

An Introduction to Processing

Variables, Data Types & Arithmetic Operators

Produced Dr. Siobhán Drohan
by: Mr. Colm Dunphy
 Mr. Diarmuid O'Connor



Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
<http://www.wit.ie/>

Topics list

1. Variables.
2. Assignment statement.
3. Data Types.
4. Java's Primitive Data Types
 1. Whole numbers.
 2. Decimal numbers.
 3. Others.
5. Arithmetic operators.

Variables

In Programming, variables:

- are created (defined) in your programs.
- are used to store data (whose value can change over time).
- have a data type.
- have a name.
- are a **VERY** important programming concept.

Variable names...

- Are case-sensitive.
- Begin with either:
 - a **letter (preferable)**,
 - the dollar sign "\$", or
 - the underscore character "_".
- Can contain letters, digits, dollar signs, or underscore characters.
- Can be any length you choose.
- Must not be a **keyword or reserved word** e.g. int, while, etc.
- Cannot contain white spaces.

Variable names should be carefully chosen

- Use full words instead of cryptic abbreviations e.g.
 - variables named **speed** and **gear** are much more intuitive than abbreviated versions, such as **s** and **g**.
- If the name consists of:
 - only one word,
 - spell that word in all lowercase letters e.g. **ratio**.
 - more than one word,
 - capitalise the first letter of each subsequent word e.g. **gearRatio** and **currentGear**.
 - This is called **camelCase**

Topics list

1. Variables.
2. Assignment statement.
3. Data Types.
4. Java's Primitive Data Types
 1. Whole numbers.
 2. Decimal numbers.
 3. Others.
5. Arithmetic operators.

Assignment Statement

- Values are stored in variables via assignment statements:

| | |
|---------|------------------------|
| Syntax | variable = expression; |
| Example | diameter = 100; |

- A variable stores a single value, so any previous value is lost.
- Assignment statements work by taking the value of what appears on the right-hand side of the operator and copying that value into a variable on the left-hand side.

Topics list

1. Variables.
2. Assignment statement.

- 3. Data Types.

4. Java's Primitive Data Types
 1. Whole numbers.
 2. Decimal numbers.
 3. Others.
5. Arithmetic operators.

Data Types

- In Java, when we define a variable, we have to give it a data type.
- The data type defines the kinds of values (data) that can be stored in the variable e.g.
 - - 456
 - 2
 - 45.7897
 - I Love Programming
 - S
 - true
- The data type also determines the operations that may be performed on it.

Data Types

- Java uses two kinds of data types:
 - **Primitive** types
 - **Object** types
- We are only looking at **Primitive** types now; we will cover Object types later in the module.

Topics list

1. Variables.
2. Assignment statement.
3. Data Types.
4. Java's Primitive Data Types
 1. Whole numbers.
 2. Decimal numbers.
 3. Others.
5. Arithmetic operators.

Java's Primitive Data Types

- Java programming language supports eight primitive data types.
- A primitive type is predefined by the language and is named by a reserved keyword.
- A primitive type is highlighted red when it is typed into the PDE e.g.

```
int numberOfItems;  
boolean bounceUp;  
float lengthOfRectangle;
```

Topics list

1. Variables.
2. Assignment statement.
3. Data Types.
4. Java's Primitive Data Types
 1. Whole numbers.
 2. Decimal numbers.
 3. Others.
5. Arithmetic operators.



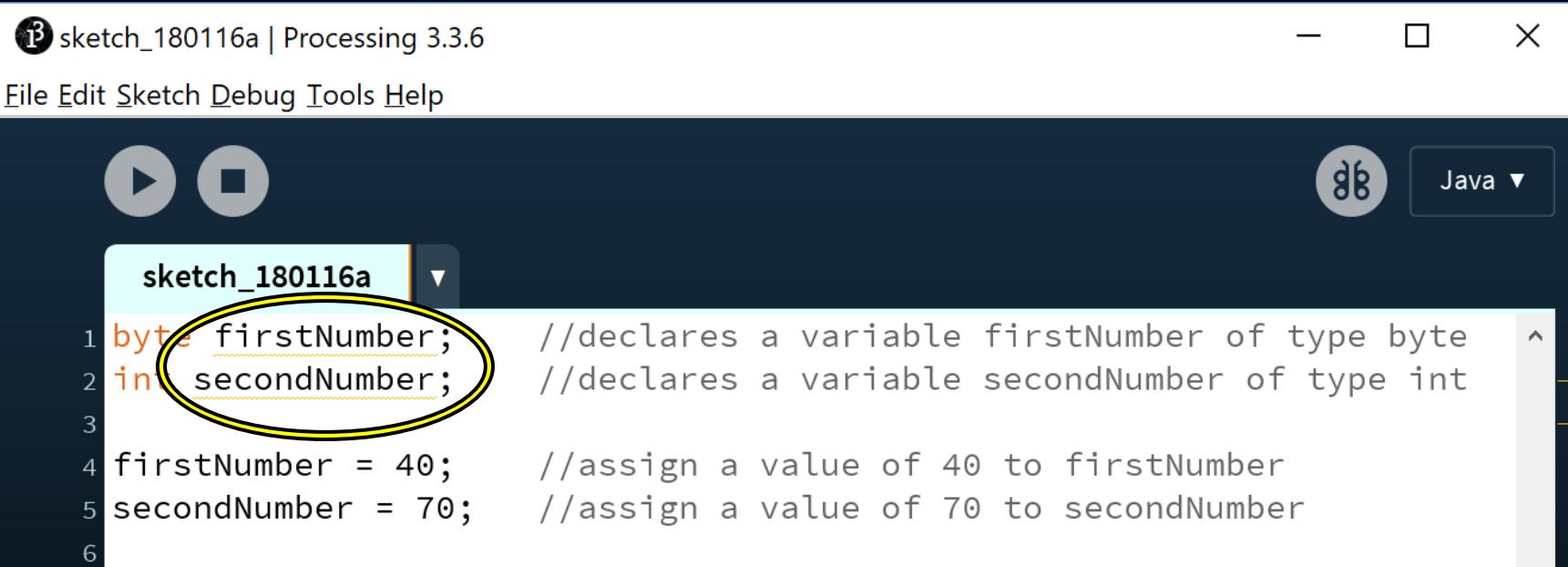
Java's Primitive Data Types (whole numbers)

| Type | Byte-size | Minimum value (inclusive) | Maximum value (inclusive) | Typical Use |
|-------|-----------|------------------------------------|-------------------------------|---|
| byte | 8-bit | -128 | 127 | Useful in applications where memory savings apply. |
| short | 16-bit | -32,768 | 32,767 | memory savings apply. |
| int | 32-bit | -2,147,483,648 | 2,147,483,647 | Default choice. |
| long | 64-bit | - 9,223,372,036, 854,775,808 | 9,223,372,036, 854,775,807 | Used when you need a data type with a range of values larger than that provided by int. |

| $2^{\text{Number of Bits}}$ | Range of values |
|-----------------------------|-----------------|
| 2^0 | 1 |
| 2^1 | 2 |
| 2^2 | 4 |
| 2^3 | 8 |
| 2^4 | 16 |
| 2^5 | 32 |
| 2^6 | 64 |
| 2^7 | 128 |
| 2^8 | 256 |

If the eight bit is used for the sign, we get a range from -128 to +127
 i.e. $256/2 = +128$ values,
 but a value is required to store 0, so range is **-128 to +127**

Declaring variables of a specific type



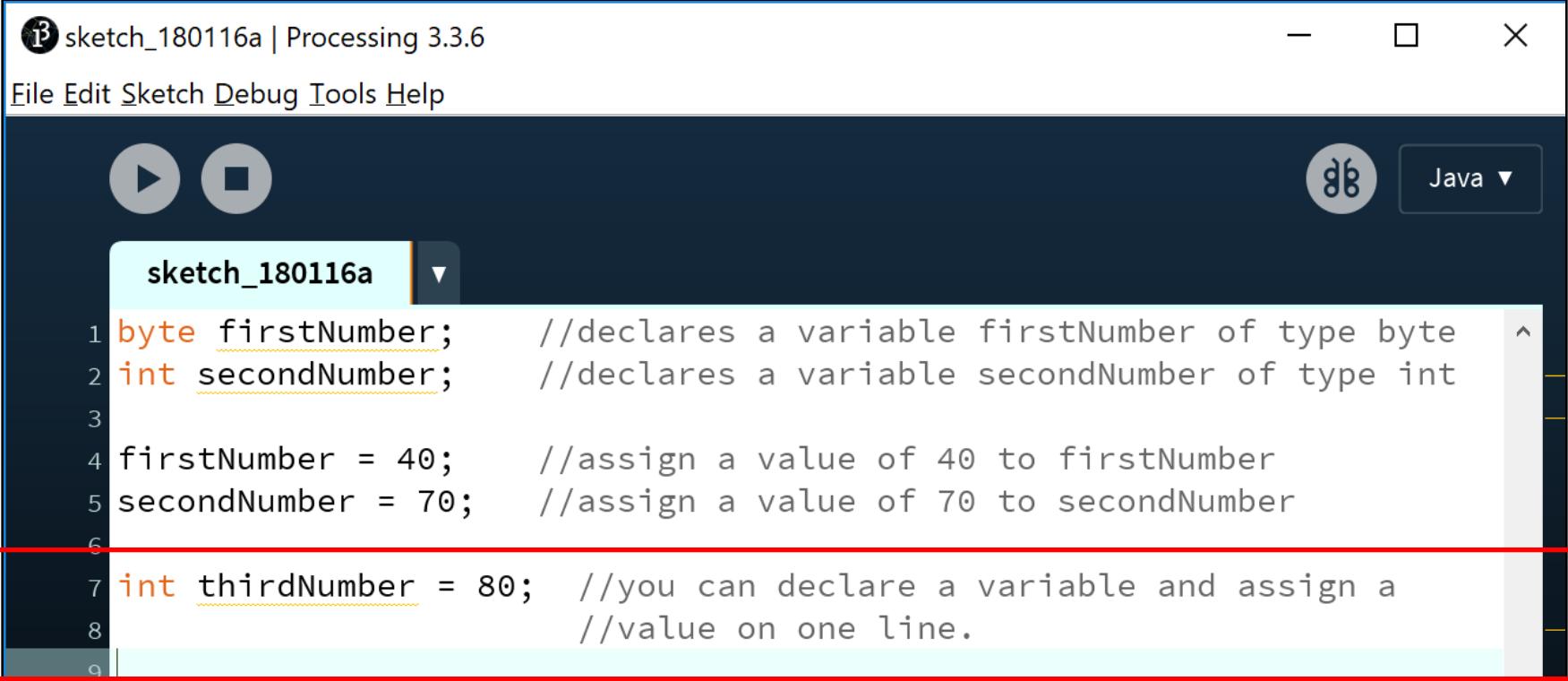
The screenshot shows the Processing IDE interface with the title "sketch_180116a | Processing 3.3.6". The code editor contains the following Java code:

```
1 byte firstNumber;           //declares a variable firstNumber of type byte
2 int secondNumber;           //declares a variable secondNumber of type int
3
4 firstNumber = 40;           //assign a value of 40 to firstNumber
5 secondNumber = 70;           //assign a value of 70 to secondNumber
6
```

A yellow oval highlights the declarations of `firstNumber` and `secondNumber`. The word `int` has a yellow underline, indicating it hasn't been used meaningfully.

YELLOW underline - indicates that the variable hasn't been used meaningfully

Declaring variables of a specific type



The screenshot shows the Processing IDE interface with the title "sketch_180116a | Processing 3.3.6". The code editor displays the following Java code:

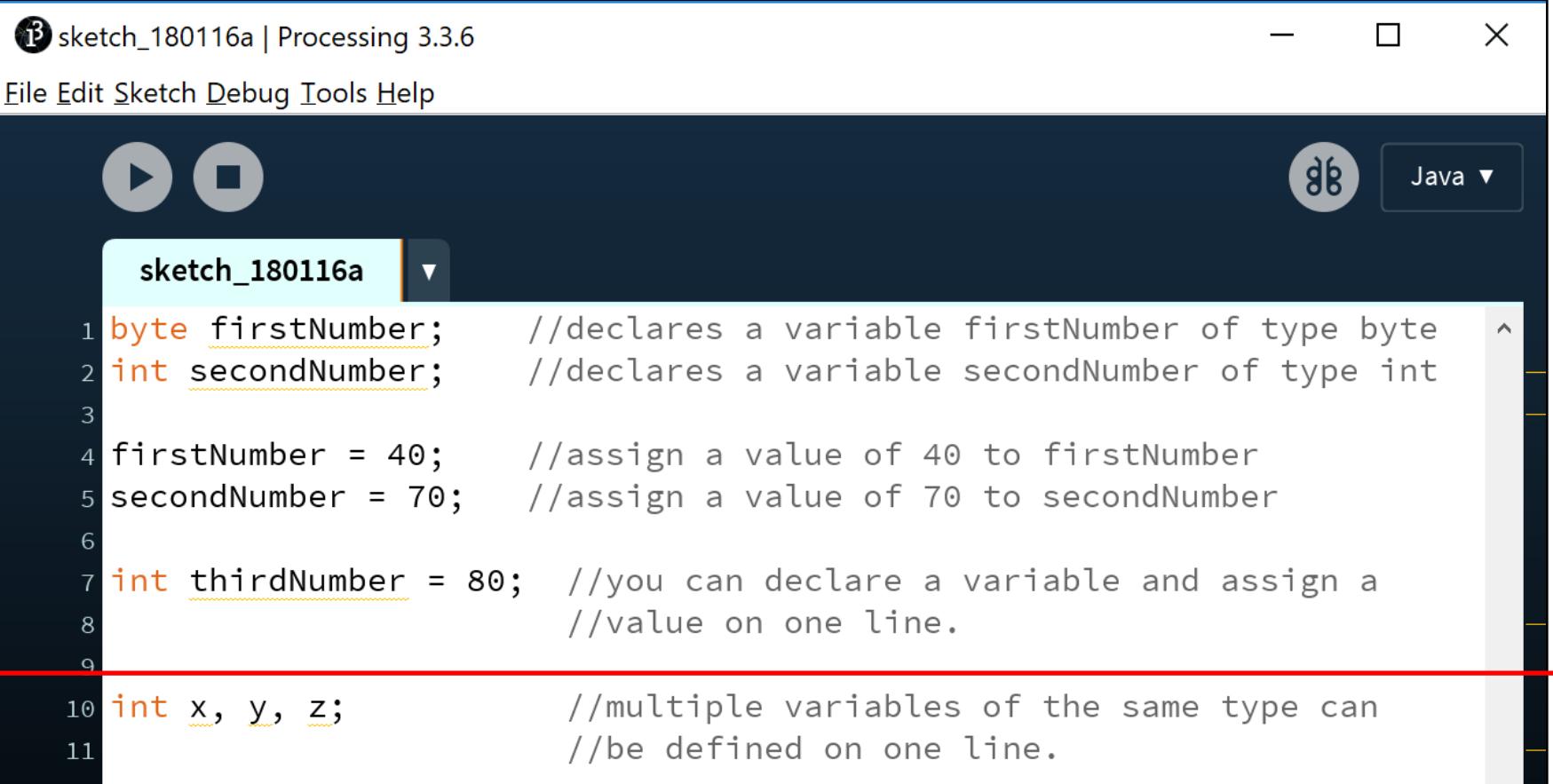
```
byte firstNumber; //declares a variable firstNumber of type byte
int secondNumber; //declares a variable secondNumber of type int

firstNumber = 40; //assign a value of 40 to firstNumber
secondNumber = 70; //assign a value of 70 to secondNumber

int thirdNumber = 80; //you can declare a variable and assign a
                     //value on one line.
```

A red rectangular box highlights the last line of code, specifically the declaration and assignment of the variable `thirdNumber`.

Declaring variables of a specific type



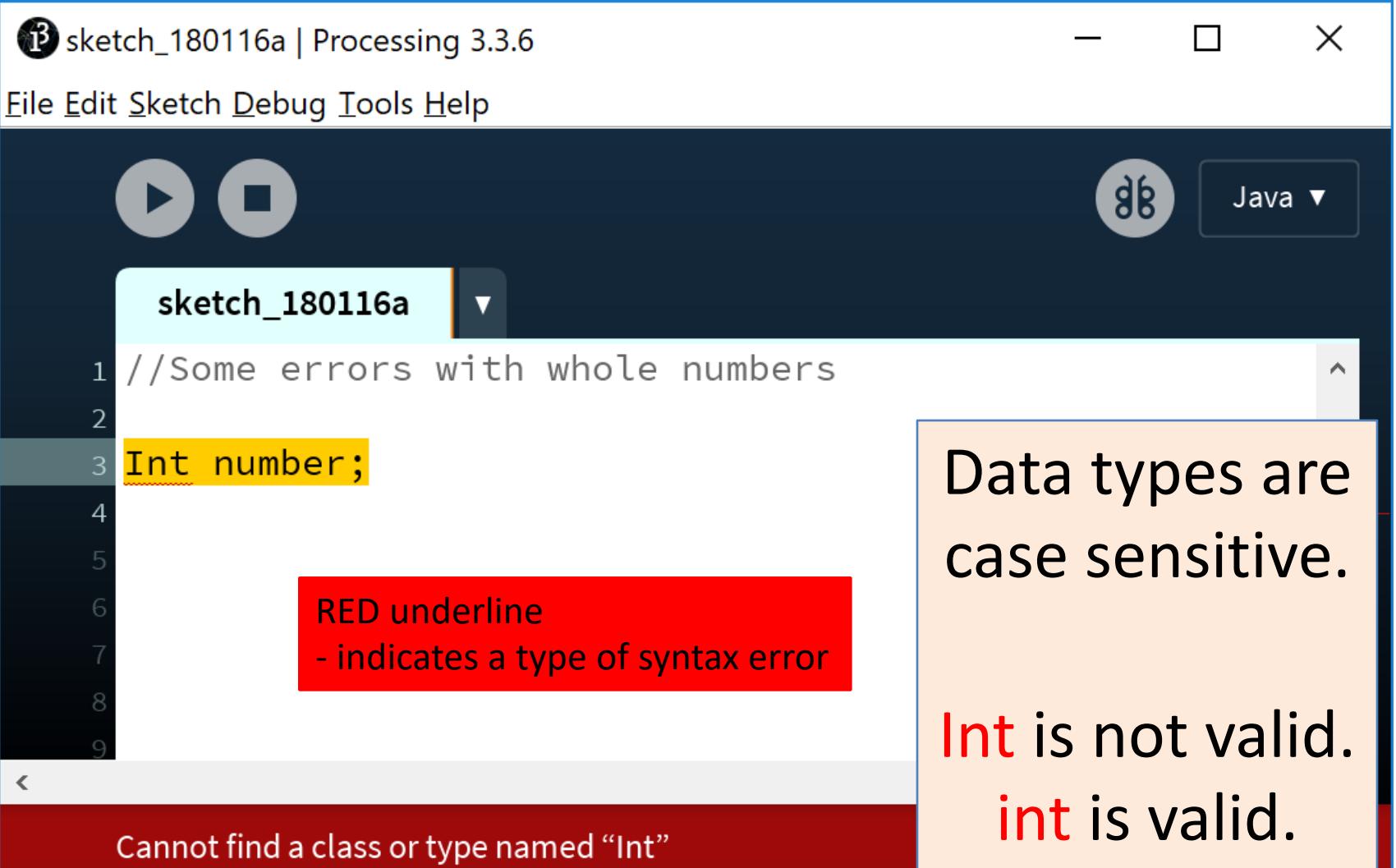
The screenshot shows the Processing IDE interface with the title bar "sketch_180116a | Processing 3.3.6". The menu bar includes File, Edit, Sketch, Debug, Tools, and Help. The toolbar features a play button, a stop button, and a Java dropdown menu. The code editor displays the following Java code:

```
sketch_180116a

1 byte firstNumber;      //declares a variable firstNumber of type byte
2 int secondNumber;      //declares a variable secondNumber of type int
3
4 firstNumber = 40;       //assign a value of 40 to firstNumber
5 secondNumber = 70;      //assign a value of 70 to secondNumber
6
7 int thirdNumber = 80;   //you can declare a variable and assign a
                           //value on one line.
8
9
10 int x, y, z;           //multiple variables of the same type can
                           //be defined on one line.
```

The code uses the Java syntax for variable declaration and assignment. Lines 10 and 11 demonstrate how multiple variables of the same type can be declared on a single line. The entire line 10 is highlighted with a red rectangle.

Declaring variables - some errors



The screenshot shows the Processing IDE interface. The title bar reads "sketch_180116a | Processing 3.3.6". The menu bar includes File, Edit, Sketch, Debug, Tools, and Help. On the left, there are play and stop buttons. On the right, there is a "Java" dropdown menu. The code editor window displays a sketch named "sketch_180116a". The code is as follows:

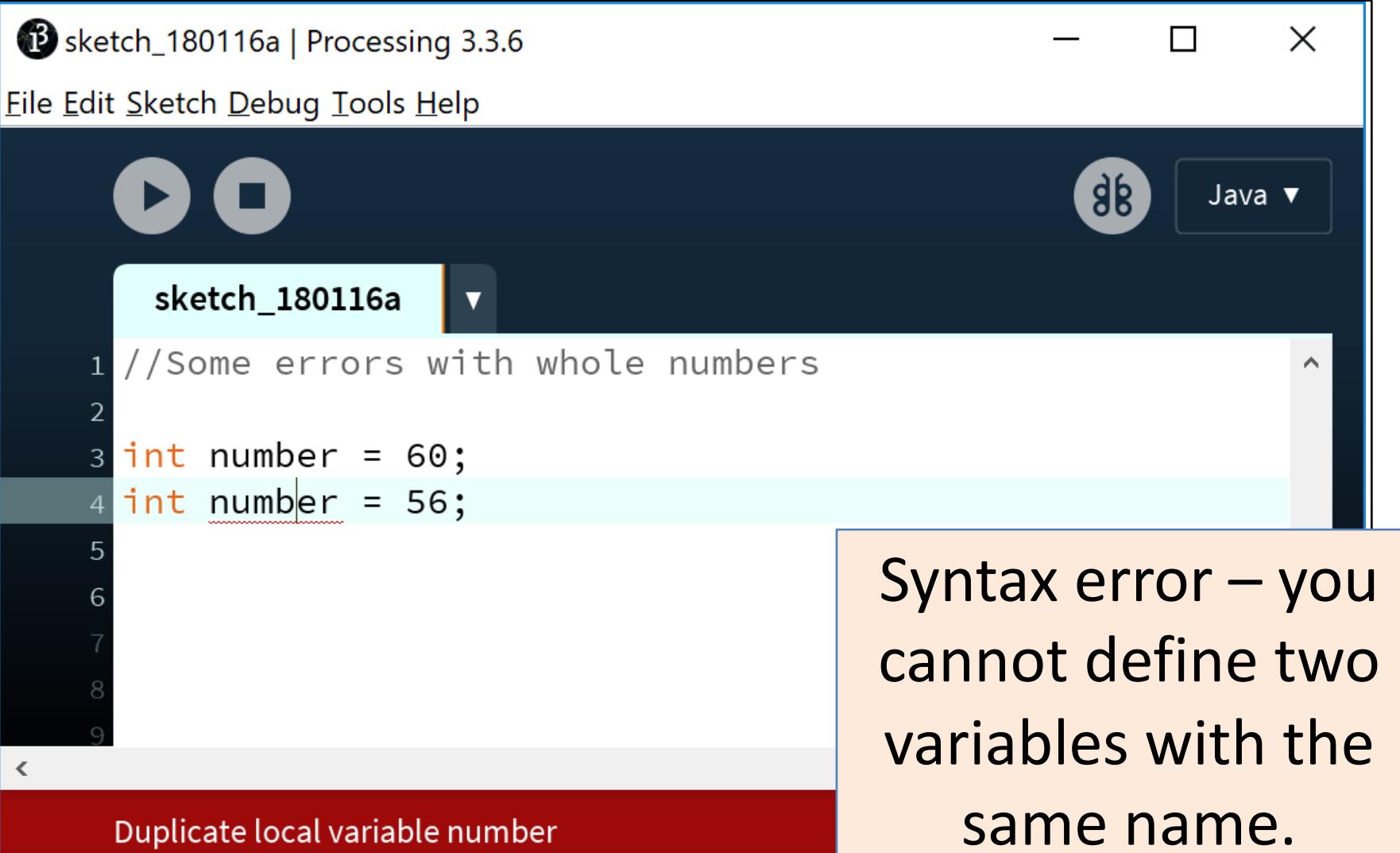
```
1 //Some errors with whole numbers
2
3 Int number;
```

A red box highlights the word "Int" in line 3, with the text "RED underline - indicates a type of syntax error" overlaid. A red banner at the bottom states "Cannot find a class or type named ‘Int’".

Data types are case sensitive.

Int is not valid.
int is valid.

Declaring variables - some errors



The screenshot shows the Processing IDE interface. The title bar reads "sketch_180116a | Processing 3.3.6". The menu bar includes "File", "Edit", "Sketch", "Debug", "Tools", and "Help". On the left, there are play and stop buttons. On the right, there is a logo and a "Java" dropdown. The code editor window displays the following code:

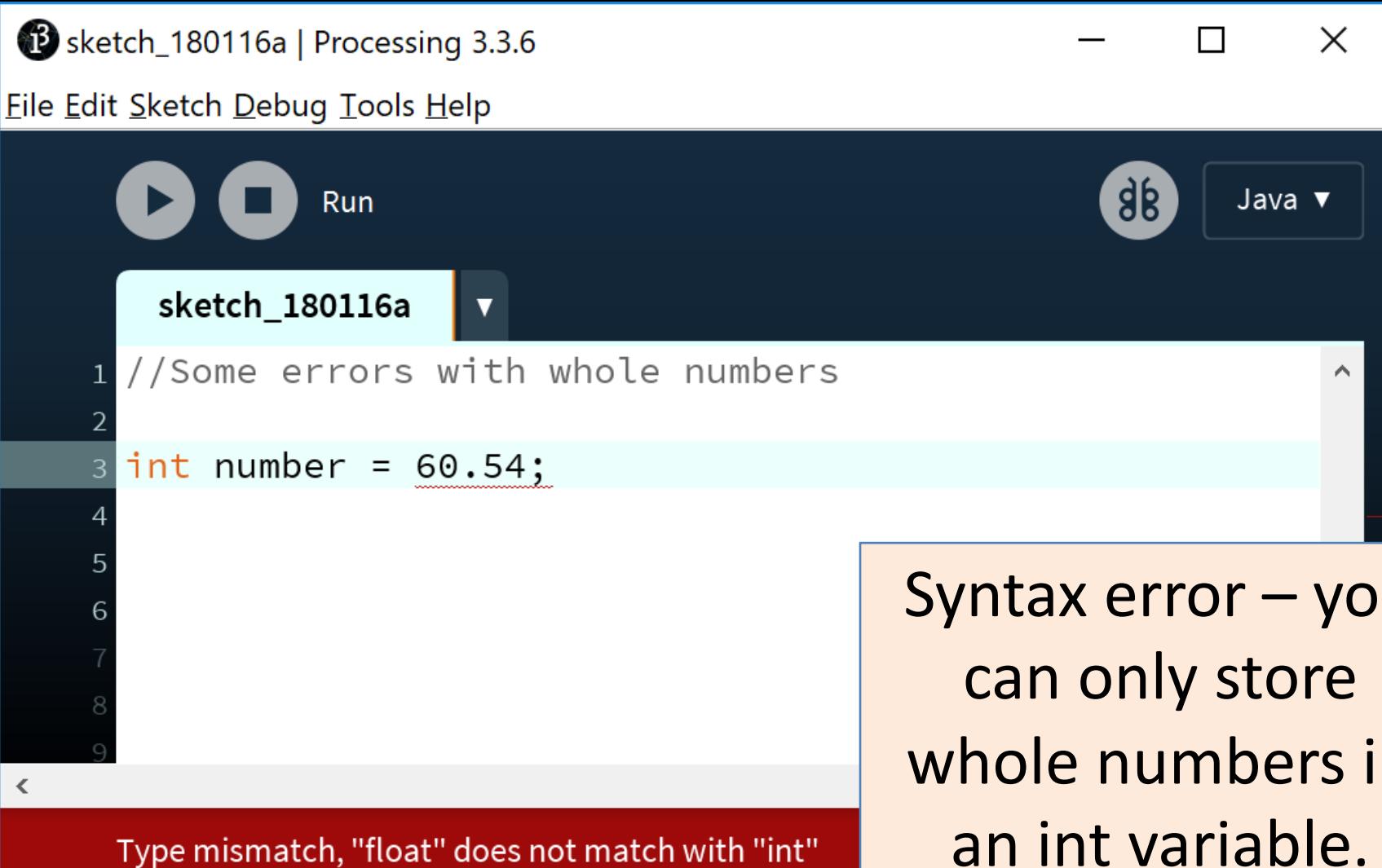
```
1 //Some errors with whole numbers
2
3 int number = 60;
4 int number = 56;
```

A red box highlights the second line of code, "int number = 56;". A red banner at the bottom of the code editor window displays the error message "Duplicate local variable number". To the right of the code editor, a callout box contains the explanatory text: "Syntax error – you cannot define two variables with the same name."

Duplicate local variable number

Syntax error – you cannot define two variables with the same name.

Declaring variables - some errors



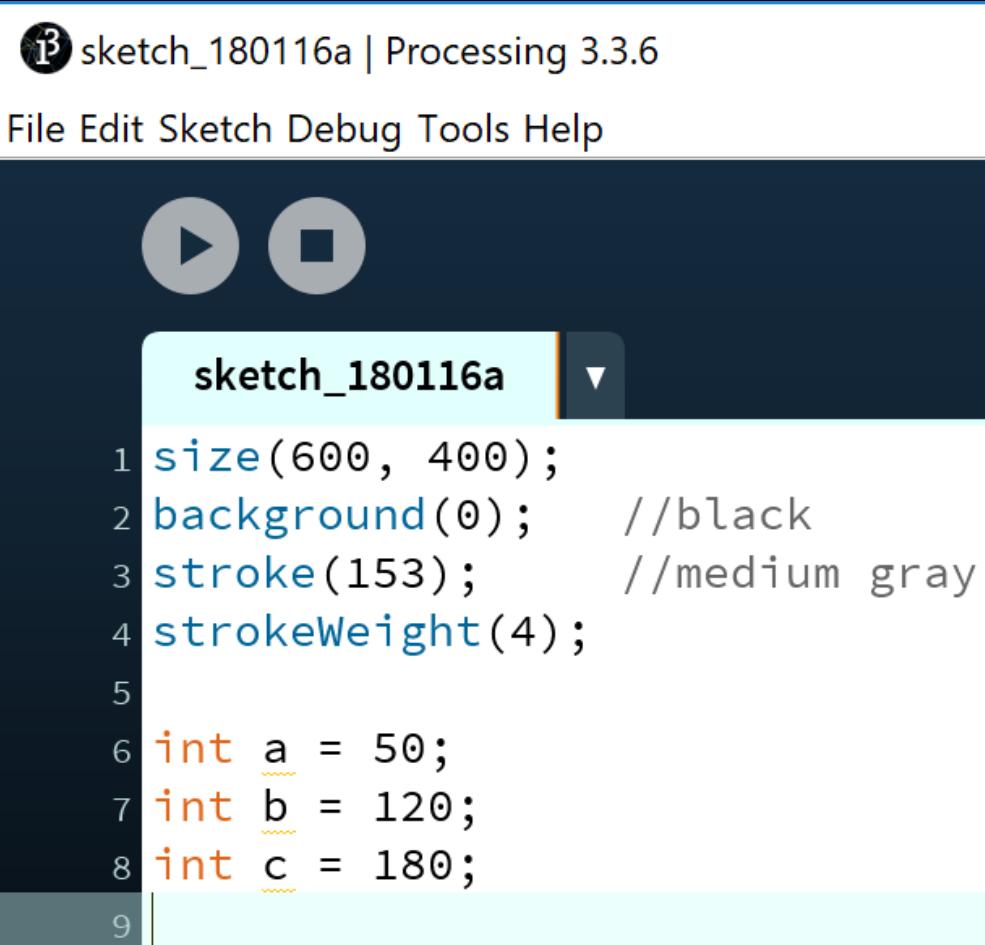
The screenshot shows the Processing IDE interface. The title bar reads "sketch_180116a | Processing 3.3.6". The menu bar includes File, Edit, Sketch, Debug, Tools, and Help. Below the menu is a toolbar with Run and Java buttons. The code editor window displays a sketch named "sketch_180116a". The code is as follows:

```
1 //Some errors with whole numbers
2
3 int number = 60.54;
```

A red error message at the bottom of the code editor states "Type mismatch, 'float' does not match with 'int'". A callout box on the right side of the screen contains the explanatory text: "Syntax error – you can only store whole numbers in an int variable."

Syntax error – you
can only store
whole numbers in
an int variable.

Java's Primitive Data Types: int example



The screenshot shows the Processing software interface with the title bar "sketch_180116a | Processing 3.3.6". Below the title bar is a menu bar with "File", "Edit", "Sketch", "Debug", "Tools", and "Help". On the left side of the interface, there are two large buttons: a play button and a square button. Below these buttons is a dropdown menu labeled "sketch_180116a" with a downward arrow. The main area contains the following Java code:

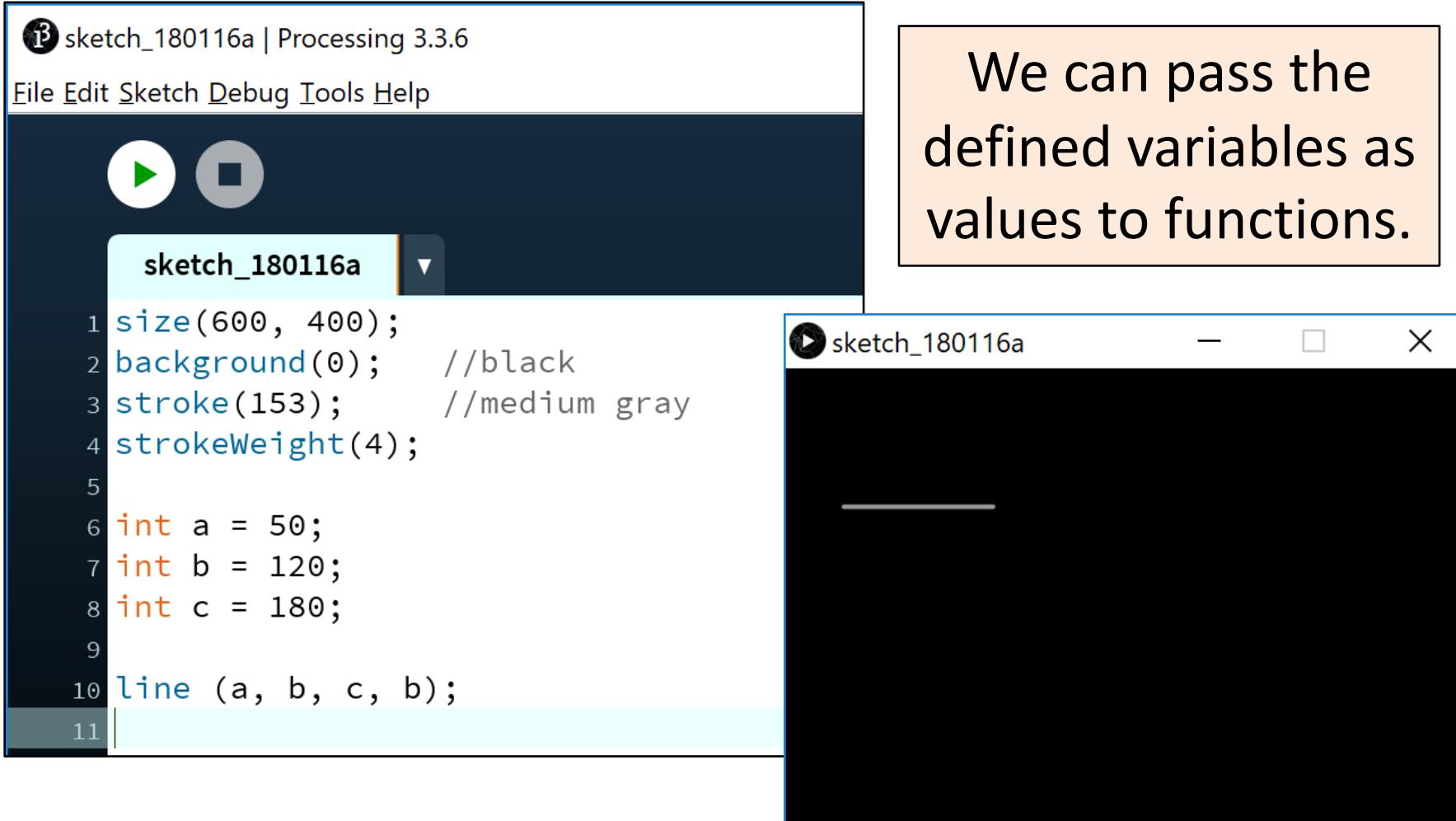
```
1 size(600, 400);
2 background(0);      //black
3 stroke(153);        //medium gray
4 strokeWeight(4);

5
6 int a = 50;
7 int b = 120;
8 int c = 180;
```

To the right of the code, a callout box with a light orange background contains the text: "In this example, we have:" followed by a bulleted list.

- defined three variables (a, b and c)
- that can hold whole numbers (int).
- and are set with a starting value.

Java's Primitive Data Types: int example



The image shows a Processing sketch window titled "sketch_180116a" running on version 3.3.6. The code in the editor is:

```
1 size(600, 400);
2 background(0);      //black
3 stroke(153);       //medium gray
4 strokeWeight(4);

5

6 int a = 50;
7 int b = 120;
8 int c = 180;

9

10 line (a, b, c, b);
11
```

The output window shows a black canvas with a single horizontal line segment drawn from point (50, 120) to point (180, 120).

We can pass the defined variables as values to functions.

Based on the Processing Example: Basics → Data → Variables

Java's Primitive Data Types: int example

```
sketch_180116a
size(600, 400);
background(0); //black
stroke(153); //medium gray
strokeWeight(4);
```

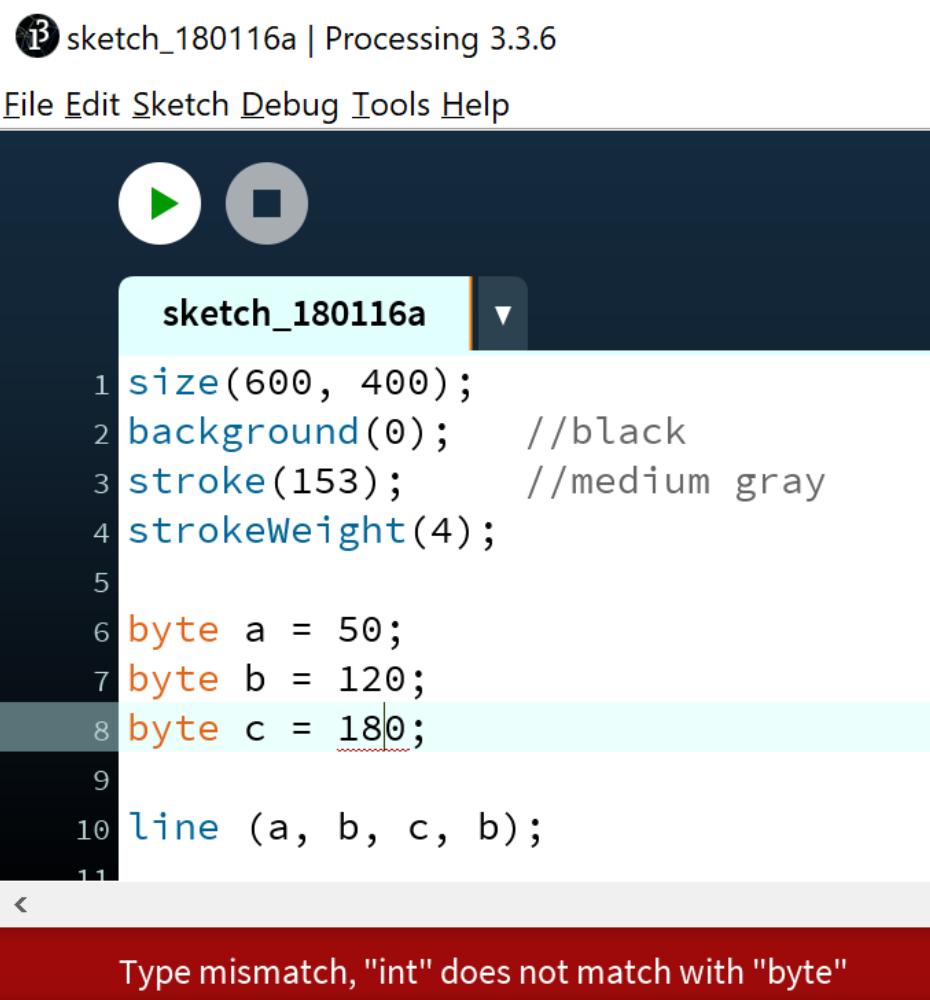
```
int a = 50;
int b = 120;
int c = 180;
```

```
line (a, b, c, b);
```

Q: Could we have used the *byte* data type instead of *int*?

| Type | Minimum value (inclusive) | Maximum value (inclusive) |
|-------|----------------------------|---------------------------|
| byte | -128 | 127 |
| short | -32,768 | 32,767 |
| int | -2,147,483,648 | 2,147,483,647 |
| long | -9,223,372,036,854,775,808 | 9,223,372,036,854,775,807 |

Java's Primitive Data Types: int example



```
sketch_180116a | Processing 3.3.6
File Edit Sketch Debug Tools Help
sketch_180116a ▾
1 size(600, 400);
2 background(0);      //black
3 stroke(153);       //medium gray
4 strokeWeight(4);
5
6 byte a = 50;
7 byte b = 120;
8 byte c = 180;
9
10 line (a, b, c, b);
11
```

Type mismatch, "int" does not match with "byte"

Q: Could we have used the **byte** data type instead of **int**?

A: For **a** and **b** we could have; 50 and 120 fall below the max value of 127. But **c** produces a syntax error; 180 cannot fit into a 127 capacity variable.

| Type | Min value | Max value |
|-------|-----------|-----------|
| byte | -128 | 127 |
| short | -32,768 | 32,767 |

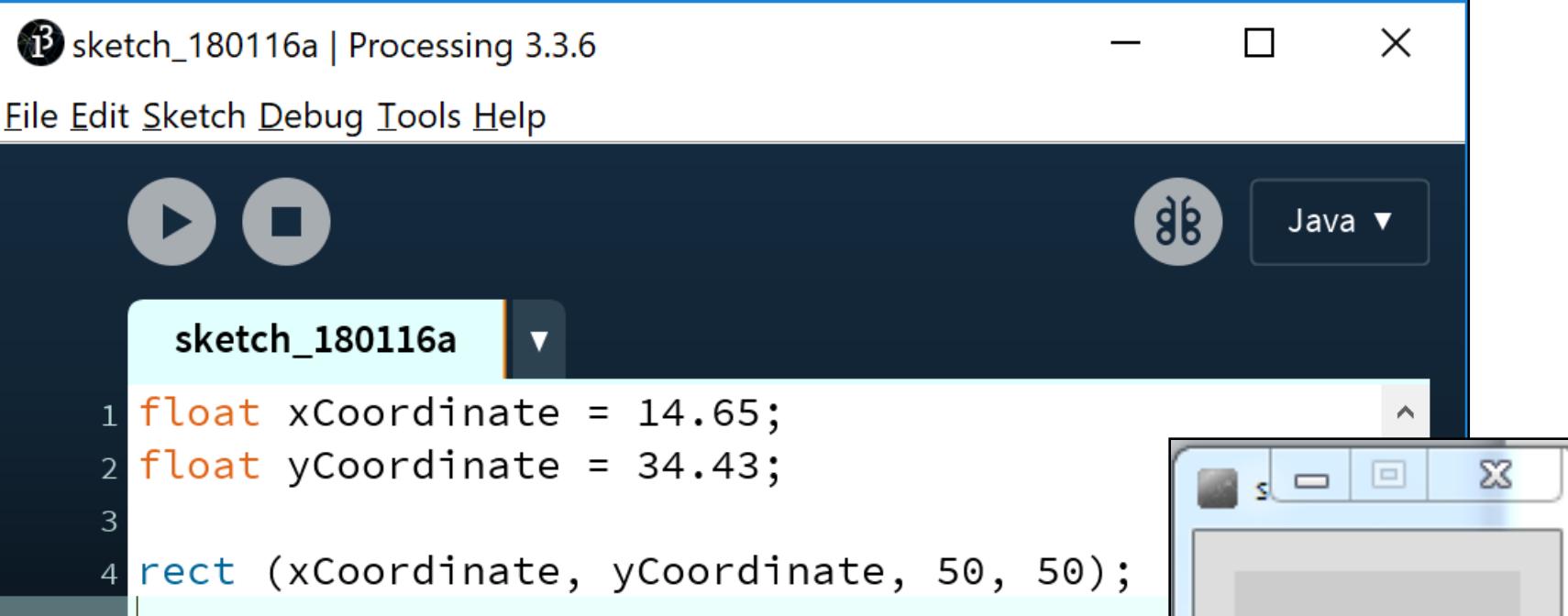
Topics list

1. Variables.
2. Assignment statement.
3. Data Types.
4. Java's Primitive Data Types
 1. Whole numbers.
 2. Decimal numbers. 
 3. Others.
5. Arithmetic operators.

Java's Primitive Data Types (decimal numbers)

| Type | Byte-size | Minimum value (inclusive) | Maximum value (inclusive) | Typical Use |
|--------|-----------|---|------------------------------|---|
| float | 32-bit | <i>Beyond the scope of this lecture .</i> <i>There is also a loss of precision in this data-type that we will cover in later lectures.</i> | | Useful in applications where memory savings apply. Default choice when using Processing . |
| double | 64-bit | | | Default choice when programming Java apps . |

Java's Primitive Data Types: float example



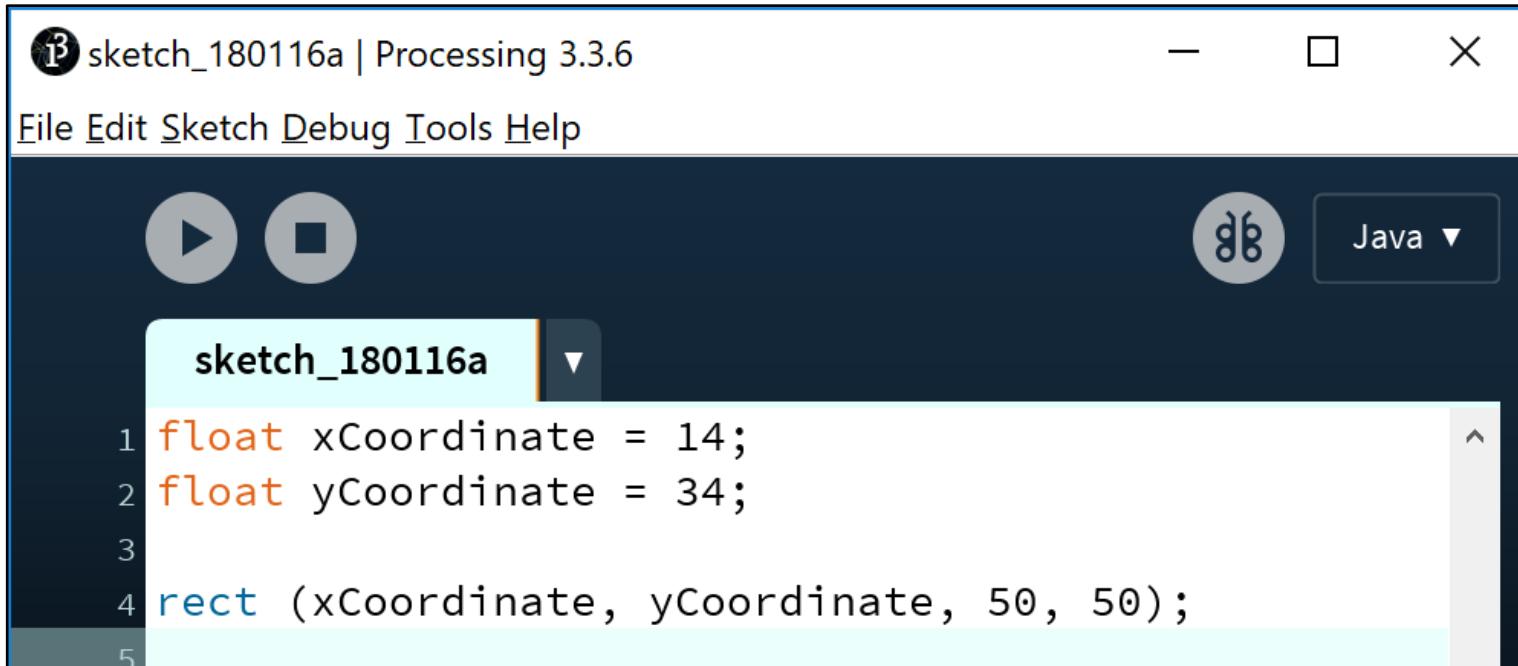
The screenshot shows the Processing IDE interface. The title bar reads "sketch_180116a | Processing 3.3.6". The menu bar includes "File", "Edit", "Sketch", "Debug", "Tools", and "Help". On the left, there are play and stop buttons. In the center, a dropdown menu shows "sketch_180116a" with a downward arrow. The code editor contains the following Java code:

```
1 float xCoordinate = 14.65;
2 float yCoordinate = 34.43;
3
4 rect (xCoordinate, yCoordinate, 50, 50);
```

To the right of the code editor is a preview window showing a small white square centered on a gray background.

We can pass the defined variables as values to functions.

Java's Primitive Data Types: float example



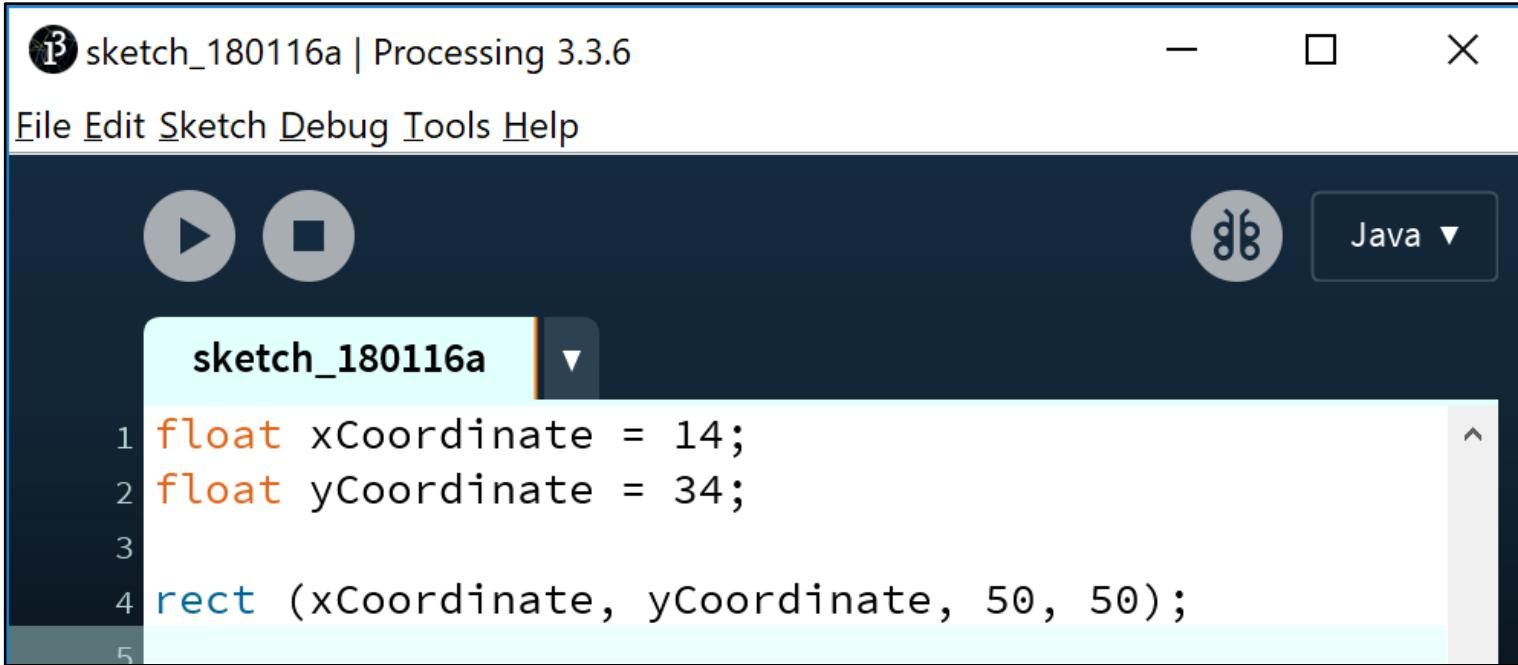
The screenshot shows the Processing IDE interface with the title "sketch_180116a | Processing 3.3.6". The menu bar includes File, Edit, Sketch, Debug, Tools, and Help. The toolbar features a play button, a square button, a logo, and a "Java" dropdown. The code editor displays the following Java code:

```
1 float xCoordinate = 14;  
2 float yCoordinate = 34;  
3  
4 rect (xCoordinate, yCoordinate, 50, 50);  
5
```

Whole numbers can be placed into a **float** variable.

Q: Why?

Java's Primitive Data Types: float example



The screenshot shows the Processing IDE interface with the title "sketch_180116a | Processing 3.3.6". The menu bar includes File, Edit, Sketch, Debug, Tools, and Help. The sketch window displays a play button, a square button, a logo, and a "Java" dropdown. The code area contains the following Java code:

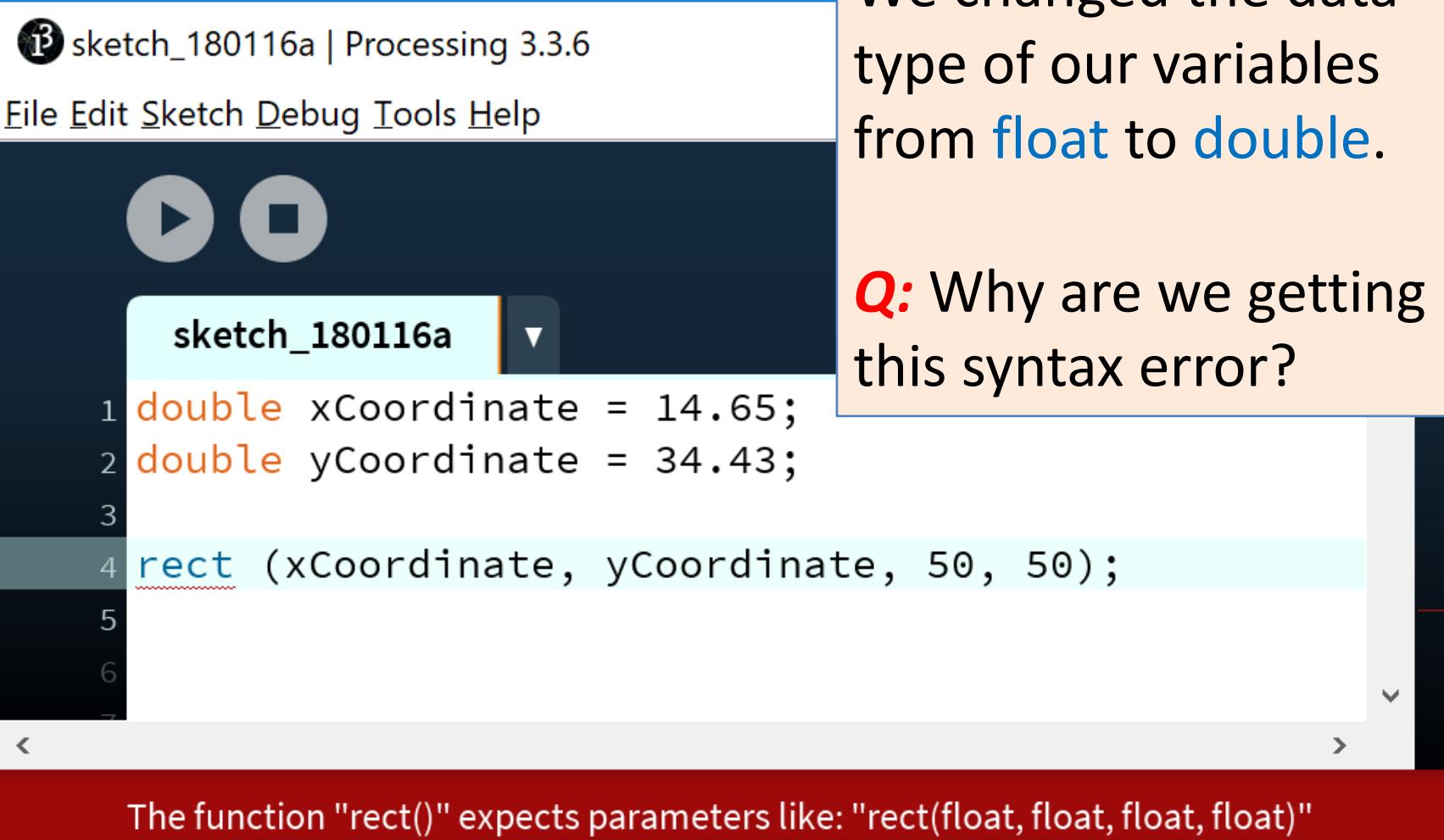
```
1 float xCoordinate = 14;  
2 float yCoordinate = 34;  
3  
4 rect (xCoordinate, yCoordinate, 50, 50);  
5
```

Whole numbers can be placed into a **float** variable.

Q: Why?

A: There is no loss of precision. We are not losing any data.

Passing variables as arguments: some errors



The screenshot shows the Processing 3.3.6 IDE interface. The title bar says "sketch_180116a | Processing 3.3.6". The menu bar includes "File Edit Sketch Debug Tools Help". Below the menu is a toolbar with play and stop buttons. The code editor window contains the following code:

```
1 double xCoordinate = 14.65;
2 double yCoordinate = 34.43;
3
4 rect (xCoordinate, yCoordinate, 50, 50);
```

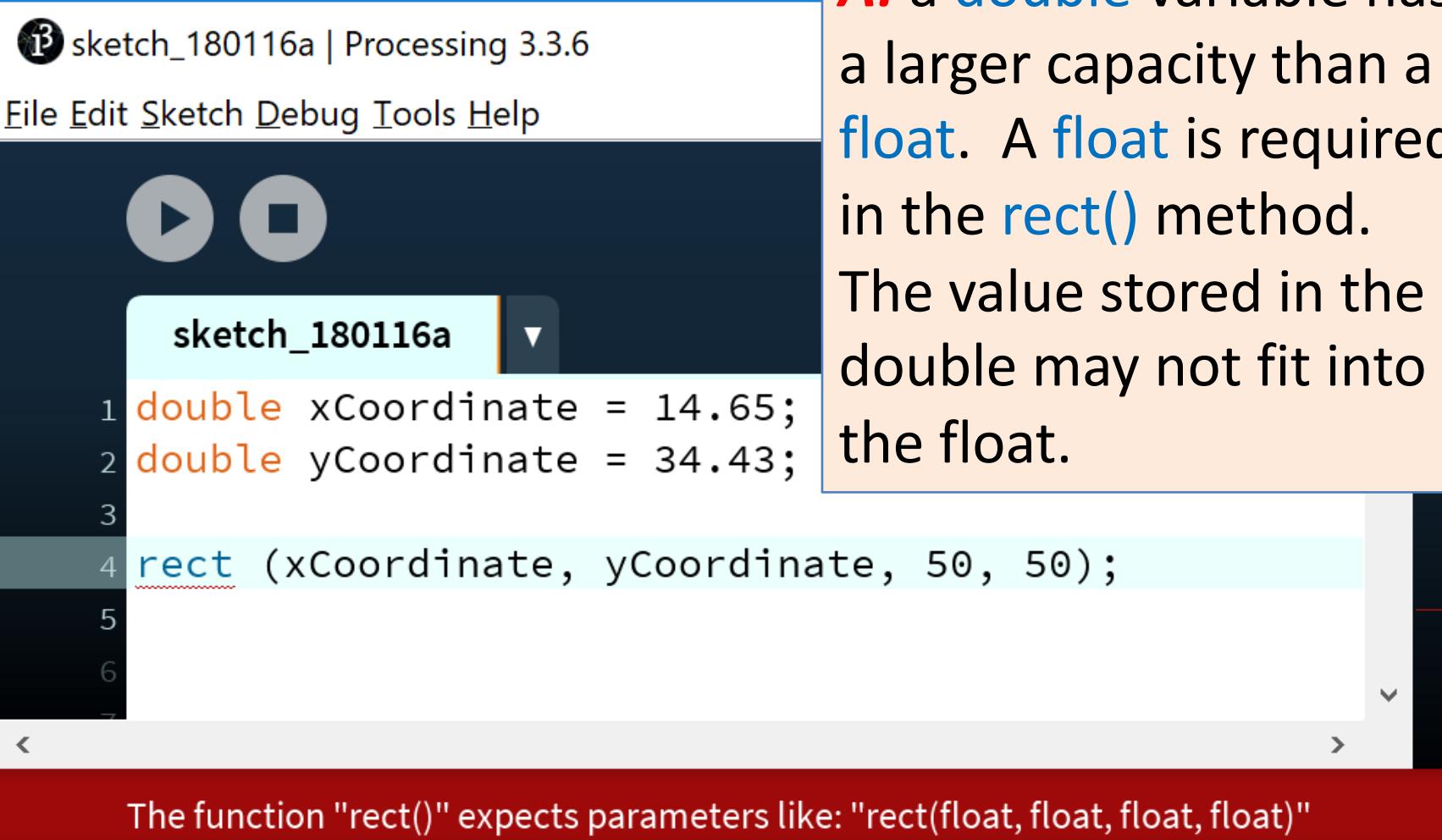
A red underline is under the word "rect" in line 4, indicating a syntax error. A tooltip box on the right side of the code editor contains the following text:

We changed the data type of our variables from **float** to **double**.

Q: Why are we getting this syntax error?

The status bar at the bottom of the IDE displays the message: "The function "rect()" expects parameters like: "rect(float, float, float, float)"

Passing variables as arguments: some errors



The screenshot shows the Processing 3.3.6 IDE interface. The title bar says "sketch_180116a | Processing 3.3.6". The menu bar includes "File Edit Sketch Debug Tools Help". Below the menu is a toolbar with play and stop buttons. The code editor window contains the following code:

```
1 double xCoordinate = 14.65;
2 double yCoordinate = 34.43;
3
4 rect (xCoordinate, yCoordinate, 50, 50);
```

A red box highlights the word "rect" in line 4. A tooltip message at the bottom of the screen reads: "The function "rect()" expects parameters like: "rect(float, float, float, float)"

A: a **double** variable has a larger capacity than a **float**. A **float** is required in the **rect()** method. The value stored in the double may not fit into the float.

Passing variables as arguments: some errors

From: https://processing.org/reference/rect_.html

Syntax

```
rect(a, b, c, d)
```

Parameters

| | |
|---|---|
| a | float: x-coordinate of the rectangle by default |
| b | float: y-coordinate of the rectangle by default |
| c | float: width of the rectangle by default |
| d | float: height of the rectangle by default |

```
double xCoordinate = 14.65;  
double yCoordinate = 34.43;
```

```
rect(xCoordinate, yCoordinate, 50, 50);
```



The function "rect()" expects parameters like: "rect(float, float, float, float)"

Topics list

1. Variables.
2. Assignment statement.
3. Data Types.
4. Java's Primitive Data Types
 1. Whole numbers.
 2. Decimal numbers.
 3. Others. 
5. Arithmetic operators.

Java's Primitive Data Types (others)

| Type | Byte-size | Minimum value (inclusive) | Maximum value (inclusive) | Typical Use |
|---------|-----------|---------------------------|---------------------------|---|
| char | 16-bit | '\u0000' (or 0) | '\uffff' (or 65,535). | Represents a Unicode character. |
| boolean | 1-bit | n/a | | Holds either true or false and is typically used as a flag. |

- We will go into more detail on these two data types in later lectures.

http://en.wikipedia.org/wiki/List_of_Unicode_characters

Java's Primitive Data Types (default values)

| Data Type | Default Value |
|-----------|---------------|
| byte | 0 |
| short | 0 |
| int | 0 |
| long | 0L |
| float | 0.0f |
| double | 0.0d |
| char | '\u0000' |
| boolean | false |

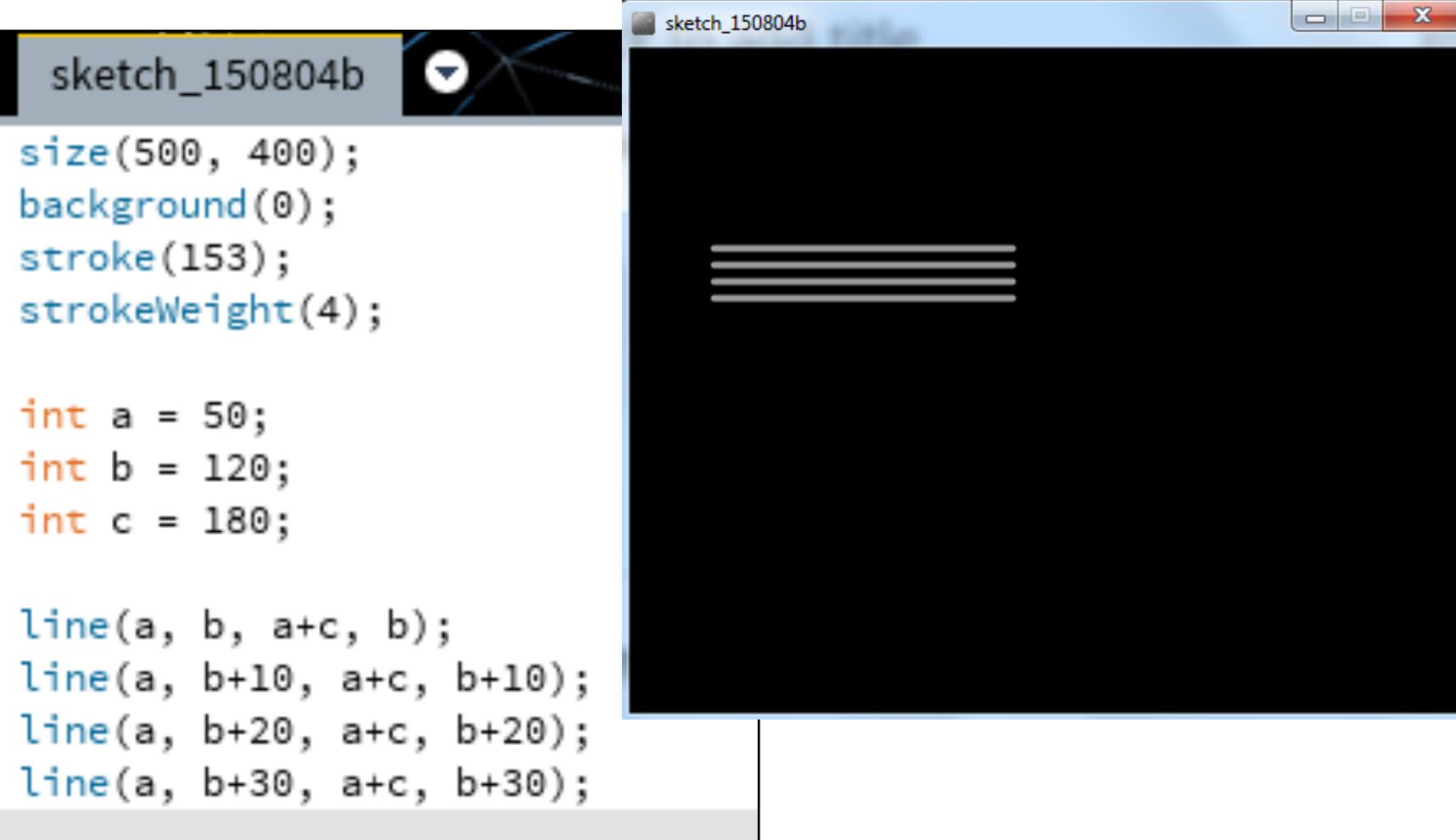
Topics list

1. Variables.
2. Assignment statement.
3. Data Types.
4. Java's Primitive Data Types
 1. Whole numbers.
 2. Decimal numbers.
 3. Others.
5. Arithmetic operators.

Arithmetic Operators

| Arithmetic Operator | Explanation | Example(s) |
|---------------------|----------------|----------------------------|
| + | Addition | $6 + 2$ amountOwed + 10 |
| - | Subtraction | $6 - 2$ amountOwed - 10 |
| * | Multiplication | $6 * 2$ amountOwed * 10 |
| / | Division | $6 / 2$ amountOwed / 10 |

Arithmetic operators: example 1



The image shows the Processing IDE interface. On the left is the code editor window titled "sketch_150804b" containing the following Pseudocode:

```
size(500, 400);
background(0);
stroke(153);
strokeWeight(4);

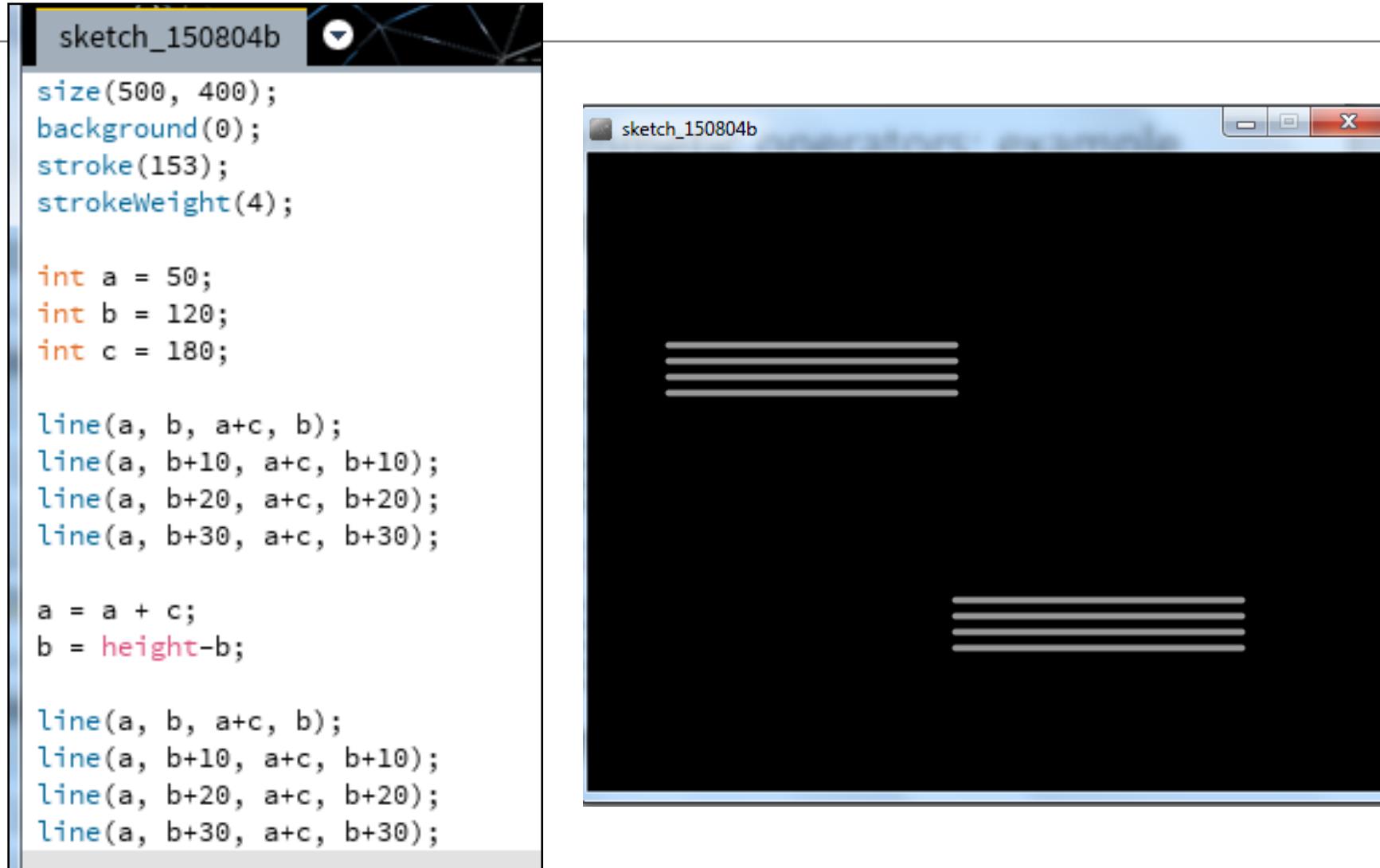
int a = 50;
int b = 120;
int c = 180;

line(a, b, a+c, b);
line(a, b+10, a+c, b+10);
line(a, b+20, a+c, b+20);
line(a, b+30, a+c, b+30);
```

On the right is the "sketch_150804b" window showing the output of the code. It displays a black canvas with four horizontal white lines. The first line is at y=120, the second at y=130, the third at y=140, and the fourth at y=150. All lines have a stroke weight of 4.

Based on the Processing Example: Basics → Data → Variables

Arithmetic operators: example 2



The image shows the Processing IDE interface. On the left is the code editor window titled "sketch_150804b" containing the following Pseudocode:

```
size(500, 400);
background(0);
stroke(153);
strokeWeight(4);

int a = 50;
int b = 120;
int c = 180;

line(a, b, a+c, b);
line(a, b+10, a+c, b+10);
line(a, b+20, a+c, b+20);
line(a, b+30, a+c, b+30);

a = a + c;
b = height-b;

line(a, b, a+c, b);
line(a, b+10, a+c, b+10);
line(a, b+20, a+c, b+20);
line(a, b+30, a+c, b+30);
```

On the right is the "sketch_150804b" window showing the visual output. It displays two sets of four horizontal white lines each, positioned in the upper-left and lower-right quadrants of the black canvas. The lines are spaced vertically at 10-unit intervals.

Based on the Processing Example: Basics → Data → Variables

Arithmetic operators: example 3

The screenshot shows the Processing IDE interface. On the left is the code editor window titled "sketch_150804b" containing the following Pseudocode:

```
size(400, 200);
background(0);
stroke(153);
strokeWeight(4);

int a = 50;
int b = 1500;
int c = 4;

line(a, b/10, a*c, b/10);
line(a, b/20, a*c, b/20);
line(a, b/30, a*c, b/30);
line(a, b/40, a*c, b/40);
line(a, b/50, a*c, b/50);
```

On the right is the preview window titled "sketch_150804b" showing a black canvas with five horizontal white lines. The lines are evenly spaced vertically, corresponding to the calculated y-values of $b/10, b/20, b/30, b/40, \text{ and } b/50$.

Based on the Processing Example: Basics → Data → Variables

Questions?

