Special Topics in Computer Science- CSC 4992

Introduction to Graphics

Bitmapped Display

- Think of the monitor screen as a rectangular grid that is divided up into tiny square pieces

- Each square piece is called a *picture element* or *pixel*

- Each pixel can be painted
  - black or white (monochrome)
  - Several shades of gray (grayscale)
  - Several (perhaps millions of) colors

- The values of the colors are represented as bits in a structure called a *bitmap*

Representing Colors: Monochrome

- Black and white: 0 = black, 1 = white, so just a single bit
- 8, 16, 128, or 256 shades of gray: 3, 4, 7, or 8 bits (why?)
- In general, N bits allows 2^N distinct color values

Representing Colors: RGB

- The *RGB system* composes each color from red, green, and blue components
- Each component is an integer in the range 0..255
- 0 means the total absence of a color component
- 255 means the highest saturation of a color component
- 256 * 256 * 256 = 16,777,216 distinct color values

Representing Colors: RGB

Think of each color value as a tuple of integers of the form (<*r*>, <*g*>, <*b*>)

RGB Value	**Color**
(0, 0, 0)	Black
(255, 0, 0)	Red
(0, 255, 0)	Green
(0, 0, 255)	Blue
(127, 127, 127)	Medium Gray
(255, 255, 255)	White

Turtle Graphics

- The turtle is an object that has a position in a drawing window

- This object can be told to turn a number of degrees, move a given distance, move to a new position, and change its color and line width

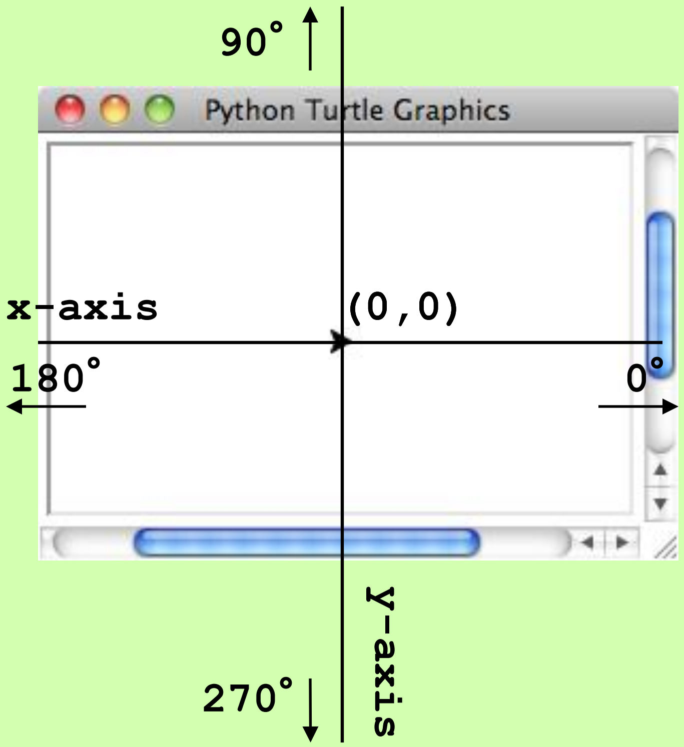
- If the turtle's pen is down, it draws a line; otherwise, it just moves without drawing

The **`turtle`** Module

- A standard Python module
- Includes a **`Turtle`** type with methods for getting turtle objects to do things

The Turtle Drawing Window

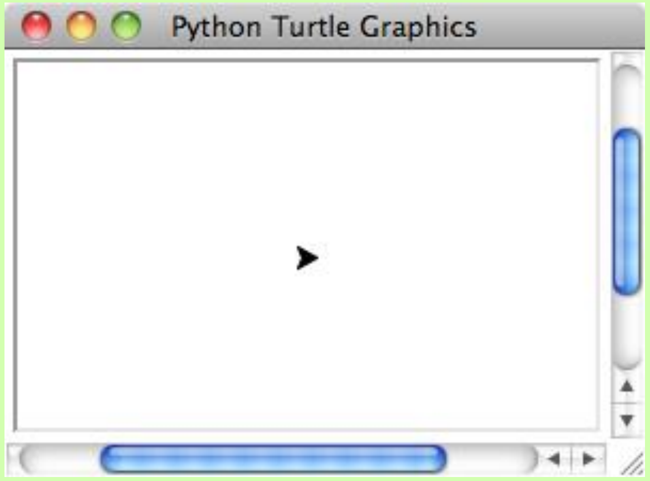
- The turtle appears as an icon
- Initial position: (0, 0)
- Initial direction: East (0°)
- Color: black
- Line width: 1 pixel
- Pen: down (ready to draw)



Instantiating a Turtle

```
>>> from turtle import Turtle

>>> sleepy = Turtle()
```

Pops up a drawing window whose drawing area has a default height and width of 500 pixels

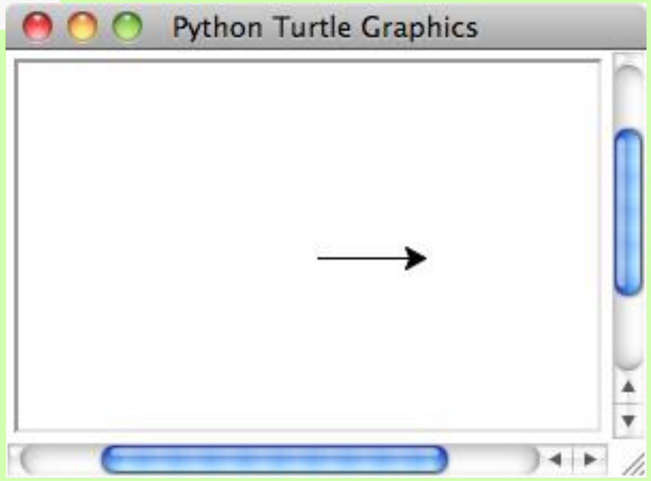


Some **`Turtle`** Methods

```
home()                 # Return to (0, 0) and 0°  (east)

down()                 # Enable drawing

up()                   # Disable drawing

forward(distance)      # Go distance in current direction

goto(x, y)             # Go to (x, y)

left(degrees)          # Add degrees to current direction

setheading(degrees)    # Make degrees the new direction

width(width)           # Set width of pen in pixels

pencolor(r, g, b)      # Red, green, blue compound
```

Move a Given Distance

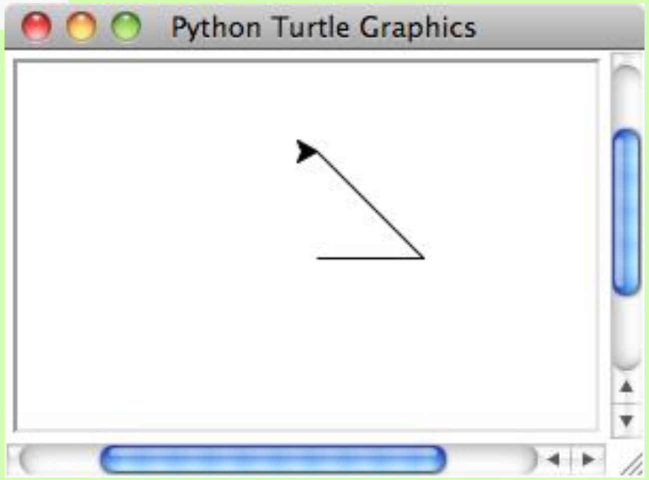
```
>>> from turtle import Turtle
>>> sleepy = Turtle()

>>> sleepy.forward(50)
```



Move 50 pixels in the current direction,
which is 0° (east)

Move to a Given Position

```
>>> from turtle import Turtle
>>> sleepy = Turtle()

>>> sleepy.goto(0, 50)
```

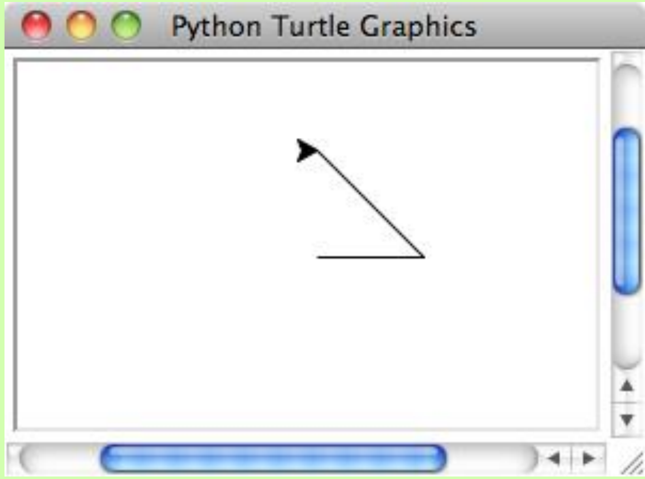


Move to location (0, 50)

Print the State of the Turtle

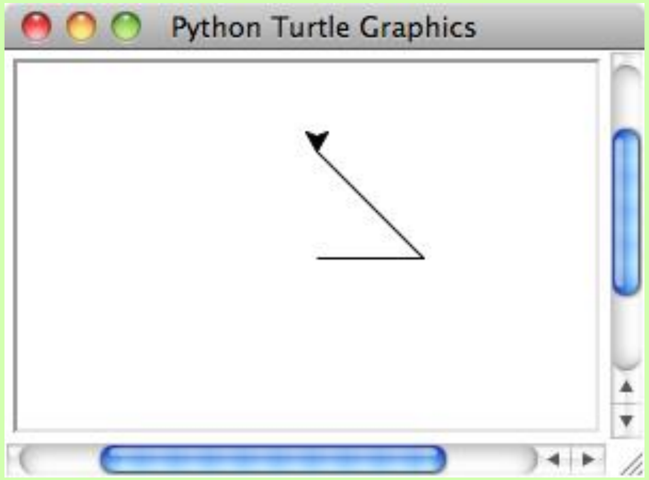
```
>>> from turtle import Turtle
>>> sleepy = Turtle()

>>> sleepy.goto(0, 50)
>>> print(sleepy.position(), sleepy.pencolor(), \
          sleepy.heading(), sleepy.isdown())
 (0.00,50.00) black 0.0 True
```

The ***state*** of an object includes the values of its attributes at any given time



Set the Direction

```
>>> sleepy.setheading(270)
```

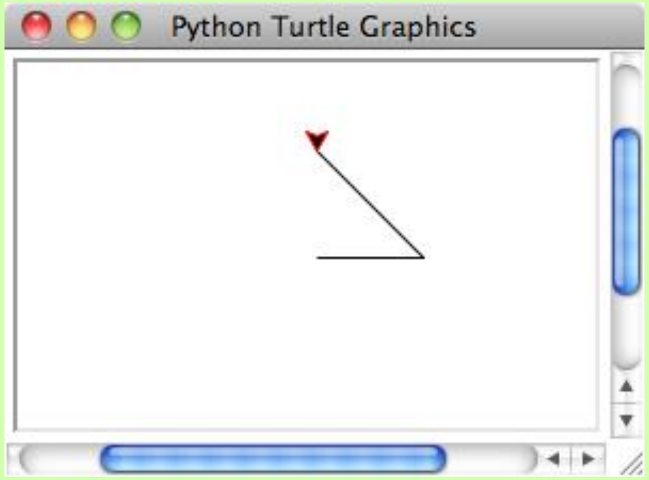


270° is due south

The turtle’s icon changes direction

Change the Pen’s Color

```
>>> sleepy.setheading(270)
>>> sleepy.pencolor(255, 0, 0)
```



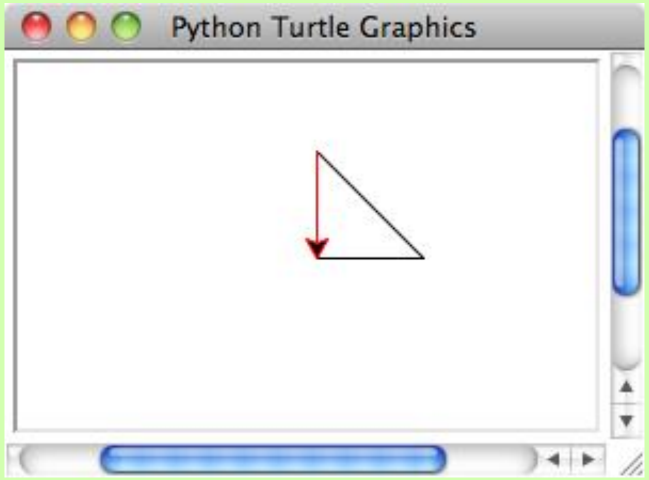
RGB value for the brightest red

Alternatively, could use “red”

Changes the icon’s outline color

Move Due South

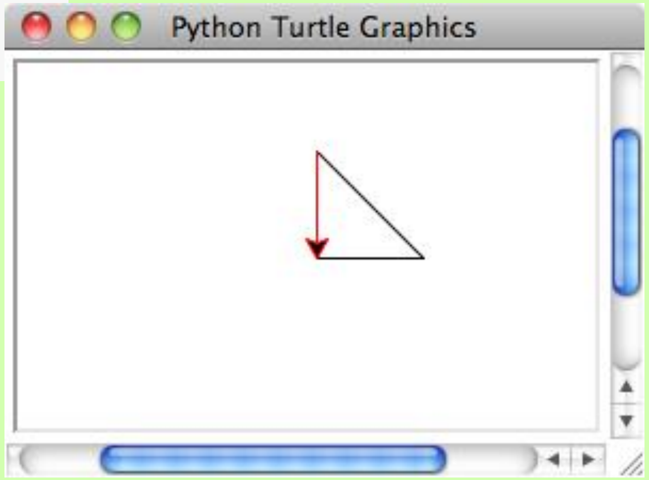
```
>>> sleepy.setheading(270)
>>> sleepy.pencolor(255, 0, 0)
>>> sleepy.forward(50)
```



Returns to the origin, drawing a red line

Pick Up the Turtle's Pen

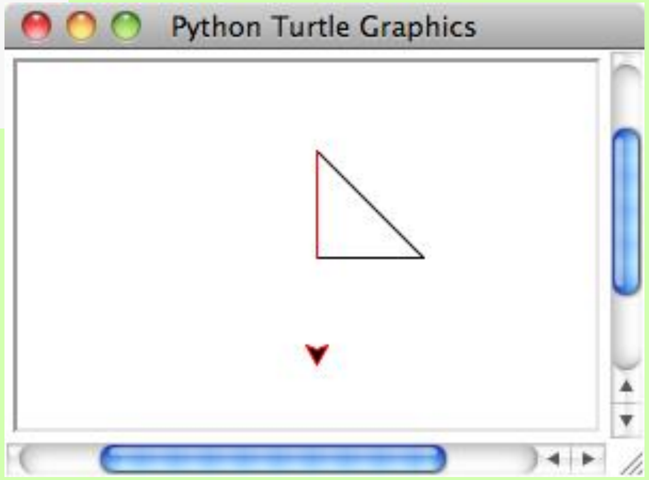
```
>>> sleepy.setheading(270)
>>> sleepy.pencolor(255, 0, 0)
>>> sleepy.forward(50)

>>> sleepy.up()
```



Won't draw when moved now

Move Without Drawing

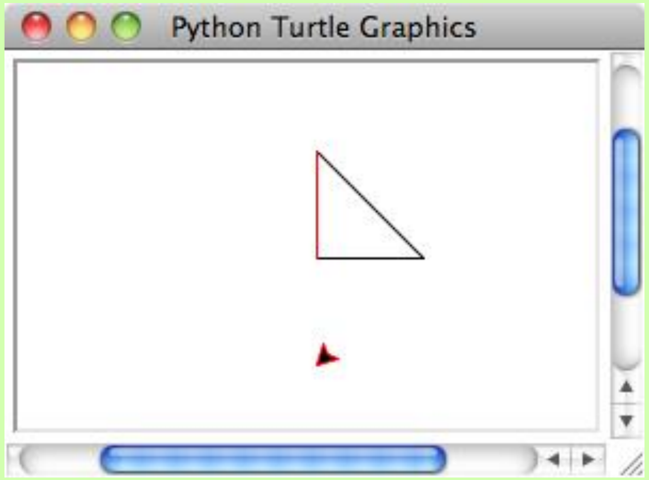
```
>>> sleepy.setheading(270)
>>> sleepy.pencolor(255, 0, 0)
>>> sleepy.forward(50)

>>> sleepy.up()
>>> sleepy.forward(50)
```



Won’t draw when moved now

Turn a Given Number of Degrees

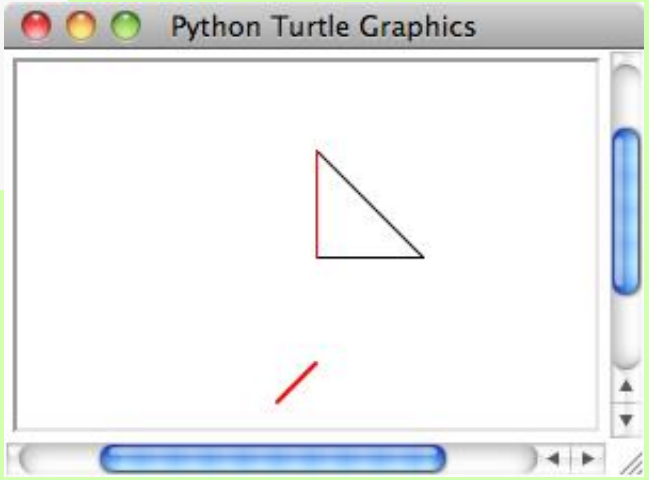
```
>>> sleepy.up()
>>> sleepy.forward(10)
>>> sleepy.right(45)
```



Subtract 45 ° from 0 ° , turning clockwise

Reset to Draw Again

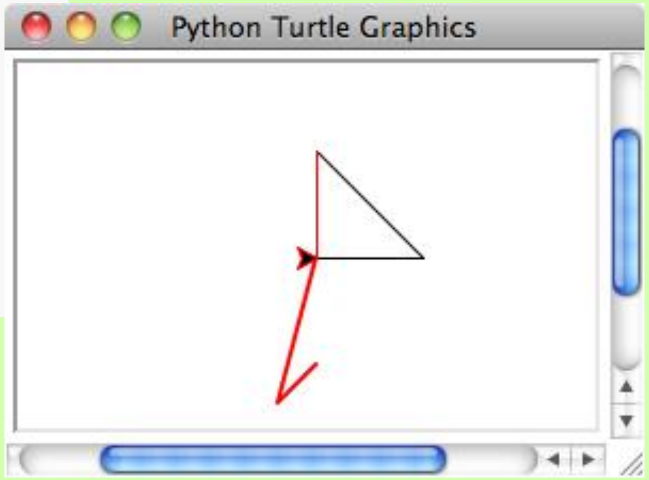
```
>>> sleepy.up()
>>> sleepy.forward(10)
>>> sleepy.right(45)
>>> sleepy.width(2)
>>> sleepy.hideturtle()
>>> sleepy.down()
>>> sleepy.forward(25)
```



- Double the pen width
- Hide the turtle's icon
- Place the pen down
- Move forward

Go Home, Sleepy, Go Home

```
>>> sleepy.up()
>>> sleepy.forward(10)
>>> sleepy.right(45)
>>> sleepy.width(2)
>>> sleepy.hideturtle()
>>> sleepy.down()
>>> sleepy.forward(25)
>>> sleepy.home()
>>> sleepy.showturtle()
```



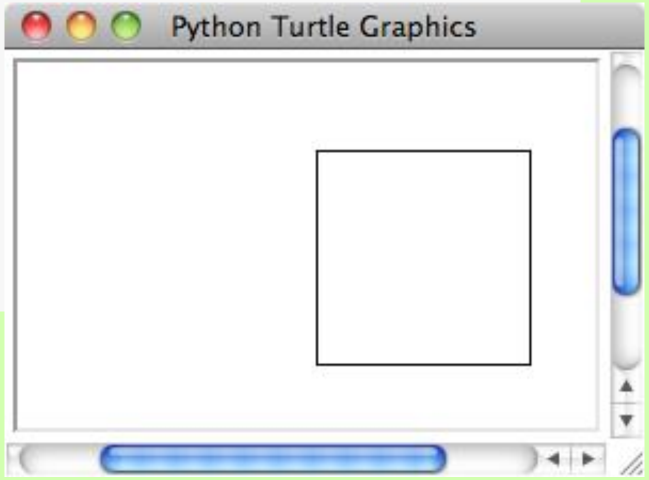
Move to the origin and face east

Define Some Functions

```
def drawSquare(t, x, y, length):
    """Use t to draw a square with corner point (x, y)
    and length."""
    t.up()
    t.goto(x, y)
    t.setheading(270)
    t.down()
    for count in range(4):
        t.forward(length)
        t.left(90)
```

Define Some Functions

```
def drawSquare(t, x, y, length):
    """Use t to draw a square with corner point (x, y)
    and length."""
    t.up()
    t.goto(x, y)
    t.setheading(270)
    t.down()
    for count in range(4):
        t.forward(length)
        t.left(90)
```



```
from turtle import Turtle
sleepy = Turtle()
sleepy.hideturtle()
drawSquare(sleepy, 0, 50, 100)
```

Define Some Functions

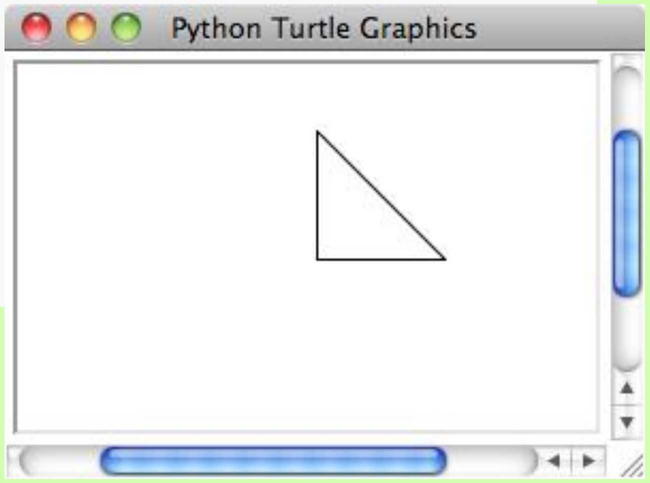
```
def drawPolygon(t, vertices):
    """Use t to draw a polygon from a list of vertices.
    The list has the form [(x_1, y_1), …, (x_n, y_n)]."""
    t.up()
    (x, y) = vertices[0]
    t.goto(x, y)
    t.down()
    for (x, y) in vertices:
        t.goto(x, y)
    (x, y) = vertices[0]
    t.goto(x, y)
```

Note that **`(x, y) = <a tuple>`** allows the values in a tuple to be assigned to distinct variables

Define Some Functions

```
def drawPolygon(t, vertices):
    """Use t to draw a polygon from a list of vertices.
    The list has the form [(x₁, y₁), …, (xₙ, yₙ)]."""
    t.up()
    (x, y) = vertices[0]
    t.goto(x, y)
    t.down()
    for (x, y) in vertices:
        t.goto(x, y)
    (x, y) = vertices[0]
    t.goto(x, y)
```

```
from turtle import Turtle
sleepy = Turtle()
sleepy.hideturtle()
drawPolygon(spleepy, [(0,0), (0, 60),
                      (60, 0)])
```



Define Some Functions

```
def drawTriangle(t, x1, y1, x2, y2, x3, y3):
    """Draws a triangle with the given vertices."""
    drawPolygon(t, [(x1, y1), (x2, y2), (x3, y3)])
```

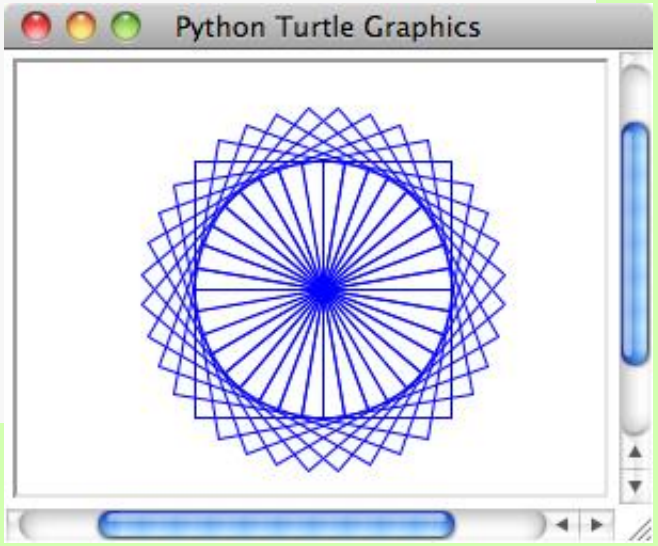
```
def drawSquare(t, x, y, length):
    """Draws a square with corner point (x, y)
    and length."""
    drawPolygon(t, [(x, y),
                    (x + length, y),
                    (x + length, y - length),
                    (x, y - length)])
```

Define Some Functions

```python
def drawFlower(t, x, y, width):
    """Draws a flower at (x, y)."""
    t.up()
    t.goto(x, y)
    t.down()
    for petals in range(36):
        for side in range(4):
            t.forward(width)
            t.left(90)
        t.left(90)
```

Define Some Functions

```
def drawFlower(t, x, y, width):
    """Draws a flower at (x, y)."""
    t.up()
    t.goto(x, y)
    t.down()
    for petals in range(36):
        for side in range(4):
            t.forward(width)
            t.left(90)
        t.left(10)
```



```
from turtle import Turtle
sleepy = Turtle()
sleepy.hideturtle()
drawFlower(sleepy, 0, 0, 60)
```