



# introduction to media computing

## week 03

# Today's topics (week 03)



- **operators & conditionals**
  - review
  - the modulo operator `' % '`
- **logical operators**
- **coding style**
- **loops I: `while()` loop**

# Today's topics (week 03)



- **operators & conditionals**
  - review
  - the modulo operator `' % '`
- **logical operators**
- **coding style**
- **loops I: `while()` loop**



- **p5.js online editor**
- **drawing text**
- **mouse click**

# Resources for review



<https://canvas.cityu.edu.hk/courses/31242>

SM1103A Introduction to Media Compt'ng

This course will teach fundamental programming concepts via creative exercises and small projects. Toward this end, students will explore the concepts of variables, sequential programming, loops, conditionals, arrays, functions with the programming of multimedia, such as image, audio, video, animation, and interactivity.

**General Information**

- [Instructors & Sessions](#)
- [Evaluation](#)
- [Readings](#)
- [Freedom](#)

**Lecture Notes**

- [Week 01](#)
- [Week 02](#)

**Assignments**

**Programming Topics**

- [Variables](#)
- [Operators](#)
- [Conditionals](#)
- [Loops](#)
- [Arrays](#)
- [Functions](#)
- [Program Flow](#)

**Resources**

- [Useful Snippets](#)
- [Tips & Tricks](#)
- [The p5.js website](#)
- [The p5.js reference](#)
- [The p5.js online editor](#)
- [The Atom editor website](#)

## Programming Topics

- [Variables](#)
- [Operators](#)
- [Conditionals](#)
- [Loops](#)
- [Arrays](#)
- [Functions](#)
- [Program Flow](#)

# Review: math. operators

```
let a = 10;  
let b = 6;  
let result;
```

```
result = a + b;
```

**addition**

```
result = 16
```

```
result = a - b;
```

**subtraction**

```
result = 4
```

```
result = a * b;
```

**multiplication**

```
result = 60
```

```
result = a / b;
```

**division**

```
result = 1.6667
```

```
result = a % b;
```

**modulo**

```
result = 4*
```

**\*Remainder of integer division**

# Review: assignment operators

```
let r;  
r = 10;  
r = r + 1;  
r += 2;  
r -= 2;  
r *= 2;  
r /= 2;  
r %= 2;
```

**assignment**

`r = 11`    `(10 + 1)`

**add. assignment**

`r = 13`    `(11 + 2)`

**sub. assignment**

`r = 11`    `(13 - 2)`

**mul. assignment**

`r = 22`    `(11 * 2)`

**div. assignment**

`r = 11`    `(22 / 2)`

**mod. assignment**

`r = 1`    `(11 % 2)` \*

\*Remainder of integer division

## Review: other operators

```
let x = 200;  
x++;
```

'++' : increment

x++ is equivalent to  $x = x + 1$

```
let x = 200;  
x--;
```

'--' : decrement

x-- is equivalent to  $x = x - 1$

## Review: if else

```
if (x == 200) {  
    // Do something  
}  
else if (x < 200) {  
    // Do something  
}  
else {  
    // Do something else  
}
```

Only ONE block of  
code will be executed.



# Review: relational operators

js

```
if (x >= 200) {  
    // Do something  
}  
else {  
    // Do something else  
}
```

operators	meaning
>	larger than
<	smaller than
>=	larger or equal to
<=	small or equal to
!=	not equal to
==	equal to



# Modulo Operator `' % '`

## Modulo Operator `' % '`

- modulo operator `' % '` computes the remainder of an integer division. Example: `5 % 2` returns `1`.
- This operator is particularly useful for some simple looping operation.

# Modulo Operator '%'

try

p5\*

JS

```
let num = 0;

function setup(){
  createCanvas(200,200);
  fill(128);
}

function draw() {
  // Loop thru 0 - 256 for color
  let brightness = num % 256;
  background(brightness);
}
```

Resources

Result



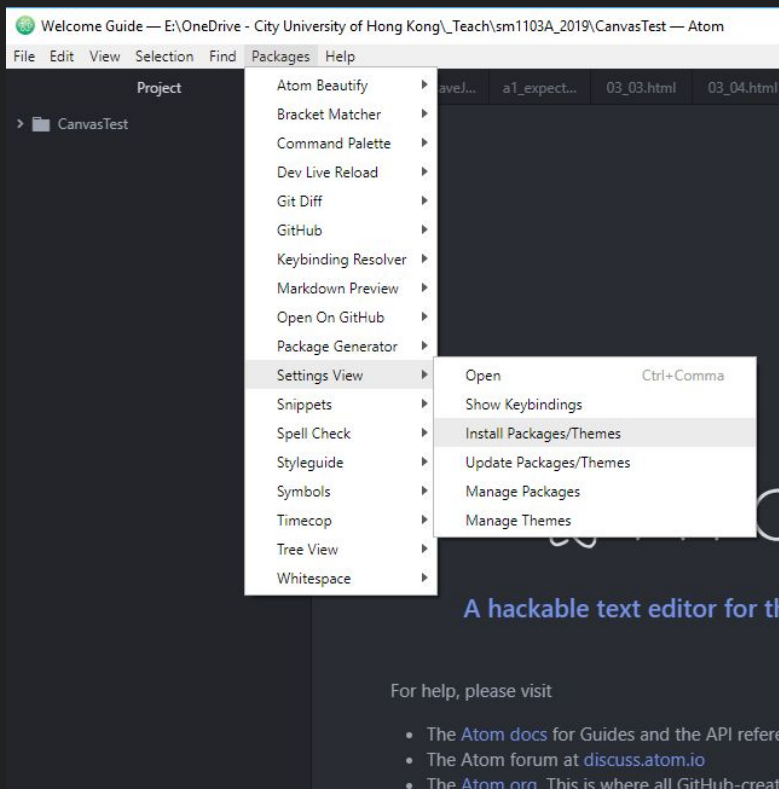
EDIT ON CODEPEN

1x 0.5x 0.25x Rerun



# Improved coding experience

# html-preview in Atom 1/3

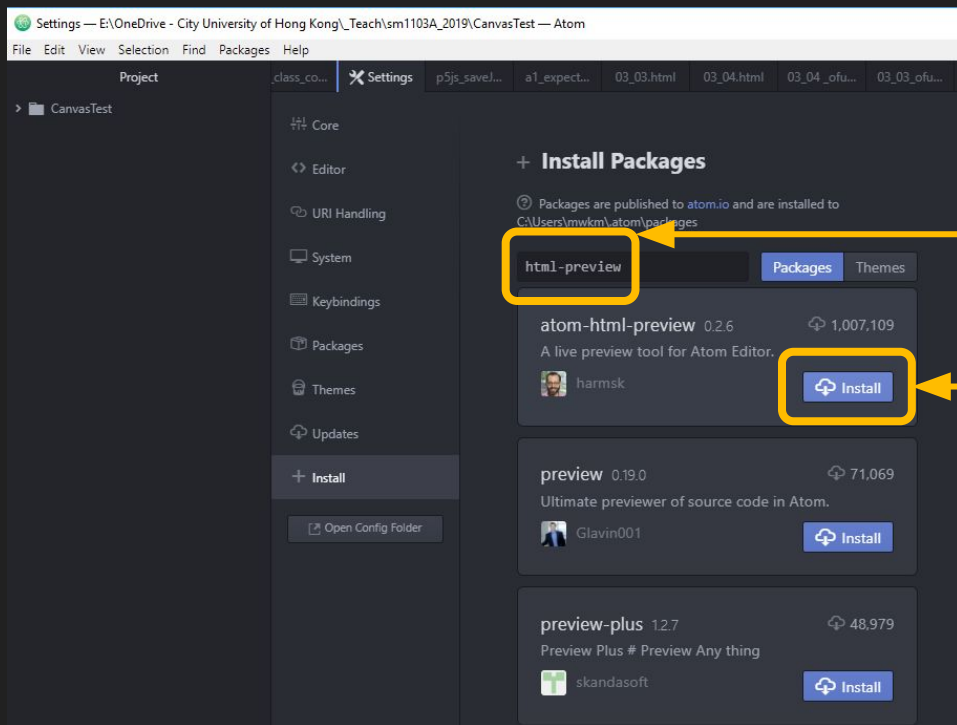


Atom has a **html-preview** package which allows **in-Atom HTML** preview.

On Atom's menu bar, choose

**Packages** → **Settings View** → **Install Packages/Themes**.

# html-preview in Atom 2/3



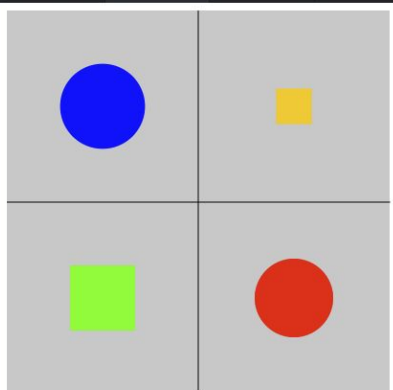
1. Type **html-preview** in the package search entry

2. Install the found **atom-html-preview** package

# html-preview in Atom 3/3

```

Welcome | 02_02.html | variables.html | operators.html | mw_01a.html | copy_button.css | copy_button.js | Welcome Guide | 02_02.html Prev... | operators.html | conditionals.htm
1 <html>
2 <head>
3   <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.9.0/p5.js"></script>
4   <script>
5
6     let w;
7     let h;
8     let x0, x1, y0, y1;
9     let _x0, _x1, _y0, _y1;
10    let offset = 500;
11    let rA = 50;
12    let rB = 50;
13    let rC = 50;
14    let rD = 50;
15
16    function setup() {
17      createCanvas(900, 400);
18      noStroke();
19      w = 400;
20      h = height;
21      x0 = w/4;
22      x1 = 3 * x0;
23      y0 = h/4;
24      y1 = 3 * y0;
25      _x0 = x0 + offset;
26      _x1 = x1 + offset;
27    }
28
29    function draw() {
30
```



Once atom-html-preview is installed. For any editor window in Atom with a .html file (The html file must have been named and saved).

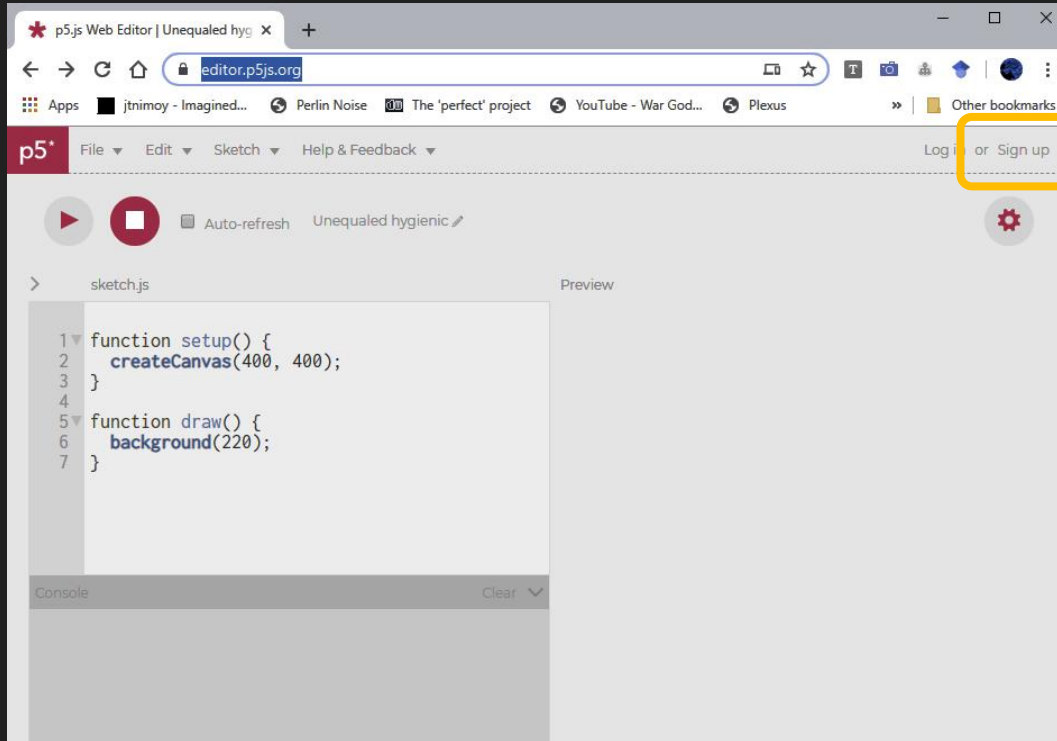
You may invoke the preview by pressing:

**CONTROL-SHIFT-H**



# p5.js online editor

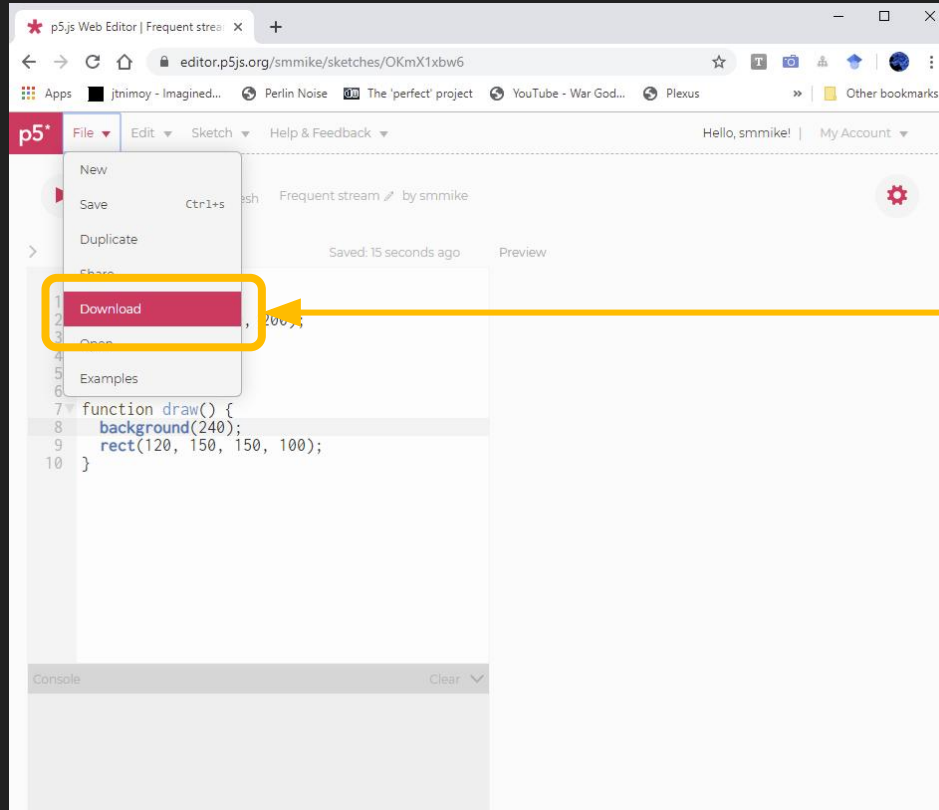
p5\*



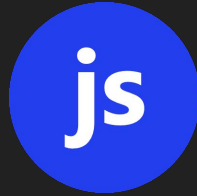
Please follow the URL and create an account so you can save your sketches, then do the exercise on the next slide.

# p5.js online editor

p5\*



Once you have saved your sketch, you may download the sketch as a zipped archive.



# Logical Operators

# Logical operators

- logical operator helps us to compose more flexible 'conditions' for various JavaScript conditionals.
- All 'conditions' evaluation in JavaScript returns a logical (boolean) value 'true' or 'false'.

```
if (x == 200) {  
    // Do something  
}
```

Simple SINGLE condition, what if we want to combine two or more conditions ?

# Logical operators

operators	meaning
<code>  </code>	Logical OR
<code>&amp;&amp;</code>	Logical AND
<code>!</code>	Logical NOT

```
if (x == 0 || x == 200) {  
    // Do something  
}
```

This block will run only if  
X equals to 0  
OR  
X equals to 200

# Logical operators

operators	meaning
	Logical OR
&&	Logical AND
!	Logical NOT

```
if (x > 1 && x != 200) {  
    // Do something  
}
```

This block will run only if  
X is larger than 1  
AND  
X is not equal to 200

# Logical operators

operators	meaning
	Logical OR
&&	Logical AND
!	Logical NOT

```
if (! (x > 1)) {  
    // Do something  
}
```

The NOT operator always inverses the result of the condition `(x > 1)`. So the block runs when `x` does not fulfill `(x > 1)`, i.e. the block runs when `x` is NOT larger than 1.

# Truth table

## Logical OR '||'

A	B	(A    B)
true	true	true
true	false	true
false	true	true
false	false	false

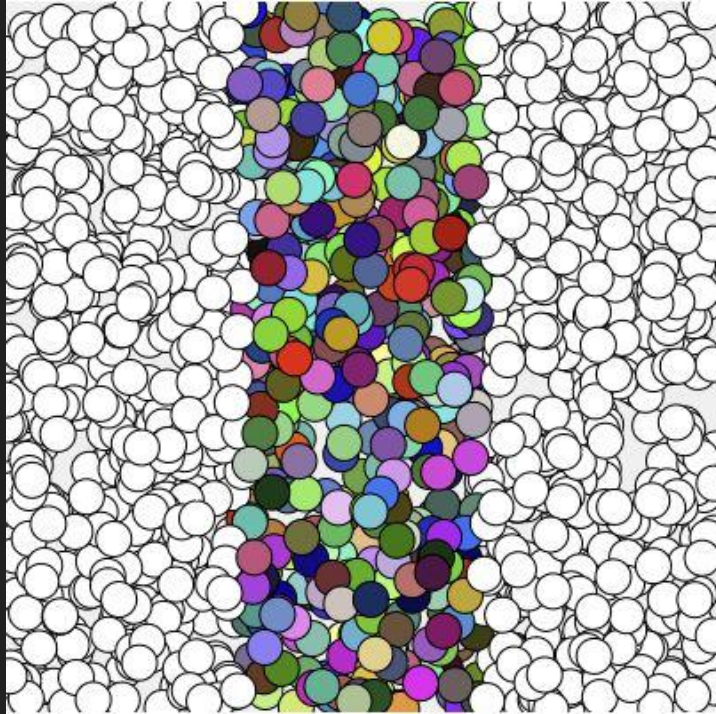
## Logical AND '&&'

A	B	(A && B)
true	true	true
true	false	false
false	true	false
false	false	false

## Logical NOT '!'

A	!(A)
true	false
false	true





1. Fill the canvas (400 x 400) with circles of size 20, each with a random position.
2. Divide the screen into 3 regions as shown in the figure. Circles in the middle region are randomly colored, and the rest are in white.
3. Use only one conditional. Hint: Use a 'logical operator'.



# Coding Style

# Coding Style

js

p5\*

```
let x=0;

function setup() {
  createCanvas(400, 400);
  background(220);
}

function draw() {
  if (x % 2 == 1) {
    fill(150);
  } else {
    fill(0);
  }
  ellipse(50 * x, 0, 50, 200);
  x++;
}
```

It is **VERY IMPORTANT** to use proper indentation and spacing in your code.

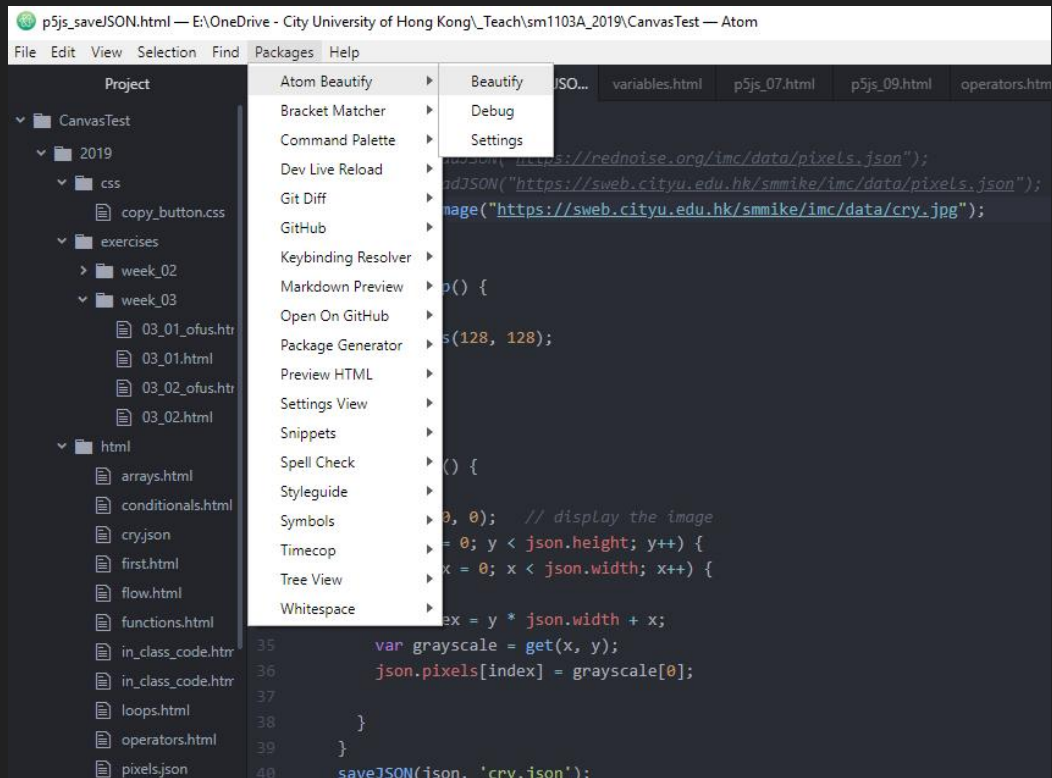
## WHY ?

- Easier for you to read
- Easier for others to read
- Help you to spot mistakes
- It shows you understand *the craft of coding*
-

# Coding Style

js

p5\*

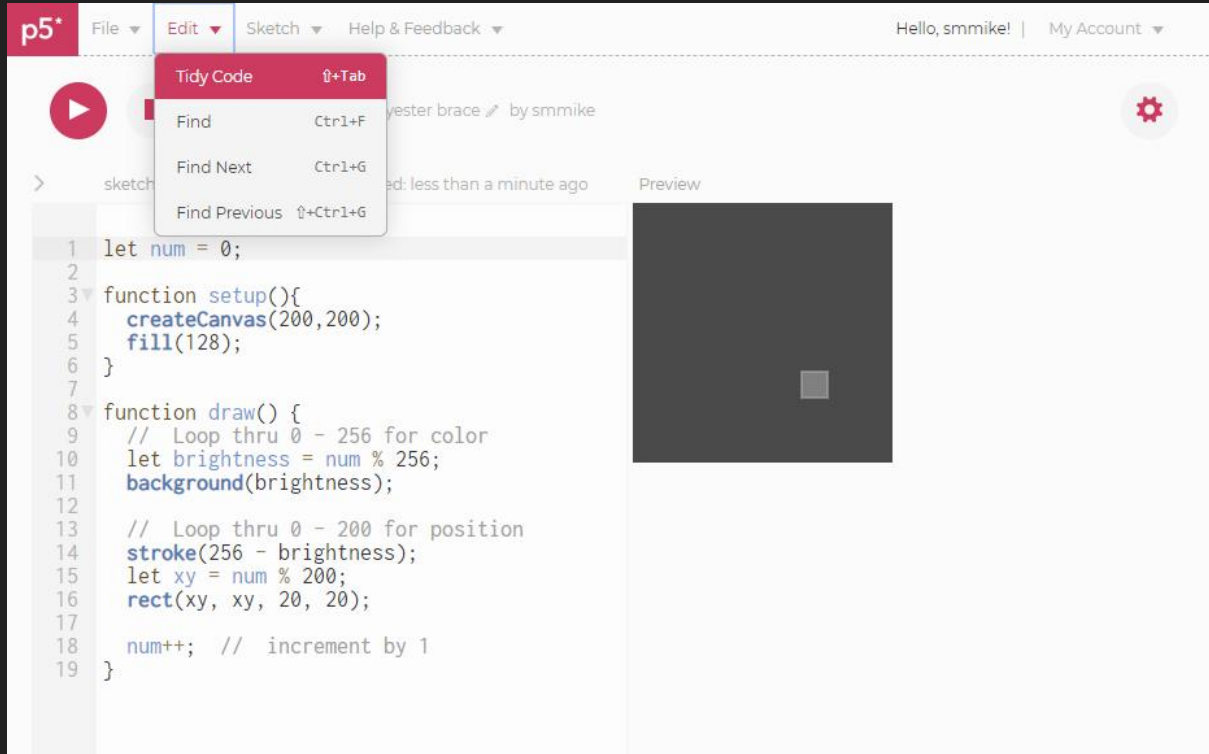


Most modern code editor like **Atom** often has code 'beautify' or 'prettify' package which helps with code formatting and syntax highlighting.

# Coding Style

js

p5\*



**p5.js editor also offers convenient code formatting functions.**

Properly formatted code is  
required in all assignments.  
Poorly formatted code will  
lead to point deduction.





## Loops 1: The 'while()' loop



## Loops 1: the `while()` loop

- A 'Loop' allows a block of code to be executed repeatedly (a.k.a. iteration).
- A `while()` loop repeats a block of code to as long as certain condition is fulfilled in each iteration.

## Loops 1: the `while()` loop

The simple `while()` loop below repeats drawing a rectangle as long as the value of variable `x` is less than 3.

```
let x = 0;
while ( x < 3 ) {
  rect( x, 0, 10, 10);
  x = x + 1;
}
```

## Loops 1: the while() loop

```
while ( <condition> ) {  
    // code run multiple times;  
}
```

**while()** loop repeats the block of code as long as the condition is being fulfilled, i.e. the condition is verified in each iteration.

```
if ( <condition> ) {  
    // code run ONCE only;  
}
```

**if()** runs the block of code if the condition is fulfilled and it only runs once.

## Loops 1: the `while()` loop

It is **VERY IMPORTANT** to note that the block of code inside a `while()` loop should always do something to fail the condition by design.

```
let x = 0;
while ( x < 3 ) {
  rect( x, 0, 10, 10);
  // x = x + 1;
}
```

**INFINITE LOOP**  
It never stops !!

## Loops 1: the while() loop

```
while (x < 10) {  
    // some code here;  
}
```

The **<condition>** is written in similar fashion as in if-else and other conditionals.

```
while (x != 10 && y == 1) {  
    // some code here;  
}
```

## Loops 1: the `while()` loop

try

p5\*

Try the following code inside `function setup()`  
of a sketch using the `p5.js` editor

```
let x = 5;
while (x > 0) {
  text(x, x * 12, 20);
  x--;
}
```

```
let y = 0;
while (y < 5) {
  text(y, 20, y * 12);
  y++;
}
```



`text()` drawing and `mouseIsPressed`

## text() drawing

- `text("hi",x,y)` draws numbers or text at a given coordinate on the canvas.
- `textSize(size)` defines the size of text to be drawn.
- Colors of the text to be drawn, like most shapes are controlled by `fill()` and `stroke()`.



## text () drawing

- `loadFont (URL)` loads a font file from the given URL or a local file. `loadFont (URL)` should be called within the `function preload ()` block of `p5.js`.
- `textFont (font)` tells P5.js what font to be used for the text to be drawn by `text ()`

## Built-in variable `mouseIsPressed`

`mouseIsPressed` is a `p5.js` built-in boolean variable (stores `true` or `false`) that you may use for detecting if the mouse is being pressed. Use it with a `if() {}` statement and execute your desired instructions when the mouse is pressed. (use it in `function draw()` block)

```
if (mouseIsPressed) {  
    // Draw or Do something  
}
```

# text() and mouseIsPressed

try

p5\*

JS


```
let font;

function preload() {
  font = loadFont("https://artixels.github.io/imc/Modak-Regular.ttf");
}

function setup() {
  createCanvas(200, 200);
  background(180);
  textFont(font);
  textSize(40);
}
```

Resources

Result



EDIT ON CODEPEN

1x 0.5x 0.25x

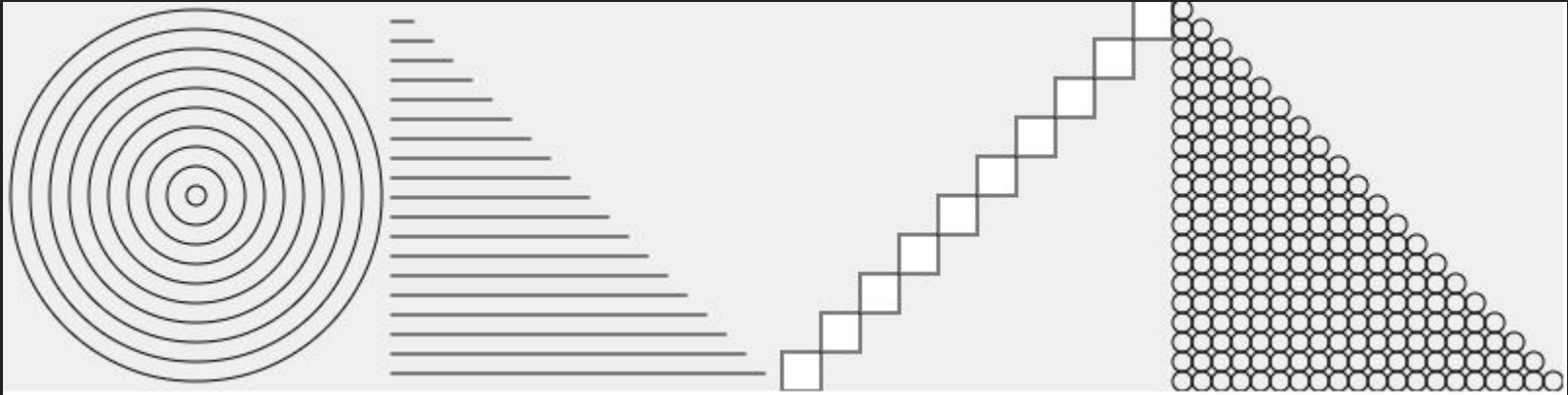
Rerun



1. Add your code to **function** `setup()`, not **function** `draw()`. Use a **while()** loop to draw 10 gray squares across a 500 x 200 canvas, exactly as shown.
2. Add the numbers as shown to your squares.

# In-class exercise 4

Try your best to reproduce the following 4 drawings as close as possible using `while()` loop. The left 3 should only take one single `while()` loop, and the right-most one may take more than one loop.



**Assignment #1** has been released on our  
Canvas page this Monday (Sept. 16).

Please study the **specification**  
and **code skeleton** carefully

**Due Date**

**Sept. 30 (Mon) 23:59**