

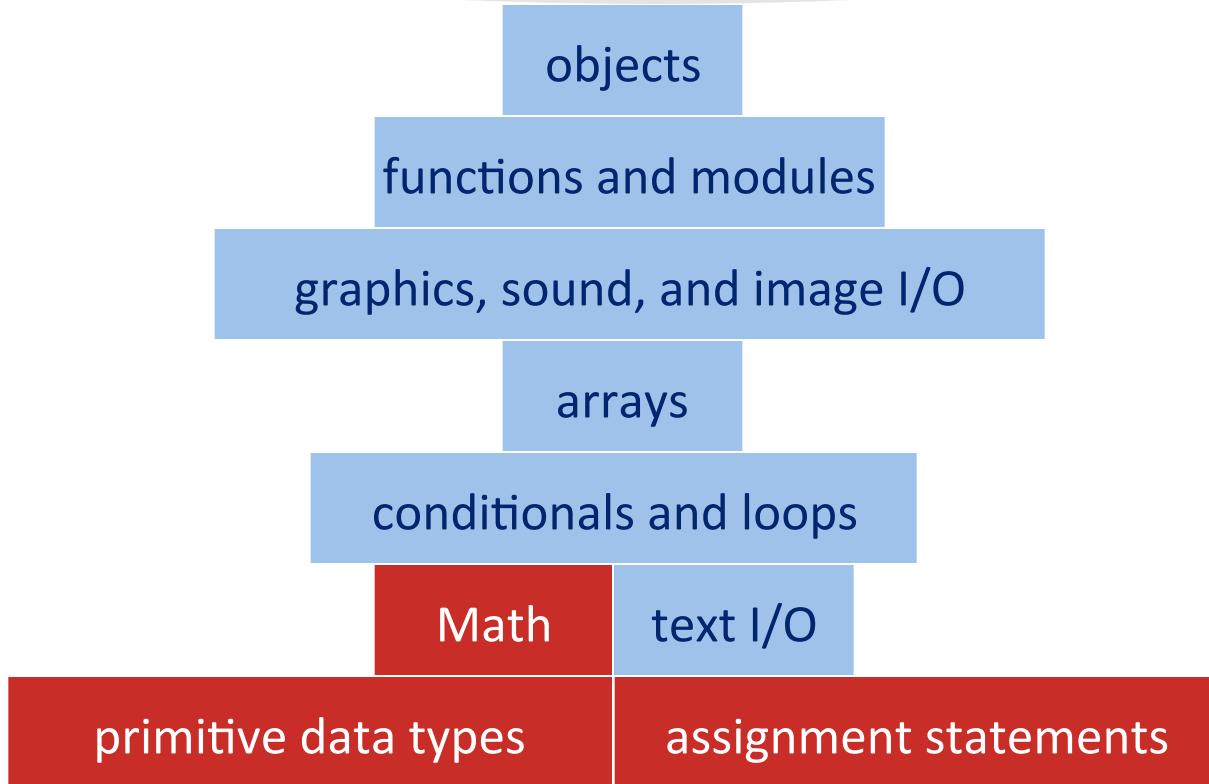
Variables

Review

- Random numbers
- mouseX, mouseY
- setup() & draw()
- frameRate(), loop(), noLoop()
- Mouse and Keyboard interaction
- Arcs, curves, bézier curves, custom shapes
- Hue-Saturation-Brightness vs. Red-Green-Blue color
- Decimal, Binary numbers, and colors

A Foundation for Programming

any program you might want to write



Variables

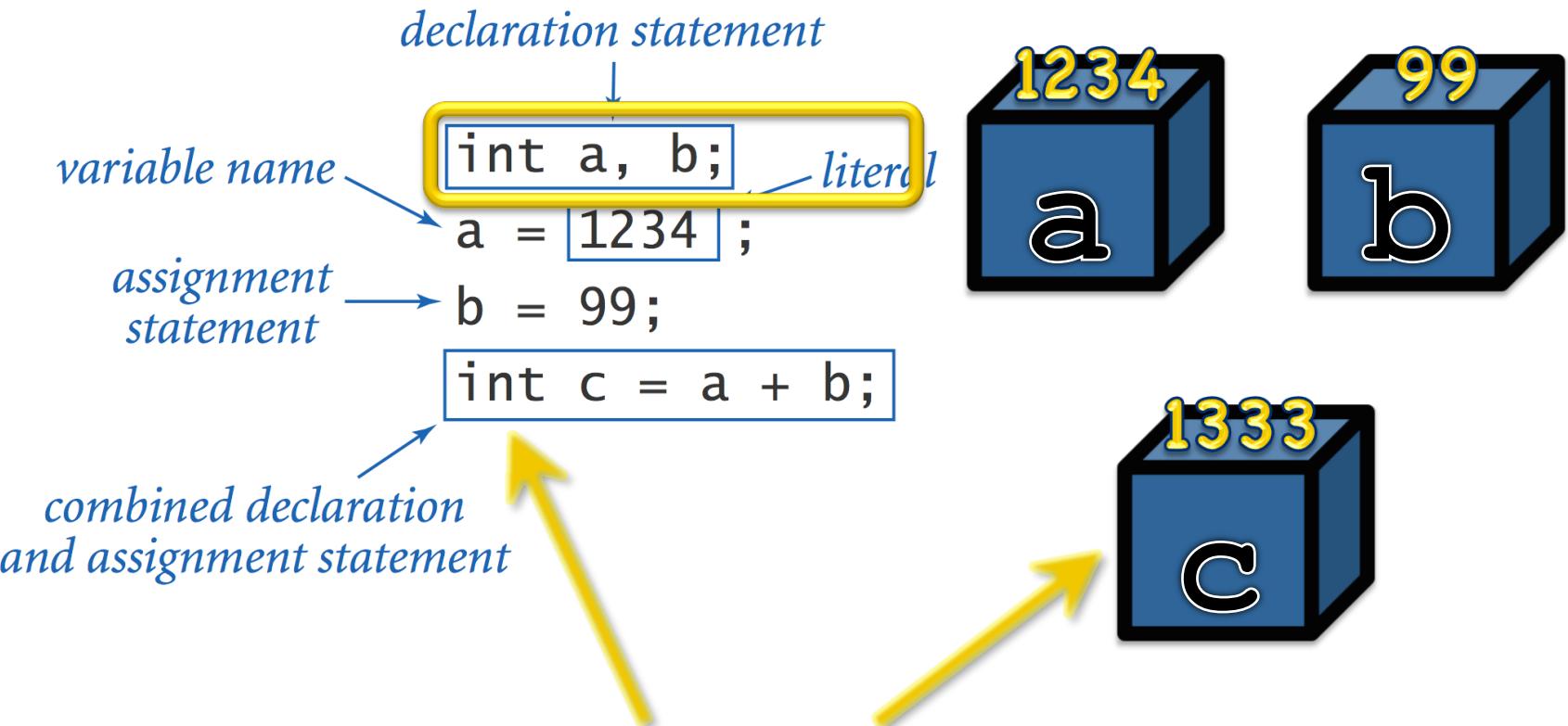
- A name to which data can be assigned
- A variable is declared as a specific data type
- Names must begin with a lowercase letter, ‘_’ or ‘\$’ and can contain letters, digits, ‘_’ and ‘\$’

```
boolean bReady = true;  
int i;  
int j = 12;  
float fSize = 10.0;  
color _red = color(255, 0, 0);  
String name123 = "Fred";  
PImage img;
```

Variable Uses

- Use a value throughout your program,
 - but allow it to be changed
- As temporary storage for a intermediate computed result
- ... etc

Variables and Types



“int” means that the variable will always hold an integer

Assignment

“=” stores a value in a variable

```
int a, b;
```

```
a = 1234;
```

```
b = 99;
```

```
int t = a;
```

```
a = b;
```

```
b = t;
```

It is not for comparison,
as in standard math



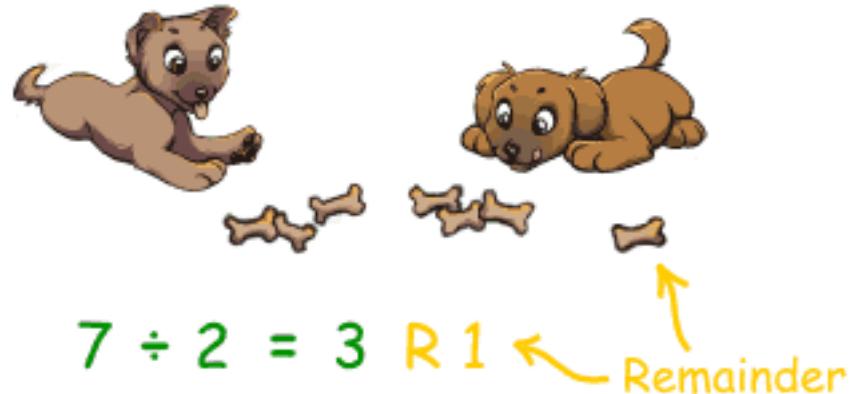
int: Integers (whole numbers)

`+, -, *, /, % (modulo), (), Integer.parseInt()`

Expression	Result?
<code>5 + 3</code>	
<code>5 - 3</code>	
<code>5 * 3</code>	
<code>5 / 3</code>	
<code>5 % 3</code>	
<code>5 % -3</code>	
<code>1 / 0</code>	
<code>3 * 5 - 2</code>	
<code>3 + 5 / 2</code>	
<code>3 - 5 / 2</code>	
<code>(3 - 5) / 2</code>	
<code>3 - (5 - 2) / 2</code>	
<code>Integer.parseInt("3")</code>	
<code>Integer.parseInt(3)</code>	

Modulo Operator (%)

$$\begin{array}{r} \text{Quotient} \\ \downarrow \\ 5 \text{ r } 1 \\ \hline 5 \sqrt{26} \\ -25 \\ \hline 1 \end{array}$$



Division gives the quotient:

$$26 / 5 == 5$$

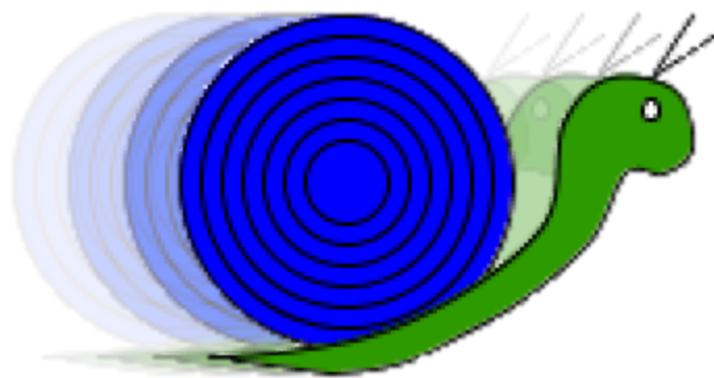
Modulo gives the remainder:

$$26 \% 5 == 1$$

Example: Determining whether an integer n is even or odd:

```
boolean isEven = (n % 2 == 0);
```

Animation



In-Class Demo

Variable Scope

Variable scope:

- That set of code statements in which the variable is known to the compiler
- Where it can be referenced in your program
- Limited to the **code block** in which it is defined
 - A **code block** is a set of code enclosed in braces (`{ }`)

double: Floating-Point (fractions)

`+, -, *, /, % (modulo), (), Double.parseDouble()`

Expression	Result?
<code>3.141 + 0.03</code>	
<code>6.02e23 / 2.0</code>	
<code>5.0 / 3</code>	
<code>(int) 5.0 / 3</code>	
<code>5.0 / (int) 3</code>	
<code>10.0 % 3.141</code>	
<code>1.0 / 0.0</code>	
<code>-1.0 / 0.0</code>	
<code>0.0 / 0.0</code>	
<code>Math.sqrt(2)</code>	
<code>Math.sqrt(-1)</code>	
<code>Math.sqrt(2) * Math.sqrt(2)</code>	
<code>Math.PI</code>	
<code>Math.pi</code>	

Java Math Library (Excerpts)

```
public class Math
```

double abs(double a)	<i>absolute value of a</i>
double max(double a, double b)	<i>maximum of a and b</i>
double min(double a, double b)	<i>minimum of a and b</i>

Note 1: abs(), max(), and min() are defined also for int, long, and float.

double sin(double theta)	<i>sine function</i>
double cos(double theta)	<i>cosine function</i>
double tan(double theta)	<i>tangent function</i>

Note 2: Angles are expressed in radians. Use toDegrees() and toRadians() to convert.

Note 3: Use asin(), acos(), and atan() for inverse functions.

double exp(double a)	<i>exponential (e^a)</i>
double log(double a)	<i>natural log ($\log_e a$, or $\ln a$)</i>
double pow(double a, double b)	<i>raise a to the bth power (a^b)</i>

long round(double a)	<i>round to the nearest integer</i>
double random()	<i>random number in [0, 1)</i>
double sqrt(double a)	<i>square root of a</i>

double E	<i>value of e (constant)</i>
double PI	<i>value of π (constant)</i>

char: Single Characters

Single characters are stored as (small) integers!

Expression	Result?
'A'	
'A' + 0	
(int) 'A'	
(char) 65	
(int) 'a'	
(int) '0'	
'3' - '0'	

Character codes are defined by
the **ASCII** and **Unicode** standards.

boolean: True/False

true, false, ==, !=, <, >, <=, >=, && (and), || (or), ! (not)

Expression	Result?
true	
!false	
'A' == 'a'	
Math.PI != 3.14	
'a' > 'b'	
1.7 <= (17 / 10)	
true && true	
true && false	
false && false	
true true	
true false	
false false	
(1 < 3) && (3 == (6 / 2))	
(1 >= 3) !(3 == (6 / 2))	

Data Type Conversion

- Some variable types can be converted to other types
- Via **casting** (from Java)

```
float f = 10.0;  
int i = (int) f;
```

- Processing includes additional type conversion functions (these don't work in standard Java):

```
// binary(...), boolean(...), byte(...),  
// char(...), float(...), str(...)
```

```
float f = 10.0;  
int i;
```

```
//i = f;           // Throws a runtime error  
i = int(f);
```

```
println( char(65) );    // Prints the character 'A'
```

Primitive Data Types

Type	Range	Default	Bytes
boolean	{ true, false }	false	?
byte	{ 0..255 }	0	1
int	{ -2,147,483,648 ... 2,147,483,647 }	0	4
long	{ -9,223,372,036,854,775,808 ... 9,223,372,036,854,775,807 }	0	8
float	{ -3.40282347E+38 ... 3.40282347E+38 }	0.0	4
double	<i>much larger/smaller</i>	0.0	8
char	<i>a single character 'a', 'b', ...</i>	'\u0000'	2

More Complex Data Types

Type	Range	Default	Bytes
String	a series of chars in quotes “abc”	null	?
PImage	an image	null	?
PFont	a font for rendering text	null	?
...			