

More useful turtle functions, loops

Oliver W. Layton

CS151: Computational Thinking: Visual Media

Lecture 05, Fall 2020

Friday September 4

Project 2

Goal: Draw a space-themed scene!

- Convert `lab2.py` into your "shape library" (`shapelib.py`): Holds all functions to draw simple and compound shapes.
- In shape library, make two more simple shapes (other than triangle — rectangles, pentagon, parallelogram, circle, etc.)
- In shape library, make two space-themed compound shape (e.g. rocket ship, planets, stars, etc.) by calling your simple shape functions and possibly adding in extra turtle commands.
- Arrange your shapes in space themed scene.

Project 2 guidelines

- Use best practices for code organization that we discussed on Wednesday (e.g. headers, docstrings, code comments, etc.).
- Use single empty lines in between blocks of code that accomplish subtasks to improve readability (analogous to new paragraphs when writing an essay).
- Use function names that make it clear what they are doing (e.g. `triangle()`).
- On rubric: **Top level code** means "main code" (not intended / part of a function).
- Grading: 30 point system (26 core project + 4 extensions) is a Colby CS department standard.

Review: Adding position and scale parameters to a compound shape function

Let's do more practice: make a house out of a triangle on top of a rectangle.

Let's also show how we can access functions in a different file.

Useful turtle functions

Filling shapes

Feel free to add color to your shapes that you draw with turtle!

```
# Set up the blue interior fill color
turtle.color('blue')
turtle.begin_fill()
# Draw a blue triangle
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
turtle.left(120)
turtle.forward(100)
turtle.left(120)
# Stop filling and change pen color back to black
turtle.end_fill()
turtle.color('black')
```

- If you change the color in a function, it is a good idea to change it back by the end of the same function.
- If you start filling in a function, it is a good idea to stop filling by the end of the same function.

Color list

[Here](#) is a list of color names that turtle can use.

Different edge and interior colors

- You can also select different edge and interior colors using `turtle.color()`. The syntax is:

`turtle.color(<edge color>, <interior color>).`

- For example, `turtle.color('yellow', 'blue')` would create a blue pentagon above with a yellow outline.
- Let's add color to our house.

Setting heading

Heading is the direction that the turtle is currently facing. Default, heading is 0 deg (east).

- Can set using `turtle.setheading(<angle>)` command.
- Example: calling `turtle.setheading(90)` will make your turtle face north, no matter what's current heading is.
- Get the current heading angle: `turtle.heading()`.
- Let's experiment with heading when drawing triangles.

Speeding things up

- Call `turtle.speed(0)`. The parameter is an int and the default is 6 (0 is fastest). It controls how fast the turtle moves on-screen.
- Values of 1, 3 are *slower* than normal.
- Values of 10, 0 are *faster* than normal.

For loops

Until now, we've had to write one line of code per command.

- Functions have helped reduce code, but for highly repetitive tasks, it is still annoying!
- Example: Draw a square, we need 4 sets of `forward` and `right` commands.
- With a **for loop**, we can *substantially* reduce the amount of code we need to write for repetitive tasks.

For loops in Swift Playgrounds

For loops in Python

In general, for loops have the following structure:

```
for <loop index> in range(<number of loop iterations>):  
    <loop body>
```

Example:

```
for i in range(10):  
    turtle.forward(50)
```