

SM2715

CREATIVE CODING

# ASSESSMENT TASKS

## Course Work

## Weighting

Attendance and Participation

10%

Coding assignments

50% (2\*25%)

Final project

40%

# SCHEDULE

Week 1	Basics: Processing, Data Type, Structure	
Week 2	Object-Oriented Programming (OOP)	
Week 3-4	Image Processing	
	Lunar New Year Break	
Week 5-6	Video Processing	
Week 7-8	Ambient Media	Assignment 1 (25%)
Week 9	Sound Library	
Week 10	Time-based Media	
Week 11	Easter Holiday	Assignment 2 (25%)
Week 12	Interactive: Games, Physics	
Week 13	Particle Systems	
Week 15	Final project presentation	
Week 16	Final project (40%) due by May 06 Mon 23:59	

# POLICIES

## Attendance:

- more than 3 absences may fail the course

## Late submission:

- 10% mark deduction per day late

## Plagiarism:

- 0 mark for the concerned assignment AND a lower course grade
- A written warning letter

## AI Generative Tool:

- Not allowed for coding assignment
- If the AI tool is used to improve the writing, you need to make proper citations and clearly describe how the AI tool is used



# RULES ON ACADEMIC HONESTY

Academic dishonesty is regarded as a serious offence in the University

Dishonesty examples

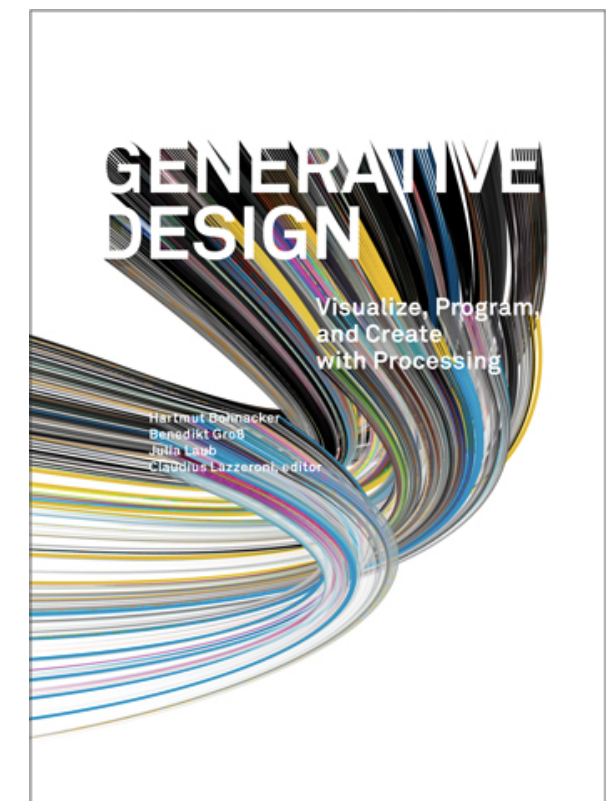
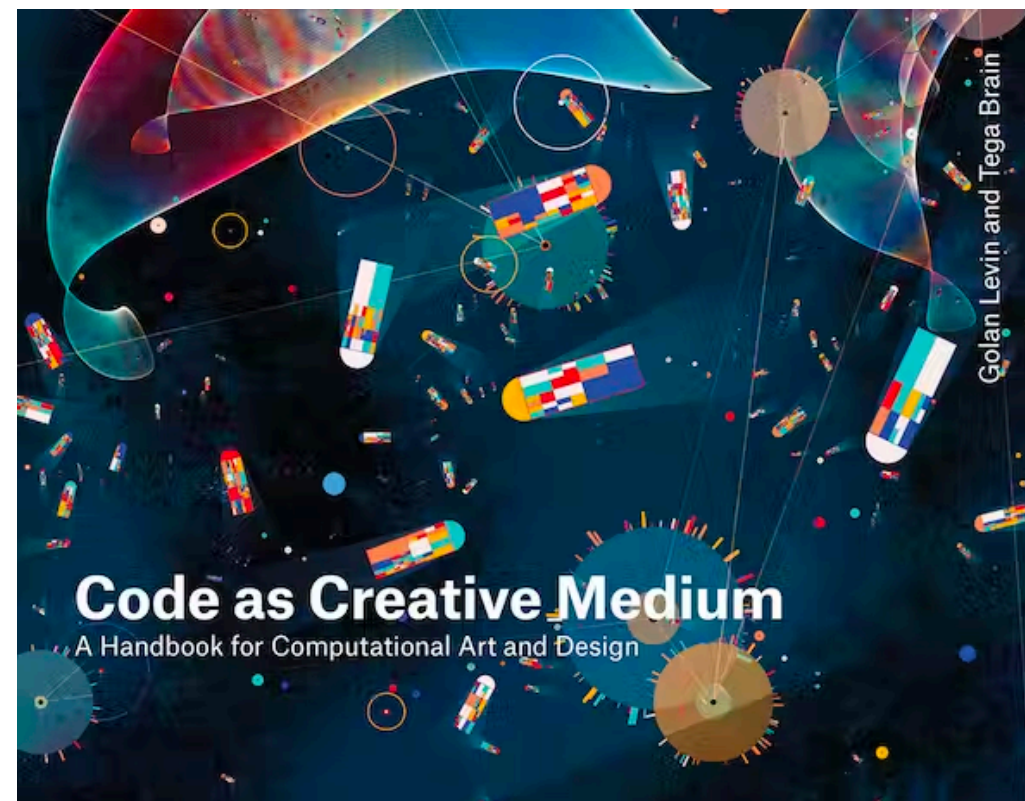
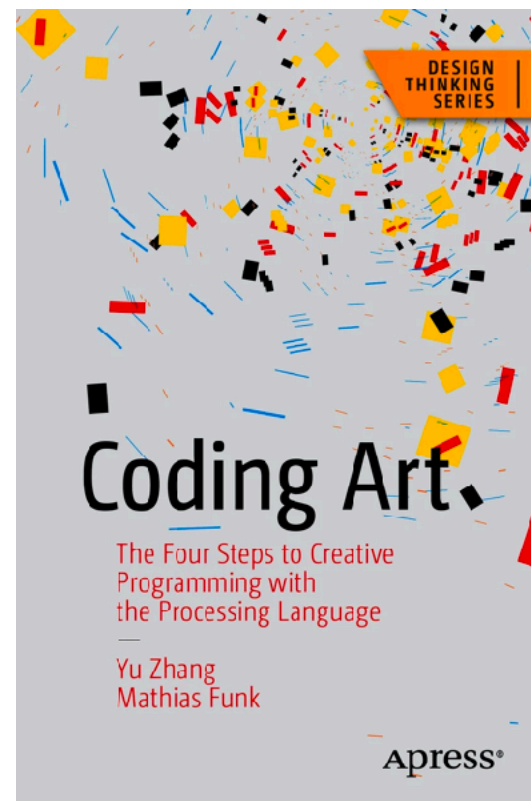
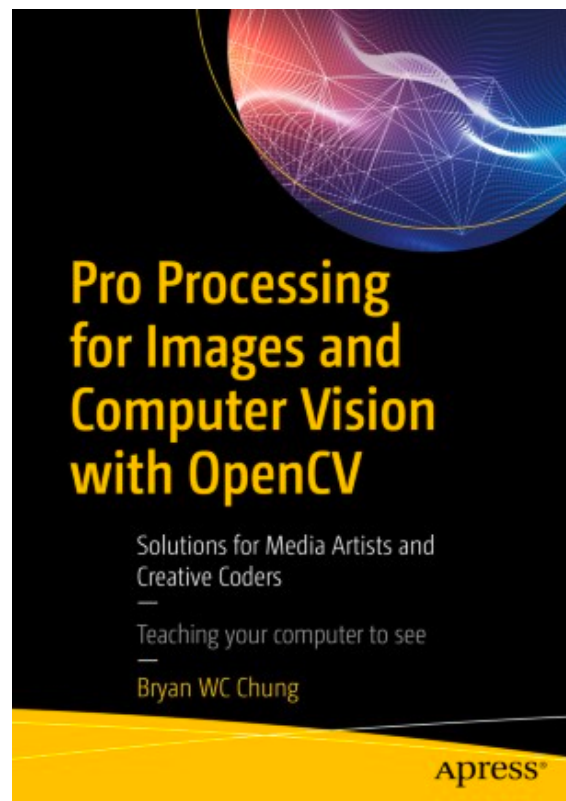
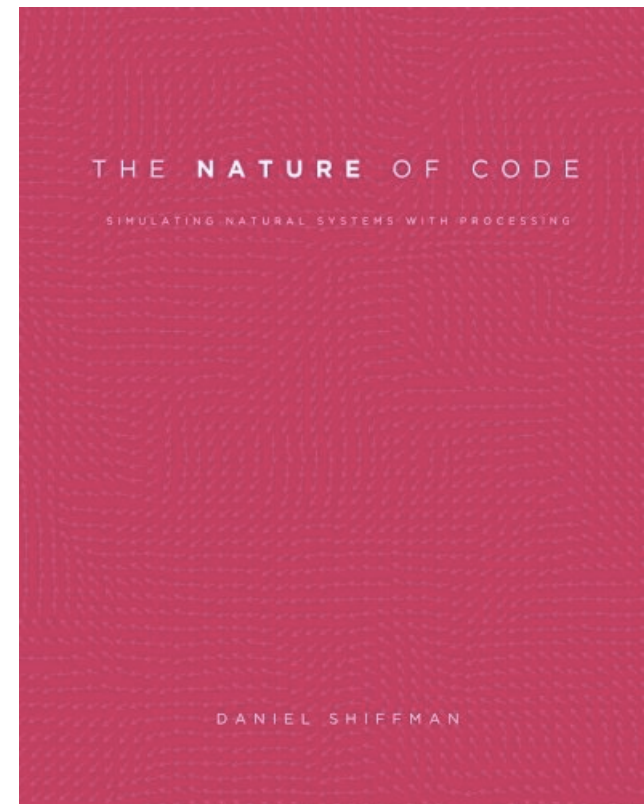
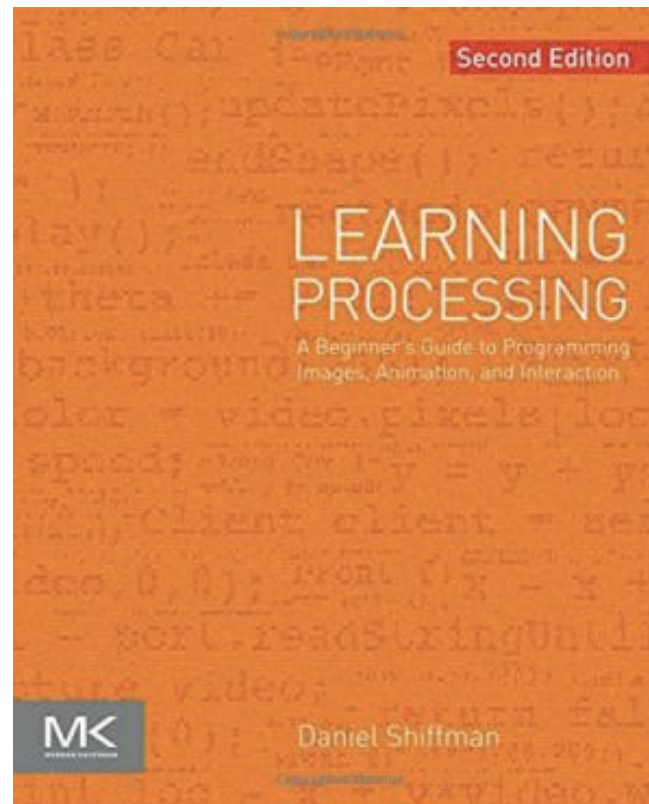
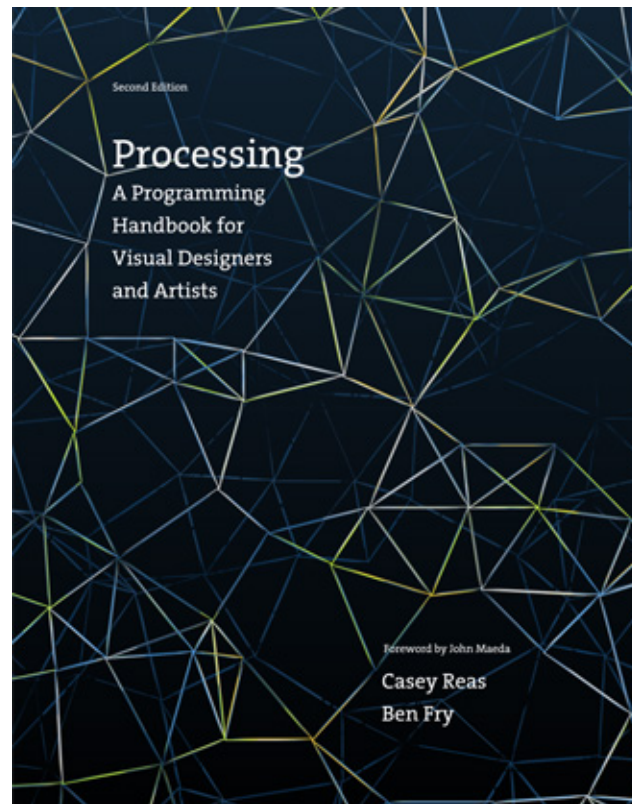
- Plagiarism
  - Intentionally or unintentionally fail to cite the source of code from others, OR
  - Copy too much from sources, with too little of your own thought
    - Whether or not you use citation and referencing
- Collusion
  - Allow another person to copy one's work

Read `Annex_1_Citing_Referencing_Code.pdf`,  
`Annex_4_Plagiarism_Programming.pdf` and  
`guidelines-generative-ai-tools.pdf`

WEEK 1

BASICS OF PROCESSING

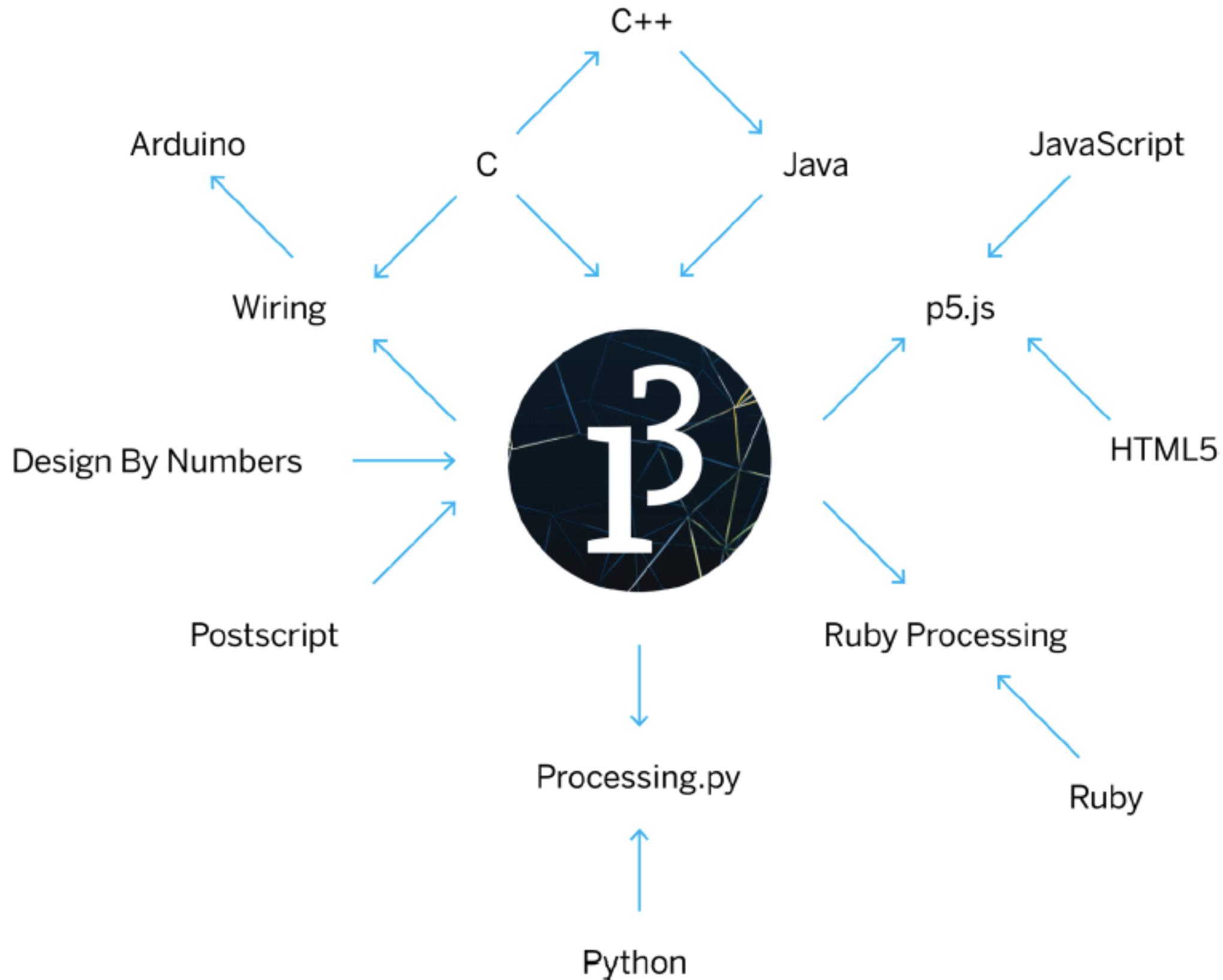
# BOOKS



<https://processing.org/books/>



# FAMILY TREE



# HELLO PROCESSING

## BY DANIEL SHIFFMAN

Hello Processing!

[1 Hello](#) [2 Shapes](#) [3 Color](#) [4 Interact](#) [5 Questions](#) [6 Goodbye](#)

[About](#) [Guide](#) [Help](#) [Hour of Code™](#)



# DOWNLOAD AND INSTALL PROCESSING



## Create with code, everywhere

Processing is open source and is available for macOS, Windows, and Linux. Projects created with Processing are also cross-platform, and can be used on macOS, Windows, Android, Raspberry Pi, and many other Linux platforms.

We need your help!



Help us continue with your generosity!


Donate

Download Processing 4.3 for macOS


macOS • Intel 64-bit • 214 MB • ⓘ

Got an M1 or M2 CPU? Download the [Apple Silicon](#) version instead.

Need another version?




Windows




macOS

Intel 64-bit ⓘ

Apple Silicon ⓘ

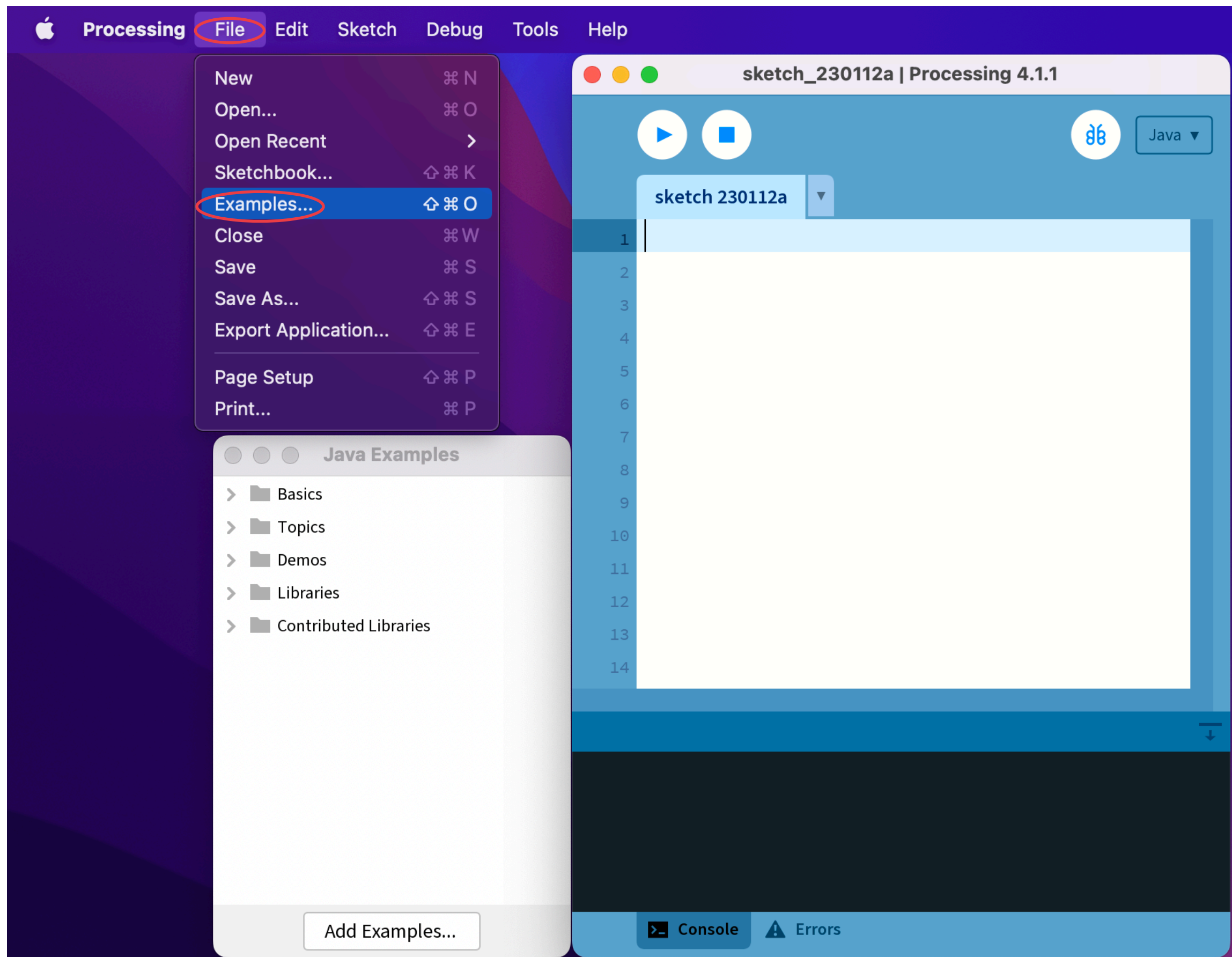


Linux



Raspberry Pi

# EXAMPLES (BUILT IN)



# EXAMPLES ON PROCESSING.ORG

[Download](#)[Documentation](#)[Learn](#)[Teach](#)[About](#)[Donate](#)[Tutorials](#)[Examples](#)[Books](#)

## Examples

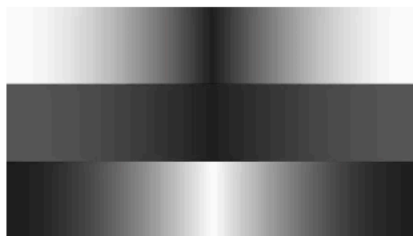
Short, prototypical programs exploring the basics of programming with Processing.

Filter by keywords...

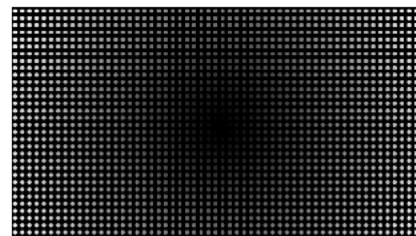
## Basics

Programs about form, data, images, color, typography, and more...

### Arrays



Array

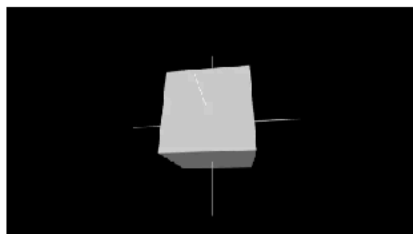


Array 2D

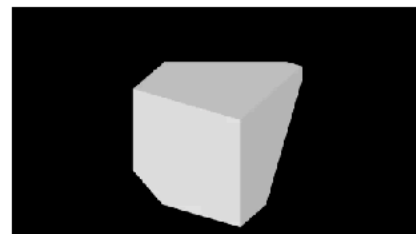


Array Objects

### Camera



Move Eye

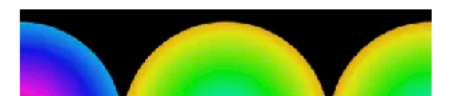


Orthographic



Perspective

### Color



We need your help!



Help us continue with your generosity!

[Donate](#)[A-Z](#)[Sort By Level](#)



# GETTING HELP

[Download](#)[Documentation](#)[Learn](#)[Teach](#)[About](#)[Donate](#)[Reference](#)[Environment](#)[Libraries](#)[Tools](#)

## Reference

Filter by keywords...

### Shortcuts

Data  
Rendering  
Output  
Structure

Input  
Image  
Color  
Control

Constants  
Shape  
Lights Camera  
Environment

Typography  
Math  
Transform

## Data

### Composite

Array

An array is a list of data

ArrayList

An `ArrayList` stores a variable number of objects

FloatDict

A simple table class to use a `String` as a lookup for a float value

FloatList

Helper class for a list of floats

HashMap

A `HashMap` stores a collection of objects, each referenced by a key

IntDict

A simple class to use a `String` as a lookup for an int value

IntList

Helper class for a list of ints

JSONArray

A JSONArray is an ordered sequence of values

We need  
your help!

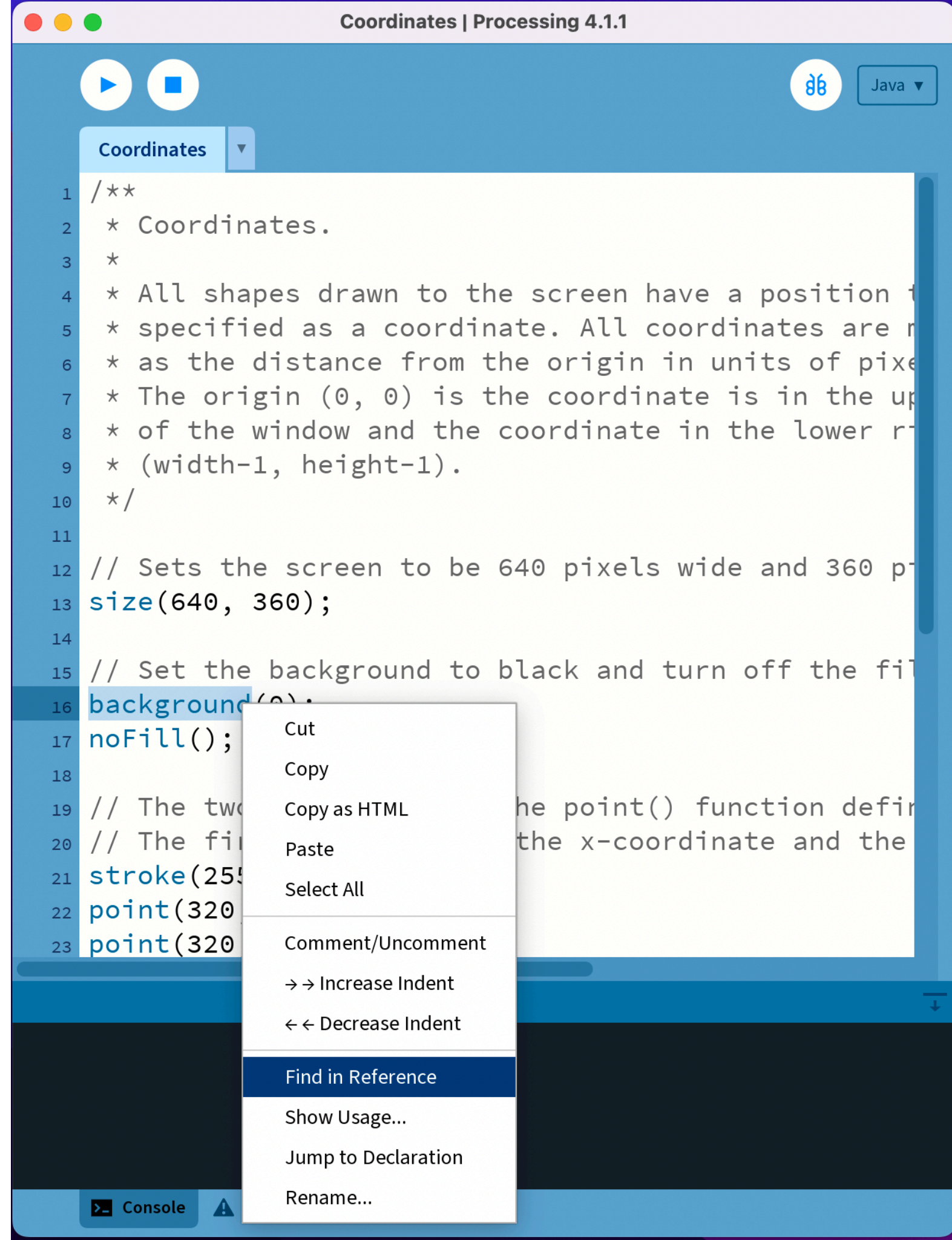


Help us continue with  
your generosity!

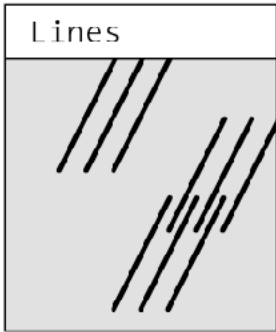
[Donate](#)

# GETTING HELP

Right click to go to  
reference page of  
the selected function



# PROCESSING DEVELOPMENT ENVIRONMENT (PDE)



Display window

Processing

File Edit Sketch Debug Tools Help

▶

■

Java ▼

Lines ▼

```
void setup() {  
  size(100, 100);  
  noLoop();  
}  
  
void draw() {  
  diagonals(40, 90);  
  diagonals(60, 62);  
  diagonals(20, 40);  
}  
  
void diagonals(int x, int y) {  
  line(x, y, x+20, y-40);  
  line(x+10, y, x+30, y-40);  
  line(x+20, y, x+40, y-40);  
}
```

Menu

Toolbar

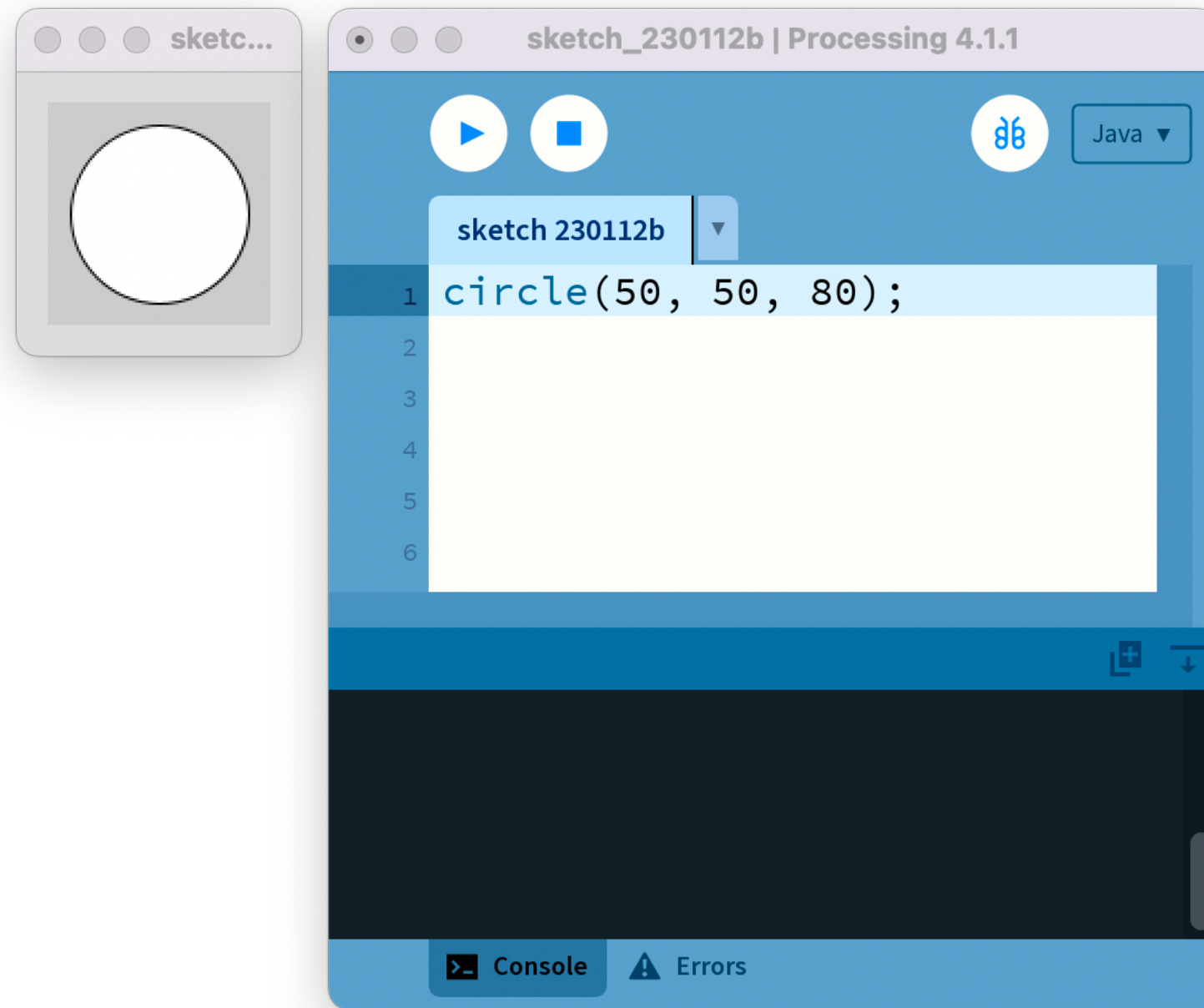
Tabs

Text editor

Message area

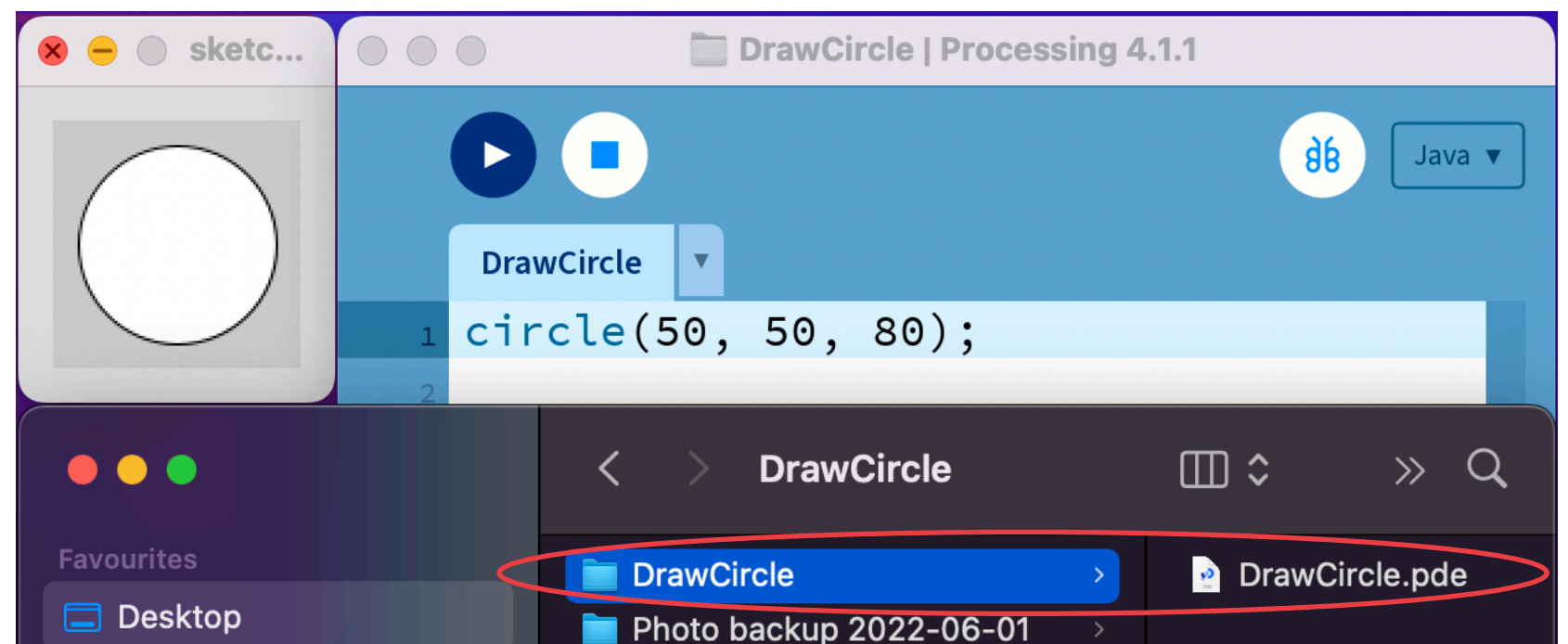
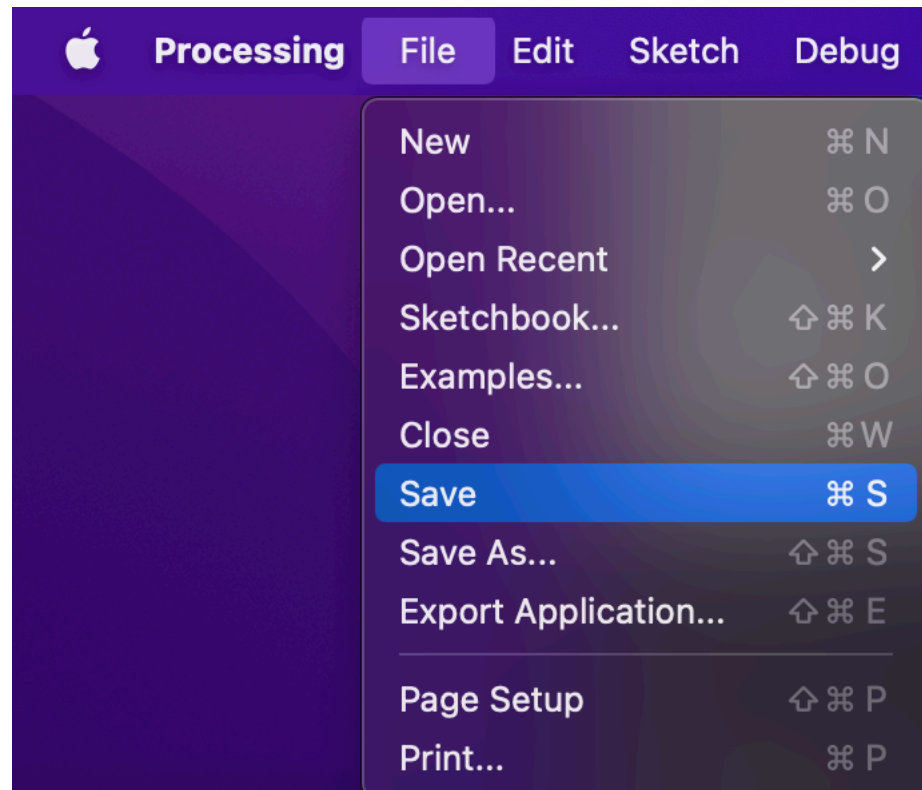
Console

# YOUR FIRST PROCESSING PROGRAM



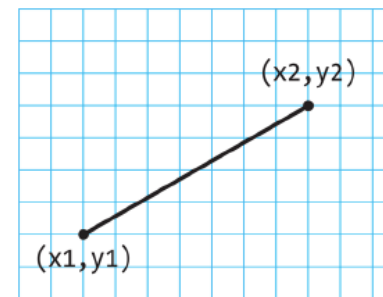


# SAVING A SKETCH

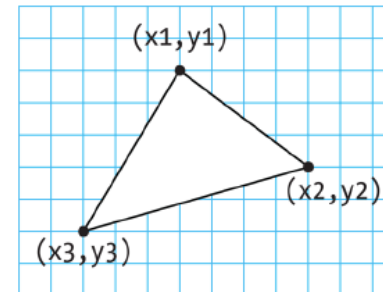


folder name = file name

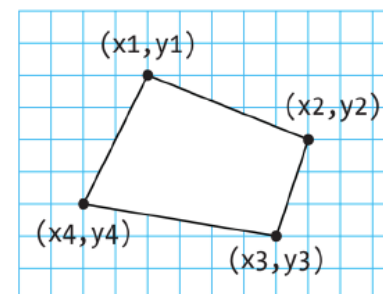
# 2D PRIMITIVE SHAPES AND THEIR COORDINATES



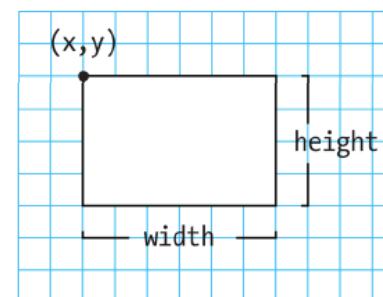
`line(x1, y1, x2, y2)`



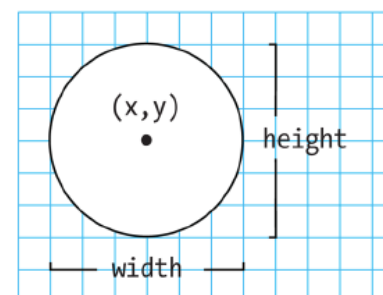
`triangle(x1, y1, x2, y2, x3, y3)`



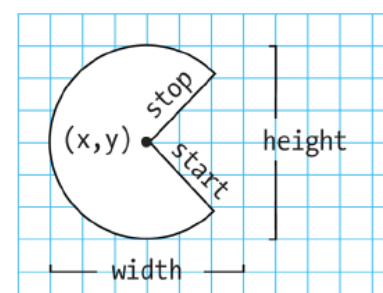
`quad(x1, y1, x2, y2, x3, y3, x4, y4)`



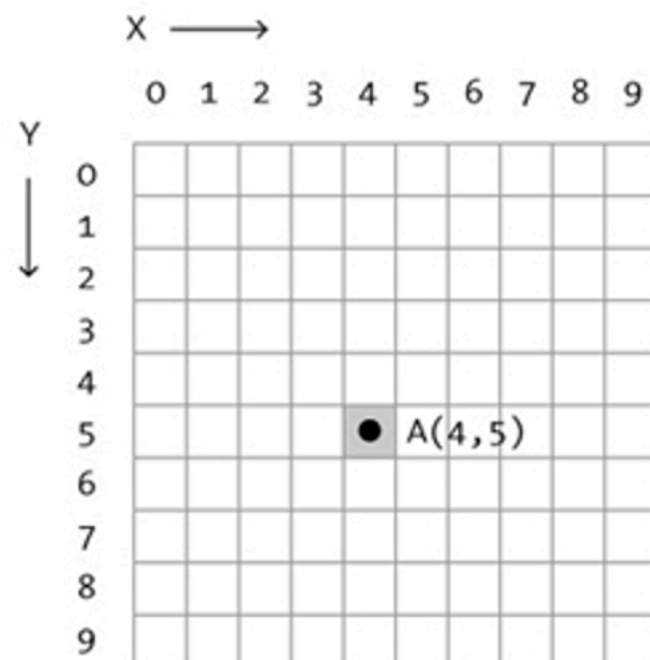
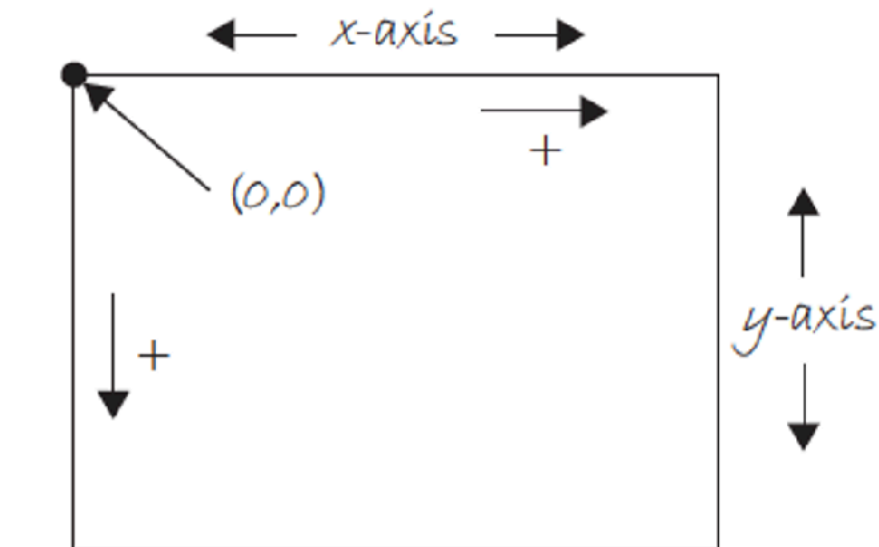
`rect(x, y, width, height)`



`ellipse(x, y, width, height)`



`arc(x, y, width, height, start, stop)`



`point(x,y);`

Example:  
`A(4,5);`

# MOVING FROM P5.JS TO PROCESSING

## p5.js

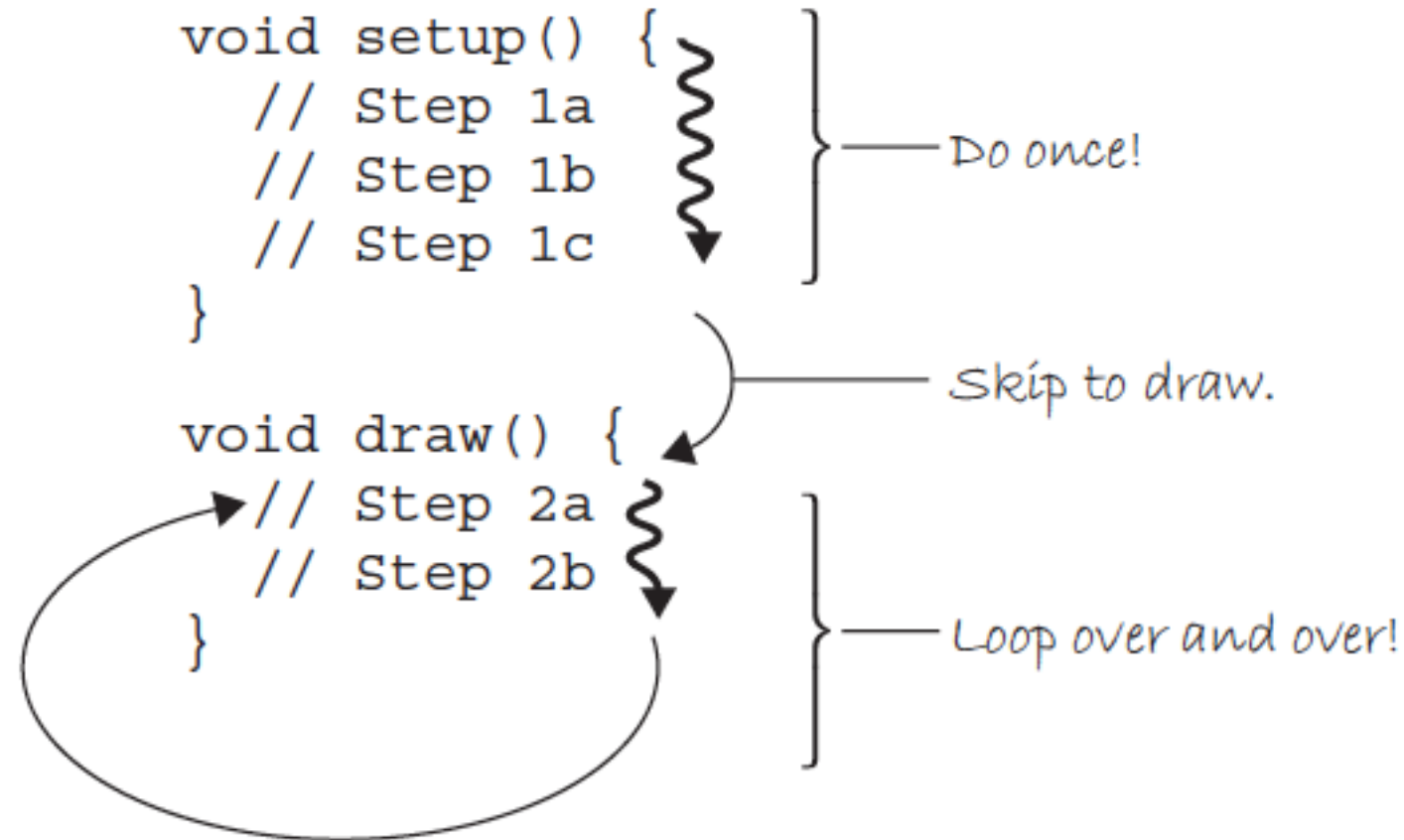
```
var yPos = 0;
function setup() {
  createCanvas(200, 400);
}
function draw() {
  background(204);
  yPos = yPos - 1;
  if (yPos < 0) {
    yPos = height;
  }
  line(0, yPos, width, yPos);
}
```

## Processing

```
int yPos = 0;
void setup() {
  size(200, 400);
}
void draw() {
  background(204);
  yPos = yPos - 1;
  if (yPos < 0) {
    yPos = height;
  }
  line(0, yPos, width, yPos);
}
```

# STRUCTURE AND FLOW OF PROGRAM

- `void setup()`: called first and once, to define initial environment properties
- `void draw()`: continuously executes the lines inside until the program is stopped or `noLoop()` is called
- `noLoop()`: stops Processing from continuously executing the code within `draw()` (to resume, use `loop()`)

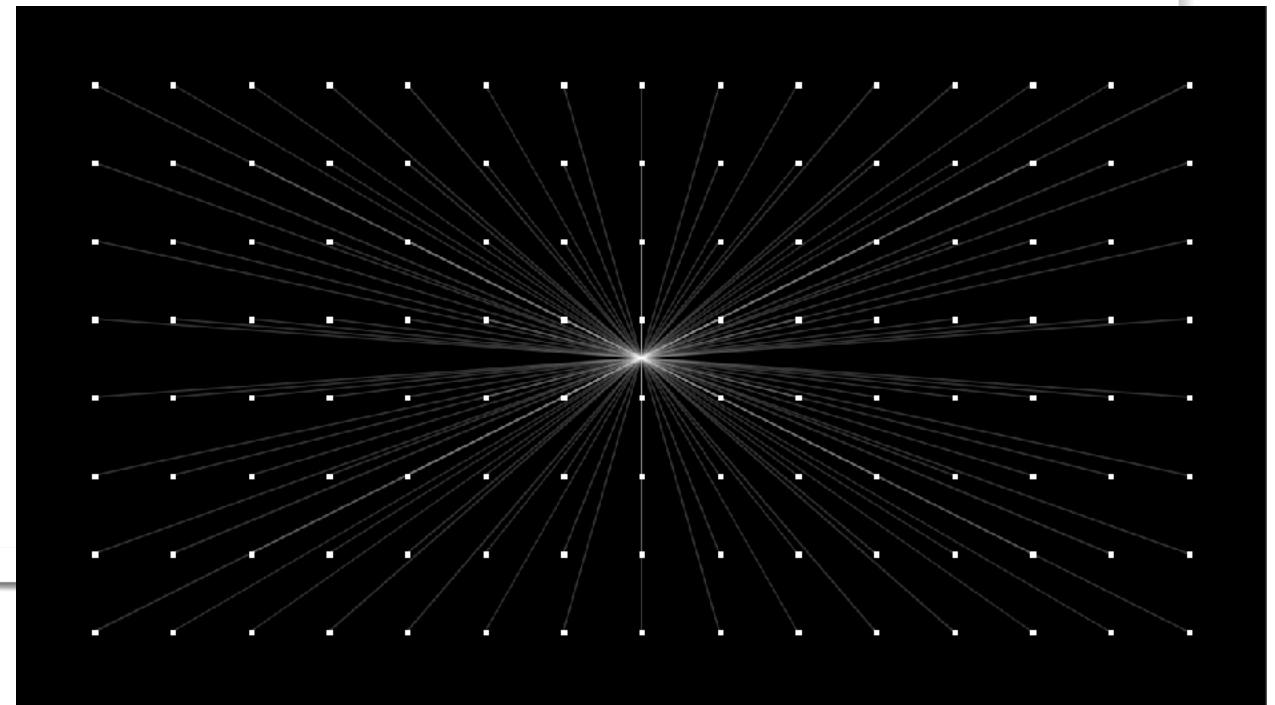
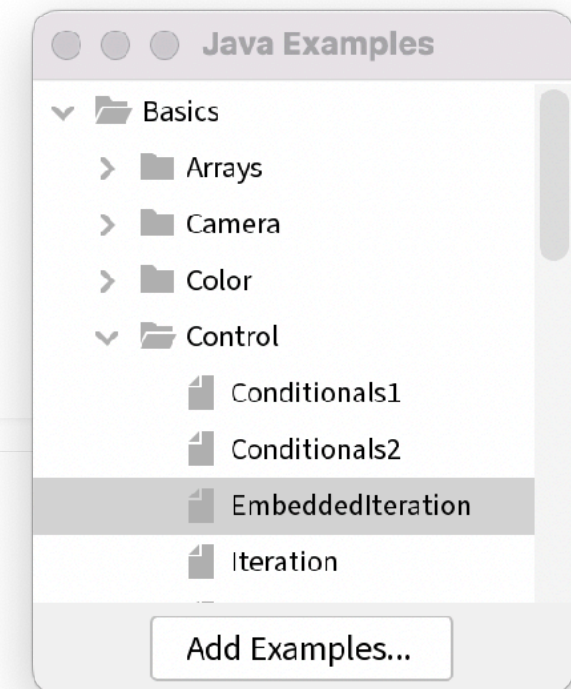




# UNSTRUCTURED PROGRAM

- Does not contain `setup()` nor `draw()` functions
- Static sketch only - no interaction, dynamics and animation

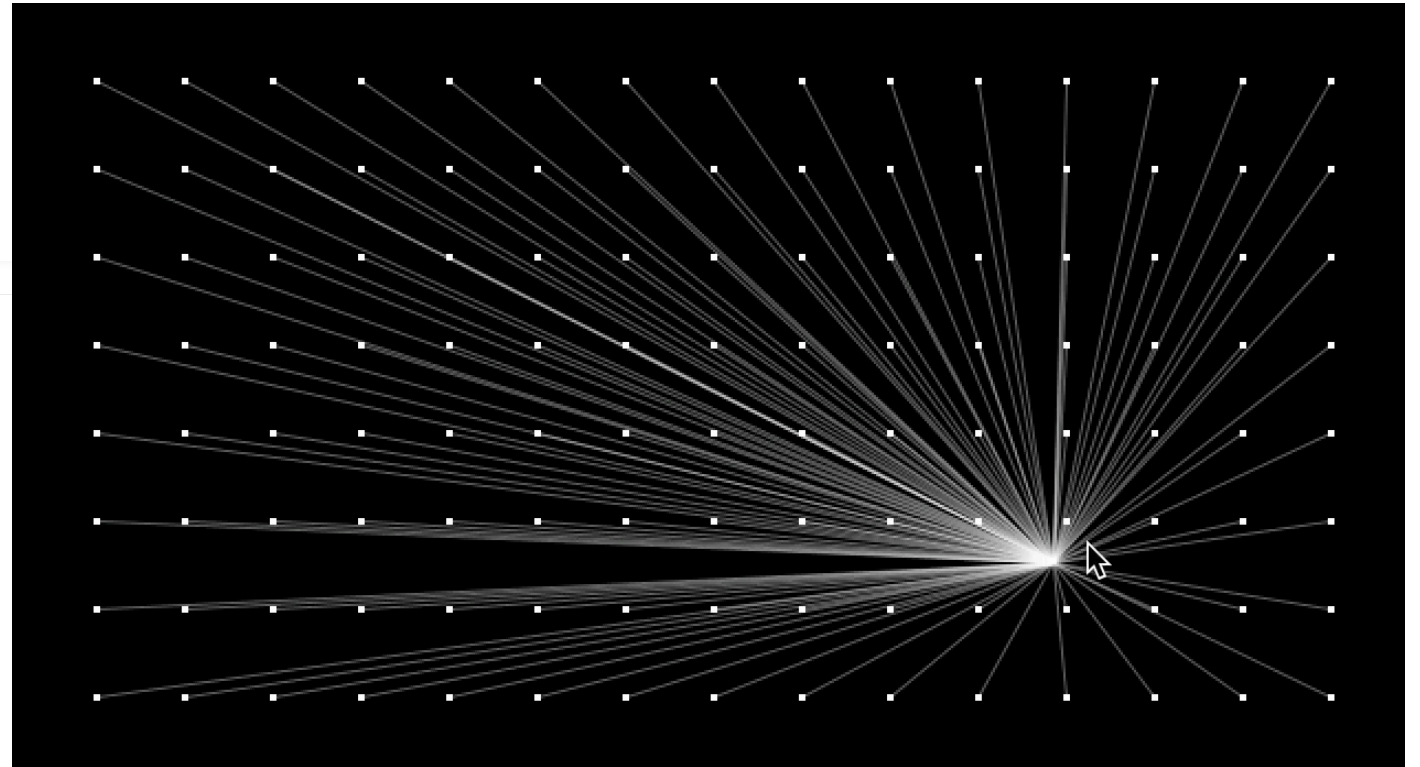
```
size(640, 360);  
background(0);  
  
int gridSize = 40;  
  
for (int x = gridSize; x <= width - gridSize; x += gridSize) {  
  for (int y = gridSize; y <= height - gridSize; y += gridSize) {  
    noStroke();  
    fill(255);  
    rect(x-1, y-1, 3, 3);  
    stroke(255, 100);  
    line(x, y, width/2, height/2);  
  }  
}
```



# STRUCTURED PROGRAM

- Contains one setup() and one draw() function
- Dynamic sketches - allow animation and interactivities

```
void setup() {  
  size(640, 360);  
}  
  
void draw() {  
  background(0);  
  
  int gridSize = 40;  
  
  for (int x = gridSize; x <= width - gridSize; x += gridSize) {  
    for (int y = gridSize; y <= height - gridSize; y += gridSize) {  
      noStroke();  
      fill(255);  
      rect(x-1, y-1, 3, 3);  
      stroke(255, 100);  
      line(x, y, mouseX, mouseY);  
    }  
  }  
}
```




# EXAMPLE 1 WHERE TO PUT BACKGROUND()?

Compare these two programs:


```
void setup() {
  size(400, 300);
  noStroke();
  fill(100);
}

void draw() {
  background(255);
  ellipse(mouseX, mouseY, 10, 10);
}
```



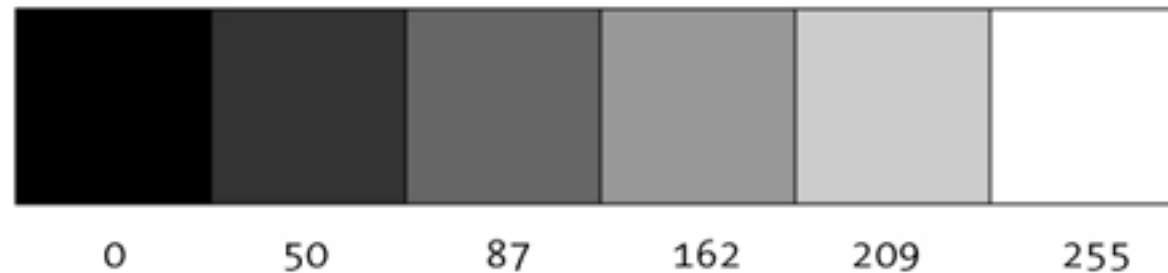
```
void setup() {
  size(400, 300);
  background(255);
  noStroke();
  fill(100);
}

void draw() {
  ellipse(mouseX, mouseY, 10, 10);
}
```

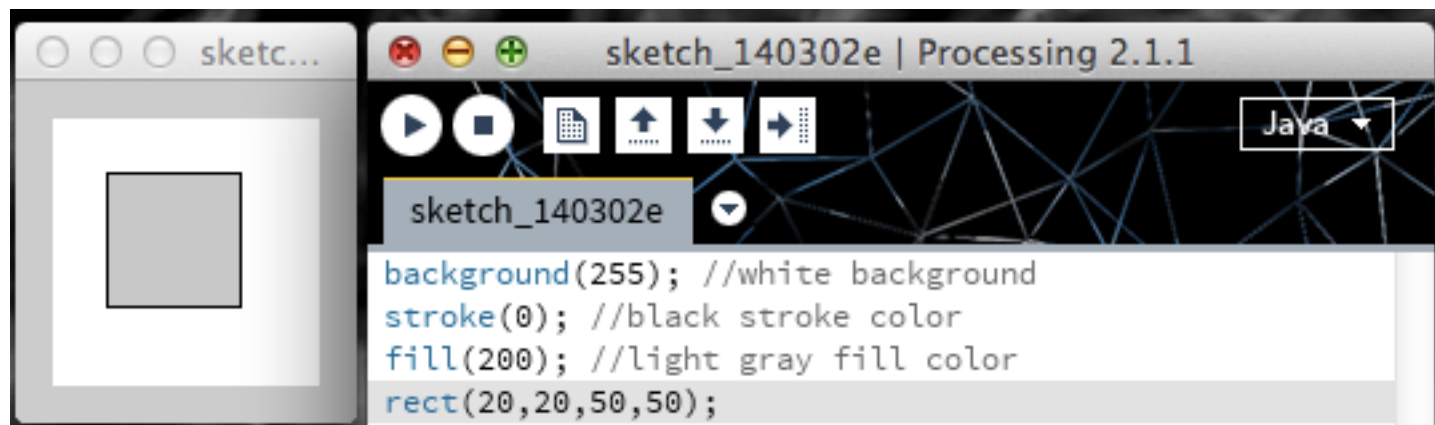


# GRAYSCALE COLOR

- 8-bit: 0-255 ( $2^8 = 256$ )
- 0 as black and 255 as white



- eg. `background(255); //white background`  
`stroke(0); //black stroke color`  
`fill(200); //light gray fill color`



## Color

### Setting

`background()`  
`clear()`  
`colorMode()`  
`fill()`  
`noFill()`  
`noStroke()`  
`stroke()`

### Creating & Reading

`alpha()`  
`blue()`  
`brightness()`  
`color()`  
`green()`  
`hue()`  
`lerpColor()`  
`red()`  
`saturation()`

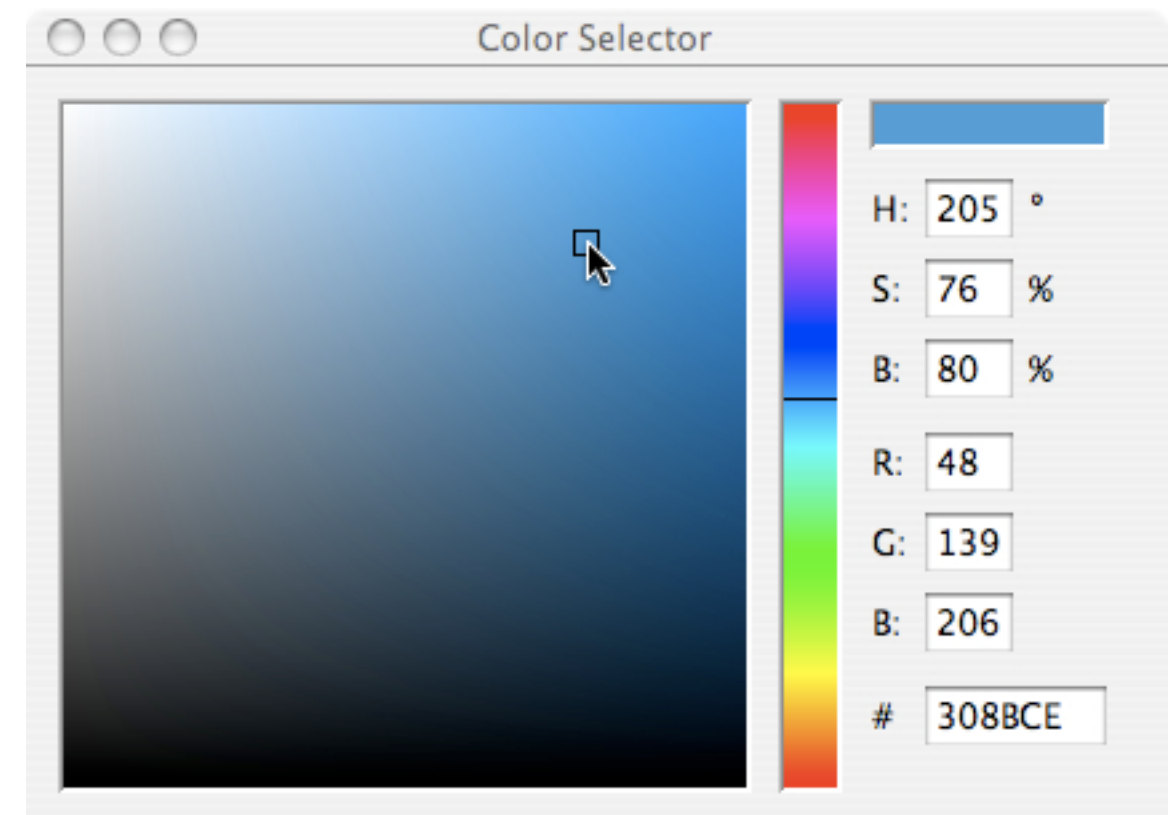
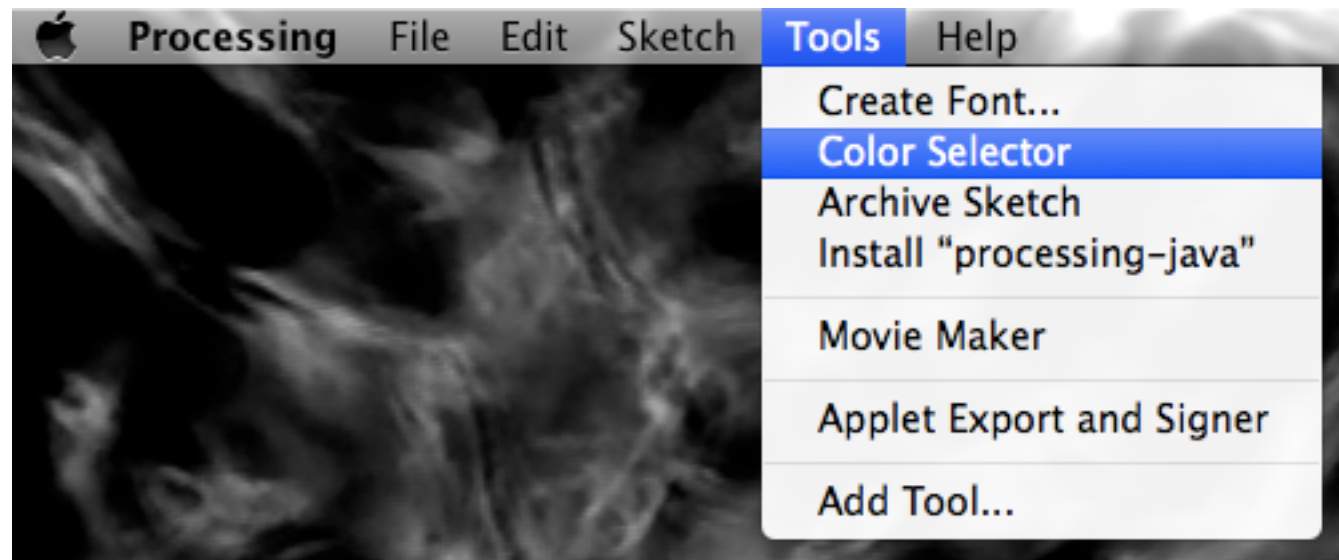
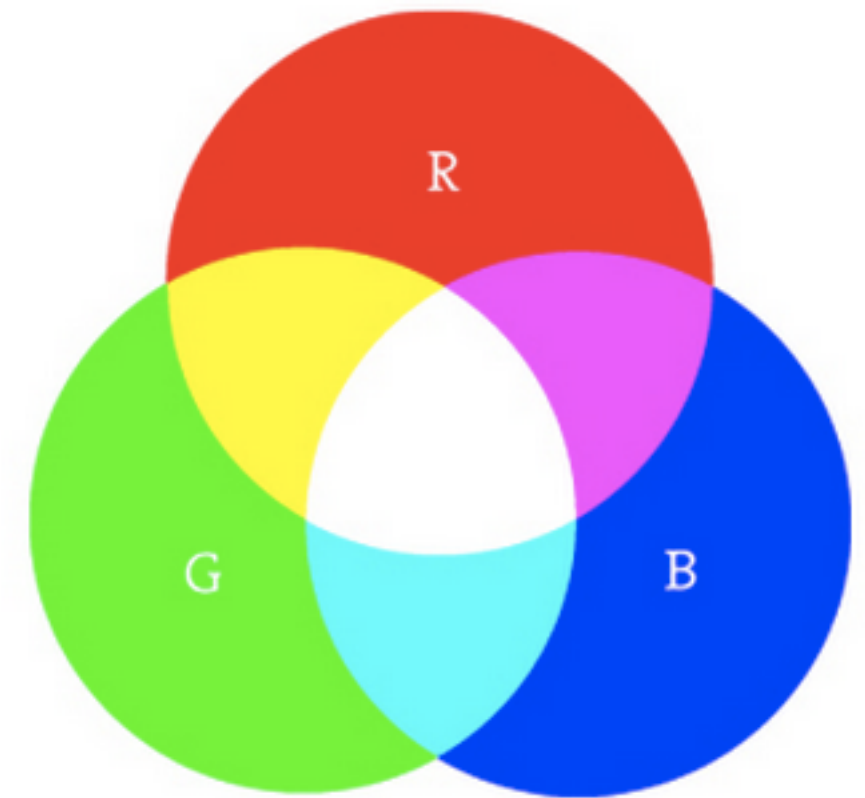
# BIT COMBINATIONS

	Examples	Possible
1-bit	<u>0</u>	$2^1 = 2$
2-bit	<u>01</u>	$2^2 = 4$
3-bit	<u>101</u>	$2^3 = 8$
4-bit	<u>0110</u>	$2^4 = 16$
5-bit	<u>10100</u>	$2^5 = 32$
6-bit	<u>001011</u>	$2^6 = 64$
7-bit	<u>1100110</u>	$2^7 = 128$
8-bit	<u>10011010</u>	$2^8 = 256$



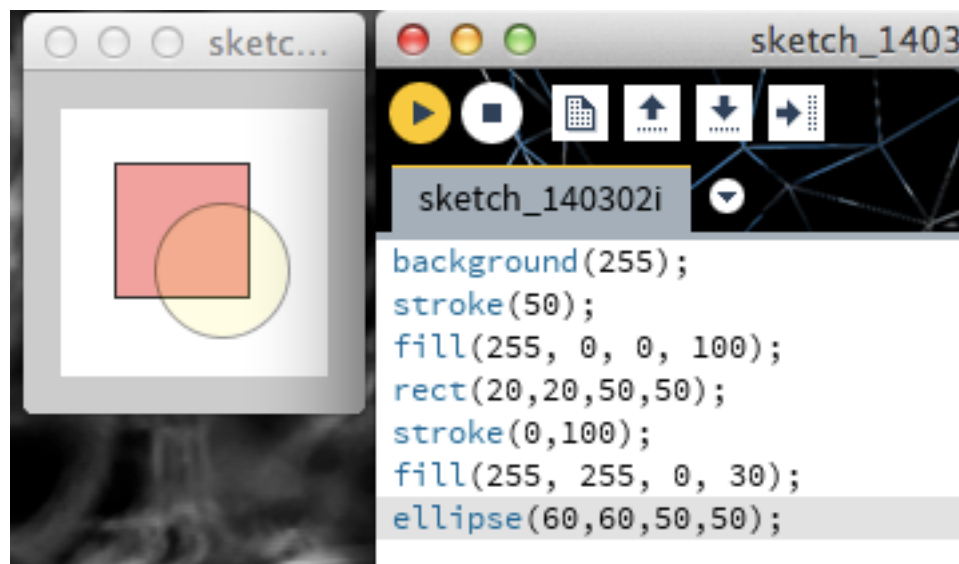
# RGB COLOR

- 24-bit full color
- 8-bit for each colour of red, green and blue (3 x 8)
- eg. `background(255,0,0); //red`  
`fill(48,139,206);`
- Color selector tool



# TRANSPARENCY (ALPHA VALUE)

- Alpha values also range from 0 to 255
- 0: completely transparent (i.e., 0% opaque)
- 255 completely opaque (i.e., 100% opaque).
- eg. `fill(0,50);`  
`stroke(255,0,0,100);`



- Transparency not applicable to background!

## Example: Alpha transparency

```
size(200,200);  
background(0);  
noStroke();
```

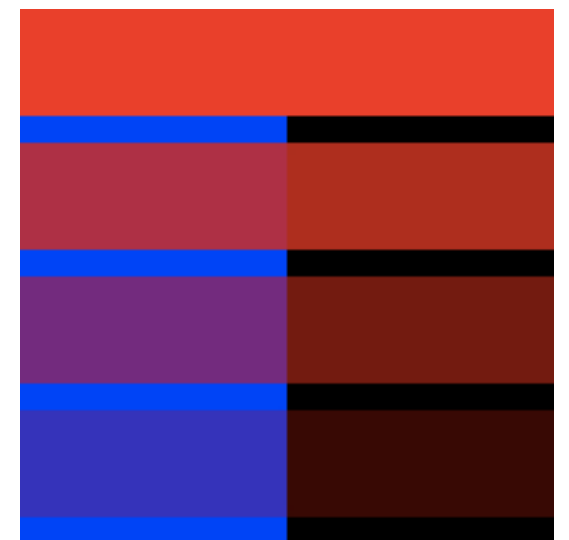
```
// No fourth argument means 100% opacity.  
fill(0,0,255);  
rect(0,0,100,200);
```

```
// 255 means 100% opacity.  
fill(255,0,0,255);  
rect(0,0,200,40);
```

```
// 75% opacity.  
fill(255,0,0,191);  
rect(0,50,200,40);
```

```
// 55% opacity.  
fill(255,0,0,127);  
rect(0,100,200,40);
```

```
// 25% opacity.  
fill(255,0,0,63);  
rect(0,150,200,40);
```



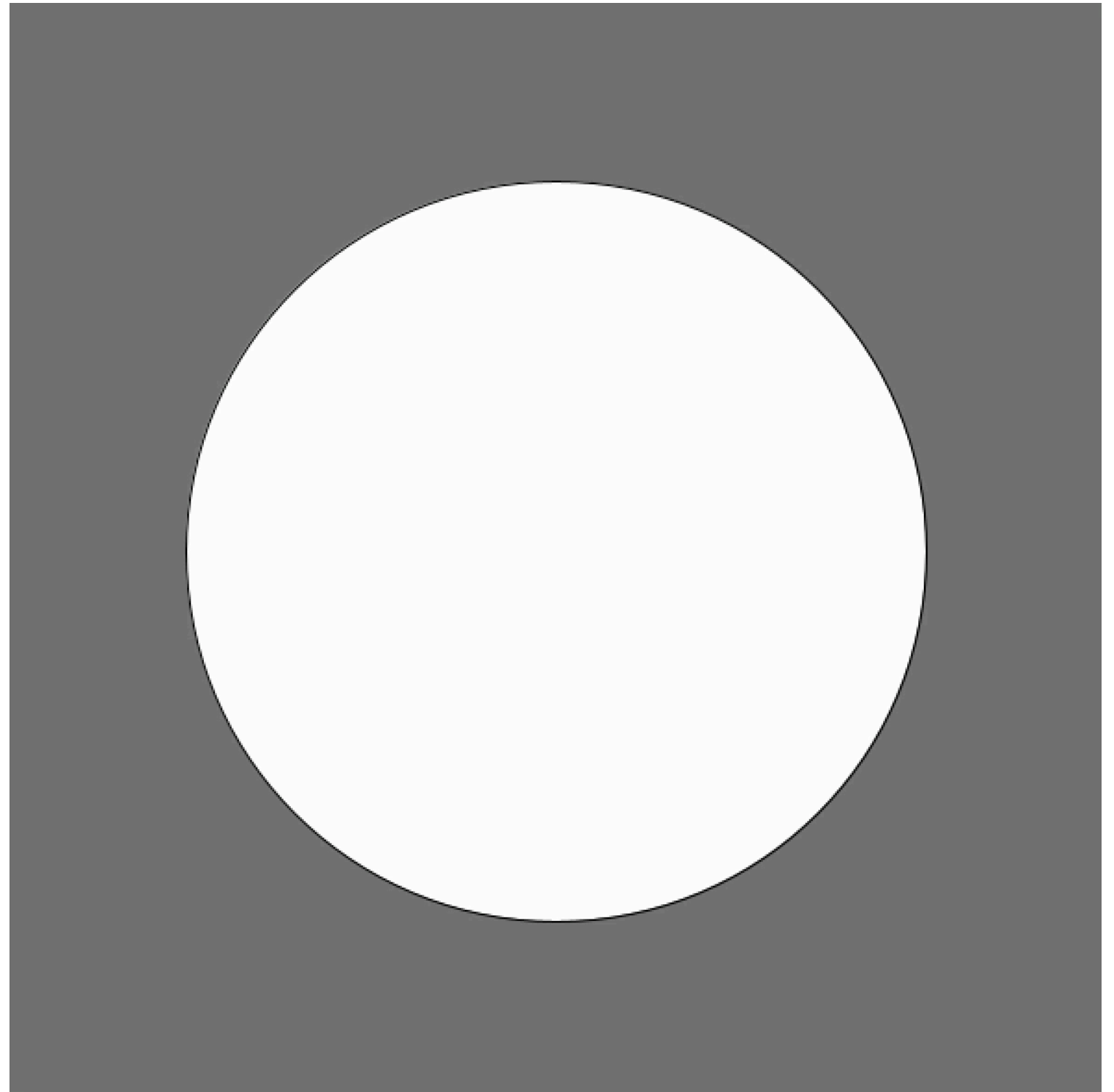
# COLOR

- `fill(50); // grayscale color`
- `fill(50, 100); // grayscale color with transparency`
- `fill(50, 100, 150); // rgb color`
- `fill(50, 100, 150, 200); // rgb color with transparency`



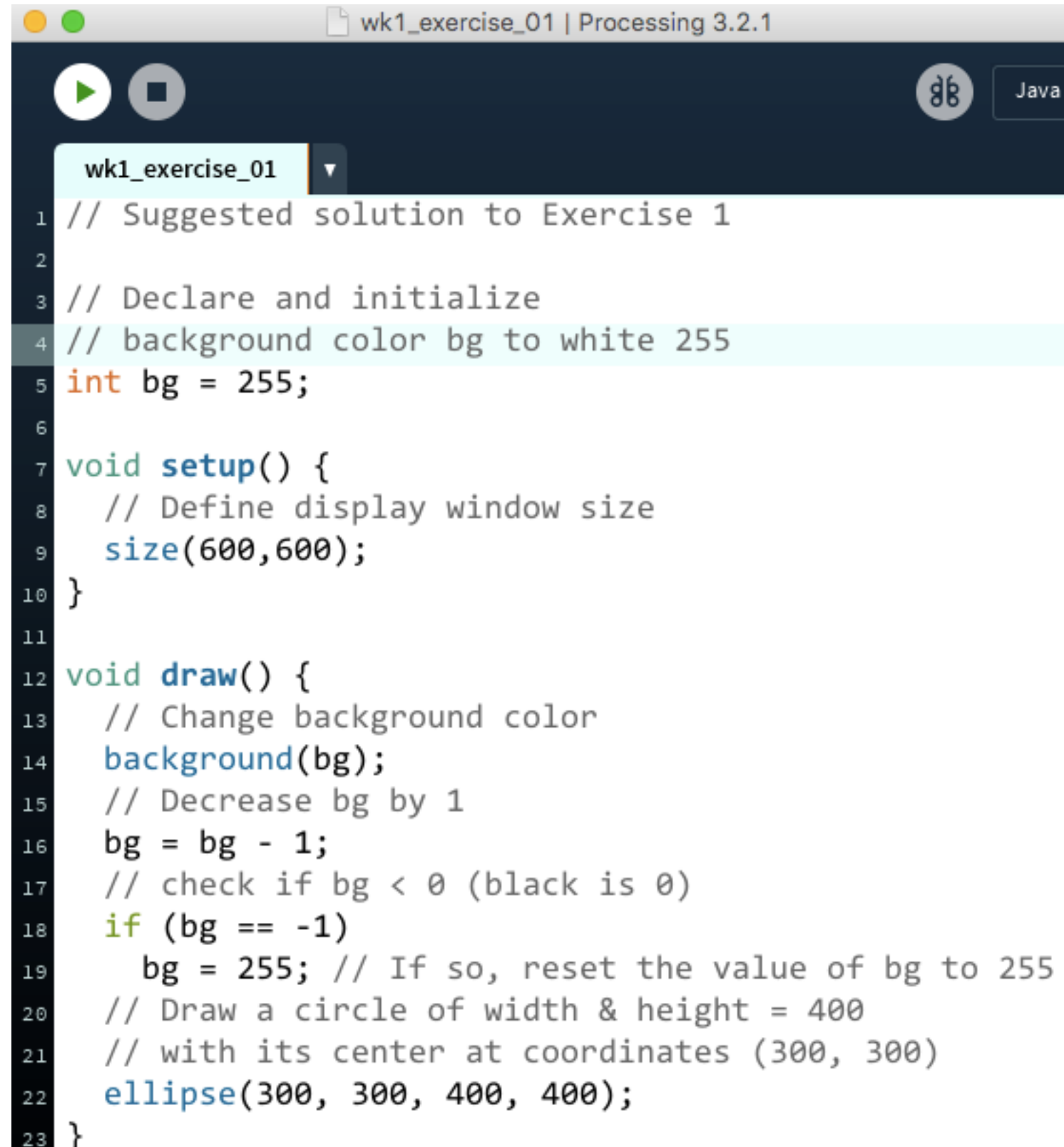
# EXERCISE 1

- Write a structured program that
  - gives a display window of dimension 600x600
  - has a changing background color from white to black, then white to black again continuously
  - draws a circle of radius 200 (width,height=400) at the middle of the canvas



# SUGGESTED CODE TO EXERCISE 1

- Background color is updated in every frame, thus `background(bg)` has to be put inside `void draw()` but not `void setup()`
- `bg = bg - 1`; Background color is decreased by 1 each time 255, 254, 253, ... , 2, 1, 0
- After drawing `background(0)` (black), `bg = 0 - 1`, then `bg = -1`; thus `bg` is reset to 255 (white) again
- The dimension of the canvas is 600x600, thus centre of the canvas is (300,300)
- Although the circle is static (not changing), but it has to be drawn again each time after the background color is updated



```
1 // Suggested solution to Exercise 1
2
3 // Declare and initialize
4 // background color bg to white 255
5 int bg = 255;
6
7 void setup() {
8   // Define display window size
9   size(600,600);
10 }
11
12 void draw() {
13   // Change background color
14   background(bg);
15   // Decrease bg by 1
16   bg = bg - 1;
17   // check if bg < 0 (black is 0)
18   if (bg == -1)
19     bg = 255; // If so, reset the value of bg to 255
20   // Draw a circle of width & height = 400
21   // with its center at coordinates (300, 300)
22   ellipse(300, 300, 400, 400);
23 }
```

# VARIABLE DECLARATION & INITIALISATION

- `data_type variable_name = init_value;`
  - `int a = 10;`
- **Variable naming rules:**
  - Must be one word (no spaces)
  - Must start with a letter
    - Can include numbers; but not start with a number
  - Can include underscore “\_”
  - Must be unique
    - **CANNOT** be the same as system variables, e.g.,
      - `mouseX`, `mouseY`, `width`, `height`, `frameCount`, `key`, `keyCode`, `keyPressed`, `mousePressed`, `mouseButton`

# DATA TYPES

Name	Value range	Example
boolean	true or false (1 bit)	boolean T = true, F = false;
char	0 to 65535 (16 bits)	char myChar1='A', myChar2='\$';
byte	-128 to 127 (8 bits)	byte b = -128;
int	-2147483648 to 2147483647 (32 bits)	int number = 800, temp = -1;
float	3.40282347E+38 to -3.40282347E+38 (32 bits)	float b = -2.984;
color	16,777,216 colors (32 bits)	color c1 = color(204,153,0), c2 = #FFCC00;

```
int x;           // Declare the variable x of type int
float y;         // Declare the variable y of type float
boolean b;       // Declare the variable b of type boolean
x = 50;          // Assign the value 50 to x
y = 12.6;        // Assign the value 12.6 to y
b = true;        // Assign the value true to b
```

# ASCII TABLE

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	<b>NUL</b> (null)	32	20	040	&#32;	<b>Space</b>	64	40	100	&#64;	<b>@</b>	96	60	140	&#96;	<b>`</b>
1	1	001	<b>SOH</b> (start of heading)	33	21	041	&#33;	<b>!</b>	65	41	101	&#65;	<b>A</b>	97	61	141	&#97;	<b>a</b>
2	2	002	<b>STX</b> (start of text)	34	22	042	&#34;	<b>"</b>	66	42	102	&#66;	<b>B</b>	98	62	142	&#98;	<b>b</b>
3	3	003	<b>ETX</b> (end of text)	35	23	043	&#35;	<b>#</b>	67	43	103	&#67;	<b>C</b>	99	63	143	&#99;	<b>c</b>
4	4	004	<b>EOT</b> (end of transmission)	36	24	044	&#36;	<b>\$</b>	68	44	104	&#68;	<b>D</b>	100	64	144	&#100;	<b>d</b>
5	5	005	<b>ENQ</b> (enquiry)	37	25	045	&#37;	<b>%</b>	69	45	105	&#69;	<b>E</b>	101	65	145	&#101;	<b>e</b>
6	6	006	<b>ACK</b> (acknowledge)	38	26	046	&#38;	<b>&amp;</b>	70	46	106	&#70;	<b>F</b>	102	66	146	&#102;	<b>f</b>
7	7	007	<b>BEL</b> (bell)	39	27	047	&#39;	<b>'</b>	71	47	107	&#71;	<b>G</b>	103	67	147	&#103;	<b>g</b>
8	8	010	<b>BS</b> (backspace)	40	28	050	&#40;	<b>(</b>	72	48	110	&#72;	<b>H</b>	104	68	150	&#104;	<b>h</b>
9	9	011	<b>TAB</b> (horizontal tab)	41	29	051	&#41;	<b>)</b>	73	49	111	&#73;	<b>I</b>	105	69	151	&#105;	<b>i</b>
10	A	012	<b>LF</b> (NL line feed, new line)	42	2A	052	&#42;	<b>*</b>	74	4A	112	&#74;	<b>J</b>	106	6A	152	&#106;	<b>j</b>
11	B	013	<b>VT</b> (vertical tab)	43	2B	053	&#43;	<b>+</b>	75	4B	113	&#75;	<b>K</b>	107	6B	153	&#107;	<b>k</b>
12	C	014	<b>FF</b> (NP form feed, new page)	44	2C	054	&#44;	<b>,</b>	76	4C	114	&#76;	<b>L</b>	108	6C	154	&#108;	<b>l</b>
13	D	015	<b>CR</b> (carriage return)	45	2D	055	&#45;	<b>-</b>	77	4D	115	&#77;	<b>M</b>	109	6D	155	&#109;	<b>m</b>
14	E	016	<b>SO</b> (shift out)	46	2E	056	&#46;	<b>.</b>	78	4E	116	&#78;	<b>N</b>	110	6E	156	&#110;	<b>n</b>
15	F	017	<b>SI</b> (shift in)	47	2F	057	&#47;	<b>/</b>	79	4F	117	&#79;	<b>O</b>	111	6F	157	&#111;	<b>o</b>
16	10	020	<b>DLE</b> (data link escape)	48	30	060	&#48;	<b>0</b>	80	50	120	&#80;	<b>P</b>	112	70	160	&#112;	<b>p</b>
17	11	021	<b>DC1</b> (device control 1)	49	31	061	&#49;	<b>1</b>	81	51	121	&#81;	<b>Q</b>	113	71	161	&#113;	<b>q</b>
18	12	022	<b>DC2</b> (device control 2)	50	32	062	&#50;	<b>2</b>	82	52	122	&#82;	<b>R</b>	114	72	162	&#114;	<b>r</b>
19	13	023	<b>DC3</b> (device control 3)	51	33	063	&#51;	<b>3</b>	83	53	123	&#83;	<b>S</b>	115	73	163	&#115;	<b>s</b>
20	14	024	<b>DC4</b> (device control 4)	52	34	064	&#52;	<b>4</b>	84	54	124	&#84;	<b>T</b>	116	74	164	&#116;	<b>t</b>
21	15	025	<b>NAK</b> (negative acknowledge)	53	35	065	&#53;	<b>5</b>	85	55	125	&#85;	<b>U</b>	117	75	165	&#117;	<b>u</b>
22	16	026	<b>SYN</b> (synchronous idle)	54	36	066	&#54;	<b>6</b>	86	56	126	&#86;	<b>V</b>	118	76	166	&#118;	<b>v</b>
23	17	027	<b>ETB</b> (end of trans. block)	55	37	067	&#55;	<b>7</b>	87	57	127	&#87;	<b>W</b>	119	77	167	&#119;	<b>w</b>
24	18	030	<b>CAN</b> (cancel)	56	38	070	&#56;	<b>8</b>	88	58	130	&#88;	<b>X</b>	120	78	170	&#120;	<b>x</b>
25	19	031	<b>EM</b> (end of medium)	57	39	071	&#57;	<b>9</b>	89	59	131	&#89;	<b>Y</b>	121	79	171	&#121;	<b>y</b>
26	1A	032	<b>SUB</b> (substitute)	58	3A	072	&#58;	<b>:</b>	90	5A	132	&#90;	<b>Z</b>	122	7A	172	&#122;	<b>z</b>
27	1B	033	<b>ESC</b> (escape)	59	3B	073	&#59;	<b>:</b>	91	5B	133	&#91;	<b>[</b>	123	7B	173	&#123;	<b>{</b>
28	1C	034	<b>FS</b> (file separator)	60	3C	074	&#60;	<b>&lt;</b>	92	5C	134	&#92;	<b>\</b>	124	7C	174	&#124;	<b> </b>
29	1D	035	<b>GS</b> (group separator)	61	3D	075	&#61;	<b>=</b>	93	5D	135	&#93;	<b>]</b>	125	7D	175	&#125;	<b>}</b>
30	1E	036	<b>RS</b> (record separator)	62	3E	076	&#62;	<b>&gt;</b>	94	5E	136	&#94;	<b>^</b>	126	7E	176	&#126;	<b>~</b>
31	1F	037	<b>US</b> (unit separator)	63	3F	077	&#63;	<b>?</b>	95	5F	137	&#95;	<b>_</b>	127	7F	177	&#127;	<b>DEL</b>

128	Ç	144	É	160	á	176	ð	192	Ł	208	ł	224	α	240	≡
129	ü	145	æ	161	í	177	ñ	193	Ł	209	ł	225	β	241	±
130	é	146	Æ	162	ó	178	■	194	Ŧ	210	ŧ	226	Γ	242	≥
131	â	147	ô	163	ú	179		195	Ŧ	211	ŧ	227	π	243	≤
132	ä	148	ö	164	ñ	180	†	196	—	212	↳	228	Σ	244	ƒ
133	à	149	ò	165	Ñ	181	‡	197	†	213	↳	229	σ	245	Ƶ
134	ã	150	û	166	ª	182	‡	198	‡	214	↳	230	μ	246	÷
135	ç	151	ù	167	º	183	¶	199	‡	215	‡	231	τ	247	≈
136	ê	152	ÿ	168	¿	184	¶	200	↳	216	‡	232	Φ	248	°
137	ë	153	Ö	169	ƒ	185	¶	201	↳	217	↳	233	Θ	249	·
138	è	154	Û	170	ƒ	186	¶	202	↳	218	↳	234	Ω	250	·
139	ï	155	◊	171	½	187	¶	203	↳	219	↳	235	δ	251	√
140	î	156	£	172	¼	188	↳	204	↳	220	↳	236	∞	252	∞
141	ì	157	¥	173	¡	189	↳	205	↳	221	↳	237	φ	253	²
142	Ä	158	£	174	«	190	↳	206	↳	222	↳	238	ε	254	■
143	Å	159	ƒ	175	»	191	↳	207	↳	223	↳	239	∩	255	

# EXAMPLE 2 DATA TYPE

```
wk1_example_02_data_type
1 boolean a = false;
2 char b = '1';
3 char c = 97;
4 byte d = -128;
5 int e = 2047;
6 float f = 3.1415;
7 float g = 3;
8 color h = color(204, 153, 0);
9 color i = #FFCC00;
10
11 println(a);
12 println(b);
13 println(c);
14 println(d);
15 println(e);
16 println(f);
17 println(g);
18 println(h);
19 println(i);
```

```
false
1
a
-128
2047
3.1415
3.0
-3368704
-13312
```

# DATA TYPES

## Data

### Primitive

boolean

byte

char

color

double

float

int

long

## Conversion

binary()

boolean()

byte()

char()

float()

hex()

int()

str()

unbinary()

unhex()

```
float f = 65.0;
int i = int(f);
println(f + " : " + i); // Prints "65.0 : 65"
```

```
char c = 'E';
i = int(c);
println(c + " : " + i); // Prints "E : 69"
```

```
boolean b = false;
byte y = -28;
char c = 'R';
float f = -32.6;
int i = 1024;
```

```
String sb = str(b);
String sy = str(y);
String sc = str(c);
String sf = str(f);
String si = str(i);
```

```
sb = sb + sy + sc + sf + si;
```

```
println(sb); // Prints 'false-28R-32.61024'
```

# MOUSE EVENTS

- mouseX, mouseY
- pmouseX, pmouseY
- mouseMoved()
- mouseDragged()
- mousePressed(), mousePressed returns true or false
- mouseClicked()
- mouseReleased()
- mouseButton returns LEFT, RIGHT, or CENTER
- cursor(), noCursor()



# KEYBOARD EVENTS

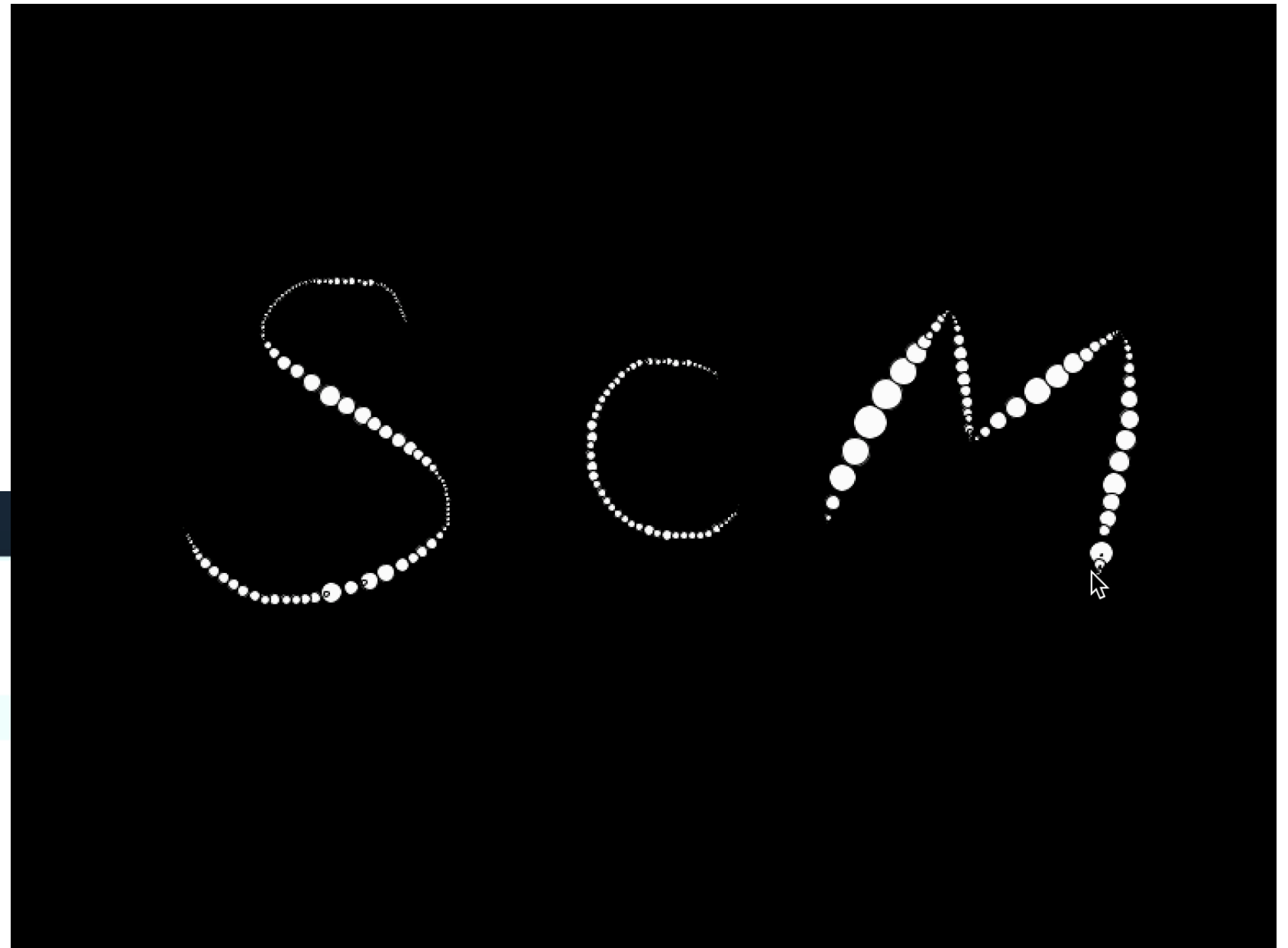
- key
- keyCode
  - To detect special keys such as UP, DOWN, LEFT, ALT, CONTROL SHIFT
- keyPressed() & keyReleased()
- keyTyped()

```
color fillVal = color(126);

void draw() {
    fill(fillVal);
    rect(25, 25, 50, 50);
}

void keyPressed() {
    if (key == CODED) {
        if (keyCode == UP) {
            fillVal = 255;
        } else if (keyCode == DOWN) {
            fillVal = 0;
        }
    } else {
        fillVal = 126;
    }
}
```

# EXAMPLE 3



```
wk1_example_03_doodling
1 void setup() {
2   size(600, 400);
3   background(0);
4   stroke(0);
5 }
6
7 void draw() {
8 }
9
10 void mouseDragged() {
11   float d = dist(mouseX, mouseY, pmouseX, pmouseY);
12   ellipse(mouseX, mouseY, d, d);
13 }
14
15 void keyPressed() {
16   background(0);
17 }
```