TURTLE GRAPHICS

TURTLE

- Graphics is the discipline that underlies the representation and display of geometric shapes in two- and three-dimensional space, as well as image processing.
- A Turtle graphics toolkit provides a simple and enjoyable way to draw pictures in a window and gives you an opportunity to run several methods with an object.
- Imagine a turtle crawling on a piece of paper with a pen tied to its tail. Commands direct the turtle as it moves across the paper and tell it to lift or lower its tail, turn some number of degrees left or right, and move a specified distance.
- Whenever the tail is down, the pen drags along the paper, leaving a trail. In this manner, it is possible to program the turtle to draw pictures ranging from the simple to the complex.

TURTLE

- In the context of a computer, of course, the sheet of paper is a window on a display screen, and the turtle is an icon, such as an arrowhead.
- At any given moment in time, the turtle is located at a specific position in the window. This position is specified with (x, y) coordinates.
- The coordinate system for Turtle graphics is the standard Cartesian system, with the origin (0, 0) at the center of a window.
- The turtle's initial position is the origin, which is also called the home.
- An equally important attribute of a turtle is its heading, or the direction in which it currently faces

TURTLE

Heading	Specified in degrees, the heading or direction increases in value as the turtle turns to the left, or counterclockwise. Conversely, a negative quantity of degrees indicates a right, or clockwise, turn. The turtle is initially facing east, or 0 degrees. North is 90 degrees.
Color	Initially black, the color can be changed to any of more than 16 million other colors.
Width	This is the width of the line drawn when the turtle moves. The initial width is 1 pixel. (You'll learn more about pixels shortly.)
Down	This attribute, which can be either true or false, controls whether the turtle's pen is up or down. When true (that is, when the pen is down), the turtle draws a line when it moves. When false (that is, when the pen is up), the turtle can move without drawing a line.

Table 7-1 Some attributes of a turtle

TURTLE

- A turtle will draw when it is moved if its pen is currently down, but it will simply move without drawing when its pen is currently up.
- Two other important classes used in Python's Turtle graphics system are Screen, which represents a turtle's associated window, and Canvas, which represents the area in which a turtle can move and draw lines.
- A canvas can be larger than its window, which displays just the area of the canvas visible to the human user.

TURTLE OPERATIONS

`t = Turtle()`	Creates a new **`Turtle`** object and opens its window.
`t.home()`	Moves `t` to the center of the window and then points `t` east.
`t.up()`	Raises `t`'s pen from the drawing surface.
`t.down()`	Lowers `t`'s pen to the drawing surface.
`t.setheading(degrees)`	Points `t` in the indicated direction, which is specified in degrees. East is 0 degrees, north is 90 degrees, west is 180 degrees, and south is 270 degrees.
`t.left(degrees)` `t.right(degrees)`	Rotates `t` to the left or the right by the specified degrees.
`t.goto(x, y)`	Moves `t` to the specified position.
`t.forward(distance)`	Moves `t` the specified distance in the current direction.
`t.pencolor(r, g, b)` `t.pencolor(string)`	Changes the pen color of `t` to the specified RGB value or to the specified string, such as **`"red"`**. Returns the current color of `t` when the arguments are omitted.

TURTLE OPERATIONS

`t.fillcolor(r, g, b)` `t.fillcolor(string)`	Changes the fill color of `t` to the specified RGB value or to the specified string, such as `"red"`. Returns the current fill color of `t` when the arguments are omitted.
`t.begin_fill()` `t.end_fill()`	Enclose a set of turtle commands that will draw a filled shape using the current fill color.
`t.clear()`	Erases all of the turtle's drawings, without changing the turtle's state.
`t.width(pixels)`	Changes the width of `t` to the specified number of pixels. Returns `t`'s current width when the argument is omitted.
`t.hideturtle()` `t.showturtle()`	Makes the turtle invisible or visible.
`t.position()`	Returns the current position `(x, y)` of `t`.
`t.heading()`	Returns the current direction of `t`.
`t.isdown()`	Returns `True` if `t`'s pen is down or `False` otherwise.

INTERFACE

- The set of methods of a given class of objects is called its interface.
- This is another important idea in object-based programming. Programmers who use objects interact with them through their interfaces. T
- Thus, an interface should contain only enough information to use an object of a given class. This information includes method headers and documentation about the method's arguments, values returned, and changes to the state of the associated objects.
- The expression dir() lists the names of methods in a class's interface.

DRAWSQUARE

- To illustrate the use of some methods with a Turtle object, let's define a function named drawSquare.
- This function expects a Turtle object, a pair of integers that indicate the coordinates of the square's upper-left corner, and an integer that designates the length of a side.
- The function begins by lifting the turtle up and moving it to the square's corner point.
- It then points the turtle due south—270 degrees—and places the turtle's pen down on the drawing surface. Finally, it moves the turtle the given length and turns it left by 90 degrees, four times

```
def drawSquare(t, x, y, length):
    """Draws a square with the given turtle t, an upper-left
    corner point (x, y), and a side's length."""
    t.up()
    t.goto(x, y)
    t.setheading(270)
    t.down()
    for count in range(4):
        t.forward(length)
        t.left(90)
```

OBJECT INSTANTIATION AND THE TURTLE MODULE

- Before you use some objects, like a Turtle object, you must create them.
- To be precise, you must create an instance of the object's class.
- The process of creating an object is called instantiation.
- The syntax for instantiating a class and assigning the resulting object to a variable is the following:

```
<variable name> = <class name>(<any arguments>)
```

- The expression on the right side of the assignment, also called a constructor, resembles a function call.
- The constructor can receive as arguments any initial values for the new object's attributes, or other information needed to create the object
- The Turtle class is defined in the turtle module (note carefully the spelling of both names).
- The following code imports the Turtle class for use in a session:
- >>> from turtle import Turtle

- The next code segment creates and returns a Turtle object and opens a drawing window. >>> t = Turtle()

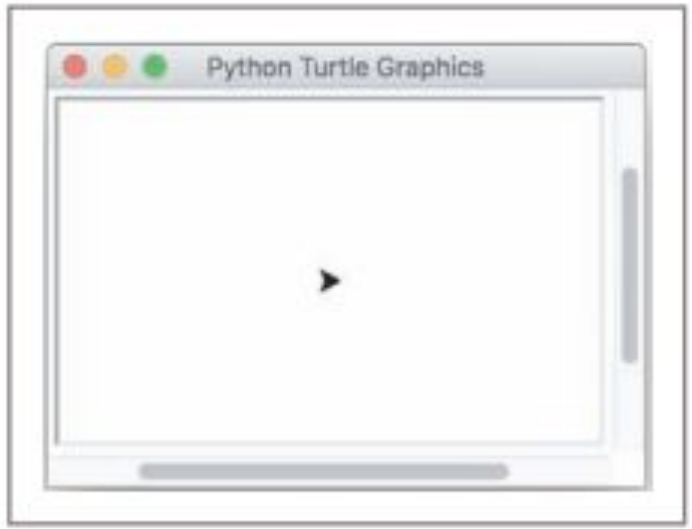
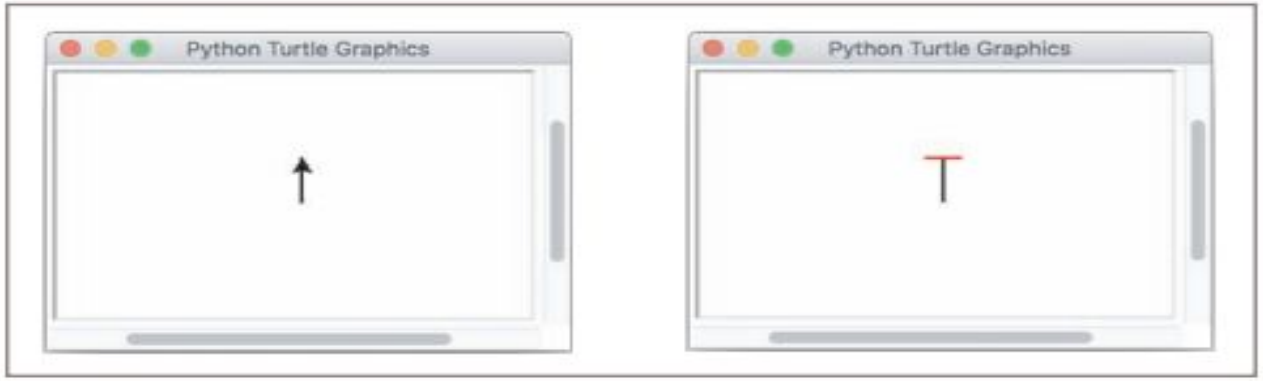


Figure 7-1 Drawing window for a turtle

- Let's continue with the turtle named t, and tell it to draw the letter T, in black and red
- It begins at the home position, accepts a new pen width of 2, turns 90 degrees left, and moves north 30 pixels to draw a black vertical line.
- Then it turns 90 degrees left again to face west, picks its pen up, and moves 10 pixels.
- The turtle next turns to face due east, changes it color from black to red, puts its pen down, and moves 20 pixels to draw a horizontal line.
- Finally, we hide the turtle

```
>>> t.width(2)              # For bolder lines
>>> t.left(90)              # Turn to face north
>>> t.forward(30)           # Draw a vertical line in black
>>> t.left(90)              # Turn to face west
>>> t.up()                  # Prepare to move without drawing
>>> t.forward(10)           # Move to beginning of horizontal line
>>> t.setheading(0)         # Turn to face east
>>> t.pencolor("red")
>>> t.down()                # Prepare to draw
>>> t.forward(20)           # Draw a horizontal line in red
>>> t.hideturtle()          # Make the turtle invisible
```



DRAWING TWO-DIMENSIONAL SHAPES

```
def square(t, length):
    """Draws a square with the given length."""
    for count in range(4):
        t.forward(length)
        t.left(90)
```

POLYGON

```
def hexagon(t, length):
    """Draws a hexagon with the given length."""
    for count in range(6):
        t.forward(length)
        t.left(60)
```

RADIAL PATTERNS

```
def radialHexagons(t, n, length):
    """Draws a radial pattern of n hexagons with the given length."""
    for count in range(n):
        hexagon(t, length)
        t.left(360 / n)
```