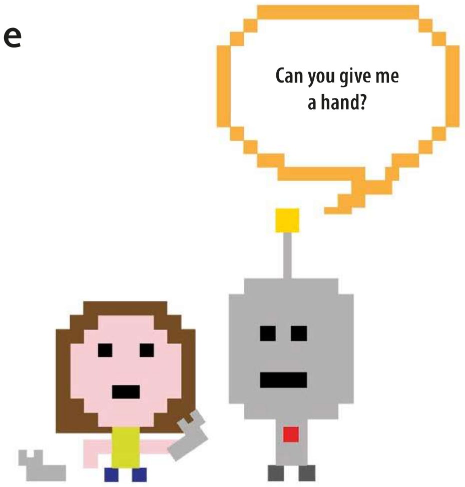
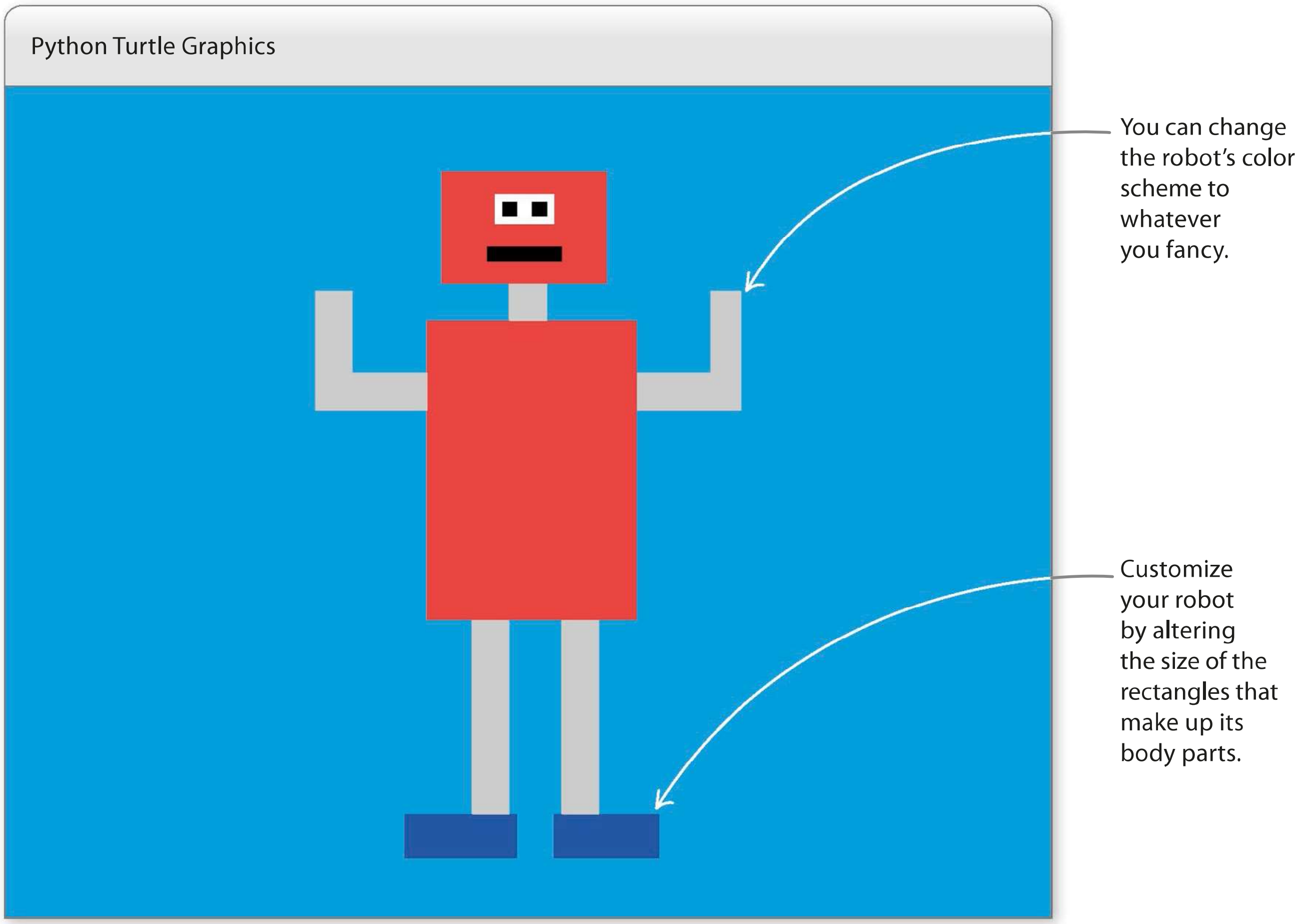
Robot Builder

Creating graphics in Python is easy. Python's `turtle` module lets you move a robot "turtle" around the screen, drawing pictures with a pen as it goes. In this project, you'll program the turtle to build more robots—or at least pictures of robots!



What happens

When you run the program, Python's turtle sets off, scuttling around the screen as it draws a friendly robot. Watch as it assembles the robot piece by piece, using different colors.


Python Turtle Graphics
You can change the robot's color scheme to whatever you fancy.
Customize your robot by altering the size of the rectangles that make up its body parts.

How it works

You'll start by writing a function that draws rectangles. Then you'll put the rectangles together to build the robot. You can change the size and color of the rectangles by altering the parameters you pass to the function. So you can have long, thin blocks for the legs, square ones for the eyes, and so on.

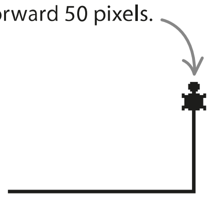
▽ Don't call me turtle!

Be careful never to name any of your `turtle` programs "turtle.py". If you do that, Python will get really confused and give you lots of error messages.

▽ Drawing with the turtle

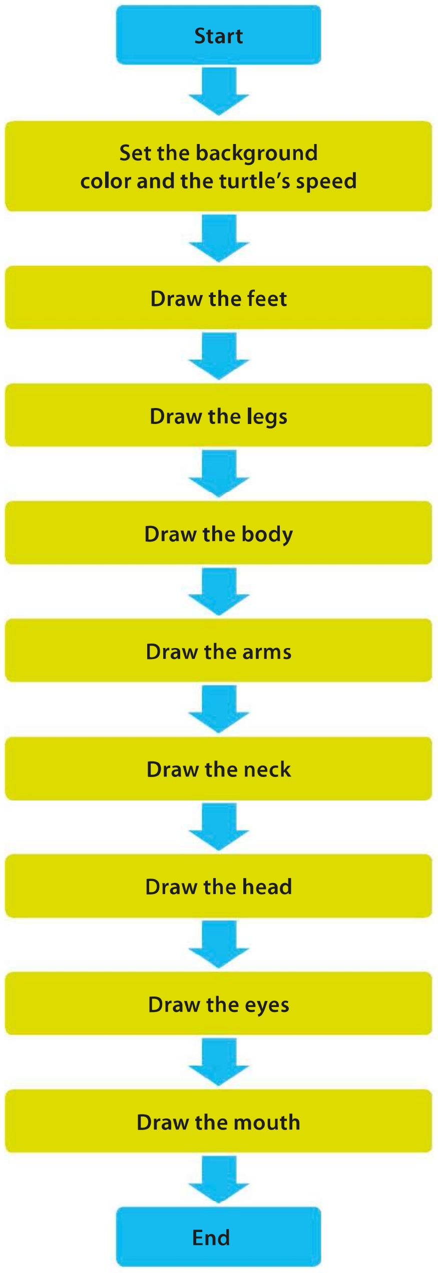
The `turtle` module allows you to control a pen-carrying robot turtle. By giving the turtle instructions on how it should move around the screen, you can draw different pictures and designs. You can also tell the turtle when to put the pen down and start drawing, or when to pull it up so it can move to a different part of the screen without leaving an untidy trail.

```
t.forward(100)
t.left(90)
t.forward(50)
```


The turtle moves forward 100 pixels, turns left 90 degrees, then moves forward 50 pixels.

▽ Robot Builder flowchart

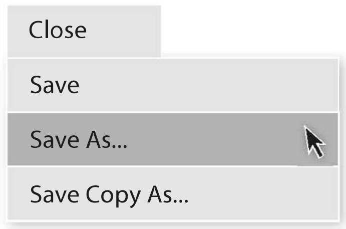
The flowchart shows how the code for this project fits together. First the program sets the background color and how fast the turtle moves. Then it draws the robot one part at a time, starting from its feet and moving up to its head.



Drawing rectangles

Let's begin by importing the `turtle` module and using it to create a function that draws rectangles.

1 Create a new file
Open IDLE and create a new file. Save it as "robot_builder.py".



2 Import the Turtle module
Type this line at the top of your program. The command `import turtle as t` lets you use functions from the `turtle` module without having to type "turtle" in full each time. It's like calling someone whose name is Benjamin "Ben" for short.

```
import turtle as t
```

This gives the Turtle module the nickname "t".

3 Create a rectangle function
Now make the function to draw the blocks that you're going to use to build your robot. The function has three parameters: the length of the horizontal side; the length of the vertical side; and color. You'll use a loop that draws one horizontal side and one vertical side each time it runs, and you'll make it run twice. Put this rectangle function under the code you added in Step 2.

Like all programming languages, Python uses the US spelling "color".

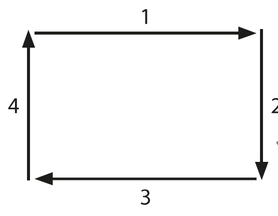
```
def rectangle(horizontal, vertical, color):
    t.pendown()
    t.pensize(1)
    t.color(color)
    t.begin_fill()
    for counter in range(1, 3):
        t.forward(horizontal)
        t.right(90)
        t.forward(vertical)
        t.right(90)
    t.end_fill()
    t.penup()
```

Put the turtle's pen down to start drawing.

Using `range(1, 3)` makes the loop run twice.

This block draws the rectangle.

Pull the turtle's pen back up to stop drawing.

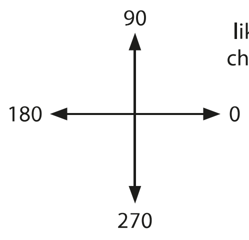


The turtle draws the sides in the order shown here.

EXPERT TIPS

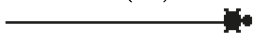
Turtle mode

You'll be using the turtle in its standard mode. This means the turtle starts off facing the right side of the screen. If you set the heading (another word for direction) to 0, it will face right. Setting the heading to 90 makes it point to the top of the screen, 180 points it to the left, and 270 makes it point to the bottom of the screen.



The turtle normally looks like an arrowhead. This line changes it to a turtle shape.

```
t.shape('turtle')
t.setheading(0)
t.forward(80)
```



EXPERT TIPS

Turtle speed

You can control how fast the turtle draws by using the `t.speed()` command to set its speed to one of these values: "slowest", "slow", "normal", "fast", and "fastest".

4 Set the background

Next get the turtle ready to start drawing, and set the background color of the window. You need the turtle to start with its pen up so that it doesn't draw lines until you want it to. It will only begin to draw when it reaches the robot's feet (Step 5). Type the following code under the code you added in Step 3.

Pull the turtle's pen up.

```
t.penup()
t.speed('slow')
t.bgcolor('Dodger blue')
```

Set the turtle's speed to slow.

Make the background of the window "Dodger blue".

Building the robot

Now you're ready to start building the robot. You're going to make it piece by piece, starting with the feet and working your way up. The whole robot will be made using rectangles of different sizes and colors, each drawn from a different starting point in the Turtle window.

5 Draw the feet

You need to move the turtle to where you want to start drawing the first foot, and then use your rectangle function to draw it. You'll need to do the same for the second foot. Type these lines under the code you added in Step 4, then run the program to see your robot's feet appear.

This comment indicates which part of the robot you're drawing.

```
# feet
t.goto(-100, -150)
rectangle(50, 20, 'blue')
t.goto(-30, -150)
rectangle(50, 20, 'blue')
```

Move the turtle to position x = −100, y = −150.

Use the rectangle function to draw a blue rectangle 50 wide and 20 high.

EXPERT TIPS

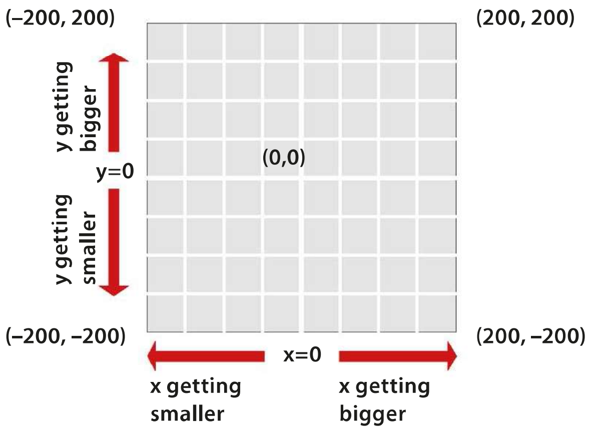
Comments

You'll notice that there are several lines in this program that start with a # symbol. The words following the # are a comment, added to make the code easier for users to read and understand. Python knows that it should ignore them.

EXPERT TIPS

Turtle coordinates

Python will adjust the Turtle window to fit your screen, but let's use an example that's 400 pixels by 400 pixels. Python uses coordinates to identify all the places in the window where the turtle could be. This means that every place on the window can be found by using two numbers. The first number, the x coordinate, shows how far to the left or right of the center the turtle is. The second number, the y coordinate, shows how far up or down from the center it is. Coordinates are written in parentheses, with the x coordinate first, like this: (x, y).



6 Draw the legs

The next bit of the program makes the turtle move to where it will start drawing the legs. Type these lines under the code you added in Step 5. Now run the code again.

```
# legs
t.goto(-25, -50)
rectangle(15, 100, 'grey')
t.goto(-55, -50)
rectangle(-15, 100, 'grey')
```

The turtle moves to position x = –25, y = –50.

Draw the left leg.

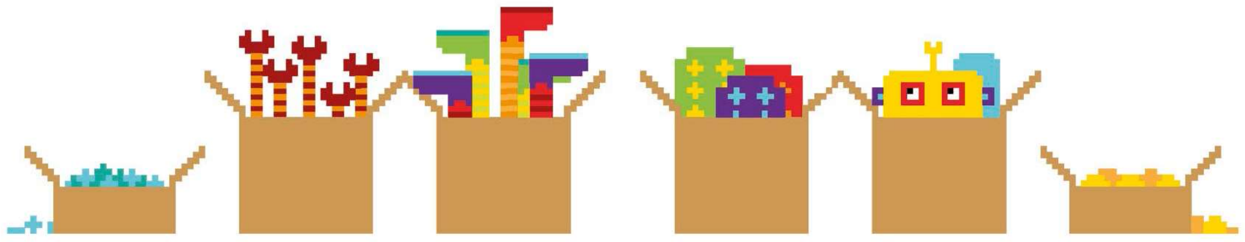
Draw the right leg.

7 Draw the body

Type this code under the code you added in Step 6. Run the program and you should see the body appear.

```
# body
t.goto(-90, 100)
rectangle(100, 150, 'red')
```

Draw a red rectangle 100 across and 150 down.



8 Draw the arms

Each arm is drawn in two parts: first the upper arm, from the robot's shoulder to its elbow; then the lower arm, from the elbow to the wrist. Type this below the code you added in Step 7, then run it to see the arms appear.

```
# arms
t.goto(-150, 70)
rectangle(60, 15, 'grey')
t.goto(-150, 110)
rectangle(15, 40, 'grey')

t.goto(10, 70)
rectangle(60, 15, 'grey')
t.goto(55, 110)
rectangle(15, 40, 'grey')
```

Upper right arm

Lower right arm

Upper left arm

Lower left arm

9 Draw the neck

Time to give your robot a neck. Type these neck-drawing commands below the code you added in Step 8.

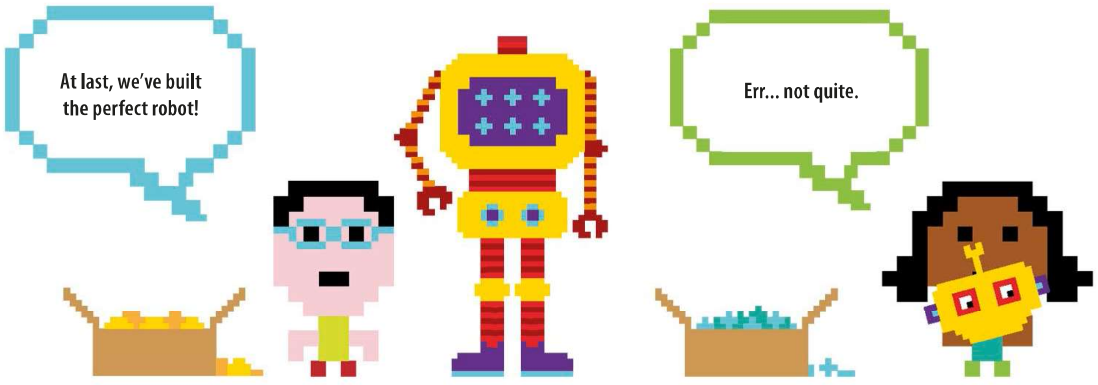
```
# neck
t.goto(-50, 120)
rectangle(15, 20, 'grey')
```

10 Draw the head

Oops—you've drawn a headless robot! To give your poor robot a head, type these commands below the code you added in Step 9.

```
# head
t.goto(-85, 170)
rectangle(80, 50, 'red')
```

Don't forget to save your work.



11 Draw the eyes

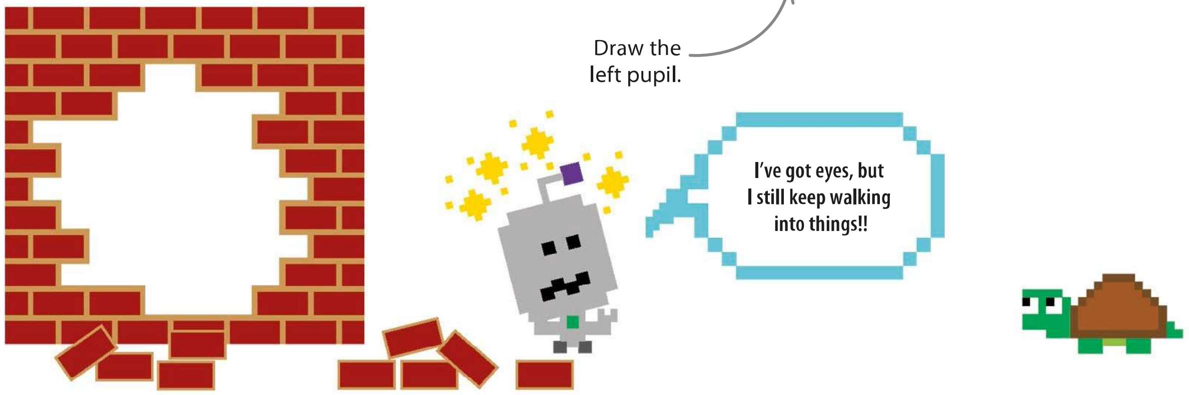
Let's add some eyes so that the robot can see where it's going. To do this, you'll draw a large white rectangle with two smaller squares inside it (for pupils). You don't have to write a new function to draw squares, since a square is a rectangle with all its sides the same length. Insert these commands under the code you added in Step 10.

```
# eyes
t.goto(-60, 160)
rectangle(30, 10, 'white')
t.goto(-55, 155)
rectangle(5, 5, 'black')
t.goto(-40, 155)
rectangle(5, 5, 'black')
```

Draw the white part of the eyes.

Draw the right pupil.

Draw the left pupil.



12 Draw the mouth

Now give the robot a mouth. Type these commands under the code you added in Step 11.

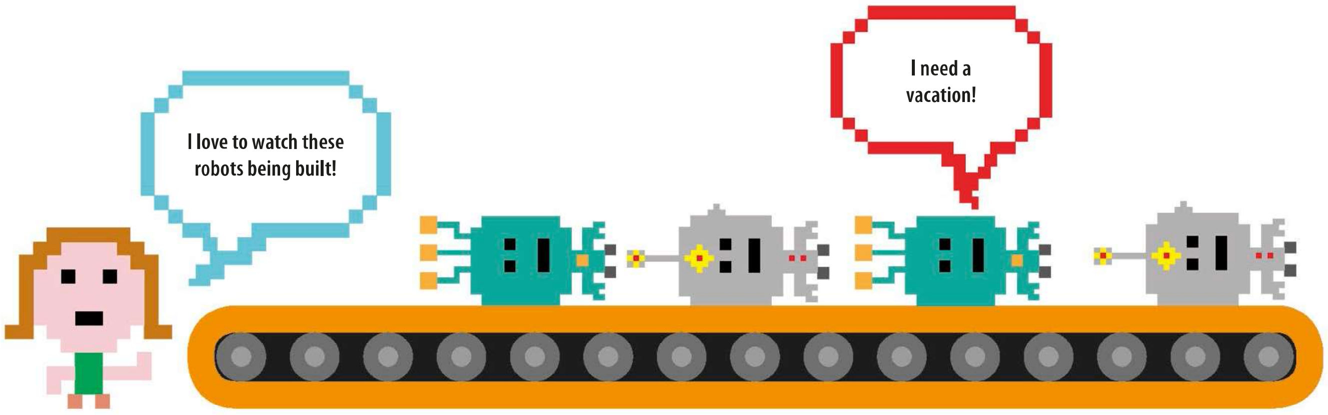
```
# mouth
t.goto(-65, 135)
rectangle(40, 5, 'black')
```

13 Hide the turtle

Finally, hide the turtle so it doesn't look odd sitting on the robot's face. Type this line after the code you added in Step 12. Run the program to see the whole robot being built.

```
t.hideturtle()
```

This makes the turtle invisible.



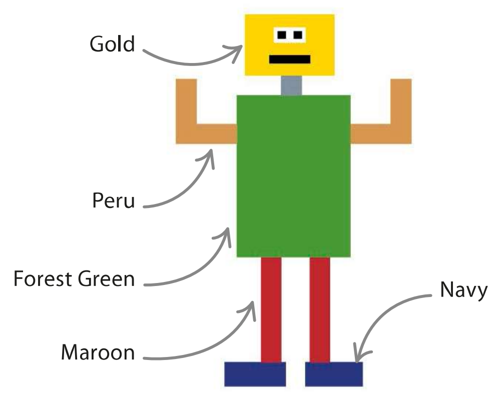
Hacks and tweaks

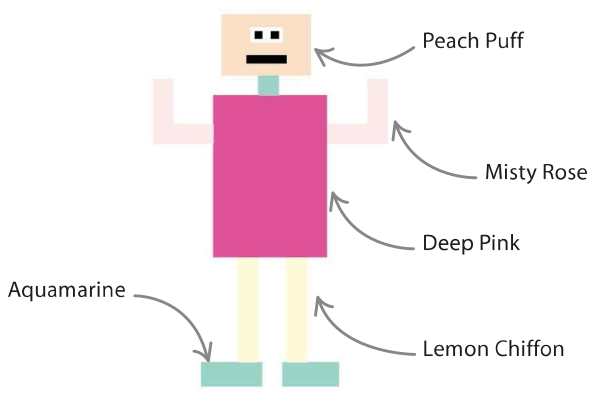
Now your project is up and running, here are some ideas for modifying the code so you can customize the robots you build.

▽ Change the colors

The robot you've created is fairly colorful, but there's definitely room for improvement. You could change the code to build a robot that matches the colors of your room or your favourite football team's shirt, or create one that's totally multicolored! On the right are some colors the turtle recognizes.

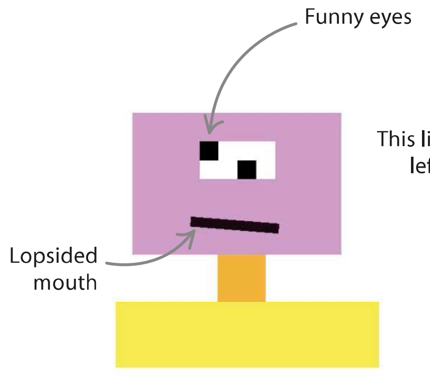
🟩 Lawn Green	⬜ Seashell	🟦 Blue
🟪 Purple	🟦 Light Blue	🟨 Yellow
🟨 Goldenrod	🟥 Hot Pink	⬜ Thistle





▷ Change the face

You can change the expression on the robot's face by rearranging its features. To give it wonky eyes and mouth, use the code on the right.



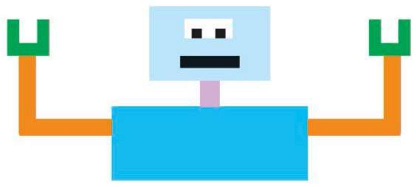
```
# eyes
t.goto(-60, 160)
rectangle(30, 10, 'white')
t.goto(-60, 160)
rectangle(5, 5, 'black')
t.goto(-45, 155)
rectangle(5, 5, 'black')

# mouth
t.goto(-65, 135)
t.right(5)
rectangle(40, 5, 'black')
```

This line moves the robot's right pupil, so it looks like the robot is rolling its eyes.

This line moves the left pupil down.

The turtle turns right slightly, which makes the mouth slope.

▷ **A helping hand**
Add this code to give your robot U-shaped gripping hands. You can reshape the hands to look like hooks, pincers, or anything else you like. Let your imagination run wild and create your own version!



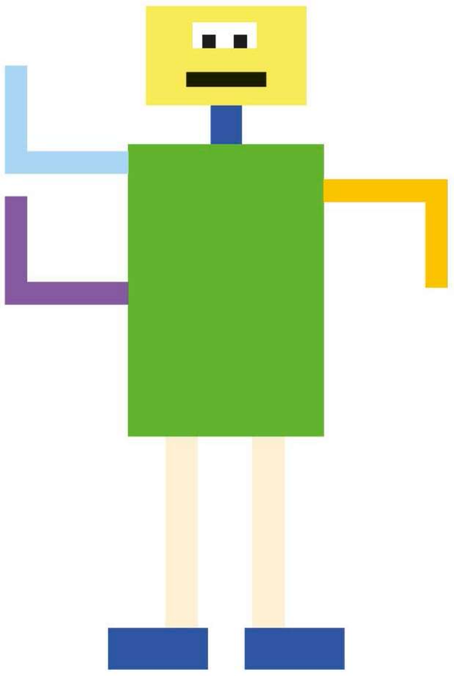
```
# hands
t.goto(-155, 130)
rectangle(25, 25, 'green')
t.goto(-147, 130)
rectangle(10, 15, t.bgcolor())
t.goto(50, 130)
rectangle(25, 25, 'green')
t.goto(58, 130)
rectangle(10, 15, t.bgcolor())
```

Draw a green square for the main part of the hand.

Draw a small rectangle in the background color to give the grip shape.

All-in-one arms

Drawing the arms in several parts makes it awkward to change their position or to add extra arms. In this hack, you'll write a function that draws an arm all in one go.



1 Create an arm function
First add this new function, which draws an arm shape and gives it color.

```
    t.end_fill()
    t.penup()

def arm(color):
    t.pendown()
    t.begin_fill()
    t.color(color)
    t.forward(60)
    t.right(90)
    t.forward(50)
    t.right(90)
    t.forward(10)
    t.right(90)
    t.forward(40)
    t.left(90)
    t.forward(50)
    t.right(90)
    t.forward(10)
    t.end_fill()
    t.penup()
    t.setheading(0)
```

This line colors in the shape formed by the following moves.

Set the color.

The turtle follows these commands to draw the arm.

Stop coloring in the shape.

Reset the turtle so it's facing right again.