

Introduction to Programming

Lecture 1: variables, functions and Processing

Ben Ruijl

Nikhef Amsterdam and Leiden University

October 4, 2016

The art of programming

- Useful skill
- Ability to solve new problems
- No fuzziness: you have to be very specific
- You learn by practicing a lot

Example problems

- Finding the first million prime numbers
- Counting words in a book
- A web shop

Keep in mind

A computer is fast, but stupid

Processing

- We are going to use the language *Processing*
- In principle the language does not matter for basic skills
- Easy visualization
- We are going to draw a lot
- If your code is wrong, you can see it immediately

Course setup

- Seven lectures on Processing, two on Pure Data
- Only Processing will be exam material
- Homework is not graded, but each week one exercise must be handed in
- Final assignment: impress me, make a game?
- Final grade: 50% exam + 50% game
- The exam score and game score must be more than 5.5

A new way of thinking

- Programming is the art of not telling a computer *what* to do, but *how* to do it.
- You have to be specific, there is no interpretation

Logic and natural language

- Logic is more strict than natural language
- Example: John says he will call you if he is ten minutes late. John calls. What conclusions can you draw?

Logic and natural language

- Logic is more strict than natural language
- Example: John says he will call you if he is ten minutes late. John calls. What conclusions can you draw?
- It is unspecified if John will call if he is not late. Thus, we may not conclude that John is ten minutes late.

Logic and natural language

- Logic is more strict than natural language
- Example: John says he will call you if he is ten minutes late. John calls. What conclusions can you draw?
- It is unspecified if John will call if he is not late. Thus, we may not conclude that John is ten minutes late.
- Fix: John will call **if and only if** he is ten minutes late.

Logic and natural language

Other examples:

- "Would you like tea or coffee?" - "Yes."
- Mary has two children vs. Mary has **exactly** two children.

Variables

- A **variable** is an object with a name and a type. It can store a value.

Type	Name	Value
Time	today	17-10-2014
Number	pi	3.14
Number	age	24
Text	name	Quetzalcoatl

Table: Some examples of variables and values

Declaration

- We can **declare** (define) a variable like this:

```
1 type name = somevalue;
```

- For example (in pseudo-code):

```
1 number pi = 3.14;  
2 number one = 1;  
3 book lotr = "The Lord of the Rings";
```

Declaration

- Processing does not know what a *book* is
- It has the following built-in types:

Type	Meaning	Example
String	text	String a = "Hello!";
int	integer	int a = 1;
float	real number	float pi = 3.14;
boolean	true or false	boolean a = true;

- If you want other types, you can keep track of several variables. A *book* would be a collection of strings.

Assignments

You can **assign** different values to declared variables:

```
1 int a = 3;  
2 a = 4; // assignment of 4 to variable a  
3 int b = 5;  
4 a = b;  
5 a = 3;
```

Note: a and b are not linked, so if a changes, b does not change.

Pitfall

= does not mean equals, but assignment. You assign to the variable on the left, the value on the right.

Code flow

Code is executed from top to bottom

```
1 int a = 3;  
2 int b = 4;  
3 a = 4;
```

What is a and what is b?

Operators

Variables can be changed. There are some built-in operations:

- Mathematical operators: $+$, $-$, $/$, $*$, $\%$
- String operators: $+$ (concatenate two strings)
- $++$, $--$: increase/decrease by 1
- $+=$, $*=$, $-=$: update the value

Examples

```
1 int a = 3;
2 int b = 4;
3 a = 3 * b + 4;
4 a--;
5 b = a % 3; // modulo is the rest term after division
```

What is a and what is b?

Pitfall

Division of two integers is always rounded down, so $3/2=1$. Use **floats**:
 $3.0/2.0 = 1.5$.

Functions

- A variable **stores** things
- A function **modifies** things

Examples:

- Function `sin(x)` transforms an angle `x` to the sine of `x`
 - It has one argument: the angle (a **float**)
 - It returns a **float**
 - `sin($\frac{1}{2}\pi$)` returns 1
- Function `rect(x,y,width,height)` draws a rectangle on the screen
 - Arguments: `x` and `y` (position), `width` and `height` (size)
 - It does not return anything
 - `rect(10, 10, 5, 5)` draws a rect at (10,10) with width 5 and height 5

Function calling

Function calling

To **call** (execute) a function, write its name and then the arguments between parentheses

Examples:

```
1 println("Hello!"); // write hello to the console
2 ellipse(4, 5, 6, 7); // draws an ellipse
```

If a function **returns** something, you can store it in a variable:

```
1 float a = sin(1/4 * PI); // a = 0.7071
```

Processing basics

- Every Processing file consists of two functions: setup and draw
- Setup is called by the system once
- Draw is called every frame
- A **frame** is a refreshment of the screen (a movie has 24 frames per second)

Minimal working example

Empty processing file:

```
1  void setup() {  
2      // setup here  
3  }  
4  
5  void draw() {  
6      // draw here  
7  }
```

Hello world

```
1  void setup() {  
2      println("Hello world!"); // print line  
3  }  
4  
5  void draw() {  
6      // draw here  
7  }
```

Examples

```
1  void setup() {  
2      int a = 5;  
3      a = a * 5;  
4      println("This is the new value: " + a);  
5  }  
6  
7  void draw() {  
8  }
```

Drawing a dot

```
1 void setup() {  
2   size(640, 640); // create a screen of 640x640 pixels  
3   background(#FFFFFF); // white background  
4   strokeWeight(5); // width in pixels of stroke  
5 }  
6  
7 void draw() {  
8   stroke(#000000); // black dot  
9   point(320, 320); // draw a dot in the centre  
10 }
```


Variable scope

- Variables inside functions lose their value after the function ends
- In general: a variable only exists between the brackets `{ }` it is defined in
- To store variables for more than one frame, store it outside of `draw`

Local variables

```
1 void setup() {  
2 }  
3  
4 void draw() {  
5     int framecount = 0;  
6     framecount++;  
7     println(framecount);  
8 }
```

Yields 1, 1, 1, 1,

Global variables

```
1  int framecount = 0;
2
3  void setup() {
4  }
5
6  void draw() {
7      framecount++;
8      println(framecount);
9  }
```

Yields 1, 2, 3....

Standard functions

Some of these functions are useful:

- `println(string)`: print text to the screen
- `rect(x, y, width, height)`: draw a rectangle
- `ellipse(x, y, width, height)`: draw an ellipse
- `background(color)`: fill screen with color
- `stroke(color)`: stroke color
- `fill(color)`: fill color

References

In order to get a firmer grasp on how to draw, see the following tutorials online:

- For drawing: <https://www.processing.org/tutorials/drawing/>
- For colours: <https://www.processing.org/tutorials/color/>
- Function reference: <https://www.processing.org/reference>
- Trigonometry: <https://processing.org/tutorials/trig/>