Function Parameters

Def drawEye(size):

Turtle.down()

Turtle.circle(size)

Turtle.up()

Size is a function parameter

To call drawEye we must now provide a value for size

In the function definition any where size is used will be replaced with the value passed into the function call

Conditionals

They choose among code blocks

Code blocks are choose based on conditions

If statement

The most basic conditional

Example:

Def reactToLight(lightColor):

If lightColor is “red”:

Print(“stop”)

Elif lightColor is “green”:

Print(“go”)

Elif lightColor is “yellow”:

Print(“floor it!”)

Else:

Print(“WTF!”)

Making a face

Def drawMouth(shape):

If shape is “smile”:

#draw smile

Elif shape is “frown”:

#draw frown

Else:

#draw grimace

What do we need to make a face:

drawFace function

parameters, eye radius, and mouth shape

draw the border

draw the mouth with given shape

draw the nose

draw the eyes with given radius

Execution statement

drawFancyFace( mouthShape := “smile”, eyeRadius := 15 )

drawBorder()

do nothing # already at bottom of face

draw a circle of radius 100

do nothing # already at bottom of face

drawMouth( mouthShape := “smile” \_

move to center of mouth

# mouthShape == “smile”

Draw a line up-left of length 30

Move to center of mouth

Draw a line up-right of length 30

Move to center of mouth

Move to bottom of face

drawNose()

move to center of face

draw a triangle of side-length 3-

move to bottom of face

drawEyes( eyeRadius := 15 )

move to bottom of left eye

drawEye(eyeRadius := 15)

draw a circle of radius eyeRadius

drawEye(eyeRadius := 15)

draw a circle of radius eyeRadius

move to bottom of face

initWorld()

turtle.hideturtle()

turtle.update()

input()

turtle.bye

Recursive Tree

Recursion

Is the act of calling function within itself

Example:

Def countdown ( start ):  
 print(start)

Countdown (start – 1)

Fix:

Def countdown ( start ):

Print(start)

If start > 0:

Countdown (start – 1)

Segments == 1

Def drawTree1(size)

Forward(size)

Forward(-size)

drawTree2(size)

forward(size)

left(45)

forward(size/2)

forward(-size/2\_

right(45)

right(45)

forward(size/2)

forward(-size/2)

forward(-size)

left(45)

forward(-size)

Recursion

drawtree2(size)

forward(size)

left(45)

drawTree1(size/2)

right(45)

right(45)

drawTree1(size/2)

left(45)

forward(-size)

def drawtree3(size)

forward(size)

left(45)

drawTree2(size/2)

right(45)

right(45)

drawTree2(size/2)

left(45)

forward(-size)

drawTree(segments, size)

if segments == 1:

forward(size)

forward(-size)

else:

forward(size)

left(45)

drawTree(segments – 1, size/2)

right(45)

right(45)

drawTree(segments – 1, size/2)

left(45)

forward(-size)