



# SM2715 (L02) Creative Coding week 01



L02, week 01

by Mike Wong. School of Creative Media

# Evaluation

## TASKS

## Weighting

Coding Assignment x 3 ( individual )	50% (15%, 15%, 20%)
Final Project ( Group: 2 persons )	40%
In-class Exercise & Participation	10%

# Policies

## Attendance & Participation:

10% of total grade

more than 3 absences may fail the course

## Assignment Late Submission:

10% mark deduction per day

## Academic Honesty / Plagiarism:

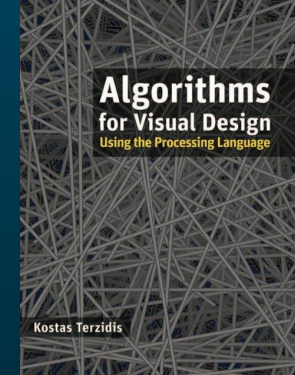
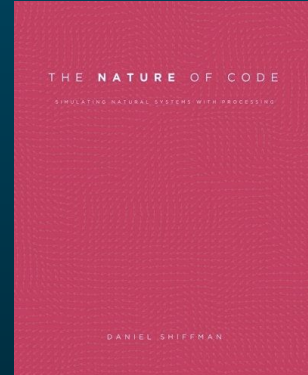
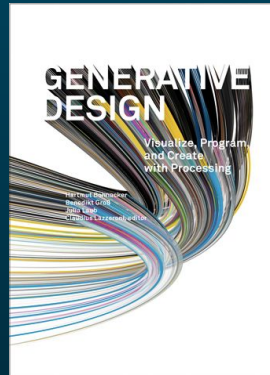
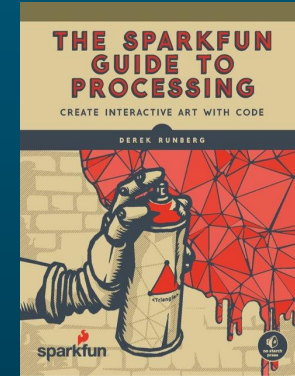
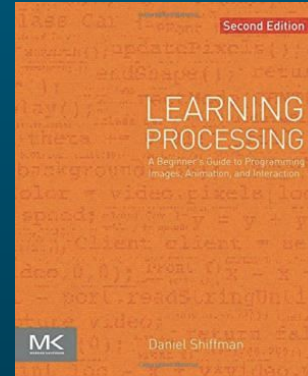
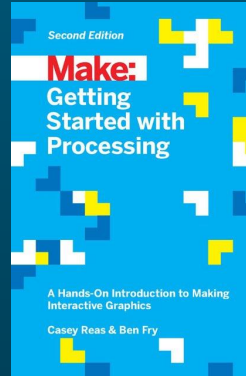
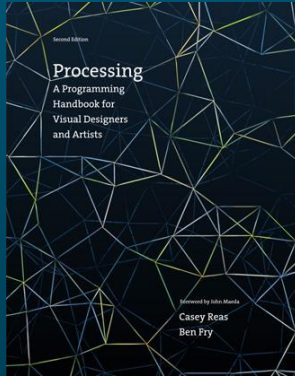
School's default penalty: **F-grade** for the course

<https://www6.cityu.edu.hk/ah/plagiarism.htm>

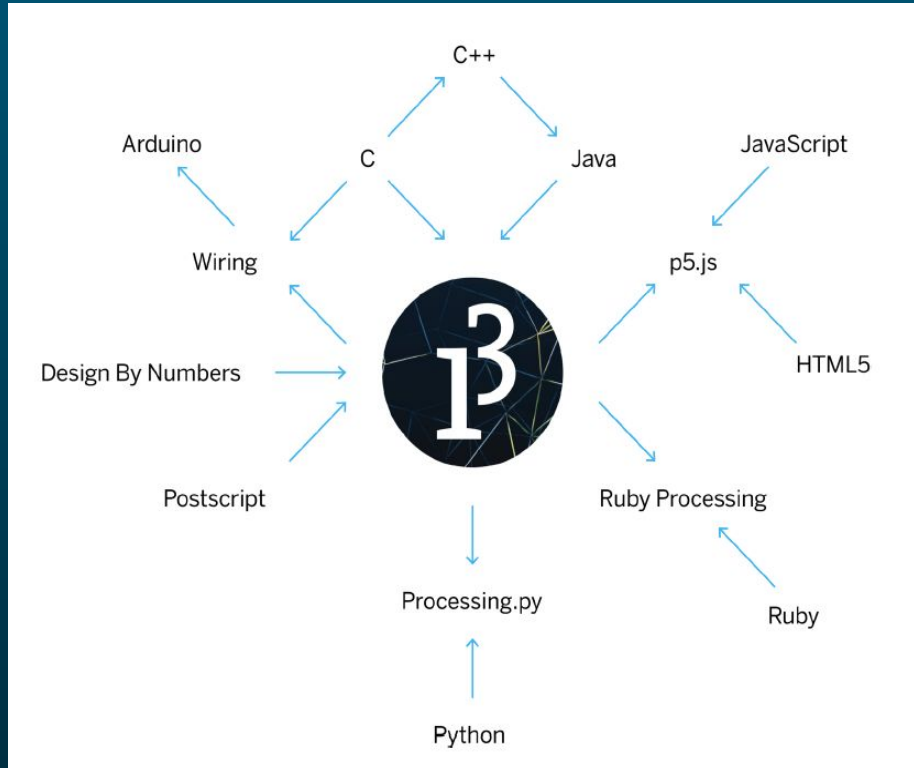
# Week 01

# Basics of Processing

# References



# Processing Family Tree

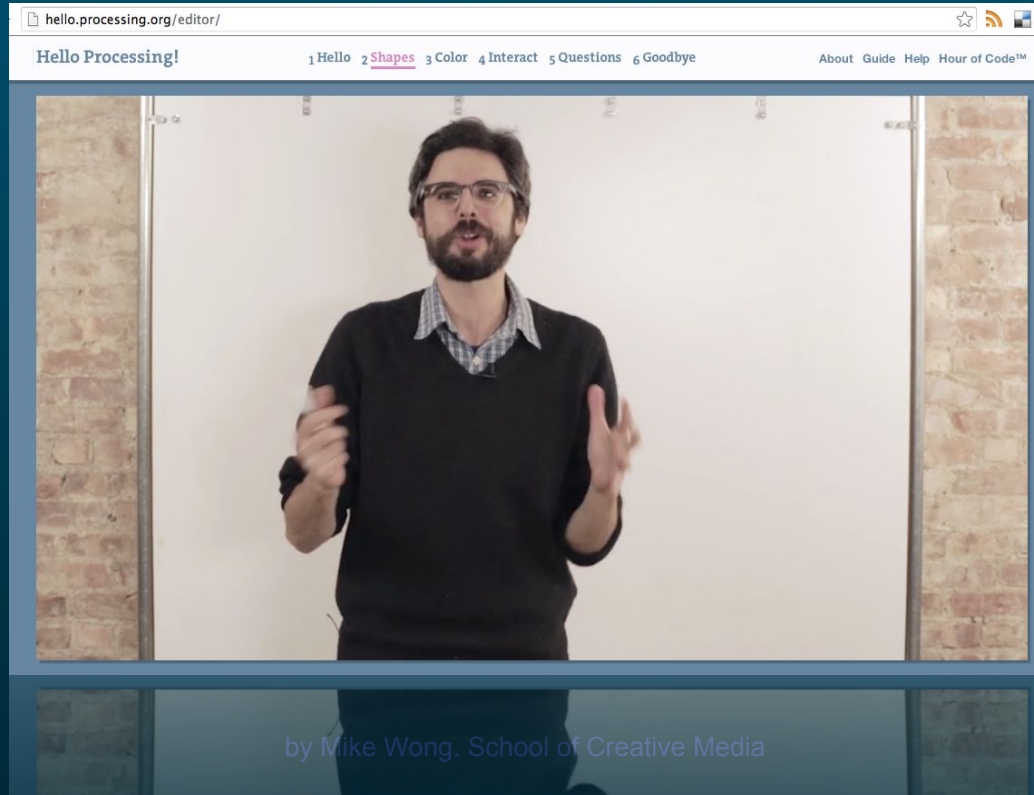


by Mike Wong

by Mike Wong. School of Creative Media

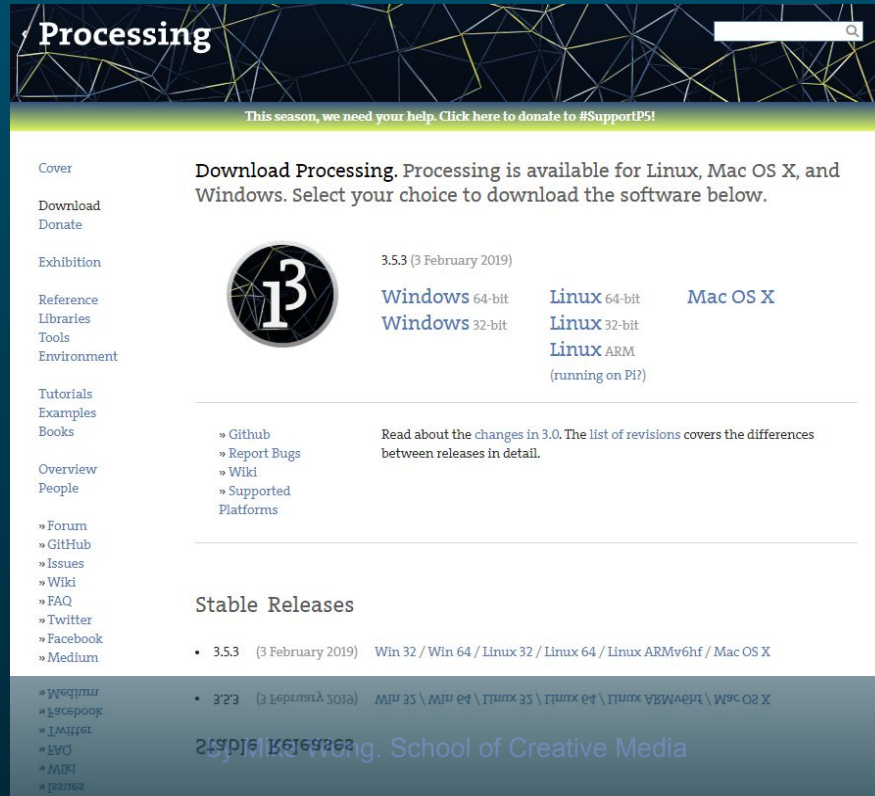
# Hello Processing by Daniel Shiffman

URL: <https://hello.processing.org>



# Download & Install Processing

URL: <https://www.processing.org/download/>




The screenshot shows the Processing.org website. The header features the 'Processing' logo and a search bar. A green banner below the header reads 'This season, we need your help. Click here to donate to #SupportPS!'. The left sidebar contains a list of navigation links: Cover, Download, Donate, Exhibition, Reference, Libraries, Tools, Environment, Tutorials, Examples, Books, Overview, People, Forum, GitHub, Issues, Wiki, FAQ, Twitter, Facebook, and Medium. The main content area is titled 'Download Processing. Processing is available for Linux, Mac OS X, and Windows. Select your choice to download the software below.' It features the Processing logo (a circle with 'P3') and the version '3.5.3 (3 February 2019)'. Below this, there are links for 'Windows 64-bit', 'Windows 32-bit', 'Linux 64-bit', 'Linux 32-bit', 'Linux ARM (running on Pi?)', and 'Mac OS X'. A section titled 'Stable Releases' lists the current version '3.5.3 (3 February 2019)' with supported platforms: 'Win 32 / Win 64 / Linux 32 / Linux 64 / Linux ARMv6hf / Mac OS X'. The footer of the page includes the Creative Commons license 'CC BY-NC-SA' and the text 'Copyright © 2015-2019. School of Creative Media'.

Processing

This season, we need your help. Click here to donate to #SupportPS!

Cover  
Download  
Donate  
Exhibition  
Reference  
Libraries  
Tools  
Environment  
Tutorials  
Examples  
Books  
Overview  
People  
» Forum  
» GitHub  
» Issues  
» Wiki  
» FAQ  
» Twitter  
» Facebook  
» Medium

Download Processing. Processing is available for Linux, Mac OS X, and Windows. Select your choice to download the software below.

 3.5.3 (3 February 2019)

Windows 64-bit  
Windows 32-bit  
Linux 64-bit  
Linux 32-bit  
Linux ARM  
(running on Pi?)  
Mac OS X

» Github  
» Report Bugs  
» Wiki  
» Supported Platforms

Read about the changes in 3.0. The list of revisions covers the differences between releases in detail.

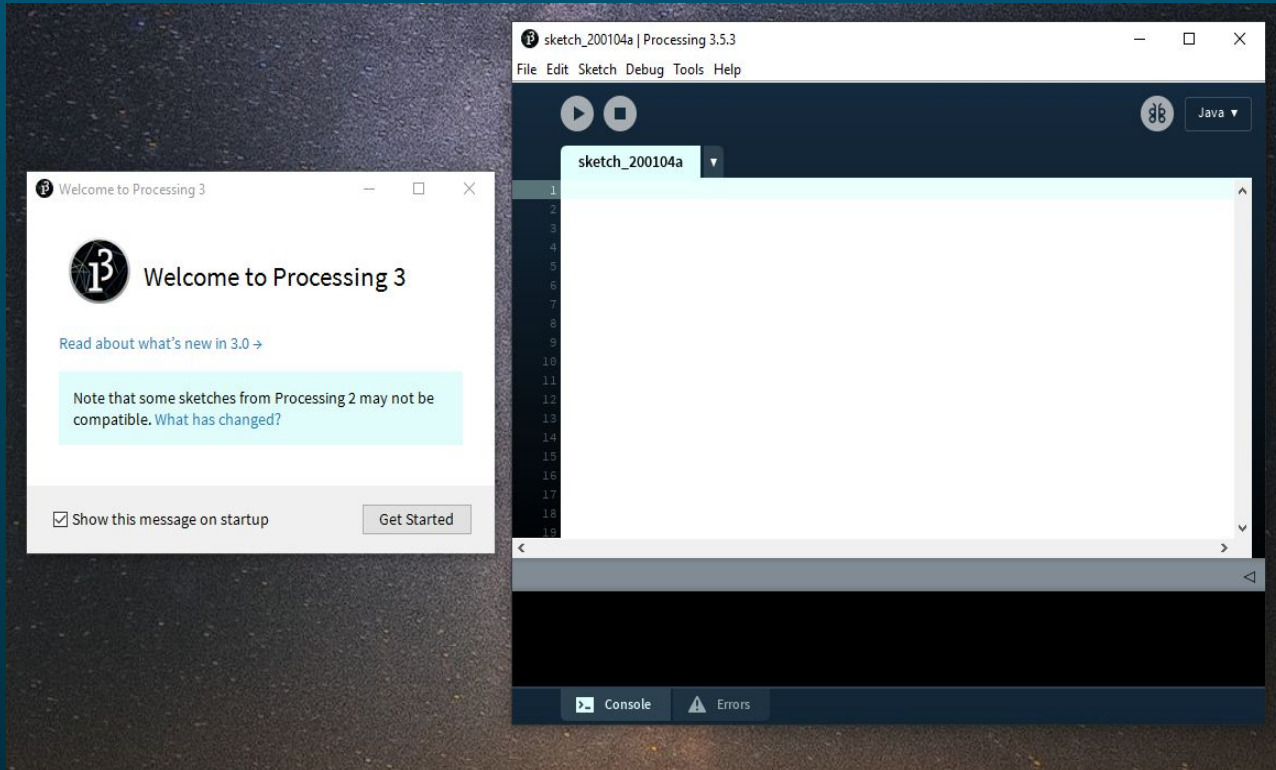
Stable Releases

- 3.5.3 (3 February 2019) Win 32 / Win 64 / Linux 32 / Linux 64 / Linux ARMv6hf / Mac OS X
- 3.5.2 (3 February 2019) Win 32 / Win 64 / Linux 32 / Linux 64 / Linux ARMv6hf / Mac OS X

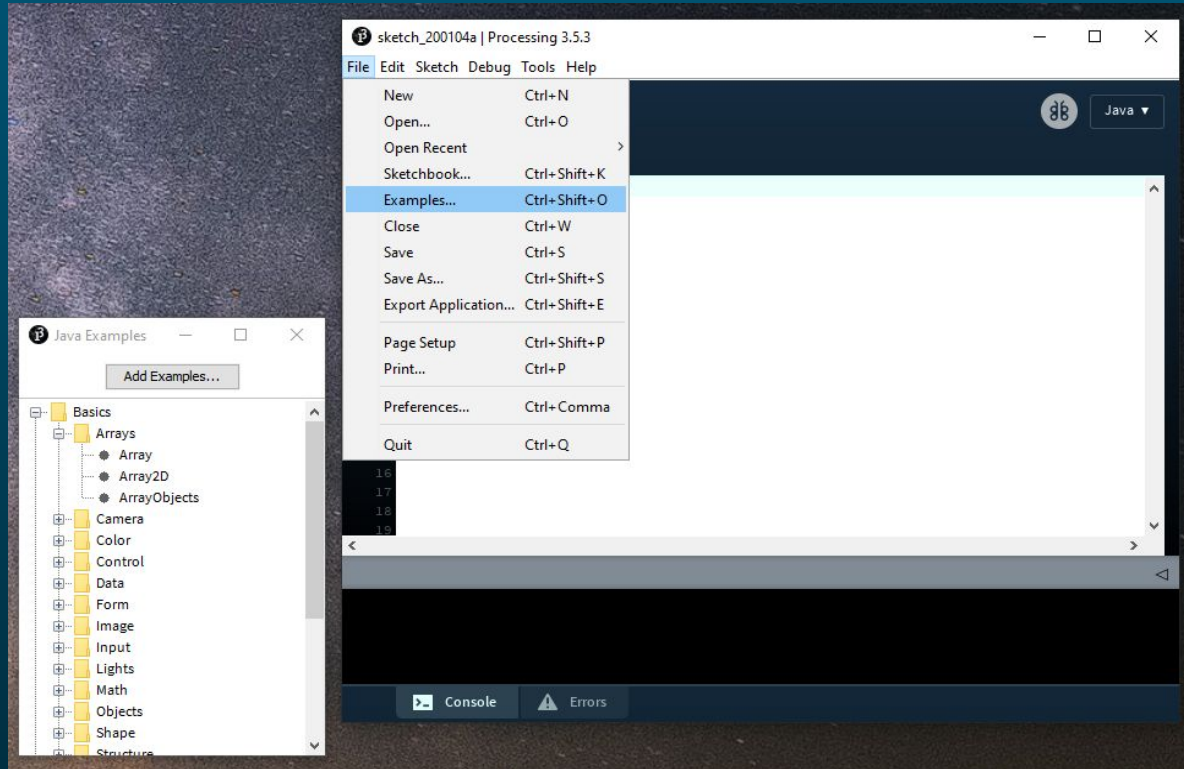
Copyright © 2015-2019. School of Creative Media



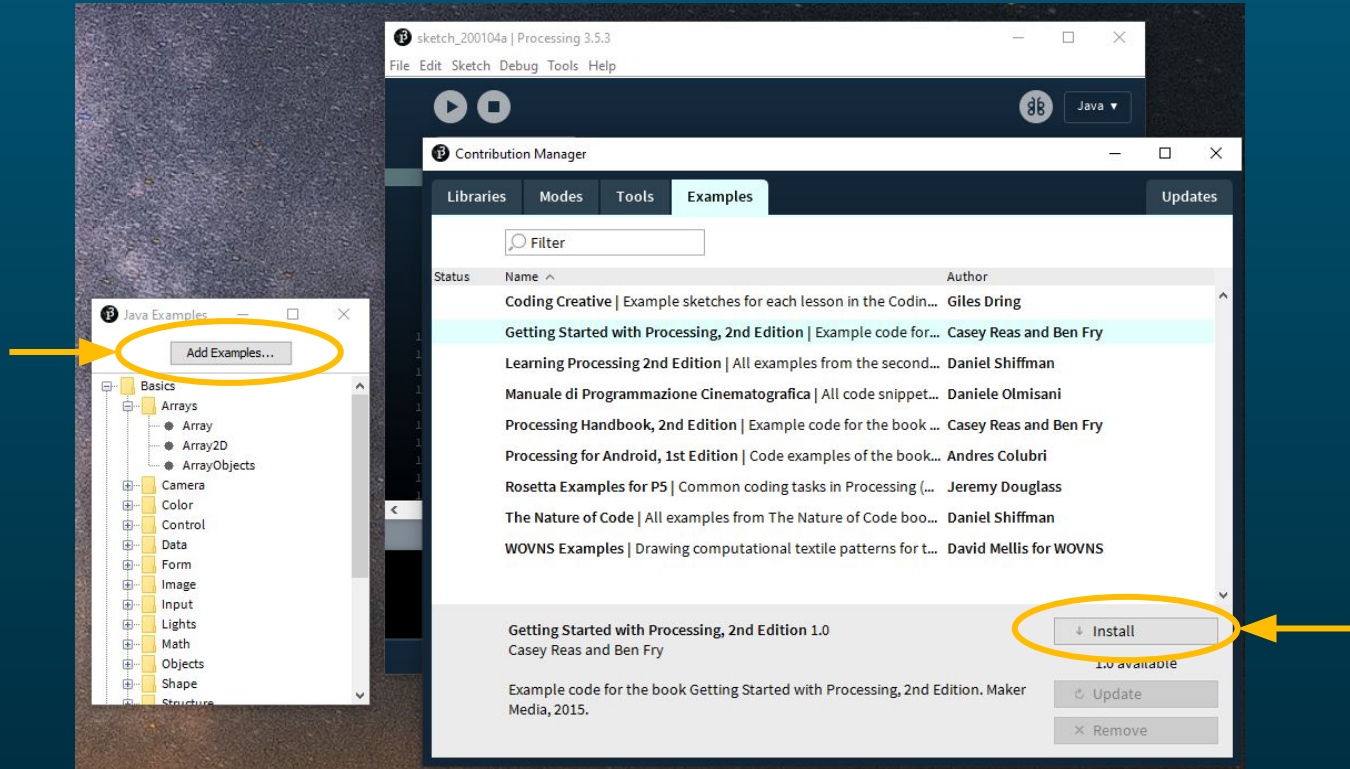
# Launch Processing



# To Learn by Examples



# To Add More Examples



# Examples on Processing.org

Processing

This season, we need your help. Click here to donate to #SupportP5!

Cover

Download

Donate

Exhibition

Reference

Libraries

Tools

Environment

Tutorials

**Examples**

Books

Overview

People

» Forum

» GitHub

» Issues

» Wiki

» FAQ

» Twitter

» Twitch

» FBO

» MIDI

» Jingles

» GitHub

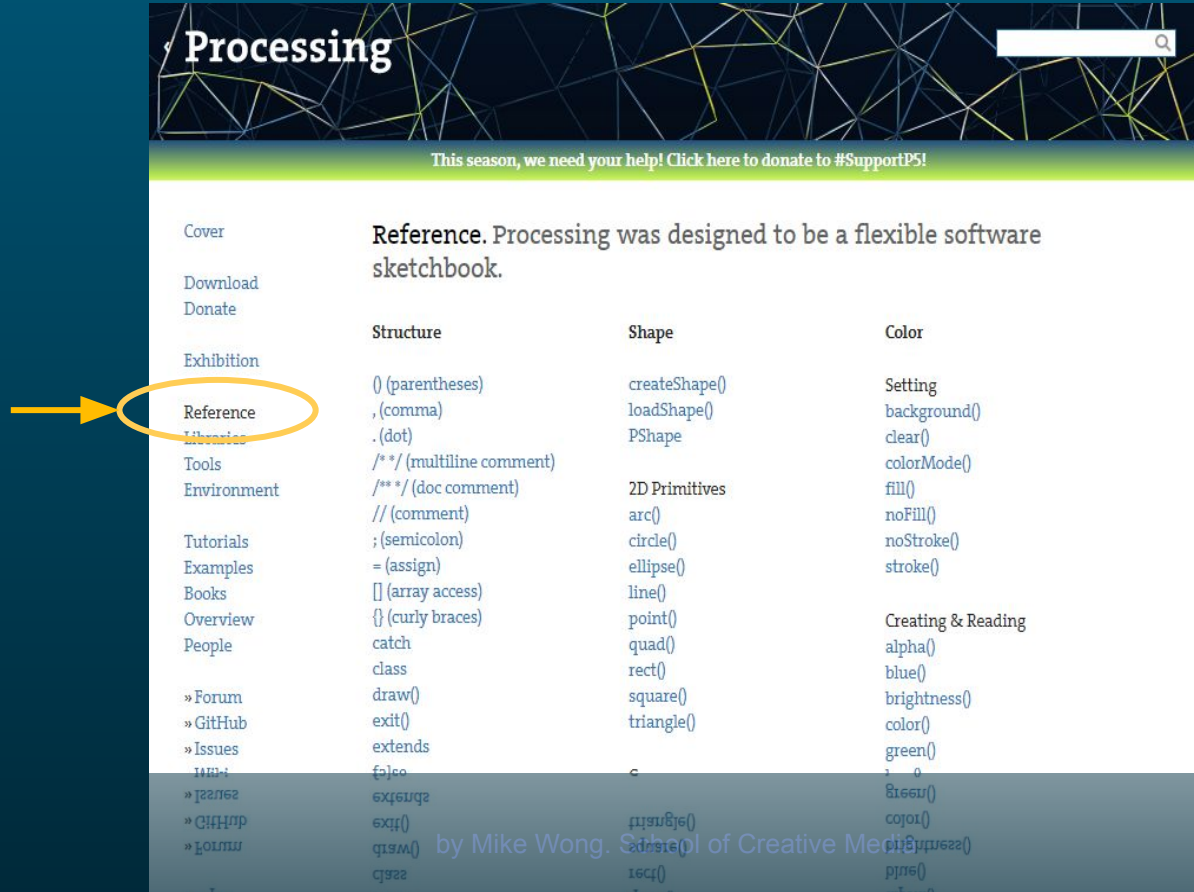
**Examples.** Short, prototypical programs exploring the basics of programming with Processing.

These examples are running online through p5.js using HTML Canvas for rendering. There are many more examples included with the Processing application; please look there if you don't find what you're looking for here.

**Basic Examples.** Programs about form, data, images, color, typography, and more...

Structure	Image	Input
Statements and Comments	Load and Display Image	Mouse 1D
Coordinates	Background Image	Mouse 2D
Width and Height	Transparency	MousePress
Setup and Draw	AlphaMask	Mouse Signals
No Loop	CreateImage	Easing
Loop	Pointillism	Constrain
Redraw	Request Image	Storing Input
Functions		Mouse Functions
Recursion	<b>Color</b>	Keyboard
CreateGraphics	Hue	Keyboard Functions
	Saturation	Milliseconds
		Clock
		Clock
		Milliseconds
		Keyboard Functions
		Keyboard
		Keyboard Functions
		Mouse Functions

# Reference on Processing.org



The screenshot shows the Processing.org website. The header features the 'Processing' logo and a search bar. Below the header is a green banner with the text 'This season, we need your help! Click here to donate to #SupportP5!'. The main content area is divided into a left sidebar and a main body. The sidebar contains links to various sections: Cover, Download, Donate, Exhibition, Reference (highlighted with a yellow arrow), Libraries, Tools, Environment, Tutorials, Examples, Books, Overview, People, » Forum, » GitHub, » Issues, » Jobs, » Gallery, » Chat, and » Forum. The main body is titled 'Reference. Processing was designed to be a flexible software sketchbook.' and contains a table of links organized into four columns: Structure, Shape, Color, and Setting. The 'Structure' column includes links for parentheses, comma, dot, multiline comment, doc comment, comment, semicolon, assign, array access, curly braces, catch, class, draw, exit, extends, file, folder, image, line, point, quad, rect, square, triangle, and vector. The 'Shape' column includes links for createShape, loadShape, PShape, 2D Primitives, arc, circle, ellipse, line, point, quad, rect, square, triangle, and vector. The 'Color' column includes links for Setting, background, clear, colorMode, fill, noFill, noStroke, stroke, Creating & Reading, alpha, blue, brightness, color, green, height, width, and zoom. The 'Setting' column includes links for Setting, background, clear, colorMode, fill, noFill, noStroke, stroke, Creating & Reading, alpha, blue, brightness, color, green, height, width, and zoom.

Processing

This season, we need your help! Click here to donate to #SupportP5!

Cover

Download

Donate

Exhibition

**Reference**

Libraries

Tools

Environment

Tutorials

Examples

Books

Overview

People

» Forum

» GitHub

» Issues

» Jobs

» Gallery

» Chat

» Forum

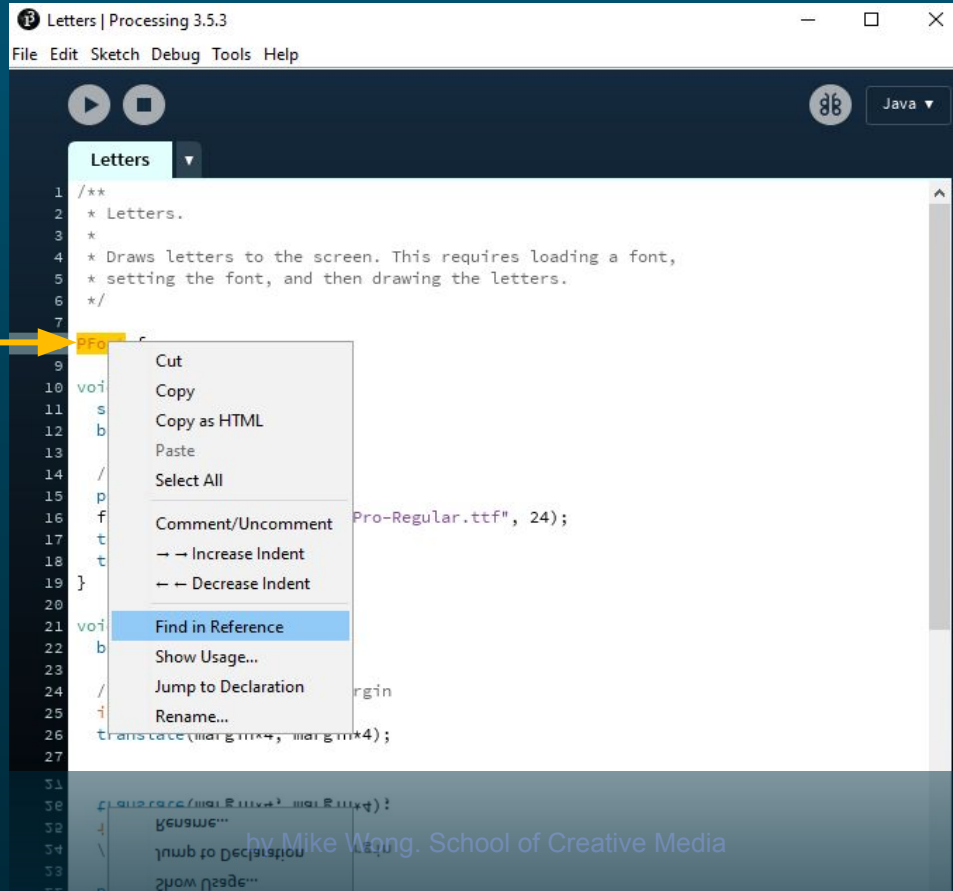
Reference. Processing was designed to be a flexible software sketchbook.

Structure	Shape	Color
<a href="#">() (parentheses)</a>	<a href="#">createShape()</a>	<a href="#">Setting</a>
<a href="#">, (comma)</a>	<a href="#">loadShape()</a>	<a href="#">background()</a>
<a href="#">. (dot)</a>	<a href="#">PShape</a>	<a href="#">clear()</a>
<a href="#">/* */ (multiline comment)</a>	<a href="#">2D Primitives</a>	<a href="#">colorMode()</a>
<a href="#">/** */ (doc comment)</a>	<a href="#">arc()</a>	<a href="#">fill()</a>
<a href="#">// (comment)</a>	<a href="#">circle()</a>	<a href="#">noFill()</a>
<a href="#">; (semicolon)</a>	<a href="#">ellipse()</a>	<a href="#">noStroke()</a>
<a href="#">= (assign)</a>	<a href="#">line()</a>	<a href="#">stroke()</a>
<a href="#">[] (array access)</a>	<a href="#">point()</a>	<a href="#">Creating &amp; Reading</a>
<a href="#">{} (curly braces)</a>	<a href="#">quad()</a>	<a href="#">alpha()</a>
<a href="#">catch</a>	<a href="#">rect()</a>	<a href="#">blue()</a>
<a href="#">class</a>	<a href="#">square()</a>	<a href="#">brightness()</a>
<a href="#">draw()</a>	<a href="#">triangle()</a>	<a href="#">color()</a>
<a href="#">exit()</a>		<a href="#">green()</a>
<a href="#">extends</a>		<a href="#">height()</a>
<a href="#">file</a>		<a href="#">width()</a>
<a href="#">folder</a>		<a href="#">zoom()</a>
<a href="#">image</a>		
<a href="#">line</a>		
<a href="#">point</a>		
<a href="#">quad</a>		
<a href="#">rect</a>		
<a href="#">square</a>		
<a href="#">triangle</a>		
<a href="#">vector</a>		

by Mike Wong. School of Creative Media

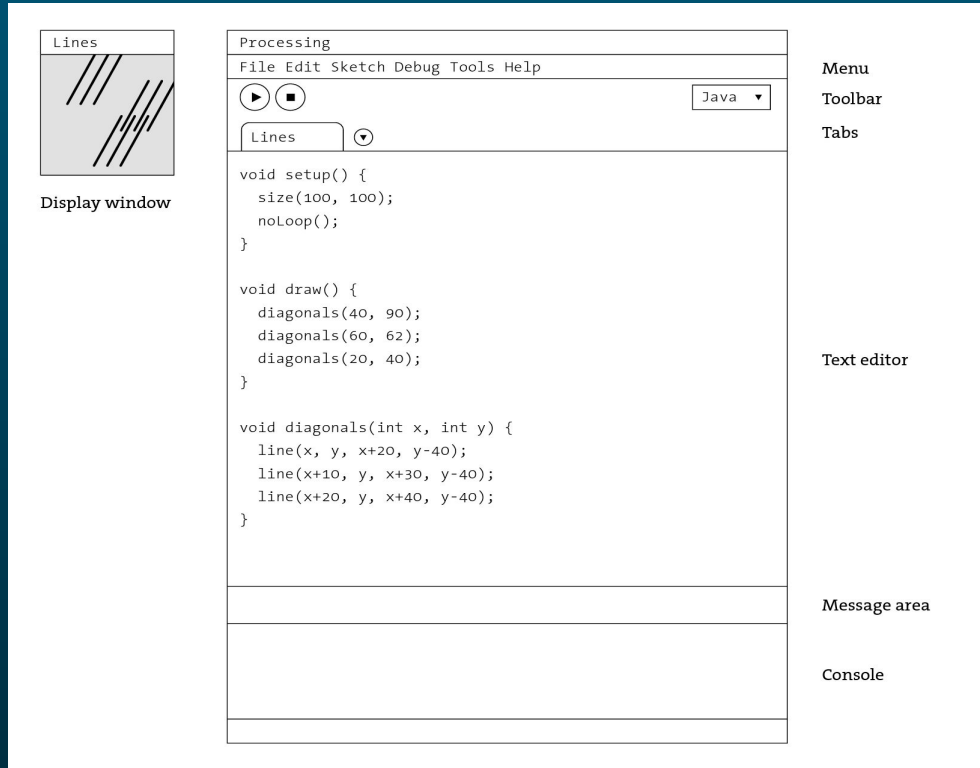
# Getting Help (off-line)

Right click on an item  
to get help from the  
offline reference page

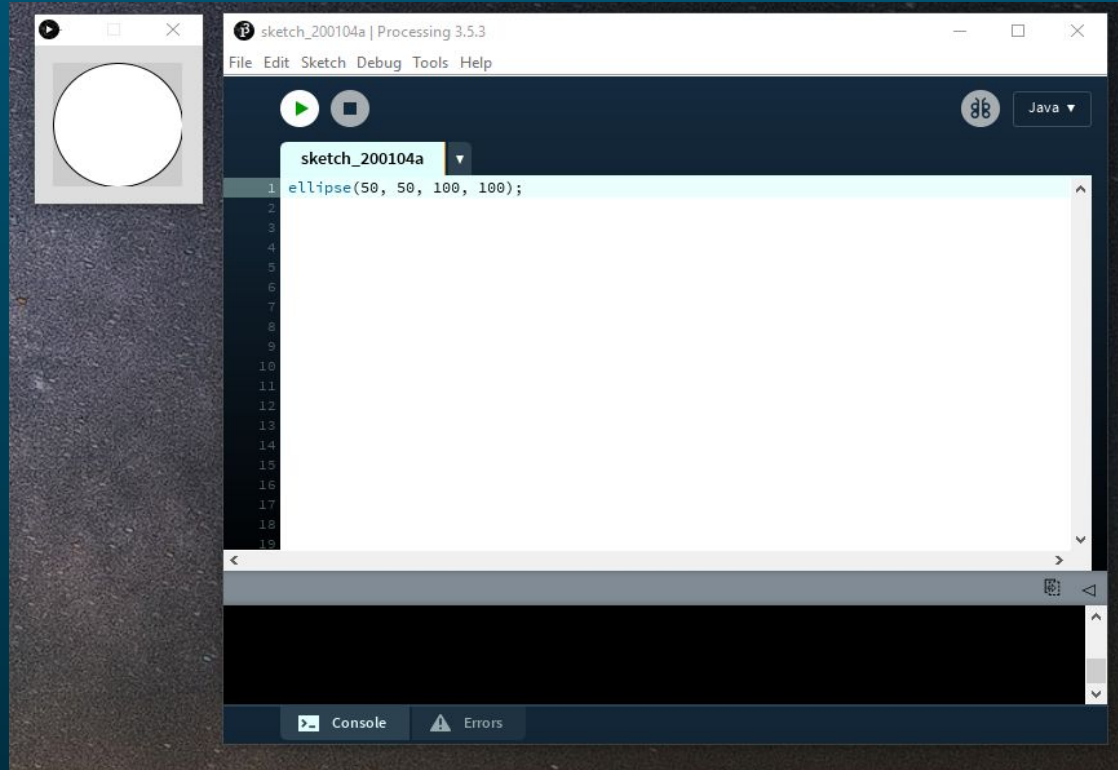




# Processing Development Env. (PDE)

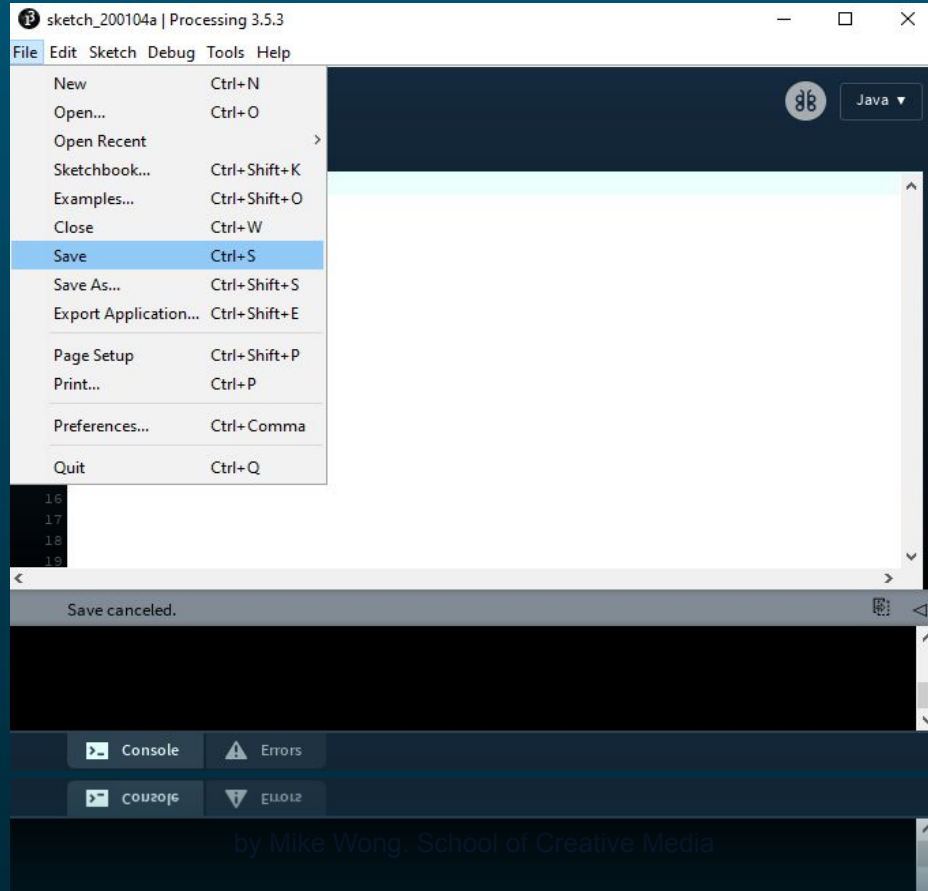


# Our First Processing Sketch

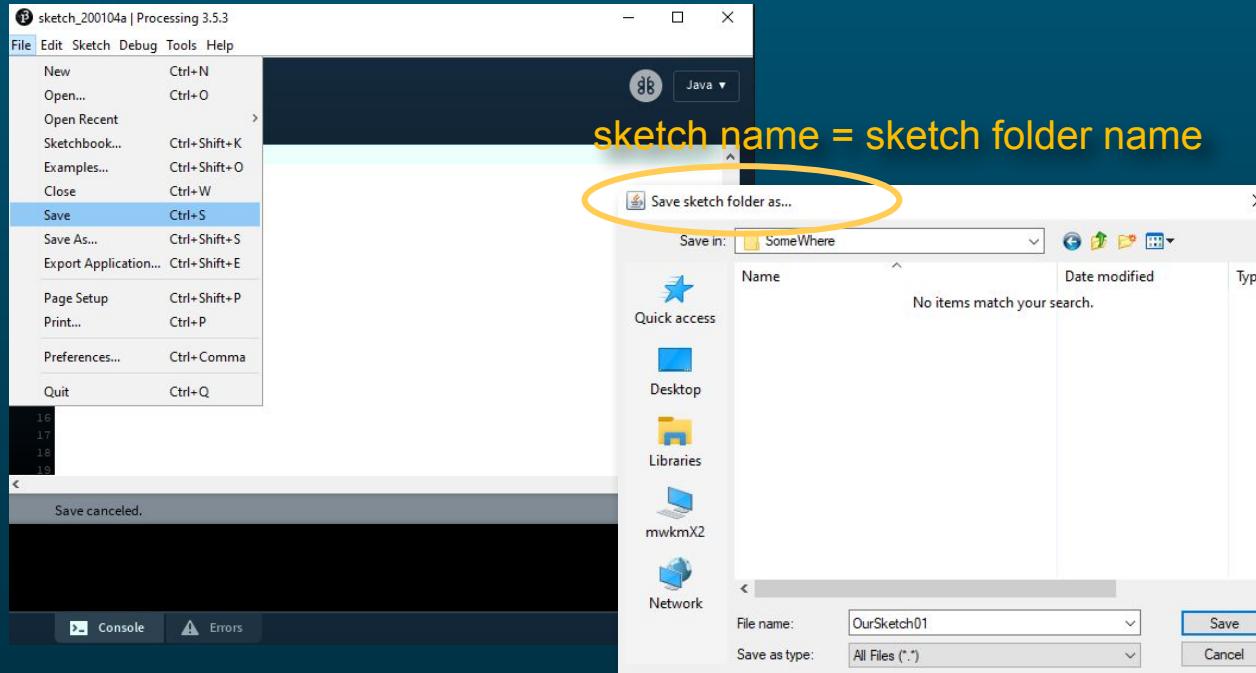




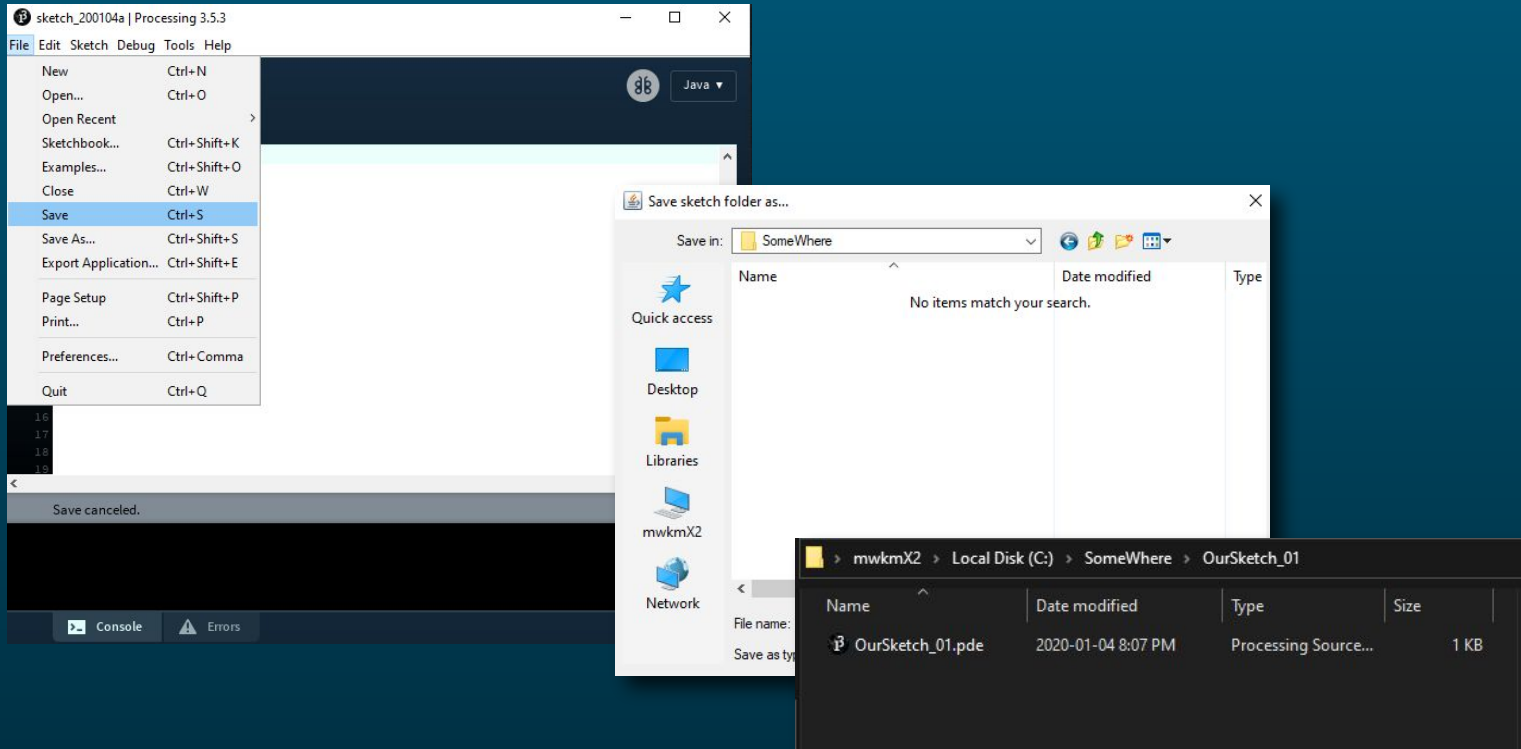
# To save our sketch



# To save our sketch

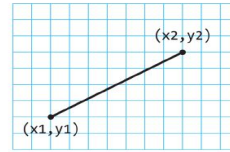


# To save our sketch

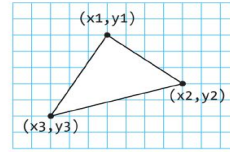


sketch name = sketch folder name = sketch .pde file name

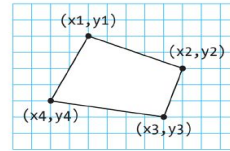
# 2D PRIMITIVE SHAPES AND THEIR COORDINATES



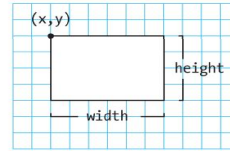
`line(x1, y1, x2, y2)`



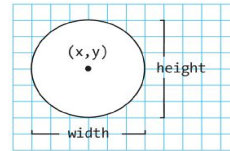
`triangle(x1, y1, x2, y2, x3, y3)`



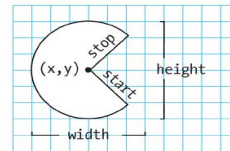
`quad(x1, y1, x2, y2, x3, y3, x4, y4)`



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



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



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

# From p5.js to Processing

## p5.js

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

# From p5.js to Processing

## p5.js

```
let 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);
}
```

# From p5.js to Processing

## p5.js

```
let 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

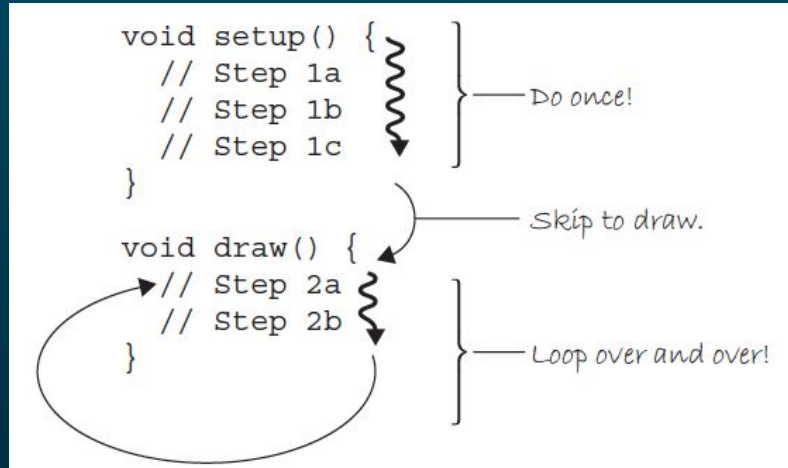
```
int yPos = 0;

void setup() {
  size(200, 400);
}

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

# Structure & 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 when `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



```
background(0);           // Set the black background           0-02
stroke(255);              // Set line value to white
strokeWeight(5);          // Set line width to 5 pixels
smooth();                // Smooth line edges
line(10, 80, 30, 40);    // Left line
line(20, 80, 40, 40);
line(30, 80, 50, 40);    // Middle line
line(40, 80, 60, 40);
line(50, 80, 70, 40);    // Right line
```

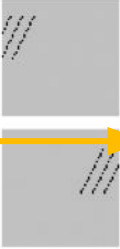


```
int x = 5;    // Set the horizontal position           0-03
int y = 60;   // Set the vertical position
line(x, y, x+20, y-40);    // Line from [5,60] to [25,20]
line(x+10, y, x+30, y-40); // Line from [15,60] to [35,20]
line(x+20, y, x+40, y-40); // Line from [25,60] to [45,20]
line(x+30, y, x+50, y-40); // Line from [35,60] to [55,20]
line(x+40, y, x+60, y-40); // Line from [45,60] to [65,20]
```

# Structured Program

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

`size(w, h);`  
it is always the first line  
in `setup()`



```
int x = 0;    // Set the horizontal position
int y = 55;   // Set the vertical position

void setup() {
  size(100, 100); // Set the window to 100 x 100 pixels
}

void draw() {
  background(204);
  line(x, y, x+20, y-40);    // Left line
  line(x+10, y, x+30, y-40); // Middle line
  line(x+20, y, x+40, y-40); // Right line
  x = x + 1;                // Add 1 to x
  if (x > 100) {             // If x is greater than 100,
    x = -40;                 // assign -40 to x
  }
}
```

0-04

# Example 1

Where to put `background(255);` ?

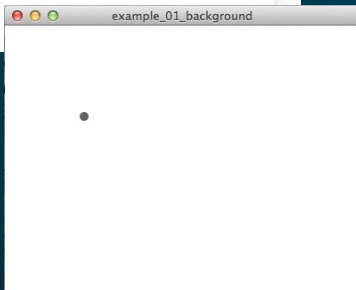
Compare these two programs:

```

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

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

```

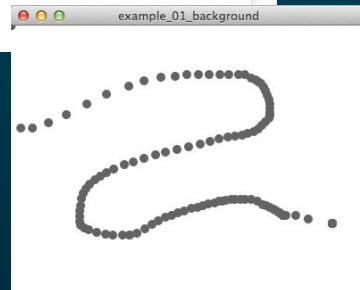


```

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

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

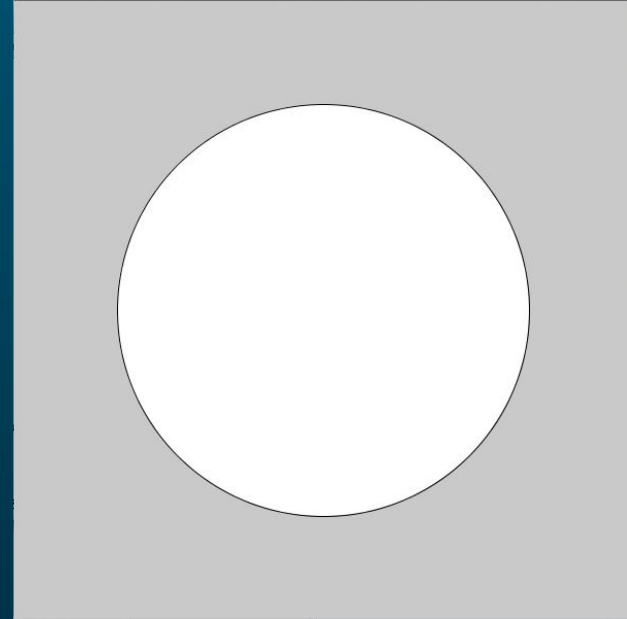
```



# In-class Exercise 1 (Canvas)

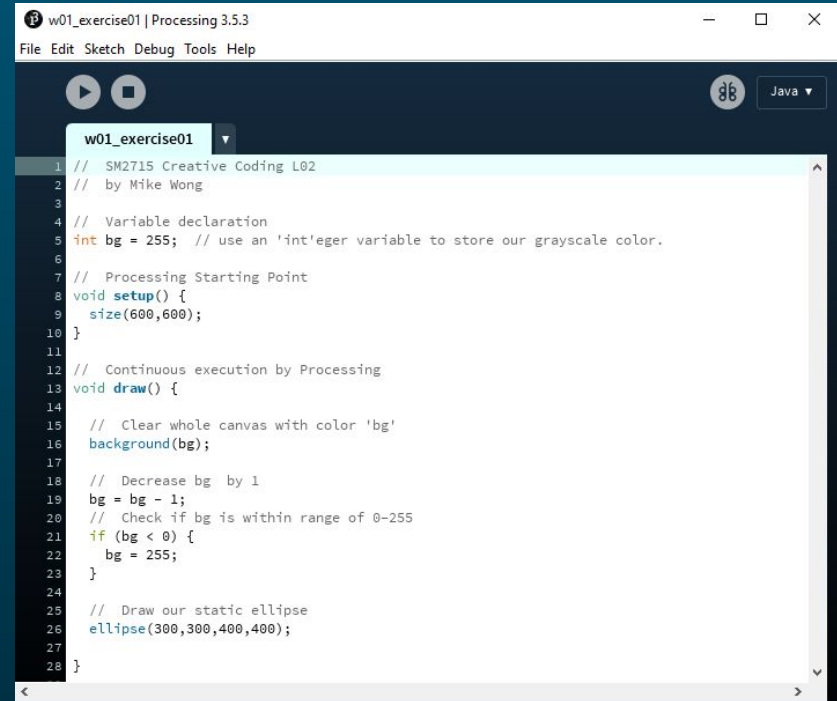


- Write a **structured program**
- Use a display window of dimension 600 x 600
- The background color has to change from white to black, then white to black again continuously
- Draws a circle of radius 200 at the middle of the canvas



# Exercise 1 reference code

- Background color `bg` is updated in every frame, thus `background(bg)` has to be put inside `void draw()`, not in `void setup()`
- `bg = bg - 1;` background color `bg` is decreased by 1 in each iteration of `draw()`, i.e. `255 -> 254 -> 253 ... 2 -> 1 -> 0`
- When background color `bg` reached a value of `-1`, we reset it 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), it has to be drawn each time after the canvas has been cleared with the background color `bg`.



```
w01_exercise01 | Processing 3.5.3
File Edit Sketch Debug Tools Help

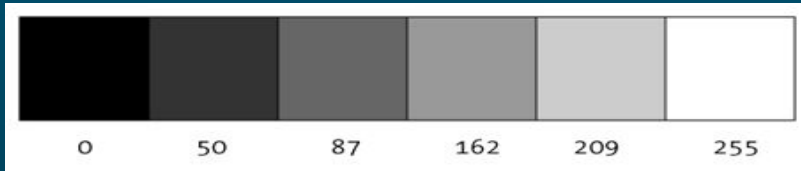
w01_exercise01
1 // SM2715 Creative Coding L02
2 // by Mike Wong
3
4 // Variable declaration
5 int bg = 255; // use an 'int'eger variable to store our grayscale color.
6
7 // Processing Starting Point
8 void setup() {
9   size(600,600);
10 }
11
12 // Continuous execution by Processing
13 void draw() {
14
15   // Clear whole canvas with color 'bg'
16   background(bg);
17
18   // Decrease bg by 1
19   bg = bg - 1;
20   // Check if bg is within range of 0-255
21   if (bg < 0) {
22     bg = 255;
23   }
24
25   // Draw our static ellipse
26   ellipse(300,300,400,400);
27
28 }
```



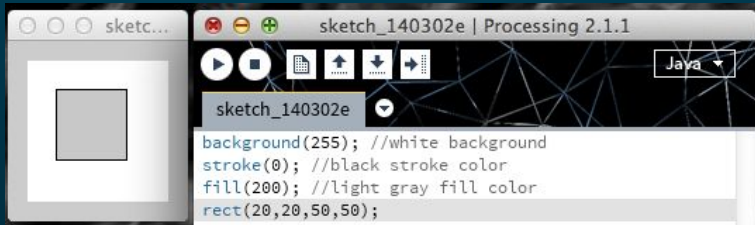
# Color and Data Types In Processing

# Grayscale Color

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



- e.g. `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

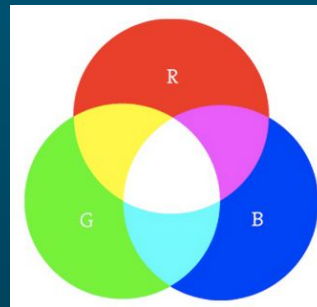
```
alpha()
blue()
brightness()
color()
green()
hue()
lerpColor()
red()
saturation()
```



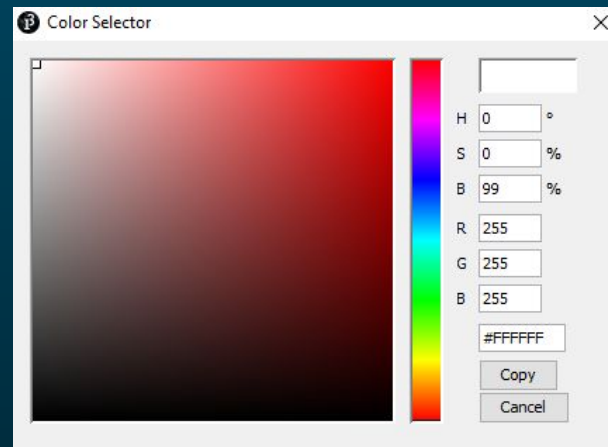
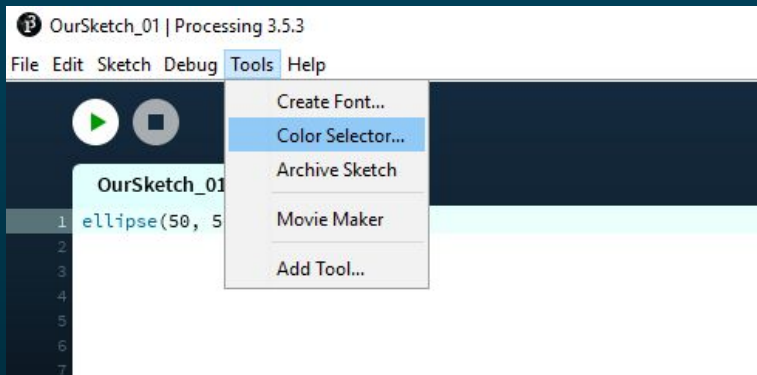
# RGB Color

- 24-bit full color
- 8-bit per-channel of red, green and blue ( 8-bit x 3 )
- Examples:

```
background(255,0,0); // red  
fill(48,139,206);
```

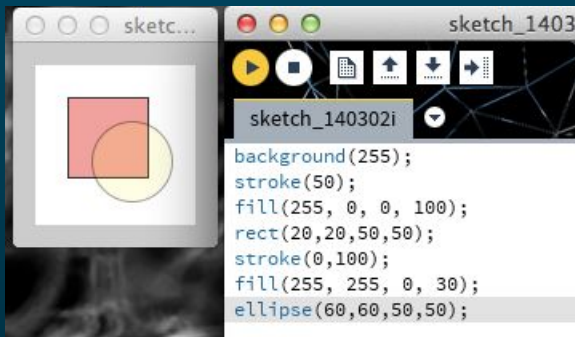


- Color selector tool in Processing PDE



# Transparency (RGBAlpha Value)

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



- Transparency is not applicable to background !

## Example: Alpha transparency

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

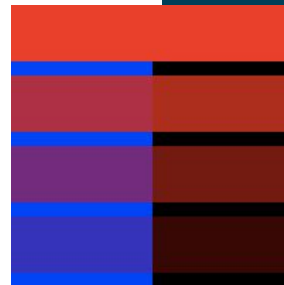
```
// No fourth argument means 100% opacity.  
fill(0,0,255);  
rect(0,0,100,200);
```

```
// 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);
```



# Variable Declaration & Initialization

- Declaration rule: `<data_type> <variable_name> = [initial_value];`
  - `int a = 10;`
- Variable naming rules:
  - Must be one word (no space)
  - Must start with a letter
    - Can include numbers; but not start with a number
  - Can include underscore “\_”
  - Must be unique
    - **CANNOT** use the name of system variables, e.g.,
      - `mouseX, mouseY, width, height, frameCount, key, keyCode, keyPressed, mousePressed, mouseButton` etc.

# Data Types

Name	Value range	Example
<b>boolean</b>	true or false	<code>boolean T = true, F = false;</code>
<b>char</b>	0 to 65535 (16-bit)	<code>char myCh1='A', myCh2='\$';</code>
<b>byte</b>	-128 to 127 (8-bit)	<code>byte b = -128;</code>
<b>int</b>	-2147483648 to 2147483647 (32-bit integer)	<code>int number = 800, temp = -1;</code>
<b>float</b>	3.40282347E+38 to -3.40282347E+38 (32 bits)	<code>float b = -2.984;</code>
<b>color</b>	16,777,216 colors RGBA 32-bit (24-bit + 8-bit)	<code>color c1 = color(204,153,0), c2 = #FFCC00;</code>

# Data Types

```
int x;           // Declare a variable named x of data type int  
float y;         // Declare a variable named y of data type float  
boolean b;       // Declare a variable named b of data type boolean  
x = 50;         // the variable x is assigned a value of 50  
y = 12.6;       // the variable y is assigned a value of 12.6  
b = true;       // the variable b is assigned a value of true
```

# Data Type Conversion\*

## 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"
```

# Data Type Conversion\*

## Data

Primitive

boolean

byte

char

color

double

float

int

long

## Conversion

binary()

boolean()

byte()

char()

float()

hex()

int()

str()

unbinary()

unhex()

```
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'
```

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

128	Ç	144	È	160	á	176	☒	192	Ł	208	Ł	224	α	240	≡
129	ù	145	é	161	í	177	☓	193	ł	209	ŧ	225	β	241	±
130	é	146	Ê	162	ó	178	■	194	ŧ	210	ŧ	226	Γ	242	≥
131	â	147	ê	163	û	179		195	ı	211	ı	227	π	243	≤
132	ä	148	ë	164	ü	180	ı	196	—	212	ı	228	Σ	244	ƒ
133	å	149	ö	165	ÿ	181	ı	197	ı	213	ı	229	σ	245	ı
134	æ	150	û	166	*	182	ı	198	ı	214	ı	230	μ	246	ı
135	ç	151	ü	167	°	183	ı	199	ı	215	ı	231	τ	247	ı
136	è	152	ÿ	168	¿	184	ı	200	ı	216	ı	232	Φ	248	°
137	é	153	Ö	169	ı	185	ı	201	ı	217	ı	233	Θ	249	ı
138	è	154	Ü	170	ı	186	ı	202	ı	218	ı	234	Ω	250	ı
139	ı	155	ı	171	½	187	ı	203	ı	219	ı	235	δ	251	√
140	ı	156	ı	172	¾	188	ı	204	ı	220	ı	236	∞	252	ı
141	ı	157	ı	173	ı	189	ı	205	ı	221	ı	237	φ	253	ı
142	Ä	158	ı	174	ı	190	ı	206	ı	222	ı	238	ε	254	ı
143	Å	159	ı	175	ı	191	ı	207	ı	223	ı	239	ı	255	ı

Source: [www.LookupTables.com](http://www.LookupTables.com)



# Example 2



```
w01_code_02 | Processing 3.5.3
File Edit Sketch Debug Tools Help

w01_code_02
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

Console Errors
```