Drawing Geometric Shapes with Turtles

You can use turtles to draw geometric shapes like squares, triangles, and rectangles.

```
from turtle import *
canvas = Screen()
canvas.setup(400,200)

sarah = Turtle()
sarah.forward(50)          # make sarah draw a square
sarah.left(90)
sarah.forward(50)
sarah.left(90)
sarah.forward(50)
sarah.left(90)
sarah.forward(50)
sarah.left(90)

canvas.exitonclick()

```

ActiveCode: 1 (a_6)

Run

Save Load

Here are some things to notice about this program:

- There are 360 degrees in a full circle. If you add up all the turns that a turtle makes, *no matter what steps occurred between the turns*, you can easily figure out if they add up to some multiple of 360. In this case sarah turns 90 + 90 + 90 + 90 = 360 degrees. This should convince you that sarah is facing in exactly the same direction as she was when she was first created.
- We could have left out the last turn for sarah, but that wouldn't be a great idea. If you're asked to draw a closed shape like a square or a rectangle, it is a good idea to complete all the turns and to leave the turtle back where it started, facing the same direction as it started in.
- Comments can be used to explain the result of a section of code. Here we use a comment to show that this section makes sarah draw a square.

Check your understanding

trl-1: True or False: If all of a turtles turns add up to a multiple of 360 it will be facing the same way it started.

- (•) a) True
- ( ) b) False

Check Me | Compare Me

Correct!! Even if you do something else like turn forward the turtle will end up facing the same way.

The **for** Loop

When sarah drew a square, it was quite tedious. She had to move then turn, move then turn, etc. etc. four times. A basic concept in computing is the ablity to repeat some code. In computer science, we refer to this as **looping** or **iteration**.

To draw a square we'd like to do the same thing four times — move the turtle forward some distance and turn 90 degrees. We previously used 8 lines of Python code to have sarah draw the four sides of a square. This next program does exactly the same thing but, with the help of the **for** statement, uses just three lines (not including the setup code). The code `[0,1,2,3]` creates a **list** with 4 items in it. The **for** statement will repeat the forward and left four times, one time for each item in the list. A **list** contains items in an order. Both the **forward** and **left** statements are in the **body of the loop**. The body of the loop is all of the statements that are indented further to the right than the **for** statement. A **for** statement must end with a colon: ':' as shown in the code below.

```
1  from turtle import *
2  wn = Screen()
3  sarah = Turtle()
4
5  for i in [0,1,2,3]:      #repeat four times
6      sarah.forward(50)
7      sarah.left(90)
8
9  wn.exitonclick()
10
```

ActiveCode: 2 (a_7)

Run

Save | Load

While writing less lines of code might be nice, it is not as important than the fact that we found a pattern here that can be repeated. If we want to change the number of times this pattern repeats we can just change the number of items in the list in the for statement.

In the previous example, there were four integers in the list. But, we could use any list that had 4 items. In the following example, each time through the loop the value of `aColor` will change to the next color in the list. We can use the value of `aColor` in the program to change the turtle's pen color.

```
1  from turtle import *
2  wn = Screen()
3  alex = Turtle()
4
5  for aColor in ["yellow", "red", "purple", "blue"]:
6     alex.color(aColor)
7     alex.forward(50)
8     alex.left(90)
9
10 wn.exitonclick()
11
```

ActiveCode: 3 (colorList)

Run

Save Load

Click on the forward button below to see how the loop changes the value of **aColor** each time through the loop. The **print()** statement prints the value in **aColor**. Notice that we used "wn" which is short for "window" to name our screen object this time instead of "canvas". It doesn't really matter what we name the screen, but it is useful to know what kind of thing it is, so using "canvas", "window", or "wn" makes sense.

```
1  for aColor in ["yellow", "red", "purple", "blue
2      print (aColor)
```

<< First < Back Step 1 of 9 Forward > Last >>

line that has just executed
next line to execute

Frames Objects

CodeLens: 1 (list1)

Check your understanding

trl-2: How does Python know which lines are in the body of the loop?

- (•) a) They are indented the same amount beneath the for statement.
- ( ) b) There is always exactly one line in the loop body.
- ( ) c) The loop body ends with a semi-colon (;) which is not shown in the code above.

Check Me | Compare Me

trl-3: How many items are in the following list: [0,1,2]

- ( ) a) 1
- ( ) b) 2
- (•) c) 3

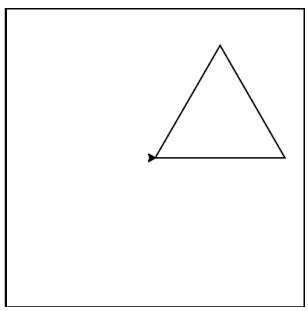
Check Me | Compare Me

trl-4: Does the for statement have to end with a colon?

- ( ) a) No
- (•) b) Yes

Check Me | Compare Me

Mixed up program



trl-5: The following program uses a turtle to draw a triangle as shown to the left, but the lines are mixed up. The program should do all necessary set-up and create the turtle. After that, iterate (loop) 3 times, and each time through the loop the turtle should go forward 175 pixels, and then turn left 120 degrees. After the loop, set the window to close when the user clicks in it.

Drag the blocks of statements from the left column to the right column and put them in the right order with the correct indention. Click on *Check Me* to see if you are right. You will be told if any of the lines are in the wrong order or are incorrectly indented.

Drag from here

Drop blocks here

```
from turtle import *
```

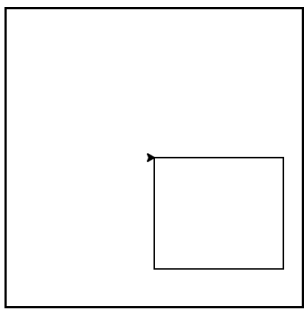
```
wn = Screen()
marie = Turtle()

# repeat 3 times
for i in [0,1,2]:
    marie.forward(175)
    marie.left(120)
wn.exitonclick()
```

Check Me | Reset

Perfect!

Mixed up program



trl-6: The following program uses a turtle to draw a rectangle as shown to the left, but the lines are mixed up. The program should do all necessary set-up and create the turtle. After that, iterate (loop) 2 times, and each time through the loop the turtle should go forward 175 pixels, turn right 90 degrees, go forward 150 pixels, and turn right 90 degrees. After the loop, set the window to close when the user clicks in it.

Drag the blocks of statements from the left column to the right column and put them in the right order with the correct indention. Click on *Check Me* to see if you are right. You will be told if any of the lines are in the wrong order or are incorrectly indented.

Drag from here

Drop blocks here

```
from turtle import *
wn = Screen()
carlos = Turtle()

# repeat 2 times
for i in [1,2]:
    carlos.forward(175)
```

```
carlos.right(90)
```

```
carlos.forward(150)
carlos.right(90)
```

```
wn.exitonclick()
```

Check Me | Reset

Perfect!

Time to Try Stuff Here is the program that draws a square again from above. Try to modify it to draw an octagon. How many times will you need to repeat the body of the loop? How much do you need to turn by each time? Remember that the total amount the turtle turns will equal 360 for any polygon.

```
1  from turtle import *
2  wn = Screen()
3  sarah = Turtle()
4
5  for i in [0,1,2,3]:      #repeat four times
6      sarah.forward(50)
7      sarah.left(90)
8
9  wn.exitonclick()
10
```

ActiveCode: 4 (a_8)

Run

Save | Load