15-104 Introduction to Computing for Creative Practice

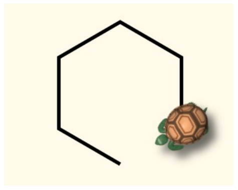
Fall 2021

26 Turtle Graphics

Instructor: Tom Cortina, tcortina@cs.cmu.edu, GHC 4117, 412-268-3514

1

Turtle Graphics



- Turtle Graphics is a conceptual model of computational thinking.
  - Based on the notion of a "first-person cursor"
  - Drawing is done by imagining a turtle moving about on the canvas with a pen.
    - We can tell the turtle to move forward, turn left, turn right, etc.
    - We can tell the turtle to raise or lower the pen.
    - All commands are given with respect to the turtle's orientation and state, as opposed to our coordinate system.

2

Historical Context

- This schema was developed in the 1960s by computer scientist and constructivist educator Seymour Papert.
  - Inspired by Jean Piaget's studies of children's learning and embodied cognition.
- Turtle Graphics is a key feature in the programming language Logo.
- Papert was interested in the use of technology to improve learning, particularly with young children.

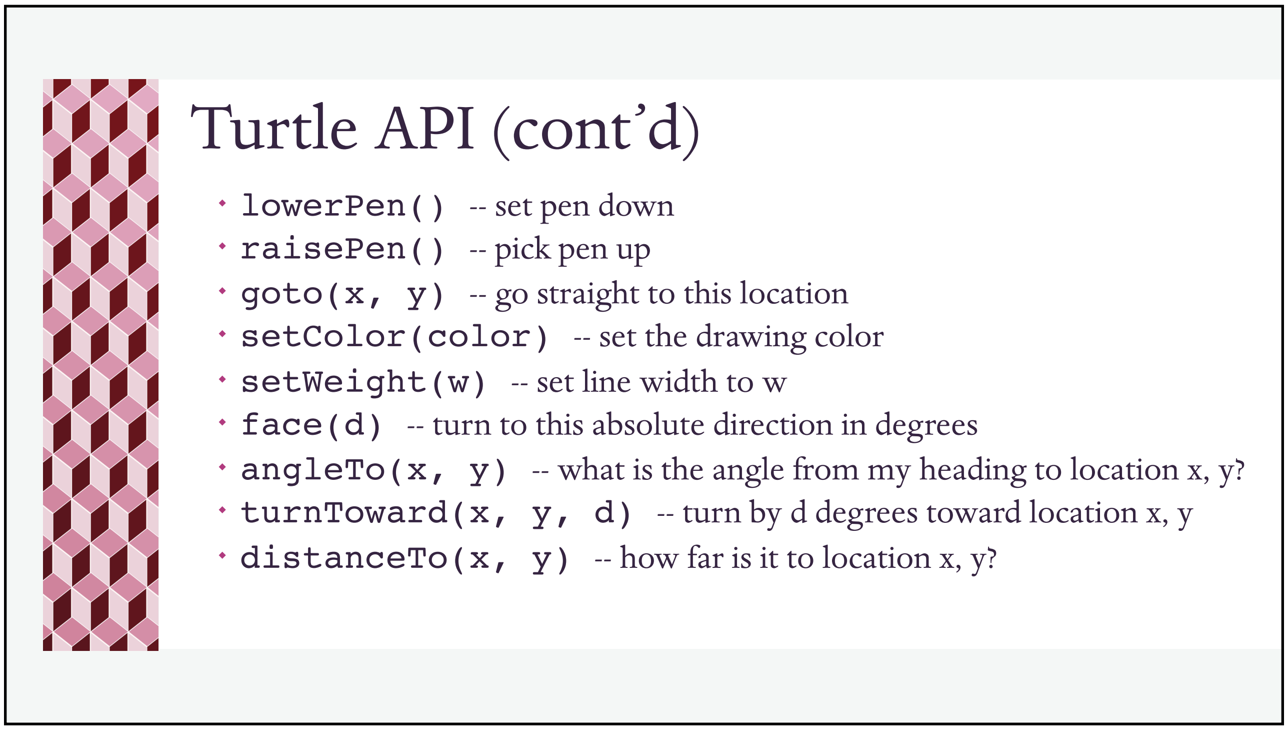
Papert (left) and Piaget (right)

3

Turtle API

- An Application Programmer's Interface (API) is a view of the methods (functions) of the object without seeing the details. The programmer can use the object just by knowing how to call the methods and what they return.
- The turtle has the following Application Programmer's Interface (API):
  - `makeTurtle(x, y)` -- make a turtle at x, y, facing right, pen down
  - `left(d)` -- turn left by d degrees
  - `right(d)` -- turn right by d degrees
  - `forward(p)` -- move forward by p pixels
  - `back(p)` -- move back by p pixels

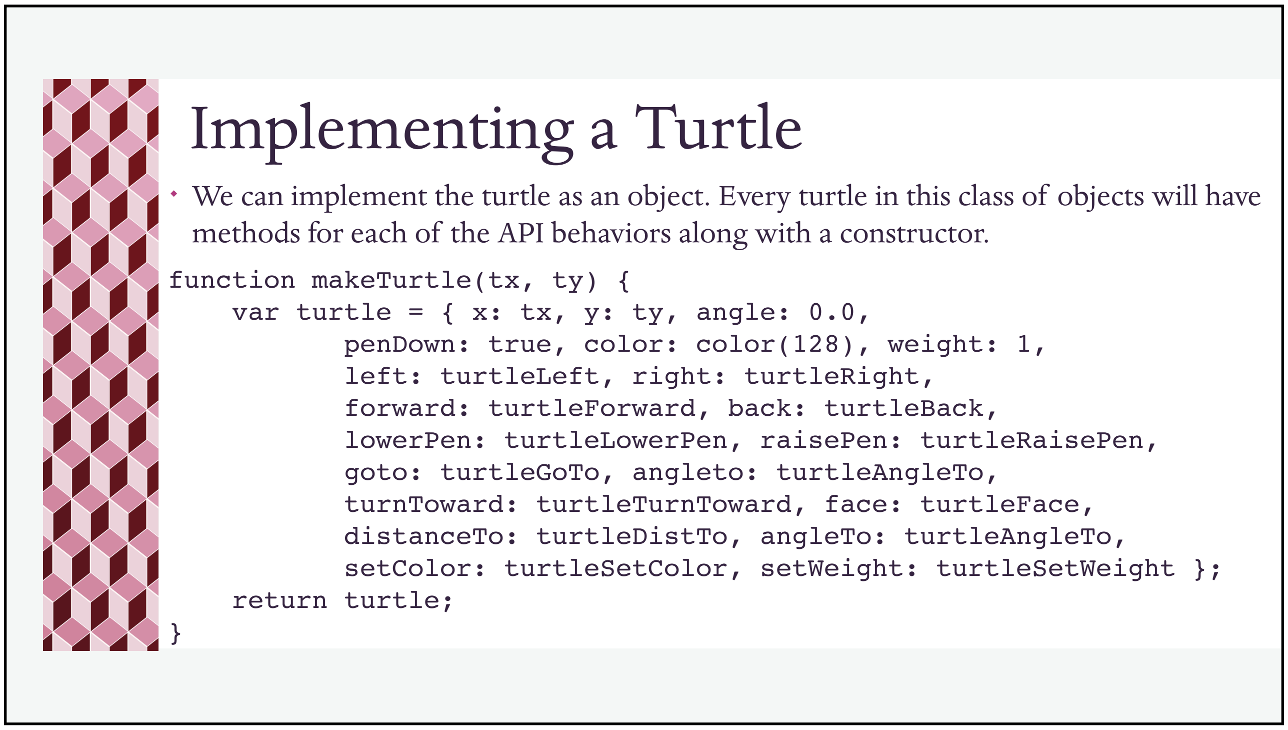
4



Turtle API (cont'd)

- `lowerPen()` -- set pen down
- `raisePen()` -- pick pen up
- `goto(x, y)` -- go straight to this location
- `setColor(color)` -- set the drawing color
- `setWeight(w)` -- set line width to w
- `face(d)` -- turn to this absolute direction in degrees
- `angleTo(x, y)` -- what is the angle from my heading to location x, y?
- `turnToward(x, y, d)` -- turn by d degrees toward location x, y
- `distanceTo(x, y)` -- how far is it to location x, y?

5

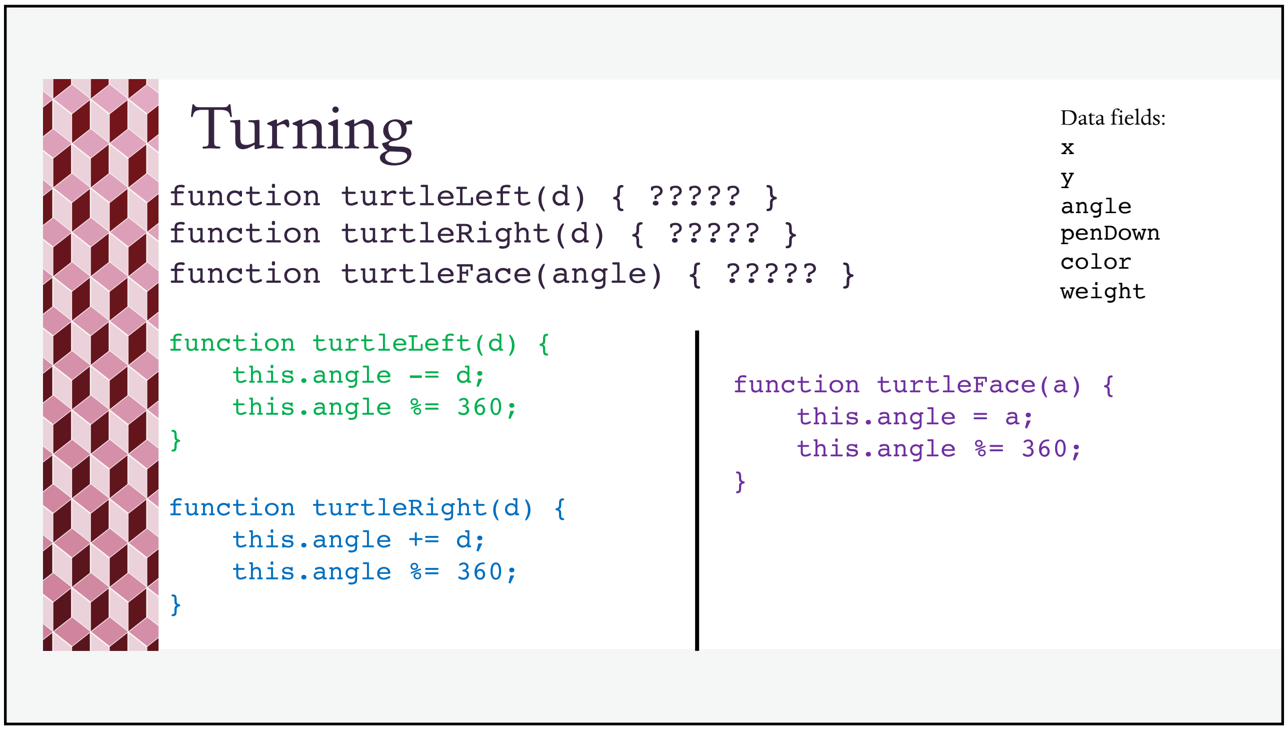


Implementing a Turtle

- We can implement the turtle as an object. Every turtle in this class of objects will have methods for each of the API behaviors along with a constructor.

```
function makeTurtle(tx, ty) {
    var turtle = { x: tx, y: ty, angle: 0.0,
           penDown: true, color: color(128), weight: 1,
           left: turtleLeft, right: turtleRight,
           forward: turtleForward, back: turtleBack,
           lowerPen: turtleLowerPen, raisePen: turtleRaisePen,
           goto: turtleGoTo, angleto: turtleAngleTo,
           turnToward: turtleTurnToward, face: turtleFace,
           distanceTo: turtleDistTo, angleTo: turtleAngleTo,
           setColor: turtleSetColor, setWeight: turtleSetWeight };
    return turtle;
}
```

6



Turning

Data fields:
```
x
y
angle
penDown
color
weight
```

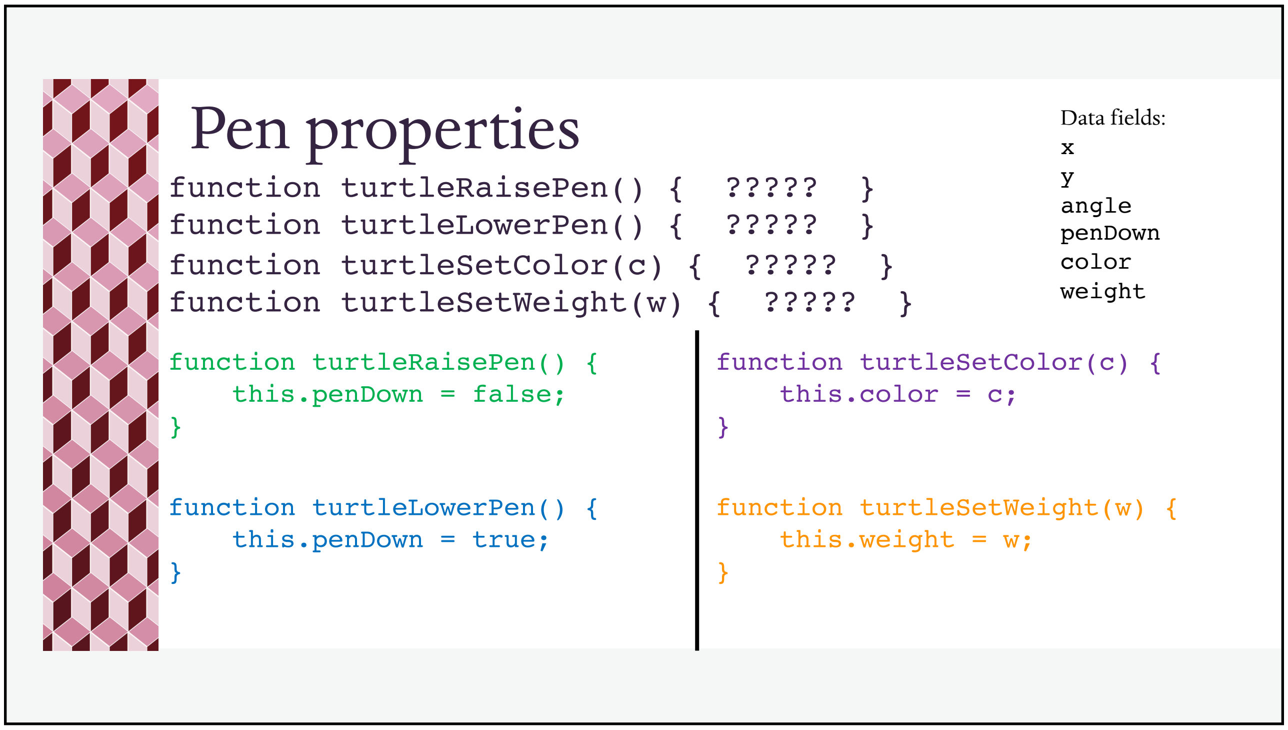
```
function turtleLeft(d) { ????? }
function turtleRight(d) { ????? }
function turtleFace(angle) { ????? }
```

```
function turtleLeft(d) {
    this.angle -= d;
    this.angle %= 360;
}

function turtleRight(d) {
    this.angle += d;
    this.angle %= 360;
}
```

```
function turtleFace(a) {
    this.angle = a;
    this.angle %= 360;
}
```

7



Pen properties

Data fields:
```
x
y
angle
penDown
color
weight
```

```
function turtleRaisePen() {  ?????  }
function turtleLowerPen() {  ?????  }
function turtleSetColor(c) {  ?????  }
function turtleSetWeight(w) {  ?????  }
```

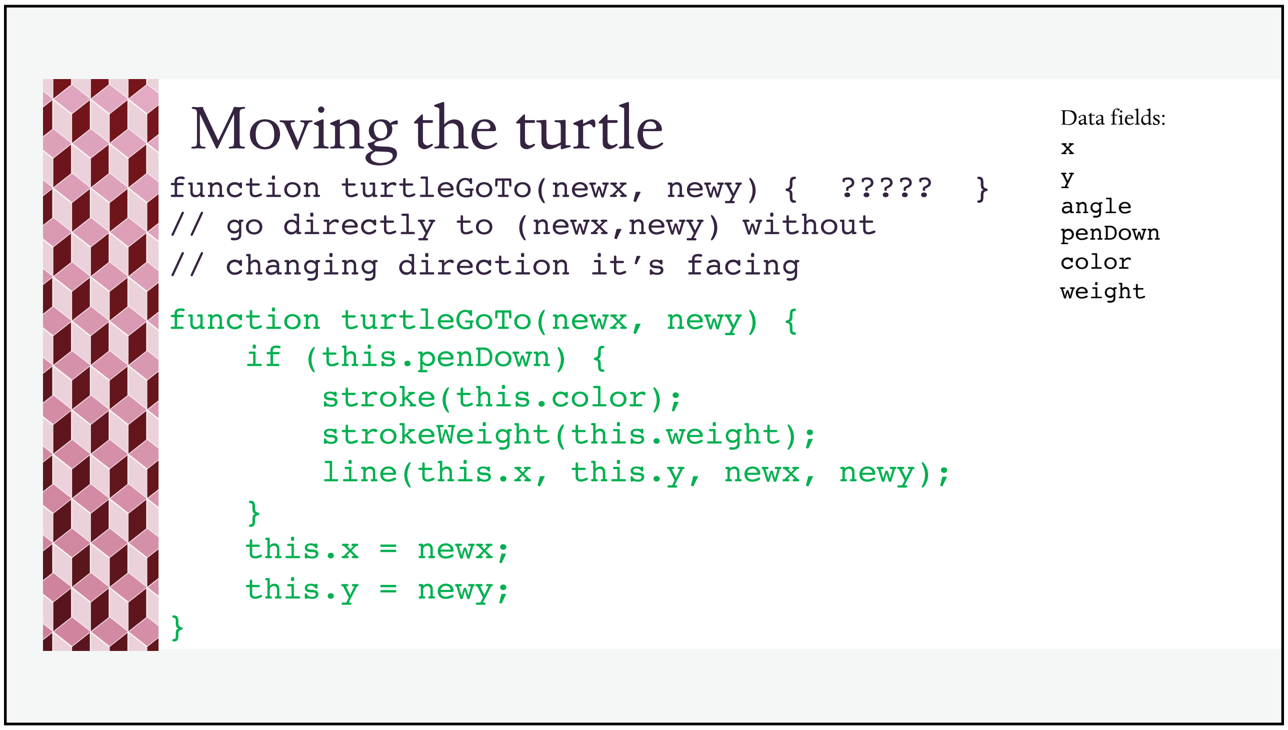
```
function turtleRaisePen() {
    this.penDown = false;
}

function turtleLowerPen() {
    this.penDown = true;
}
```

```
function turtleSetColor(c) {
    this.color = c;
}

function turtleSetWeight(w) {
    this.weight = w;
}
```

8



Moving the turtle

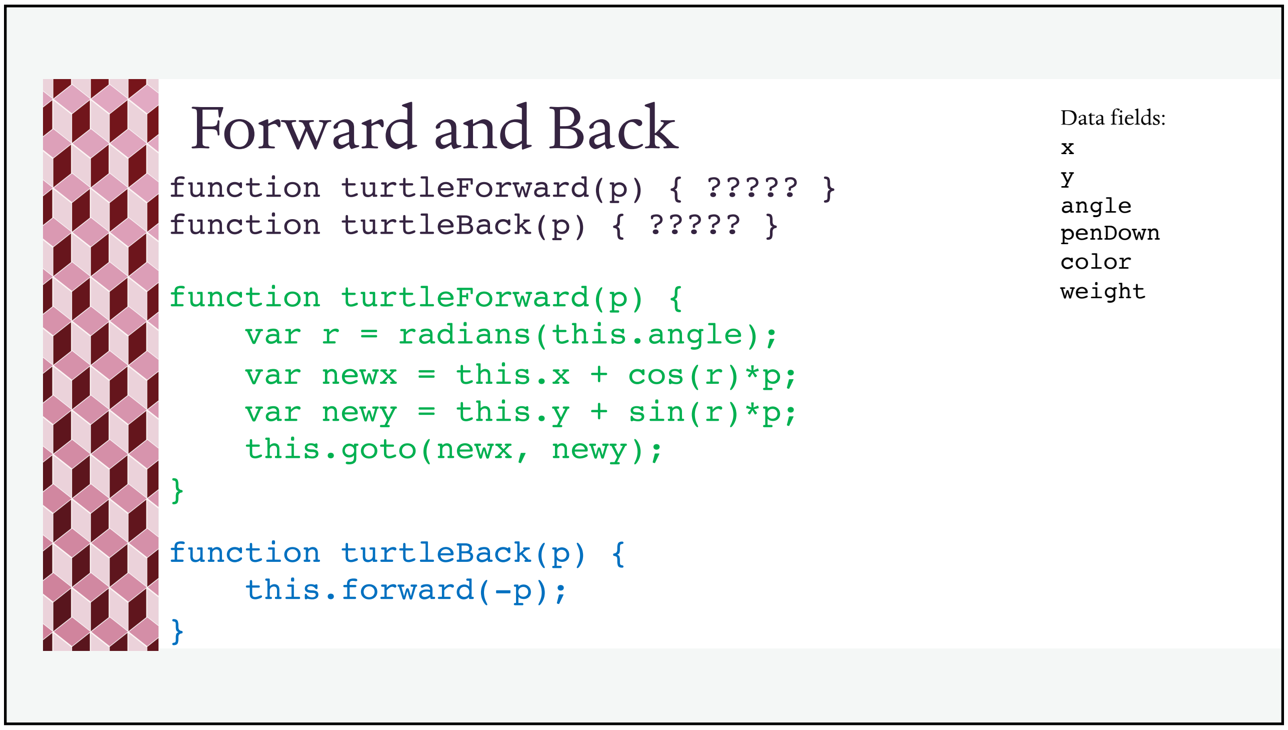
```
function turtleGoTo(newx, newy) {  ?????  }
// go directly to (newx,newy) without
// changing direction it's facing
```

```
function turtleGoTo(newx, newy) {
    if (this.penDown) {
        stroke(this.color);
        strokeWeight(this.weight);
        line(this.x, this.y, newx, newy);
    }
    this.x = newx;
    this.y = newy;
}
```

Data fields:

```
x
y
angle
penDown
color
weight
```

9



Forward and Back

```
function turtleForward(p) { ????? }
function turtleBack(p) { ????? }
```

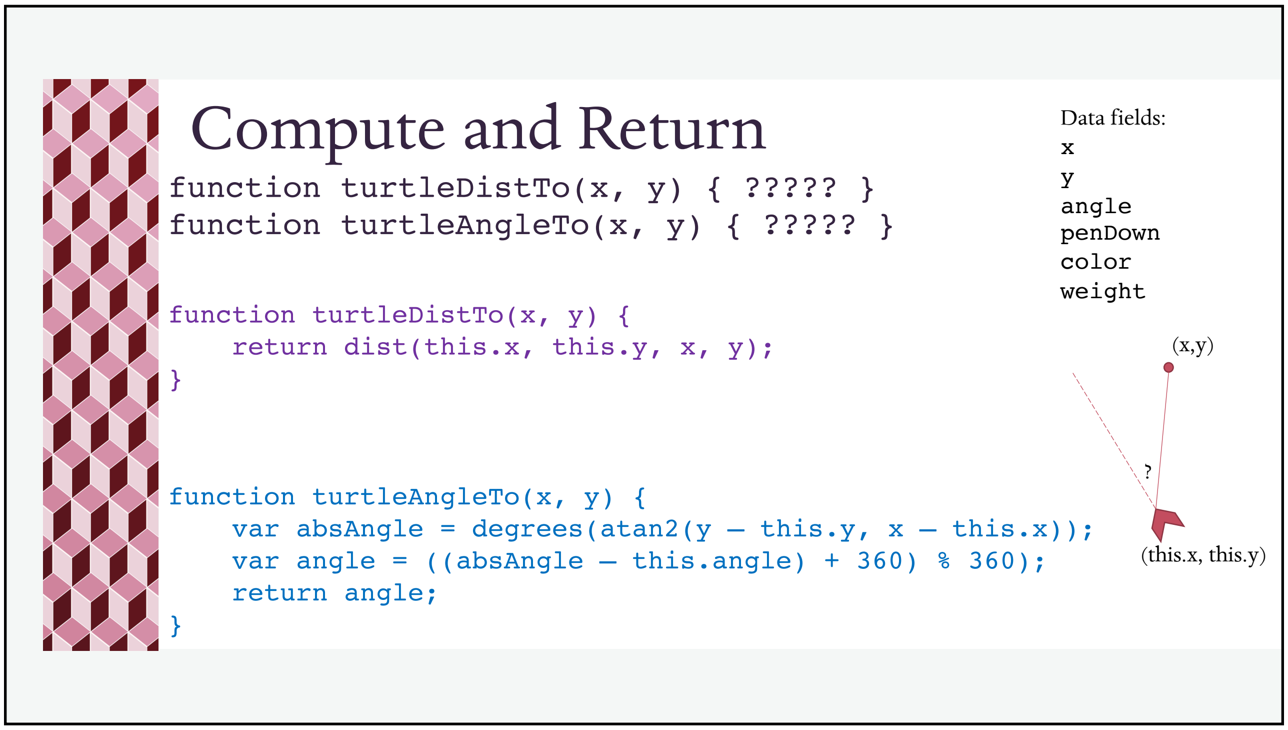
```
function turtleForward(p) {
    var r = radians(this.angle);
    var newx = this.x + cos(r)*p;
    var newy = this.y + sin(r)*p;
    this.goto(newx, newy);
}
```

```
function turtleBack(p) {
    this.forward(-p);
}
```

Data fields:

```
x
y
angle
penDown
color
weight
```

10



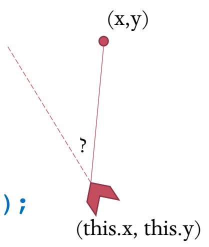
Compute and Return

```
function turtleDistTo(x, y) { ????? }
function turtleAngleTo(x, y) { ????? }
```

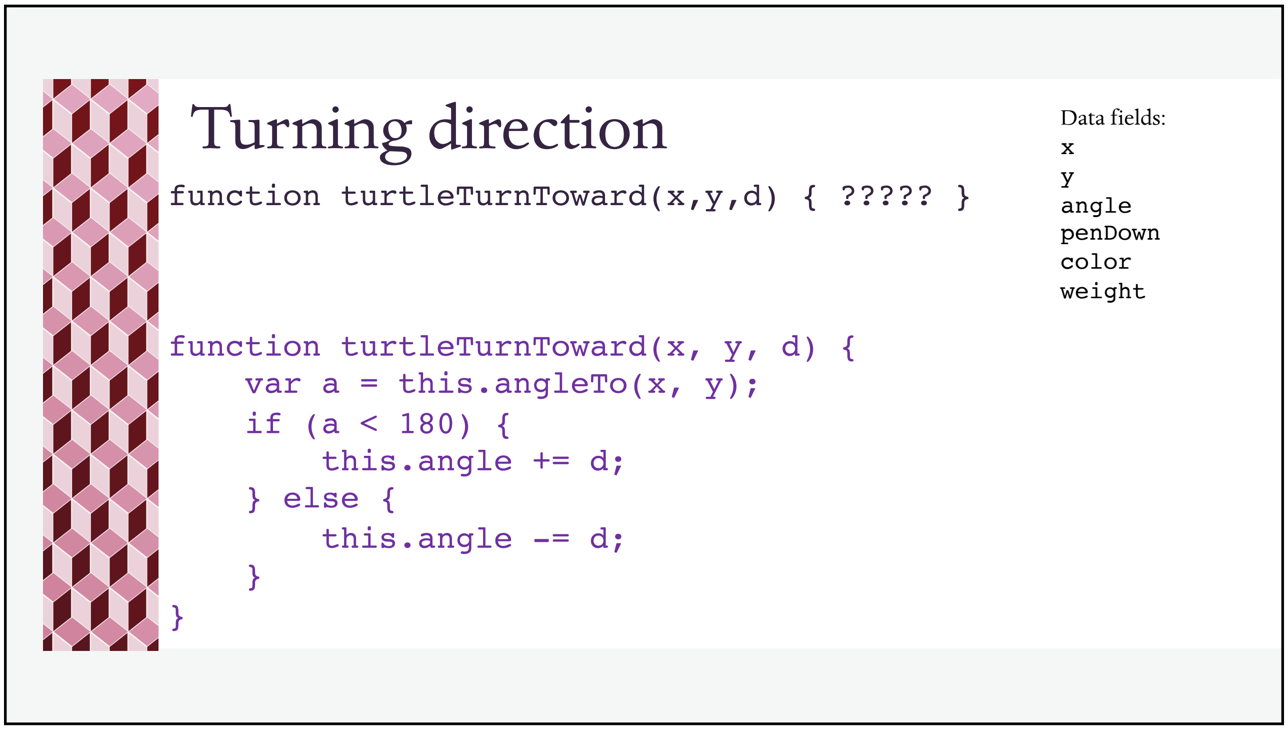
```
function turtleDistTo(x, y) {
    return dist(this.x, this.y, x, y);
}
```

```
function turtleAngleTo(x, y) {
    var absAngle = degrees(atan2(y - this.y, x - this.x));
    var angle = ((absAngle - this.angle) + 360) % 360);
    return angle;
}
```

Data fields:
```
x
y
angle
penDown
color
weight
```



11



Turning direction

```
function turtleTurnToward(x,y,d) { ????? }
```

```
function turtleTurnToward(x, y, d) {
    var a = this.angleTo(x, y);
    if (a < 180) {
        this.angle += d;
    } else {
        this.angle -= d;
    }
}
```

Data fields:
```
x
y
angle
penDown
color
weight
```

12

Using Turtles

- In order to use Turtle Graphics, we must include the code for this object and its methods in our programs.
- Copy and paste this into each example you work on.
- Put a comment at the top that says
  `// DO NOT EDIT BELOW THIS LINE`
- Then write your program code **ABOVE** this comment line.
- When you scroll through your code, you will see your code first and then the turtle code afterwards. You do not need to edit the Turtle code, so it's easier to leave it at the end of the file.

13

Examples

- Demo
  Basic drawing commands
- Spirograph
  Some cool patterns with only a few commands!

14