Appendix F: Turtle Methods

13

This documentation was generated from the Python documentation available by typing

```
from turtle import *
help(Turtle)
```

in the Python shell. In the documentation found here the variable *turtle* is a reference to a Turtle object. This is a subset of that documentation. To see complete documentation use the Python help system as described above.

Method	Description

turtle.back(distance)	Aliases: backward bk
	
	Argument:
	distance – a number
	
	Move the turtle backward by distance, opposite to the direction the turtle is headed. Do not change the turtle's heading.
	
	Example (for a Turtle instance named turtle):
	>>> turtle.position()
	(0.00, 0.00)
	>>> turtle.backward(30)
	>>> turtle.position()
	(-30.00, 0.00)

turtle.begin_fill()	Called just before drawing a shape to be filled.
	
	Example (for a Turtle instance named turtle):
	>>> turtle.color("black", "red")
	>>> turtle.begin_fill()
	>>> turtle.circle(60)
	>>> turtle.end_fill()

K.D. Lee, *Python Programming Fundamentals*,
Undergraduate Topics in Computer Science, DOI 10.1007/978-1-4471-6642-9_13

Method	Description
turtle.begin_poly()	Start recording the vertices of a polygon. Current turtle position is first point of polygon. Example (for a Turtle instance named turtle): >>> turtle.begin_poly()

turtle.circle(radius, extent=None, steps=None)	Arguments:
	radius – a number
	extent (optional) – a number
	steps (optional) – an integer
	
	Draw a circle with given radius. The center is radius units left of the turtle; extent - an angle - determines which part of the circle is drawn. If extent is not given, draw the entire circle. If extent is not a full circle, one endpoint of the arc is the current pen position. Draw the arc in counterclockwise direction if radius is positive, otherwise in clockwise direction. Finally the direction of the turtle is changed by the amount of extent.
	
	As the circle is approximated by an inscribed regular polygon, steps determines the number of steps to use. If not given, it will be calculated automatically. Maybe used to draw regular polygons.
	
	call: circle(radius) # full circle
	–or: circle(radius, extent) # arc
	–or: circle(radius, extent, steps)
	–or: circle(radius, steps=6) # 6-sided polygon
	
	Example (for a Turtle instance named turtle):
	>>> turtle.circle(50)
	>>> turtle.circle(120, 180) # semicircle

turtle.clear()	Delete the turtle's drawings from the screen. Do not move turtle. State and position of the turtle as well as drawings of other turtles are not affected. Examples (for a Turtle instance named turtle): >>> turtle.clear()
turtle.color(*args)	Arguments: Several input formats are allowed. They use 0, 1, 2, or 3 arguments as follows: color()

Method	Description

	Return the current pencolor and the current fillcolor
	as a pair of color specification strings as are returned
	by pencolor and fillcolor.
	color(colorstring), color((r,g,b)), color(r,g,b)
	inputs as in pencolor, set both, fillcolor and pencolor,
	to the given value.
	color(colorstring1, colorstring2),
	color((r1,g1,b1), (r2,g2,b2))
	equivalent to pencolor(colorstring1) and fillcolor(colorstring2)
	and analogously, if the other input format is used.
	If turtleshape is a polygon, outline and interior of that polygon
	is drawn with the newly set colors.
	For mor info see: pencolor, fillcolor
	
	Example (for a Turtle instance named turtle):
	>>>turtle.color('red', 'green')
	>>> turtle.color()
	('red', 'green')
	>>> colormode(255)
	>>> color((40, 80, 120), (160, 200, 240))
	>>> color()
	('#285078', '#a0c8f0')

turtle.degrees()	Set the angle measurement units to degrees.
	
	Example (for a Turtle instance named turtle):
	>>> turtle.heading()
	1.5707963267948966
	>>> turtle.degrees()
	>>> turtle.heading()
	90.0

turtle.dot(size=None, *color)	Optional arguments:
	size – an integer > = 1 (if given)
	color – a colorstring or a numeric color tuple
	
	Draw a circular dot with diameter size, using color.
	If size is not given, the maximum of pensize+4 and 2*pensize is used.
	
	Example (for a Turtle instance named turtle):
	>>> turtle.dot()
	>>> turtle.fd(50); turtle.dot(20, "blue"); turtle.fd(50)

turtle.end_fill()	Fill the shape drawn after the call begin_fill(). Example (for a Turtle instance named turtle): >>> turtle.color("black", "red")

Method	Description

	>>> turtle.begin_fill()
	>>> turtle.circle(60)
	>>> turtle.end_fill()

turtle.end_poly()	Stop recording the vertices of a polygon. Current turtle position is last point of polygon. This will be connected with the first point. Example (for a Turtle instance named turtle): >>> turtle.end_poly()

	turtle.filling()
	Return fillstate (True if filling, False else).
	
	Example (for a Turtle instance named turtle):
	>>> turtle.begin_fill()
	>>> if turtle.filling():
	turtle.pensize(5)
	else:
	turtle.pensize(3)

turtle.fillcolor(*args)	Return or set the fillcolor.
	
	Arguments:
	Four input formats are allowed:
	- fillcolor()
	Return the current fillcolor as color specification string,
	possibly in hex-number format (see example).
	May be used as input to another color/pencolor/fillcolor call.
	- fillcolor(colorstring)
	s is a Tk color specification string, such as "red" or "yellow"
	- fillcolor((r, g, b))
	a tuple of r, g, and b, which represent, an RGB color,
	and each of r, g, and b are in the range 0..colormode,
	where colormode is either 1.0 or 255
	- fillcolor(r, g, b)
	r, g, and b represent an RGB color, and each of r, g, and b
	are in the range 0..colormode
	
	If turtleshape is a polygon, the interior of that polygon is drawn
	with the newly set fillcolor.
	
	Example (for a Turtle instance named turtle):
	>>> turtle.fillcolor('violet')
	>>> col = turtle.pencolor()
	>>> turtle.fillcolor(col)
	>>> turtle.fillcolor(0, .5, 0)

Method Description

turtle.forward(distance)

Aliases: fd

Argument:
distance – a number (integer or float)

Move the turtle forward by the specified distance, in the direction the turtle is headed.

Example (for a Turtle instance named turtle):

```
>>> turtle.position()
(0.00, 0.00)
>>> turtle.forward(25)
>>> turtle.position()
(25.00,0.00)
>>> turtle.forward(-75)
>>> turtle.position()
(-50.00,0.00)
```

turtle.get_poly()

Return the lastly recorded polygon.

Example (for a Turtle instance named turtle):

```
>>> p = turtle.get_poly()
>>> turtle.register_shape("myFavouriteShape", p)
```

turtle.get_shapepoly()

Return the current shape polygon as tuple of coordinate pairs.

Examples (for a Turtle instance named turtle):

```
>>> turtle.shape("square")
>>> turtle.shapetransform(4, -1, 0, 2)
>>> turtle.get_shapepoly()
((50, -20), (30, 20), (-50, 20), (-30, -20))
```

turtle.getscreen()

Return the TurtleScreen object, the turtle is drawing on.
So TurtleScreen-methods can be called for that object.

Example (for a Turtle instance named turtle):

```
>>> ts = turtle.getscreen()
>>> ts
<turtle.TurtleScreen object at 0x0106B770>
>>> ts.bgcolor("pink")
```

turtle.goto(x, y=None)

Aliases: setpos setposition

Arguments:
x – a number or a pair/vector of numbers
y – a number None

call: goto(x, y) # two coordinates

Method	Description

	–or: goto((x, y)) # a pair (tuple) of coordinates
	–or: goto(vec) # e.g. as returned by pos()
	
	Move turtle to an absolute position. If the pen is down,
	a line will be drawn. The turtle's orientation does not change.
	
	Example (for a Turtle instance named turtle):
	>>> tp = turtle.pos()
	>>> tp
	(0.00, 0.00)
	>>> turtle.setpos(60,30)
	>>> turtle.pos()
	(60.00,30.00)
	>>> turtle.setpos((20,80))
	>>> turtle.pos()
	(20.00,80.00)
	>>> turtle.setpos(tp)
	>>> turtle.pos()
	(0.00,0.00)

turtle.heading()	Return the turtle's current heading.
	
	Example (for a Turtle instance named turtle):
	>>> turtle.left(67)
	>>> turtle.heading()
	67.0

turtle.hideturtle()	Makes the turtle invisible. Aliases: ht It's a good idea to do this while you're in the middle of a complicated drawing, because hiding the turtle speeds up the drawing observably. Example (for a Turtle instance named turtle): >>> turtle.hideturtle()

turtle.isdown()	Return True if pen is down, False if it's up.
	
	Example (for a Turtle instance named turtle):
	>>> turtle.penup()
	>>> turtle.isdown()
	False
	>>> turtle.pendown()
	>>> turtle.isdown()
	True

Method Description

turtle.isvisible()

Return True if the Turtle is shown, False if it's hidden.

Example (for a Turtle instance named turtle):
>>> turtle.hideturtle()
>>> print(turtle.isvisible())
False

turtle.left(angle)

Aliases: lt

Argument:
angle – a number (integer or float)

Turn turtle left by angle units. (Units are by default degrees,
but can be set via the degrees() and radians() functions.)
Angle orientation depends on mode. (See this.)

Example (for a Turtle instance named turtle):
>>> turtle.heading()
22.0
>>> turtle.left(45)
>>> turtle.heading()
67.0

turtle.onclick(fun, btn=1, add=None)

Bind fun to mouse-click event on this turtle on canvas.

Arguments:
fun – a function with two arguments, to which will be assigned
the coordinates of the clicked point on the canvas.
num – number of the mouse-button defaults to 1 (left mouse button).
add – True or False. If True, new binding will be added, otherwise
it will replace a former binding.

Example for the anonymous turtle, i. e. the procedural way:

```
>>> def turn(x, y):
        turtle.left(360)

>>> onclick(turn) # Now clicking into the turtle will turn it.
>>> onclick(None) # event-binding will be removed
```

turtle.ondrag(fun, btn=1, add=None)

Bind fun to mouse-move event on this turtle on canvas.

Arguments:
fun – a function with two arguments, to which will be assigned
the coordinates of the clicked point on the canvas.

Method	Description
	num – number of the mouse-button defaults to 1 (left mouse button). Every sequence of mouse-move-events on a turtle is preceded by a mouse-click event on that turtle. Example (for a Turtle instance named turtle): >>> turtle.ondrag(turtle.goto) ### Subsequently clicking and dragging a Turtle will ### move it across the screen thereby producing handdrawings ### (if pen is down).
turtle.onrelease(fun, btn=1, add=None)	Bind fun to mouse-button-release event on this turtle on canvas. Arguments: fun – a function with two arguments, to which will be assigned the coordinates of the clicked point on the canvas. num – number of the mouse-button defaults to 1 (left mouse button).

turtle.pencolor(*args)	Return or set the pencolor.
	
	Arguments:
	Four input formats are allowed:
	- pencolor()
	Return the current pencolor as color specification string, possibly in hex-number format (see example).
	May be used as input to another color/pencolor/fillcolor call.
	- pencolor(colorstring)
	s is a Tk color specification string, such as "red" or "yellow"
	- pencolor((r, g, b))
	a tuple of r, g, and b, which represent, an RGB color, and each of r, g, and b are in the range 0..colormode, where colormode is either 1.0 or 255
	- pencolor(r, g, b)
	r, g, and b represent an RGB color, and each of r, g, and b are in the range 0..colormode
	
	If turtleshape is a polygon, the outline of that polygon is drawn with the newly set pencolor.
	
	Example (for a Turtle instance named turtle):
	>>> turtle.pencolor('brown')
	>>> tup = (0.2, 0.8, 0.55)
	>>> turtle.pencolor(tup)
	>>> turtle.pencolor()
	'#33cc8c'

Method	Description
turtle.pendown()	Pull the pen down – drawing when moving. Aliases: pd down Example (for a Turtle instance named turtle): >>> turtle.pendown()
turtle.pensize(width=None)	Set or return the line thickness. Aliases: width Argument: width – positive number Set the line thickness to width or return it. If resizemode is set to "auto" and turtleshape is a polygon, that polygon is drawn with the same line thickness. If no argument is given, current pensize is returned. Example (for a Turtle instance named turtle): >>> turtle.pensize() 1 turtle.pensize(10) # from here on lines of width 10 are drawn
turtle.penup()	Pull the pen up – no drawing when moving. Aliases: pu up Example (for a Turtle instance named turtle): >>> turtle.penup()

turtle.radians()	Set the angle measurement units to radians.
	
	Example (for a Turtle instance named turtle):
	>>> turtle.heading()
	90
	>>> turtle.radians()
	>>> turtle.heading()
	1.5707963267948966

turtle.reset()	Delete the turtle's drawings from the screen, re-center the turtle and set variables to the default values.
	
	Example (for a Turtle instance named turtle):
	>>> turtle.position()
	(0.00,-22.00)
	>>> turtle.heading()

Method	Description

	100.0
	>>> turtle.reset()
	>>> turtle.position()
	(0.00,0.00)
	>>> turtle.heading()
	0.0

turtle.setheading(to_angle)	Set the orientation of the turtle to to_angle.
	
	Aliases: seth
	
	Argument:
	to_angle – a number (integer or float)
	
	Set the orientation of the turtle to to_angle.
	Here are some common directions in degrees:
	
	standard - mode: logo-mode:
	————————————————
	0 - east 0 - north
	90 - north 90 - east
	180 - west 180 - south
	270 - south 270 - west
	
	Example (for a Turtle instance named turtle):
	>>> turtle.setheading(90)
	>>> turtle.heading()
	90

turtle.shape(name=None)	Set turtle shape to shape with given name / return current shapename.
	
	Optional argument:
	name – a string, which is a valid shapename
	
	Set turtle shape to shape with given name or, if name is not given,
	return name of current shape.
	Shape with name must exist in the TurtleScreen's shape dictionary.
	Initially there are the following polygon shapes:
	'arrow', 'turtle', 'circle', 'square', 'triangle', 'classic'.
	To learn about how to deal with shapes see Screen-method register_shape.
	
	Example (for a Turtle instance named turtle):
	>>> turtle.shape()
	'arrow'
	>>> turtle.shape("turtle")
	>>> turtle.shape()
	'turtle'

Method	Description

turtle.showturtle()	Makes the turtle visible.
	
	Aliases: st
	
	Example (for a Turtle instance named turtle):
	>>> turtle.hideturtle()
	>>> turtle.showturtle()

turtle.speed(speed=None)	Return or set the turtle's speed. Optional argument: speed – an integer in the range 0..10 or a speedstring (see below) Set the turtle's speed to an integer value in the range 0 .. 10. If no argument is given: return current speed. If input is a number greater than 10 or smaller than 0.5, speed is set to 0. Speedstrings are mapped to speedvalues in the following way: 'fastest' : 0 'fast' : 10 'normal' : 6 'slow' : 3 'slowest' : 1 speeds from 1 to 10 enforce increasingly faster animation of line drawing and turtle turning. Attention: speed = 0 : *no* animation takes place. forward/back makes turtle jump and likewise left/right make the turtle turn instantly. Example (for a Turtle instance named turtle): >>> turtle.speed(3)

turtle.undo()	Undo (repeatedly) the last turtle action.
	Number of available undo actions is determined by the size of
	the undobuffer.
	
	Example (for a Turtle instance named turtle):
	>>> for i in range(4):
	turtle.fd(50); turtle.lt(80)
	
	>>> for i in range(8):
	turtle.undo()

Method	Description

turtle.write(arg, move=False, align='left', font=('Arial', 8, 'normal'))	Write text at the current turtle position.
	
	Arguments:
	arg – info, which is to be written to the TurtleScreen
	move (optional) – True/False
	align (optional) – one of the strings "left", "center" or right"
	font (optional) – a triple (fontname, fontsize, fonttype)
	
	Write text - the string representation of arg - at the current turtle position according to align ("left", "center" or right") and with the given font.
	If move is True, the pen is moved to the bottom-right corner of the text. By default, move is False.
	
	Example (for a Turtle instance named turtle):
	>>> turtle.write('Home = ', True, align="center")
	>>> turtle.write((0,0), True)

turtle.xcor()	Return the turtle's x coordinate.
	
	Example (for a Turtle instance named turtle):
	>>> reset()
	>>> turtle.left(60)
	>>> turtle.forward(100)
	>>> print(turtle.xcor())
	50.0

turtle.ycor()	Return the turtle's y coordinate
	
	Example (for a Turtle instance named turtle):
	>>> reset()
	>>> turtle.left(60)
	>>> turtle.forward(100)
	>>> print(turtle.ycor())
	86.6025403784