http://dextro.org

Processing as a coding platform

VA345 Creative Coding

Course Instructor : Assist. Prof. Dr. Selcuk ARTUT

Food for thought : Casey Reas

Computer Programming (Coding – sounds cool)

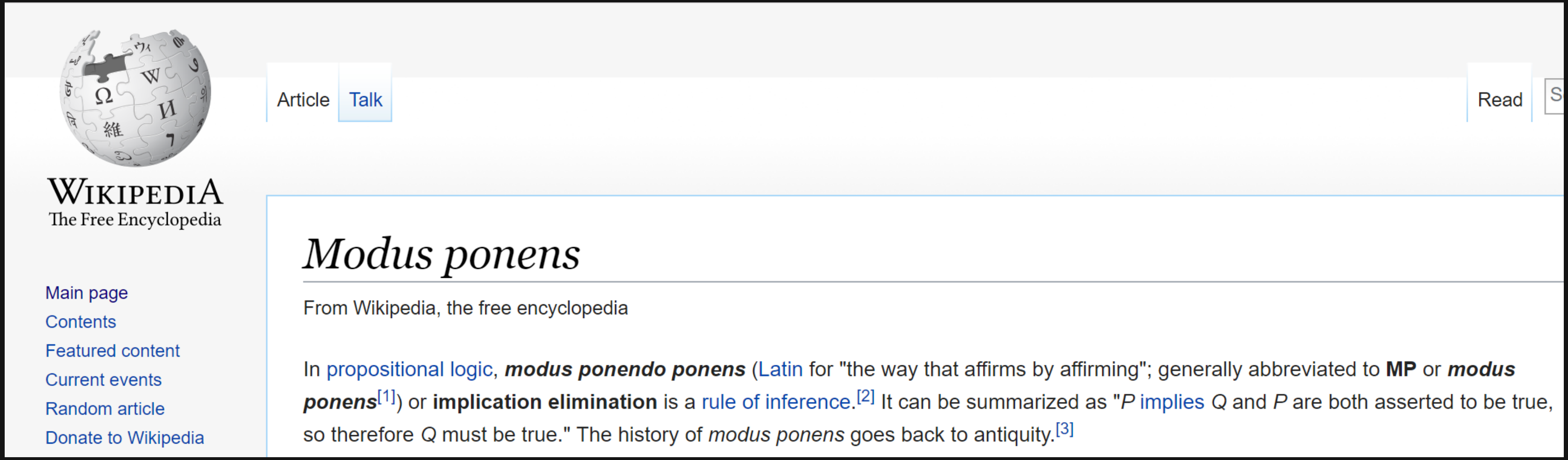
Software

Rule based systems

Earliest examples Modus ponens

If you believe that A is the case, and believe that A implies B, then you ought to believe that B is the case

Rituals, laws, political systems etc



Modus ponens

From Wikipedia, the free encyclopedia

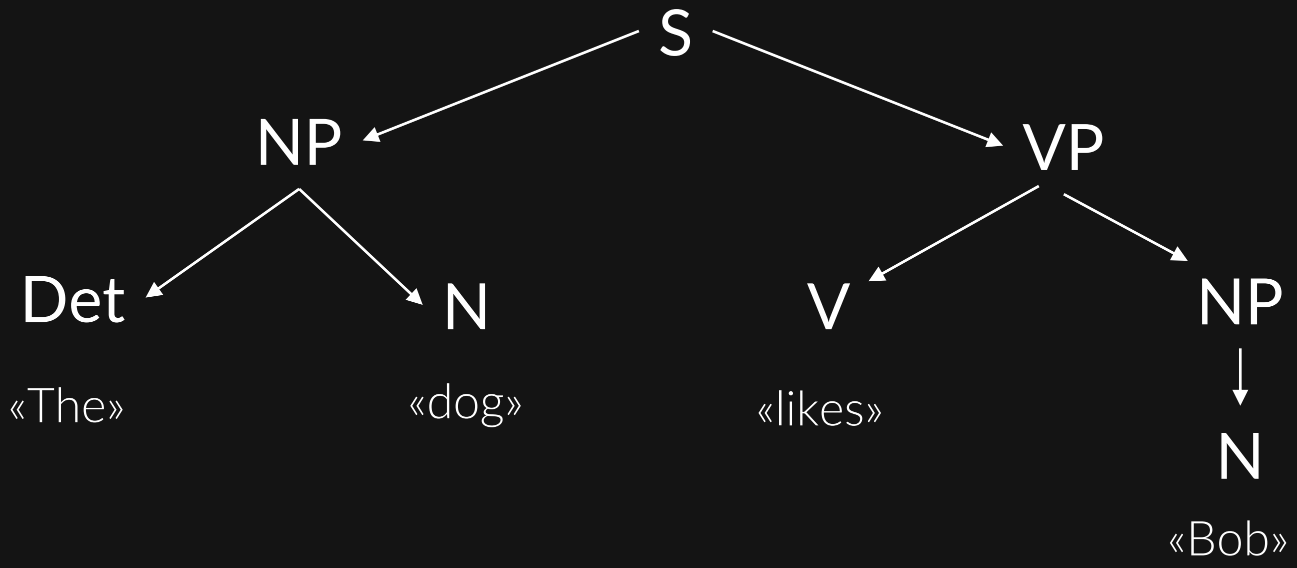
In propositional logic, ***modus ponendo ponens*** (Latin for "the way that affirms by affirming"; generally abbreviated to **MP** or ***modus ponens***[1]) or **implication elimination** is a rule of inference.[2] It can be summarized as "*P* implies *Q* and *P* are both asserted to be true, so therefore *Q* must be true." The history of *modus ponens* goes back to antiquity.[3]

Rule based systems

- Formal Languages vs Natural Languages
- It is easier to define a formal language than a natural language, because formal language is synthetic, expressive, compact etc.
- Language = grammar + dictionary
- Computer languages : Backus-Naur form (1959) - Algol60

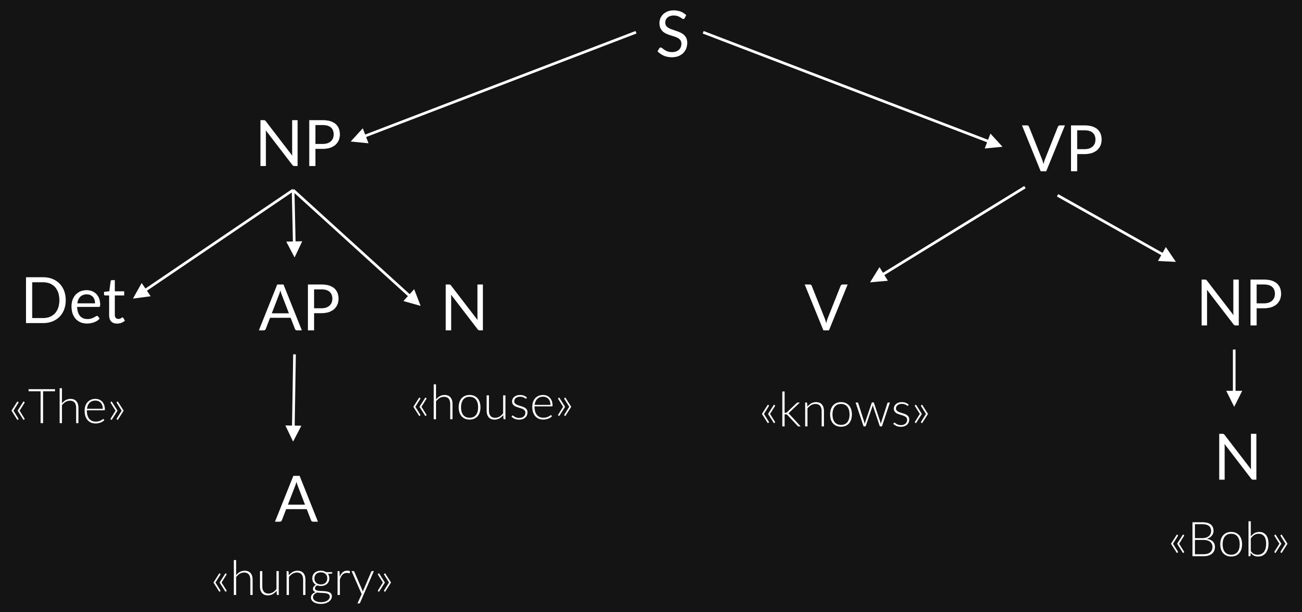
Generative Grammar for Syntax

- We can formalize a syntax of a language using a grammar

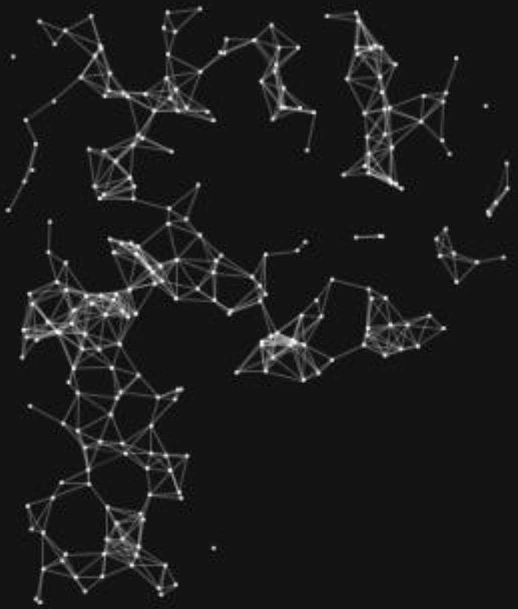


S : Sentence
NP : Noun phrase
Det : Determinant
N : Noun
V : Verb
VP : Verbal Phrase

A sentence can be syntaxically correct but semantically wrong

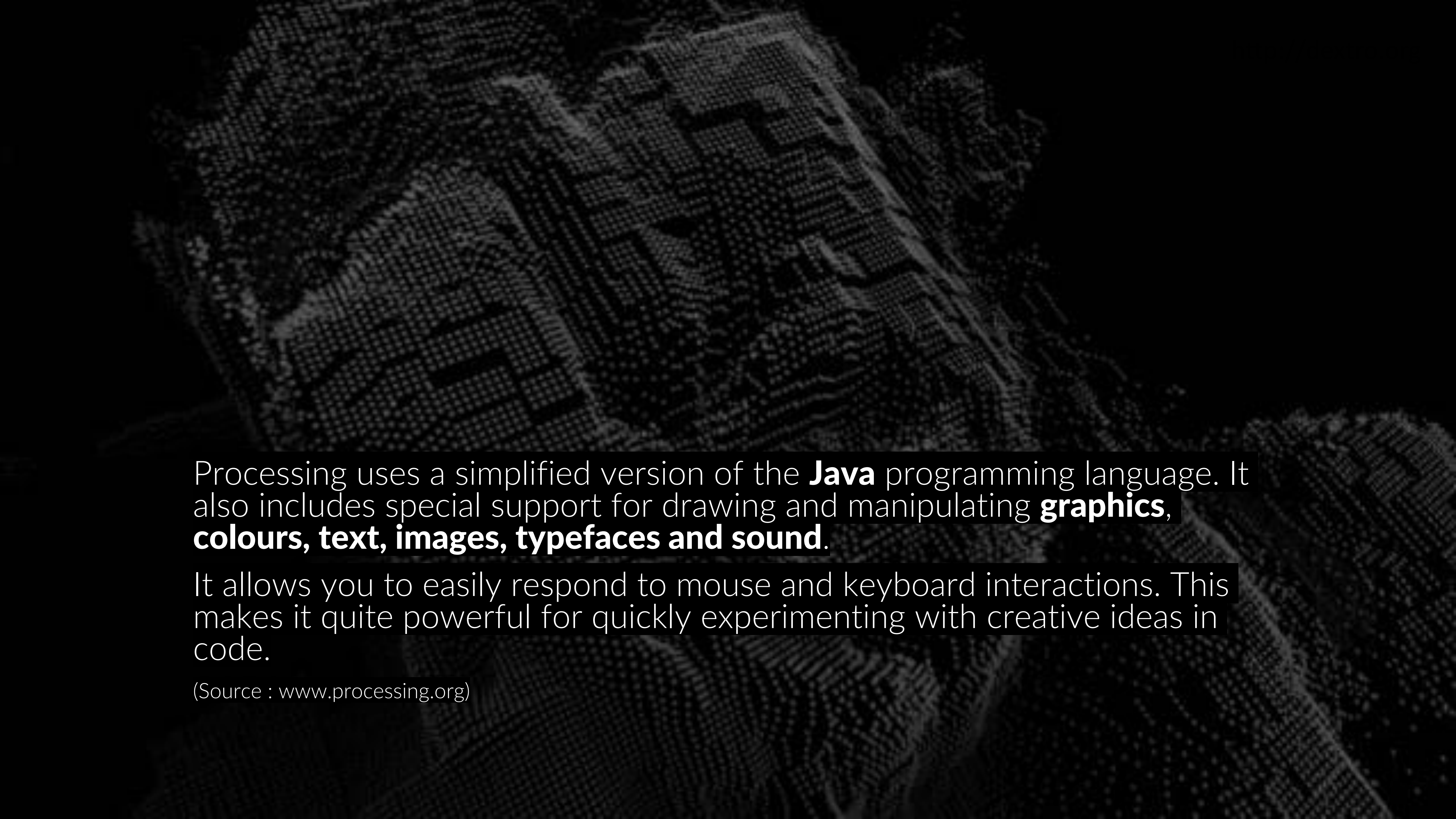


S : Sentence
NP : Noun phrase
Det : Determinant
N : Noun
V : Verb
VP : Verbal Phrase
AP : Adjective Phrase



Processing is a flexible software sketchbook and a language for learning how to code within the context of the visual arts. Since 2001, Processing has promoted software literacy within the visual arts and visual literacy within technology. There are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning and prototyping.

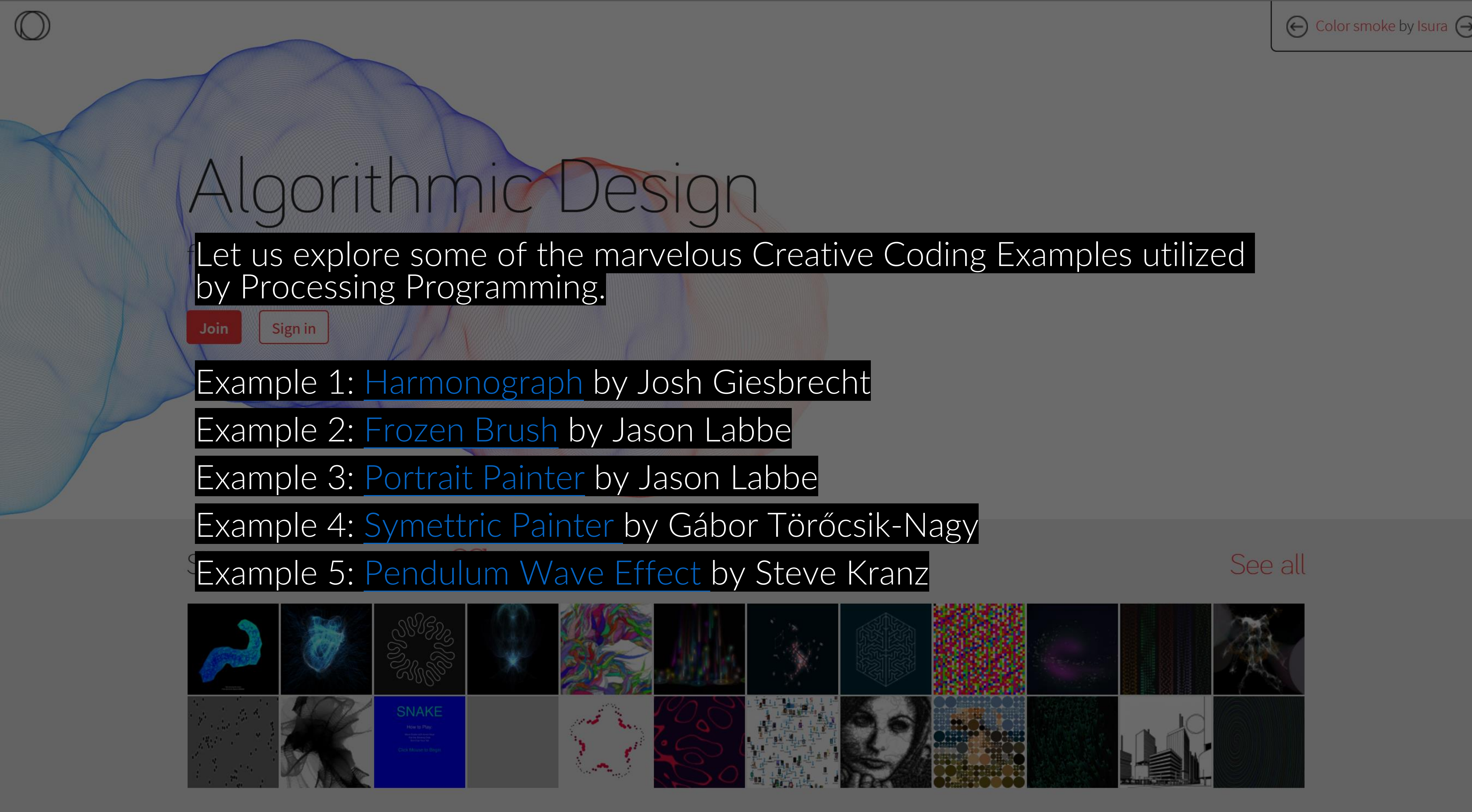
www.processing.org



Processing uses a simplified version of the **Java** programming language. It also includes special support for drawing and manipulating **graphics**, **colours, text, images, typefaces and sound**.

It allows you to easily respond to mouse and keyboard interactions. This makes it quite powerful for quickly experimenting with creative ideas in code.

(Source : www.processing.org)



Algorithmic Design

Let us explore some of the marvelous Creative Coding Examples utilized by Processing Programming.

Example 1: Harmonograph by Josh Giesbrecht

Example 2: Frozen Brush by Jason Labbe

Example 3: Portrait Painter by Jason Labbe

Example 4: Symettric Painter by Gábor Törőcsik-Nagy

Example 5: Pendulum Wave Effect by Steve Kranz

Installing Processing

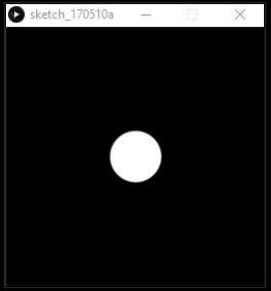
1) Follow the link www.processing.org

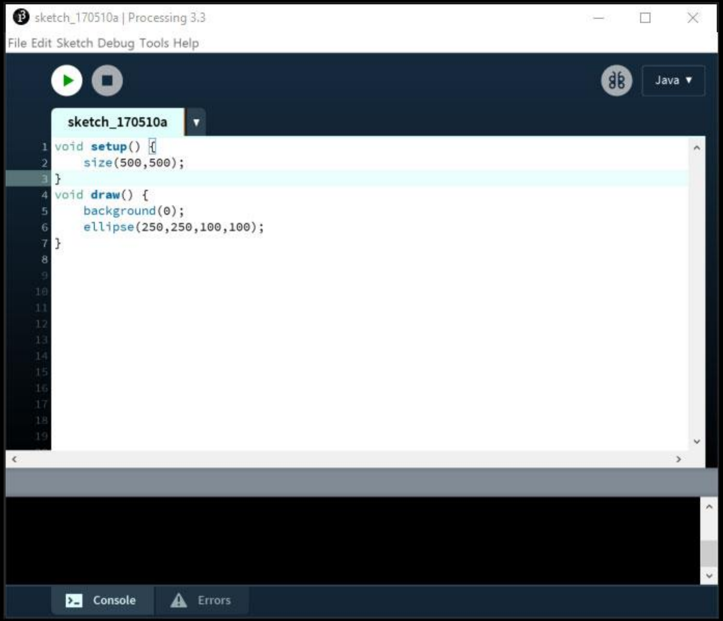
2) Download appropriate version: Windows 64bit 32 bit/Mac/Linux

3) Install

4) Try this code

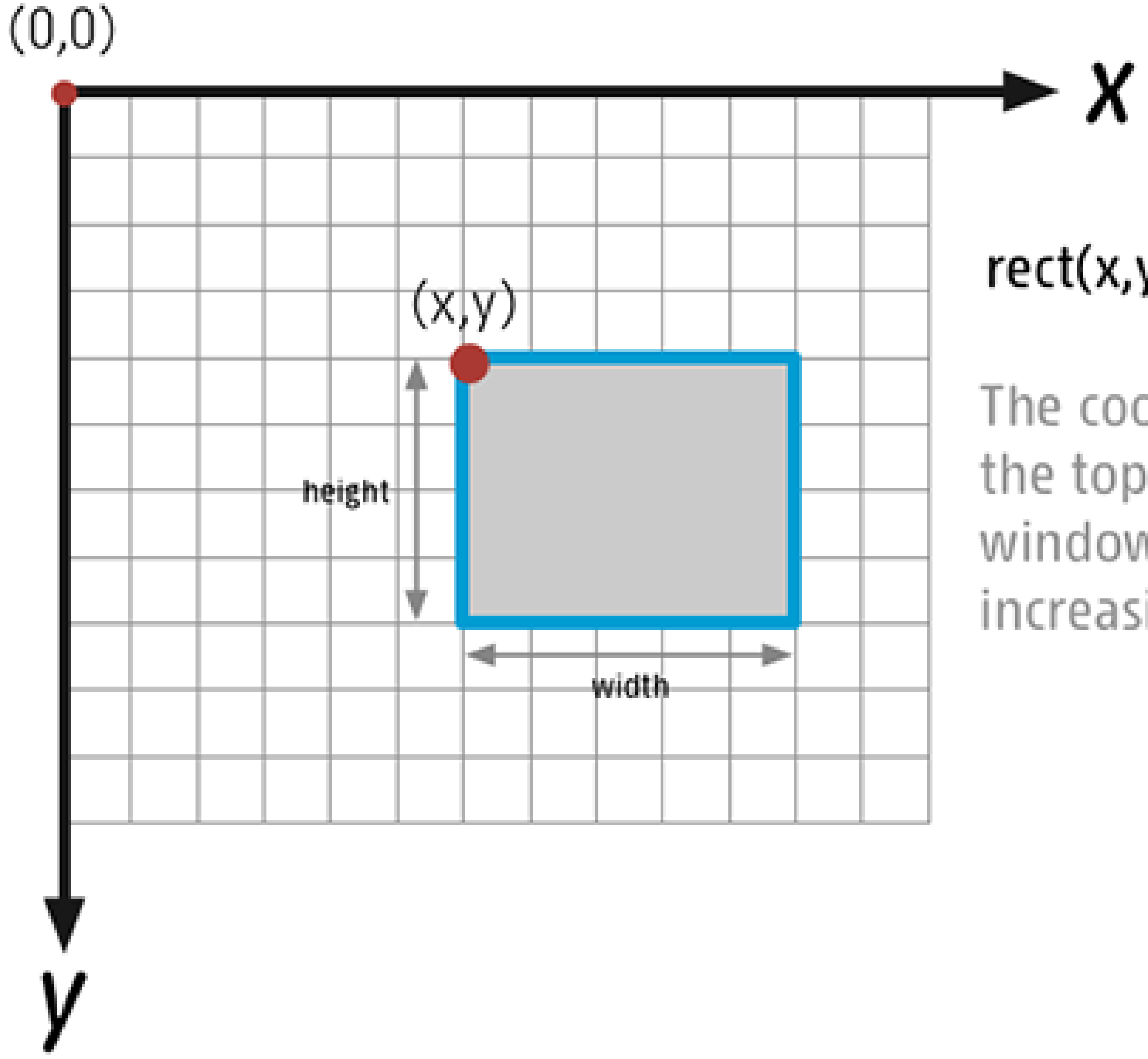
```
void setup() {
    size(500,500);
}
void draw() {
    background(0);
    ellipse(250,250,100,100);
}
```




sketch_170510a | Processing 3.3
File Edit Sketch Debug Tools Help
Java
sketch_170510a
Console
Errors

```
void setup() {
    size(500,500);
}
void draw() {
    background(0);
    ellipse(250,250,100,100);
}
```

SCREEN COORDINATES IN PROCESSING



rect(x,y,width,height)

The coordinate origin (0,0) is at the top left of the display window. Increasing ***x*** moves right, increasing ***y*** moves down.

Analyzing Processing Coding Environment

```
void setup() {
        size(500,500);
}
void draw() {
        background(0);
        ellipse(250,250,100,100);
}
```

https://processing.org/reference/ellipse_.html

Processing Commands are Case Sensitive!!

size(width, height, optional : Renderer)

size(640,480,JAVA2D);	The standard renderer, this is used if nothing else has been specified
size(640,480,P2D);	Processing 2D Renderer, this is quicker but less accurate
size(640,480,P3D);	Processing 3D Renderer, this is quick and its reliable on the Web
size(640,480, OPENGL);	OpenGL renderer, this uses OpenGL-compatible graphic hardware

Ref: Generative Design p 172

Using Comments

Comments are to be used by humans only. So computer will ignore those lines, and compile the uncommented lines.

Comment Types

Block Comment : /* …………… */ (Using forward slash asterix …. asterix forward slash)

Line Comment : // (Using two forward slash)

Alternative usage : You may want to ignore a block or a line of code to check for debugging.

Drawing Basic Shapes

Some simple shapes;

ellipse : https://processing.org/reference/ellipse_.html

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

line : https://processing.org/reference/line_.html

arc : https://processing.org/reference/arc_.html

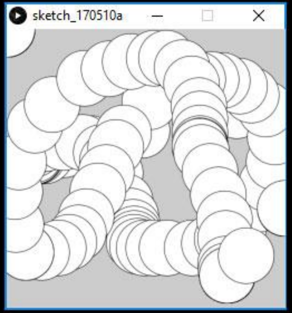
vertex : https://processing.org/reference/vertex_.html

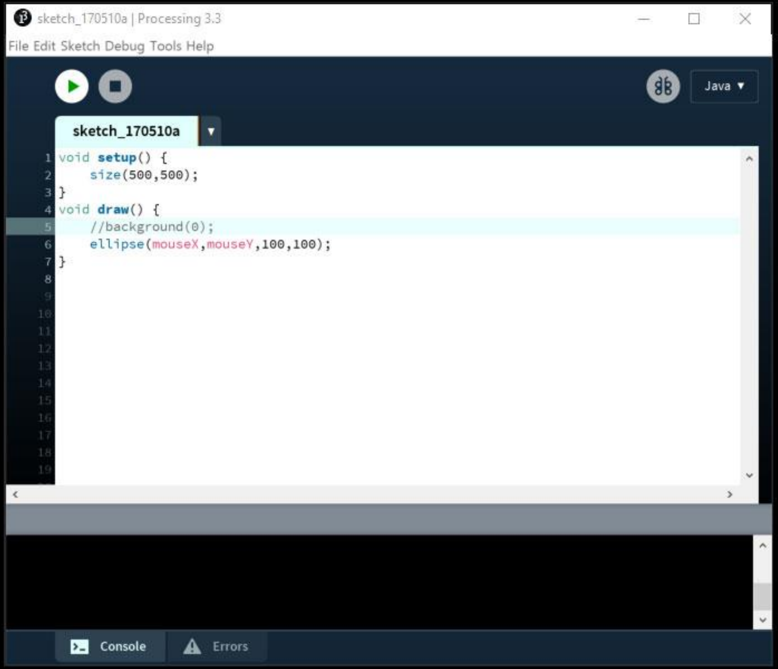
Free Coding Time (40 minutes)

Play with the code!! Feel free to experiement.

Adding a bit of an Interaction

```
void setup() {
        size(500,500);
}
void draw() {
        //we do not call the background
        // function so that it adds each time
        // draw is called
        ellipse(mouseX,mouseY,100,100);
}
```



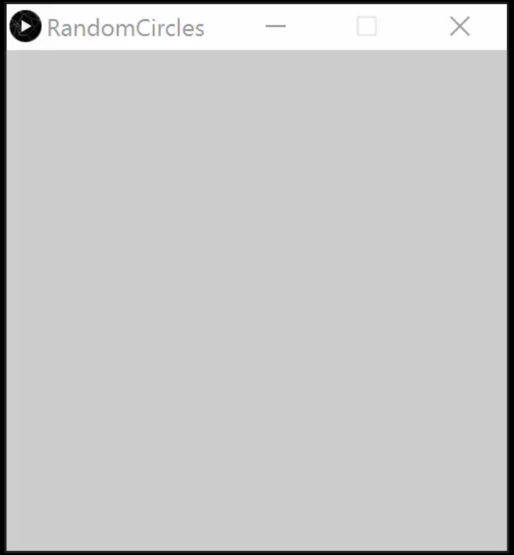

sketch_170510a | Processing 3.3
File Edit Sketch Debug Tools Help
Java ▾
sketch_170510a
Console
Errors

```
void setup() {
    size(500,500);
}
void draw() {
    //background(0);
    ellipse(mouseX,mouseY,100,100);
}
```

Challenge (15 min)

Play with the same code, but draw circles with variable sizes.

- Hint: Try randomness
- Ambitious Hint: Use pmouseX



Colors

color() : Creates colors for storing in variables of the color datatype. The parameters are interpreted as RGB or HSB values depending on the current colorMode(). The default mode is RGB values from 0 to 255 and, therefore, color(255, 204, 0) will return a bright yellow color.

Syntax

color(gray)

color(gray, alpha)

color(v1, v2, v3)

color(v1, v2, v3, alpha)

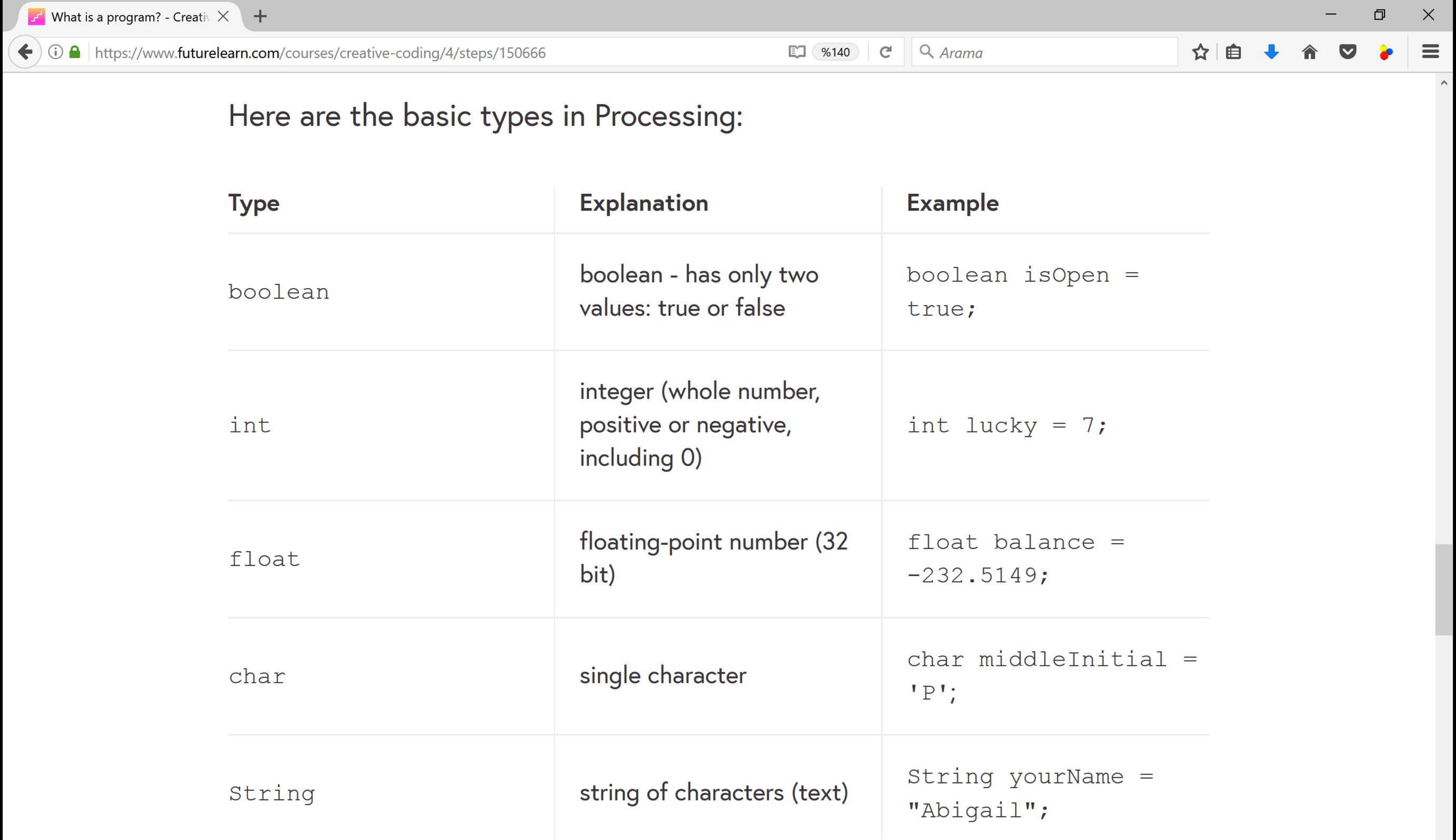
Data Types, Variables

In order to create dynamic structures, we are required to have data carriers that can change its value at a particular moment. So variables are **great** tools for **Creative Coding.**

When defining a variable, the user must specify its type, meaning what kind of information it will save.

Variable Types

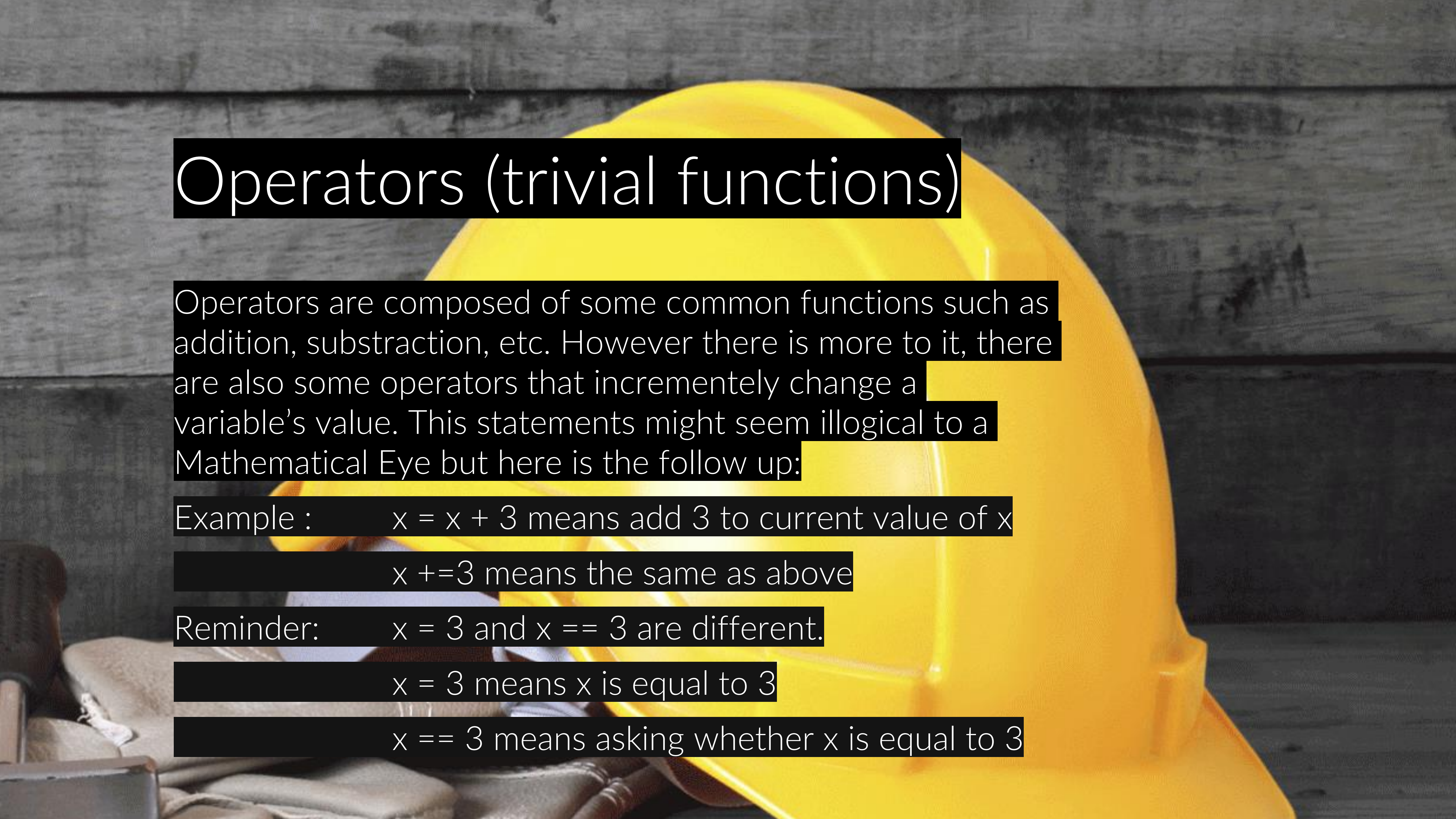
```
boolean myBoolean = true;

int myInteger = 5;

float myFloat = 3.14;

char myChar = ‘A’;

String myString = ‘’this is my string’’
```


What is a program? - Creati

https://www.futurelearn.com/courses/creative-coding/4/steps/150666

Here are the basic types in Processing:

Type	Explanation	Example
`boolean`	boolean - has only two values: true or false	`boolean isOpen = true;`
`int`	integer (whole number, positive or negative, including 0)	`int lucky = 7;`
`float`	floating-point number (32 bit)	`float balance = -232.5149;`
`char`	single character	`char middleInitial = 'P';`
`String`	string of characters (text)	`String yourName = "Abigail";`



Operators (trivial functions)

Operators are composed of some common functions such as addition, substraction, etc. However there is more to it, there are also some operators that incrementely change a variable's value. This statements might seem illogical to a Mathematical Eye but here is the follow up:

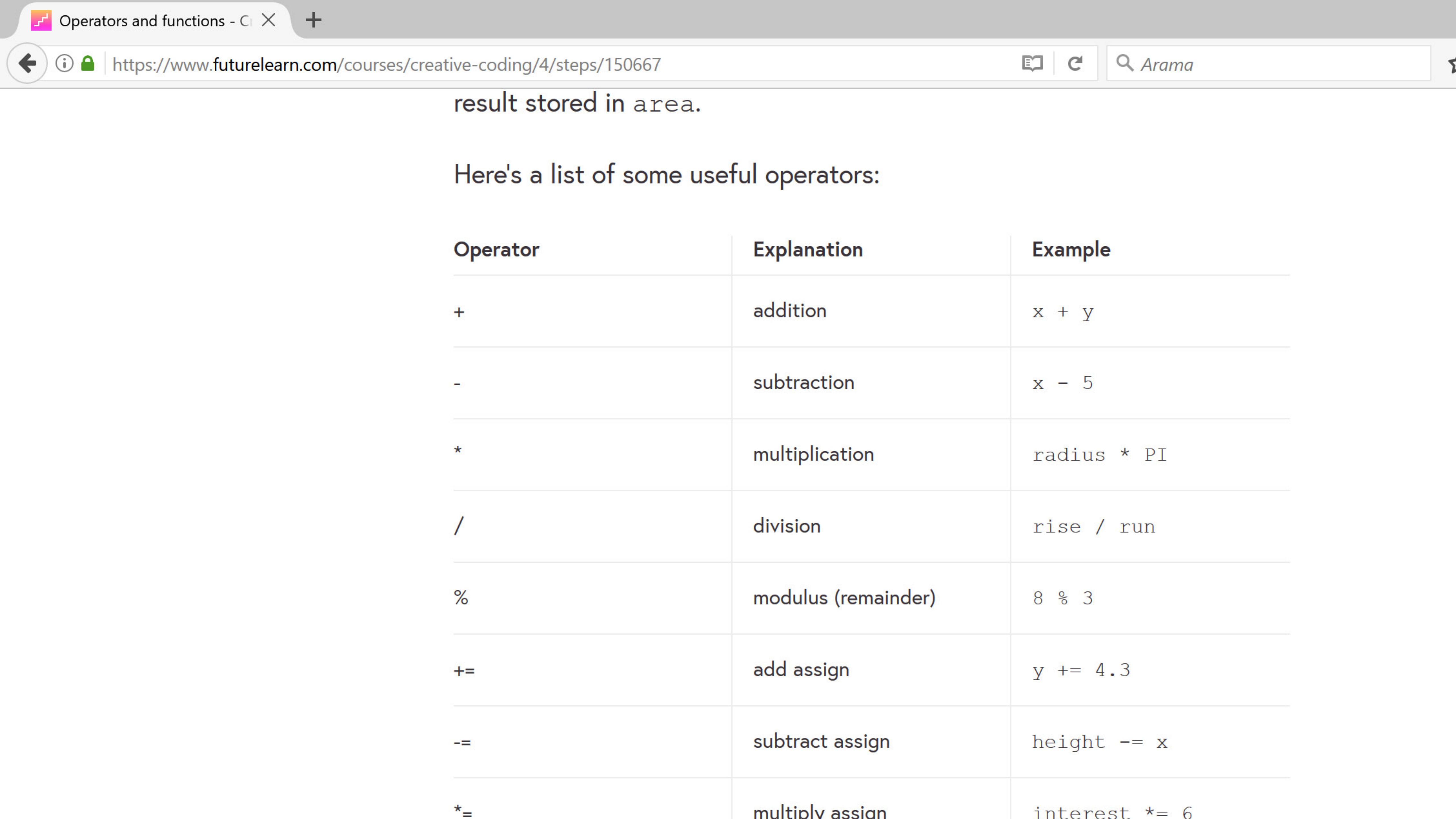
Example : x = x + 3 means add 3 to current value of x

x +=3 means the same as above

Reminder: x = 3 and x == 3 are different.

x = 3 means x is equal to 3

x == 3 means asking whether x is equal to 3


Operators and functions - C
https://www.futurelearn.com/courses/creative-coding/4/steps/150667
Arama

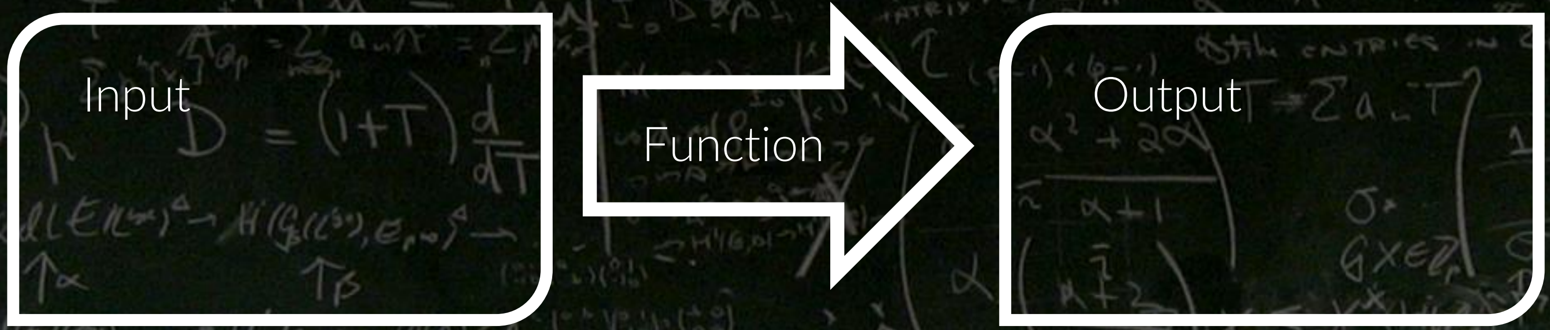
result stored in `area`.

Here's a list of some useful operators:

Operator	Explanation	Example
+	addition	`x + y`
-	subtraction	`x - 5`
*	multiplication	`radius * PI`
/	division	`rise / run`
%	modulus (remainder)	`8 % 3`
+=	add assign	`y += 4.3`
-=	subtract assign	`height -= x`
*=	multiply assign	`interest *= 6`

Functions

Functions are building blocks for achieving a specific purpose. They have inputs and outputs.



Functions

Example:

```
void setup(){
        drawStar();
}

void drawStar(){
        line(0,-10,0,10);
        line(-8,-5,8,5);
        line(-8,5,8,-5);
}
```

Here a function was defined that contains the drawing commands. This function can be called from anywhere in the program in order to execute the code it contains.

Functions May Return or Not

Functions may be called to return an output, or accomplish a certain task.

Example No Return :

```
void setup() {
		size(300,300);
}
```

Example Return :

```
float calculateHypotenuse(float x, float y) {
    return sqrt(x * x + y * y);
}
```

Functions

Example:

```
void setup(){
        size(300,300);
        println(findHypothenuse(3,4));
}

int findHypothenuse(int x,int y){
        int hyp = sqrt(sq(x)+sq(y));
        return hyp;
}
```

Values can be passed in functions, which can also return a value as a result.

Chance Operations &Randomness

Randomness is observed in a sequence when order and patterns seem to be lacking (Philip Pasquier)

Given a variable X defined over a discrete set of options, its domain D_x, a random draw is giving equal probability to all the option from D_x (equiprobability)

Artwork Credit : Candaş Şişman
Pattern of Possibilities
2015
Installation
23.716 Piece Backgammon Dices,
Wood, Video and Sound
130x130 cm

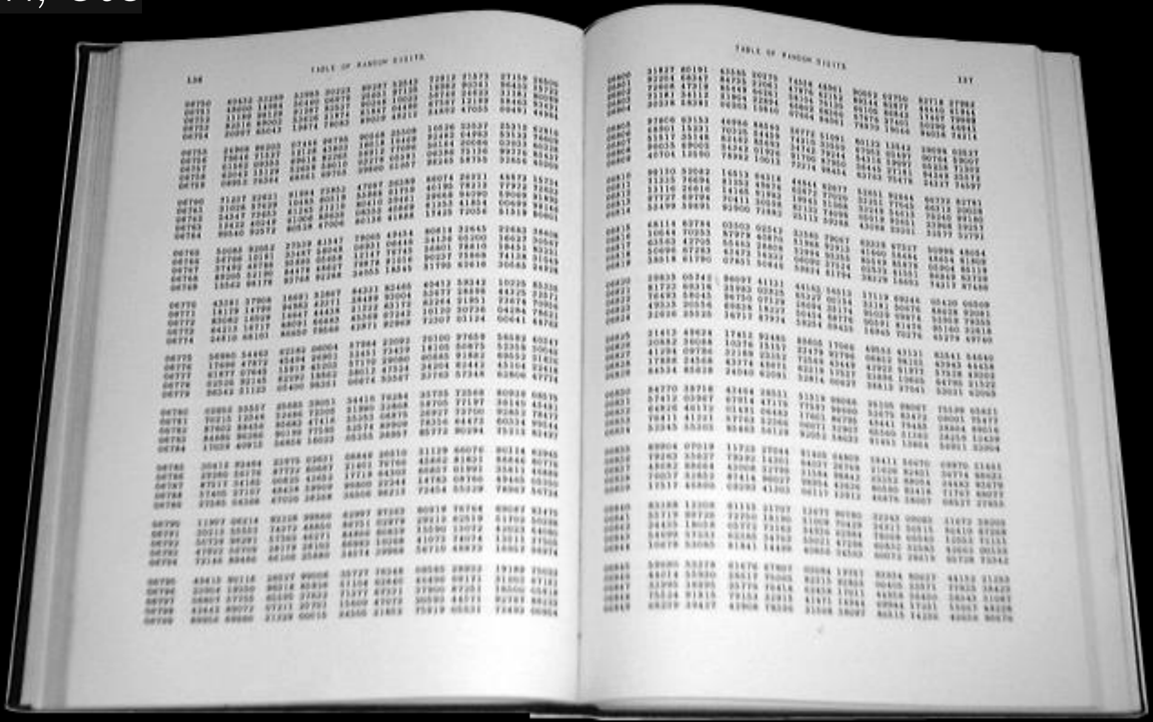
How to get random numbers

Randomness is not random at all.

One Million Random Book

Natural Sources of Entropy : Atmospheric Noise,

Thermal Noise, Cosmic Background Radiation, etc



Randomness in Computers

A pseudorandom process is a process that appears to be random but is not. Pseudorandom sequences typically exhibit statistical randomness while being generated by an entirely deterministic causal process. (wikipedia)

To generate truly random numbers would require precise, accurate, and repeatable system measurements of absolutely non-deterministic processes. Linux uses, for example, various system timings (like user keystrokes, I/O, or least-significant digit voltage measurements) to produce a pool of random numbers

Randomness in Processing

Let’s visit > https://processing.org/reference/random_.html

By default, random() produces different results each time the program is run. Set the seed parameter to a constant to return the same pseudo-random numbers each time the software is run.

```
void setup(){
 size(600,400);
}
void draw(){
 background(0);
 fill(255);
 float x =
random(width);

ellipse(x,height/2,
40,40);
}
```

Pseudorandom number generators | Computer Science | Khan Academy

https://www.youtube.com/watch?v=GtOt7EBNEwQ

Noise?

Noise is a series of random numbers, typically arranged in a line or a grid.

Some of the basic 1D/2D noise generators are:

Use random numbers directly for the output.

Use random numbers as parameters for sines and cosines, which are used for the output.

Use random numbers as parameters with varying density and distribution functions, which are used for the output. For instance this is used in Perlin Noise, Brownian Noise, Pink Noise, White Noise etc

Perlin Noise?

An algorithm known as “Perlin noise,” named for its inventor Ken Perlin, takes this concept into account. Perlin developed the noise function while working on the original Tron movie in the early 1980s; it was designed to create procedural textures for computer-generated effects. In 1997 Perlin won an Academy Award in technical achievement for this work. Perlin noise can be used to generate various effects with natural qualities, such as clouds, landscapes, and patterned textures like marble.

Perlin noise has a more organic appearance because it produces a naturally ordered (“smooth”) sequence of pseudo-random numbers.

Using Perlin Noise in Processing

noise()

Returns the Perlin noise value at specified coordinates. Perlin noise is a random sequence generator producing a more natural, harmonic succession of numbers than that of the standard random() function.

https://processing.org/reference/noise_.html

```
float xoff = 0.0;

void setup(){
 size(600,400);
}

void draw(){
 background(0);
 fill(255);
 xoff = xoff + .01;
 float x = noise(xoff) * width;
 ellipse(x,height/2, 40,40);
}
```

Assignment 002

Use Perlin Noise to create a Processing Sketch that moves basic shapes smoothly overtime.