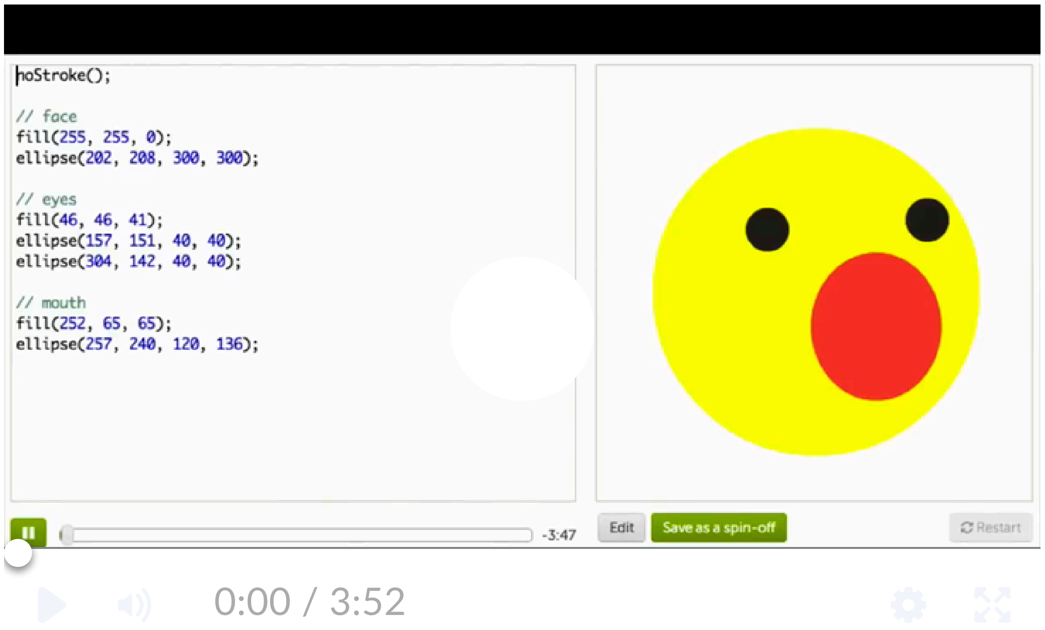
VF

2.3 Variables And Data Types

Introduction To Variables. The video *Intro to Variables* provides a clear and engaging explanation of what variables are and why they are essential in programming. It defines a variable as a name that represents an unknown or changeable value. The video emphasizes that variables allow computer scientists to write general code that can apply to many situations, making problem-solving more flexible and powerful. By using simple examples and relatable scenarios, the video demonstrates how variables are used in expressions in a drawing program.



```
noStroke();

// face
fill(255, 255, 0);
ellipse(202, 208, 300, 300);

// eyes
fill(46, 46, 41);
ellipse(157, 151, 40, 40);
ellipse(304, 142, 40, 40);

// mouth
fill(252, 65, 65);
ellipse(257, 240, 120, 136);
```

Test Your Understanding. Try these questions to test your understanding.

- In the following code, we define two variables, w and h, and use them to draw an ellipse and a rectangle:

  ```
  var w = 20;
  var h = 15;
  rect(10, 10, w, h);
  ellipse(10, 10, w, h);
  ```

  How tall is each one of the shapes?   **Check Your Answer (https://trycodelab.com/lessons/03.4-Variables-and-Data-Types/02a-Challenge-Variables-Exercise-1.md)**

  - 20
  - 15

  - 10

- In the following code, we define a variable and use it to draw a rectangle:

  ```
  var rectWidth = 20;
  rect(10, 10, rectWidth, rectWidth);
  ```

  How wide is the rectangle? **Check Your Answer (https://trycodelab.com/lessons/03.4-Variables-and-Data-Types/02b-Challenge-Variables-Exercise-2.md)**

  - 20
  - 15
  - 10

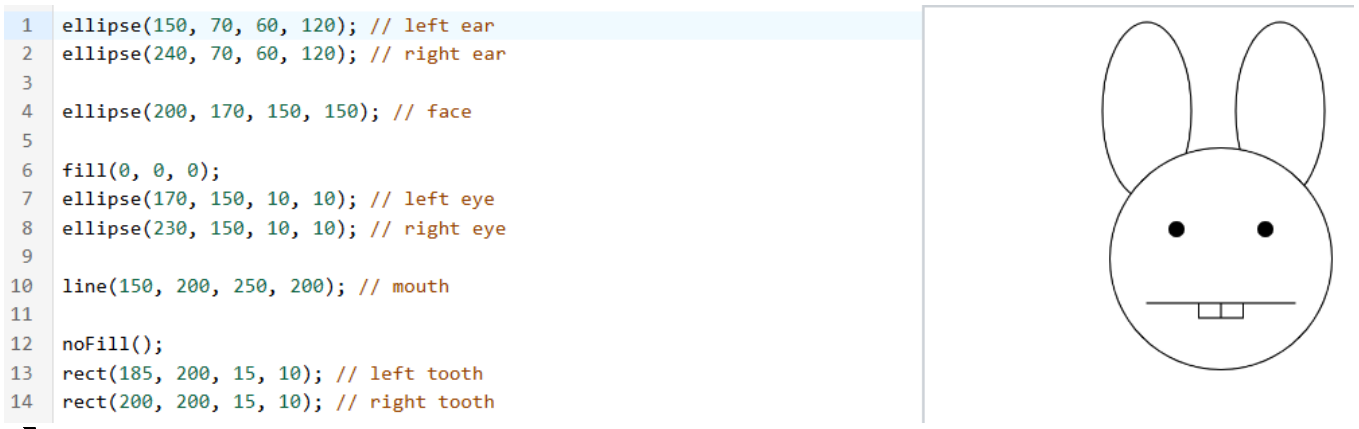
- The following code draws a face with a mouth (with no eyes!), using two variables:

  ```
  var faceS = 100;
  var mouthS = 30;

  ellipse(200, 200, faceS, faceS);
  ellipse(200, 220, mouthS, mouthS);
  ```

  How wide is the second ellipse that draws the mouth? **Check Your Answer (https://trycodelab.com/lessons/03.4-Variables-and-Data-Types/02c-Challenge-Variables-Exercise-3.md)**

  - 200
  - 100
  - 30

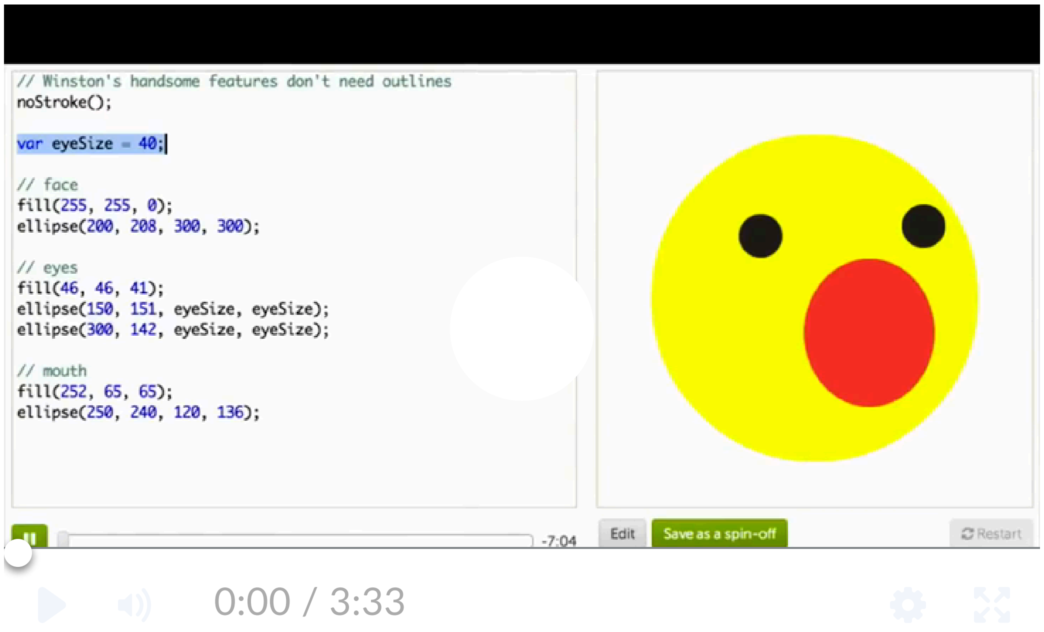
Coding Practice: Bucktooth Bunny. This challenge will have you draw using variables, which make it easier to adjust sizes. Click on the image to do the challenge.



```
ellipse(150, 70, 60, 120); // left ear
ellipse(240, 70, 60, 120); // right ear

ellipse(200, 170, 150, 150); // face

fill(0, 0, 0);
ellipse(170, 150, 10, 10); // left eye
ellipse(230, 150, 10, 10); // right eye

line(150, 200, 250, 200); // mouth

noFill();
rect(185, 200, 15, 10); // left tooth
rect(200, 200, 15, 10); // right tooth
```

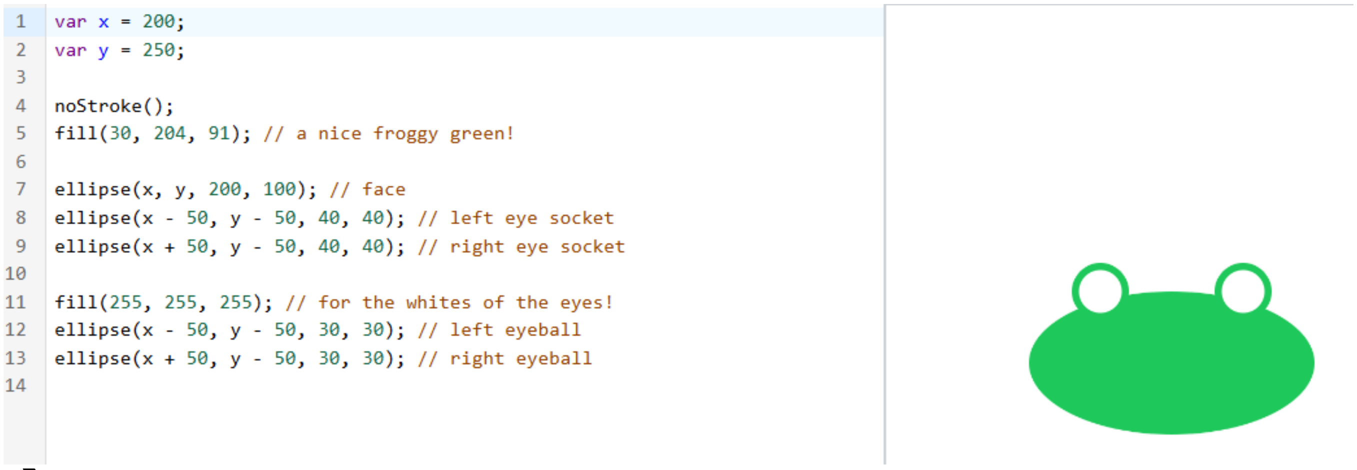
(https://trycodelab.com/lessons/03.4-Variables-and-Data-Types/03-challenge-bucktooth-bunny.md)

More on Variables. This video explains the basic idea of variables in coding, describing them as names used to store information that can change. It uses clear examples to show how variables help control different parts of a program, such as the size or position of shapes. By assigning values to variables, programmers can easily update and reuse information throughout their code. The video highlights how variables make programs more organized, flexible, and easier to understand—especially when changes are needed. This introduction helps viewers see why variables are a key part of writing effective computer programs.



```
// Winston's handsome features don't need outlines
noStroke();

var eyeSize = 40;

// face
fill(255, 255, 0);
ellipse(200, 208, 300, 300);

// eyes
fill(46, 46, 41);
ellipse(150, 151, eyeSize, eyeSize);
ellipse(300, 142, eyeSize, eyeSize);

// mouth
fill(252, 65, 65);
ellipse(250, 240, 120, 136);
```

Coding Practice: Funky Frog. This challenge has you draw a frog using variables. Click the image to do the challenge.



```
var x = 200;
var y = 250;

noStroke();
fill(30, 204, 91); // a nice froggy green!

ellipse(x, y, 200, 100); // face
ellipse(x - 50, y - 50, 40, 40); // left eye socket
ellipse(x + 50, y - 50, 40, 40); // right eye socket

fill(255, 255, 255); // for the whites of the eyes!
ellipse(x - 50, y - 50, 30, 30); // left eyeball
ellipse(x + 50, y - 50, 30, 30); // right eyeball
```

(https://trycodelab.com/lessons/03.4-Variables-and-Data-Types/05-challenge-funky-frog.md)

Variables And Computer Memory

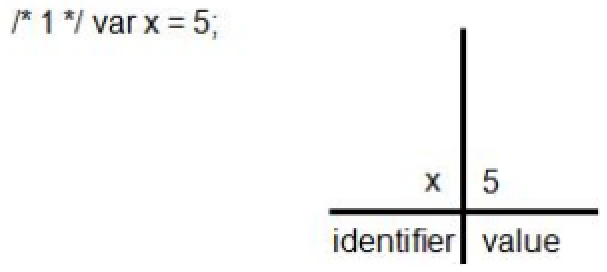
0:00 / 3:19

Memory diagrams are used to help trace code. They are typically hand-written diagrams that reflect the program’s memory over time. Memory diagrams are drawn as “inverted Ts” with two columns: the program’s identifier for the memory (also called name), and the value in the memory (also called stack). Examples are provided below.

As you trace the code from top to bottom, you add memory that the program allocates bottom up to the memory diagram. Consider this example in Javascript where the keyword var allocates a new variable and line numbers have been added as comments:

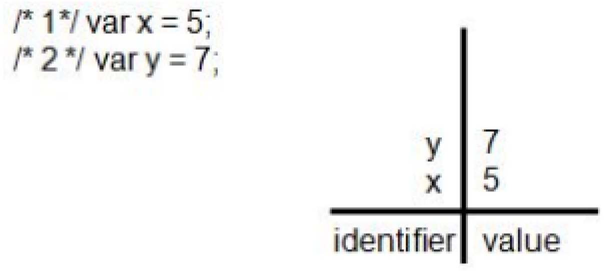
```
/* 1 */ var x = 5;
/* 2 */ var y = 7;
/* 3 */ x = 9;
/* 4 */ var total = x+y;
/* 5 */ println(total);
```

The program prints 16 to the console. When tracing this code, you go from top to bottom. The memory diagram for line 1 is:



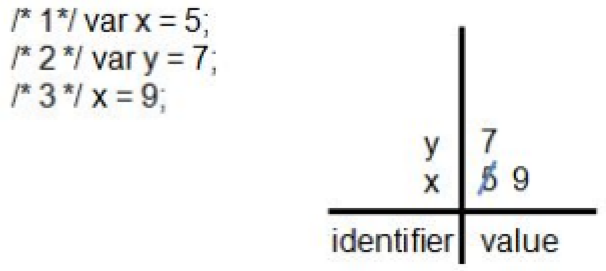
```
/* 1 */ var x = 5;
```

Note that you add x in the diagram because of the JavaScript var in the code line. var means allocate a new variable. Next trace line 2 by adding identifier y and its value. Do this above x and its value:



```
/* 1*/ var x = 5;
/* 2 */ var y = 7;
```

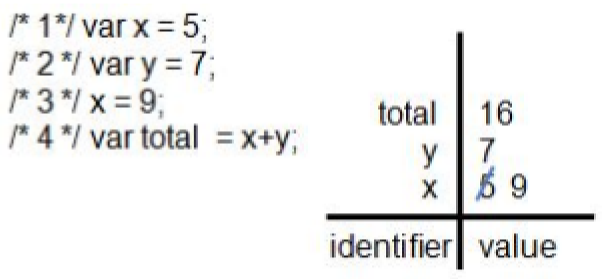
Next trace line 3. Since there is no new var, there is no new memory, so you overwrite the value of x. You do this by crossing out the old value of 5 and writing the new value of 9 so that we can see the history of x's values.



```
/* 1*/ var x = 5;
/* 2 */ var y = 7;
/* 3 */ x = 9;
```

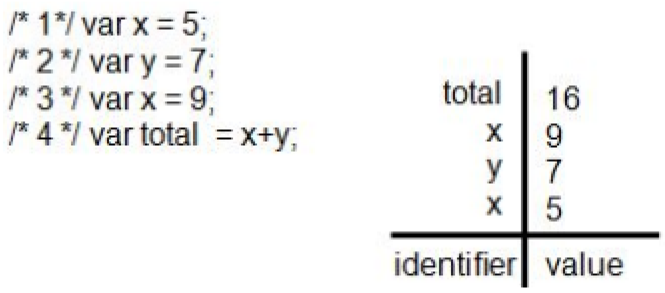
Tracing line 4 adds a new memory location for total. You evaluate the expression adding

Tracing line 4 adds a new memory location for total. You evaluate the expression adding the values for x and y that you get from the diagram, which yields the value 16. Write this 16 in the diagram for total.



```
/* 1*/ var x = 5;
/* 2 */ var y = 7;
/* 3 */ x = 9;
/* 4 */ var total  = x+y;
```

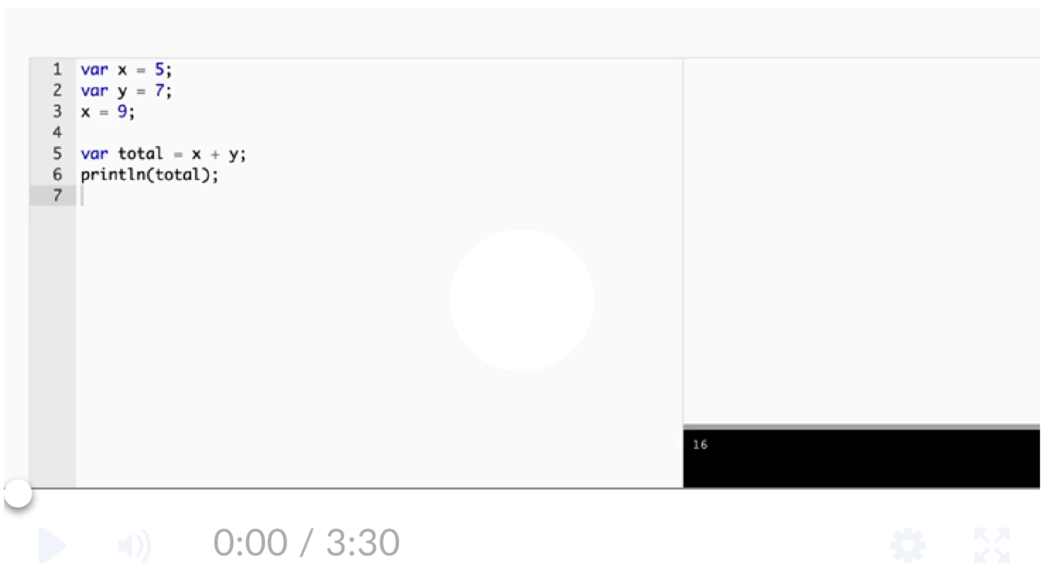
Tracing line 5 prints to the console, which does not affect memory, so you don't show it in the memory diagram. When tracing code, to find the current value of a variable, read down the memory diagram from top to bottom and use the non-crossed out value for the variable. If the example program had put a var in front of the x on line 3, then the program would have allocated another memory cell and called it x. This is show in this memory diagram:



```
/* 1*/ var x = 5;
/* 2 */ var y = 7;
/* 3 */ var x = 9;
/* 4 */ var total  = x+y;
```

This program also prints 16 to the console. Note that x appears twice in the memory diagram. Since you read the diagram from top to bottom, the x whose value is 9 is used by the program. The x whose value is 5 is wasted memory (sometimes called garbage) - the program will never access it again.

This video demonstrates the creation of a memory diagram from code that has variables.



```
var x = 5;
var y = 7;
x = 9;

var total = x + y;
println(total);
```

```
16
```

Edit HTML Reflect in ePortfolio Download

Print Open with docReader

Activity Details Learning Objectives Completion Summary

Visible

Required: Automatic

View this topic to complete the activity

Options

Reflecting in ePortfolio is enabled

2.3 Variables Abd Data Types.html - Last Modified Sep 5, 2025 8:42 AM