-Buttons. OK gibt den eingegebenen Float zurück (falls kein Float, wird Dialog neu angezeigt). Abbrechen od. Schliessen beendet das Programm
inputFloat(prompt, False)	dasselbe, aber Abbrechen/Schliessen beendet das Programm nicht, sondern gibt None zurück
inputString(prompt)	öffnet einen modalen Dialog mit OK/Abbrechen-Buttons. OK gibt den eingegeben String zurück. Abbrechen od. Schliessen beendet das Programm
inputString(prompt, False)	dasselbe, aber Abbrechen/Schliessen beendet das Programm nicht, sondern gibt None zurück
input(prompt)	öffnet einen modalen Dialog mit OK/Abbrechen-Buttons. OK gibt Eingabe als Integer, Float oder String zurück. Abbrechen od. Schliessen beendet das Programm
input(prompt, False)	dasselbe, aber Abbrechen/Schliessen beendet das Programm nicht, sondern gibt None zurück
askYesNo(prompt)	öffnet einen modalen Dialog mit Ja/Nein-Buttons. Ja gibt True, Nein gibt False zurück. Schliessen beendet das Programm
askYesNo(prompt, False)	dasselbe, aber Schliessen beendet das Programm nicht, sondern gibt None zurück

[2] weiteres: http://www.tigerjython.ch/index.php?inhalt_links=navigation.inc.php&inhalt_mitte=turtle/entrydialogdoc.html