Variables and Types

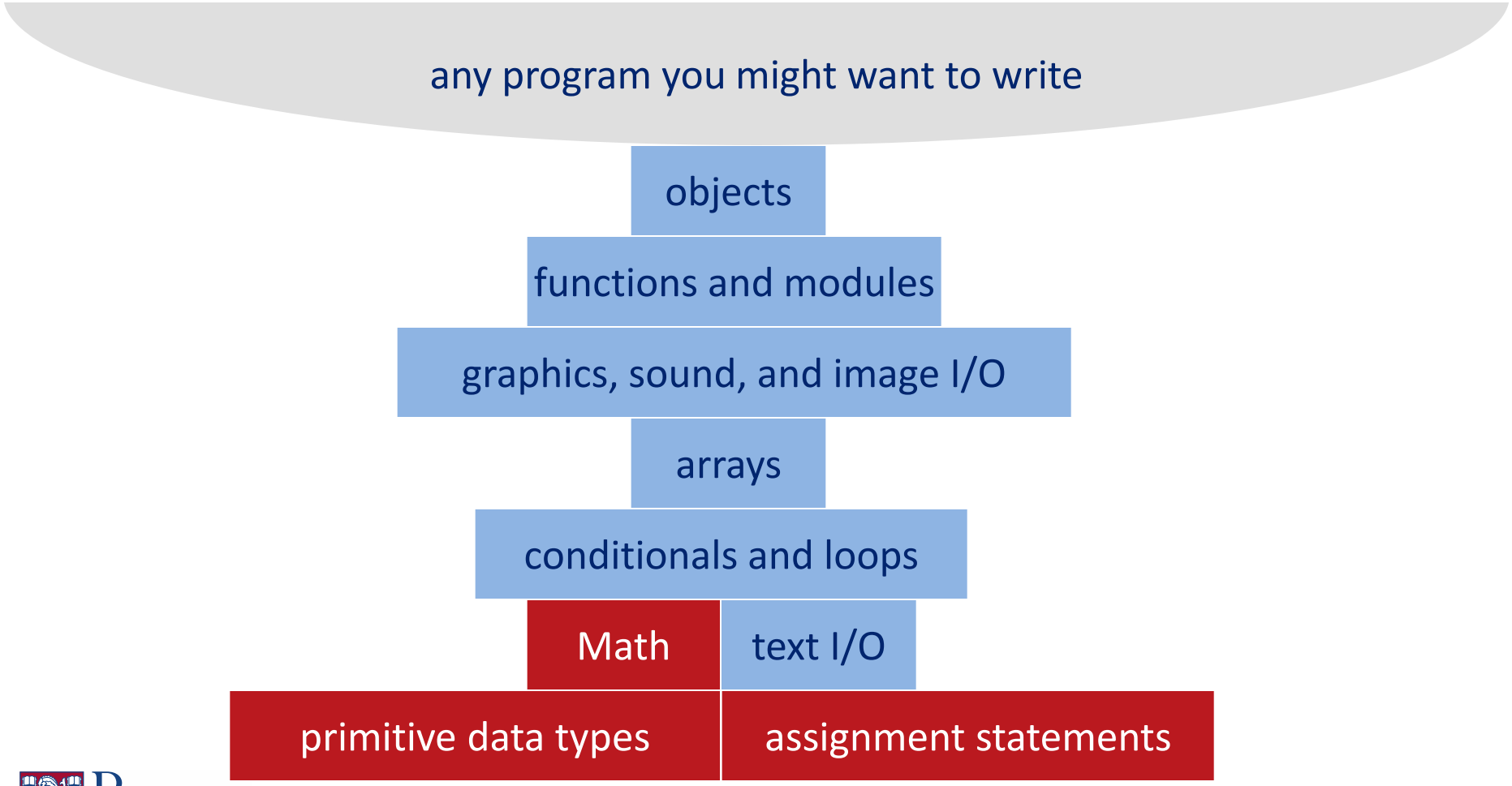
Penn Engineering

Section 1.2

Review

- Drawing primitives – lines, ellipses, etc
- setup() & draw()
- Random numbers
- mouseX, mouseY
- Colors – RGB values

A Foundation for Programming



Penn Engineering

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 '$' and can contain letters, digits, '_' and '$'

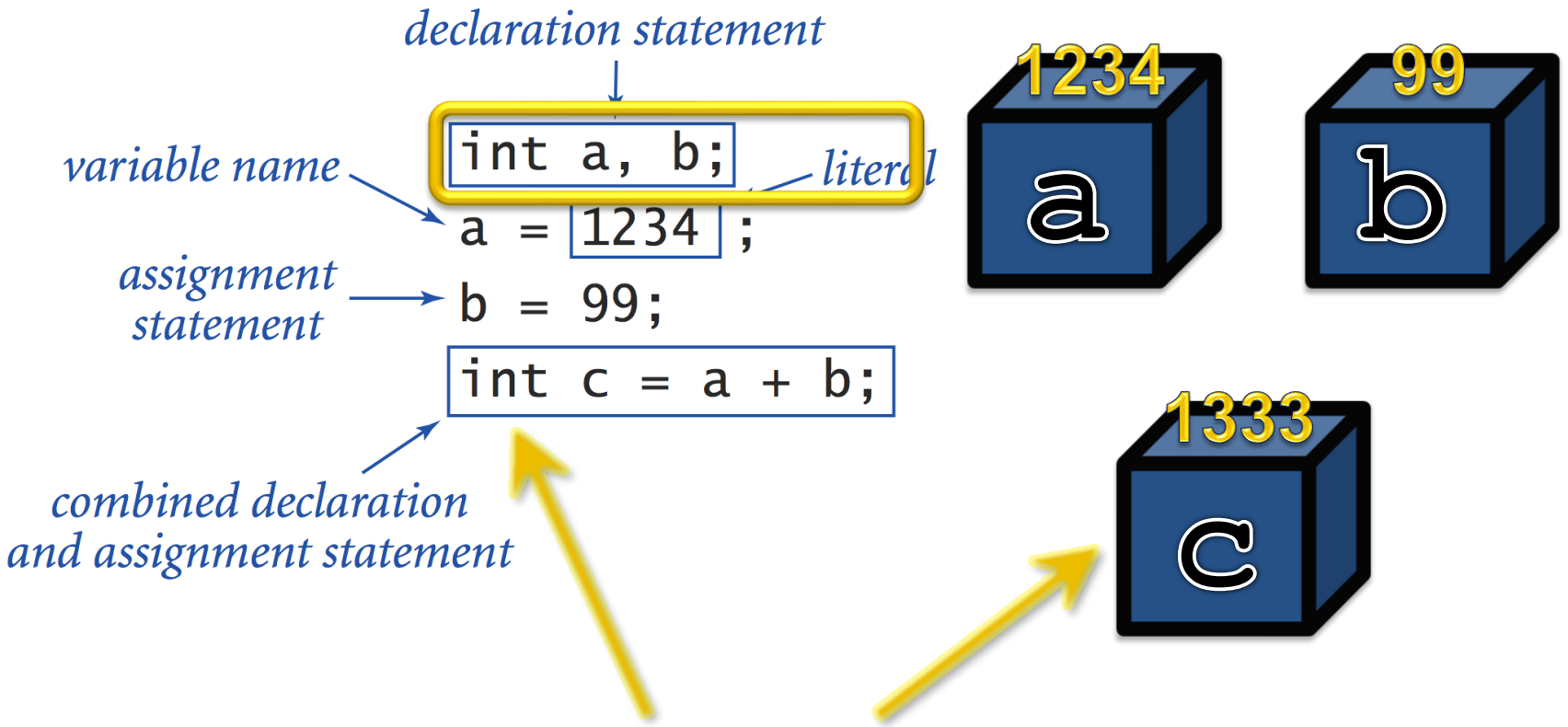
```
boolean bReady = true;
int i;
int j = 12;
float fSize = 10.0;
int _red = color(255, 0, 0);  // encodes color in int
String name123 = "Fred";
PImage img;
```

Penn Engineering

Variable Uses

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

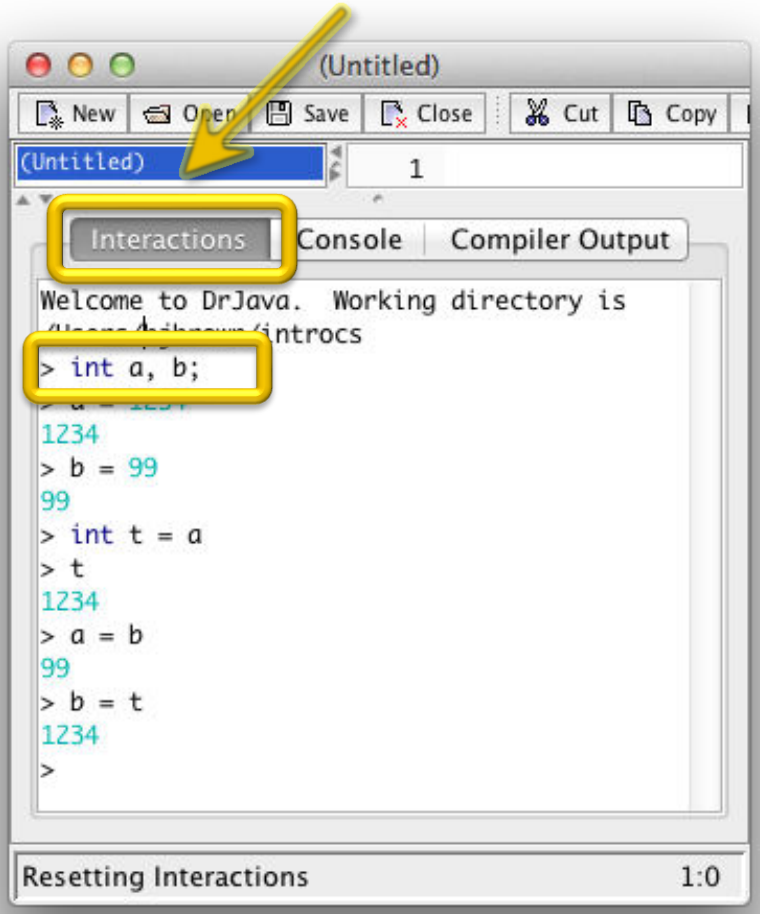
Variables and Types



"int" means that the variable will always hold an integer

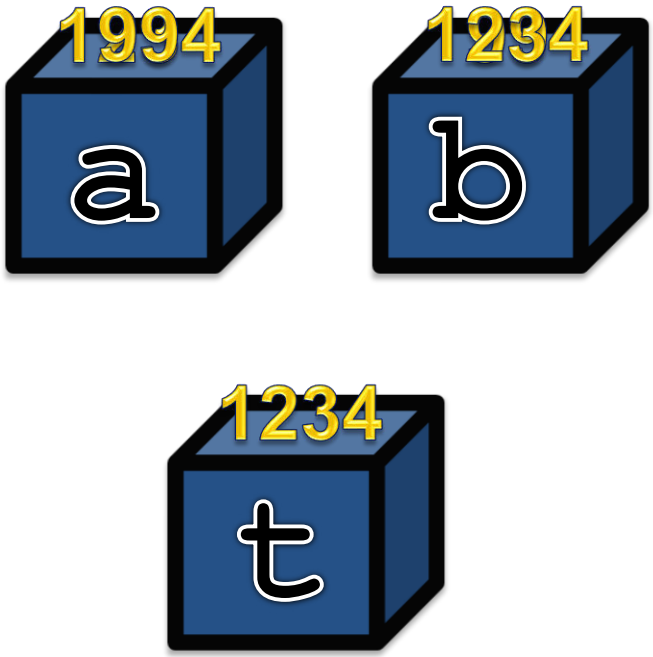
Assignment

Test with "pseudo-java"



"=" *stores* a value in a variable

It is <u>not</u> for comparison,
as in standard math



Penn Engineering

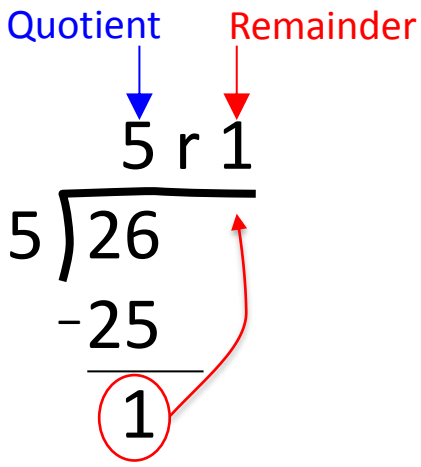
`int`: Integers (whole numbers)

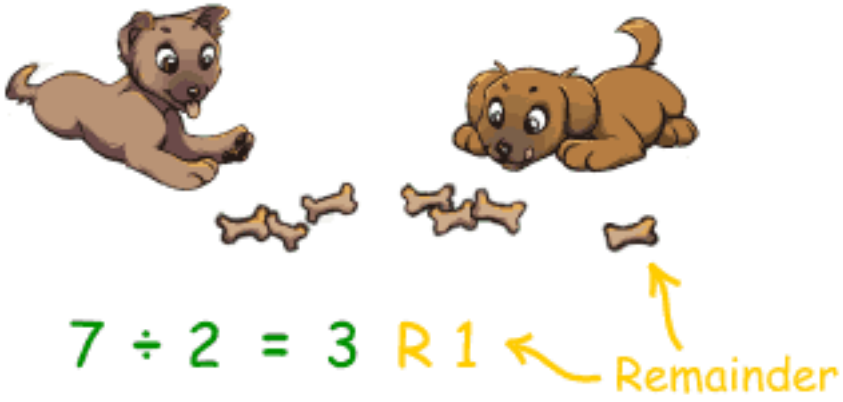
`+`, `-`, `*`, `/`, `%` (modulo), `()`, `Integer.parseInt()`

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

Penn Engineering

Modulo Operator (%)





Division gives the quotient:

26 / 5 == 5

Modulo gives the remainder:

26 % 5 == 1

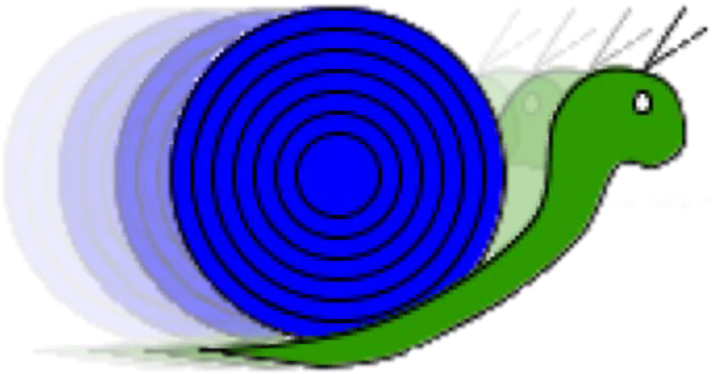
Example: Determining whether an integer `n` is even or odd:

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

Penn Engineering

Dog image from http://www.mathsisfun.com/

Animation



In-Class Demo

Animation

- Draw the same thing in slightly different position
- Frame rate
- Putting the background command in setup or draw?

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

Doubles: Example Program

/Users/bjbrown/introcs/Quadratic.java

New | Open | Save | Close | Cut | Copy | Paste | Undo

```
1  public class Quadratic {
2     public static void main(String[] args) {
3
4        // parse coefficients from command-line
5        double b = Double.parseDouble(args[0]);
6        double c = Double.parseDouble(args[1]);
7
8        // calculate roots
9        double discriminant = b*b - 4.0*c;
10       double d = Math.sqrt(discriminant);
11       double root1 = (-b + d) / 2.0;
12       double root2 = (-b - d) / 2.0;
13
14       // print them out
15       System.out.println(root1);
16       System.out.println(root2);
17    }
18 }
19
```

Interactions | Console | Compiler Output

```
> java Quadratic 2 1
-1.0
-1.0
> java Quadratic 1 2
NaN
NaN
>
```

Editing /Users/bjbrown/introcs/Quadratic.java 19:0

Download Quadratic.java from booksite, section 1.2

Solve : $x^2 + bx + c = 0$

Quadratic Formula : $\frac{-b \pm \sqrt{b^2 - 4c}}{2}$

Penn Engineering

Section 1.2

Java Math Library (Excerpts)

`public class Math`

`double abs(double a)`	*absolute value of a*
`double max(double a, double b)`	*maximum of a and b*
`double min(double a, double b)`	*minimum of a and b*

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

`double sin(double theta)`	*sine function*
`double cos(double theta)`	*cosine function*
`double tan(double theta)`	*tangent function*

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)`	*exponential* (e^a)
`double log(double a)`	*natural log* ($\log_e a$, *or* $\ln a$)
`double pow(double a, double b)`	*raise a to the bth power* (a^b)
`long round(double a)`	*round to the nearest integer*
`double random()`	*random number in* [0, 1)
`double sqrt(double a)`	*square root of a*
`double E`	*value of e (constant)*
`double PI`	*value of* π *(constant)*

Penn Engineering

`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.

Penn Engineering

`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))	

Penn Engineering

`String`: Text

Expression	Result?
"This is a string literal."	
"1" + "2"	
1 + " + " + 2 + " = " + 3	
'1' + "2"	
0 + '1' + "2"	
"" + Math.sqrt(2)	
(String) Math.sqrt(2)	
`(string) Math.sqrt(2)`	
"A" == "A"	
"A".equals("A")	
"B" < "A"	
"B".compareTo("A")	
"B".compareTo("B")	
"B".compareTo("C")	

Penn Engineering

Strings: Example Program

/Users/bjbrown/introcs/Ruler.java

New Open Save Close Cut Copy Paste Undo

```
1  public class Ruler {
2     public static void main(String[] args) {
3        String ruler1 = "1";
4        String ruler2 = ruler1 + " 2 " + ruler1;
5        String ruler3 = ruler2 + " 3 " + ruler2;
6        String ruler4 = ruler3 + " 4 " + ruler3;
7        System.out.println(ruler4);
8     }
9  }
10
```

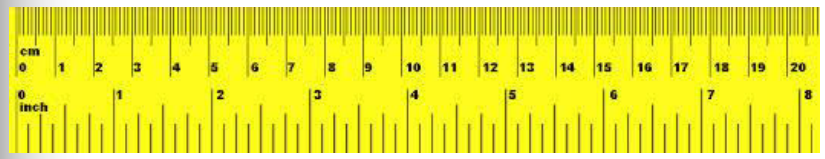
Interactions | Console | Compiler Output

```
Welcome to DrJava.  Working directory is /Users/bjbrown/introcs
> java Ruler
1 2 1 3 1 2 1 4 1 2 1 3 1 2 1
>
```

Editing /Users/bjbrown/introcs/Ruler.java 9:0

Download `Ruler.java` from booksite, section 1.2

1 2 1 3 1 2 1 4 1 2 1 3 1 2 1



Penn Engineering

Data Type Conversion

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

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

- The Processing Library 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'
```

Penn Engineering