

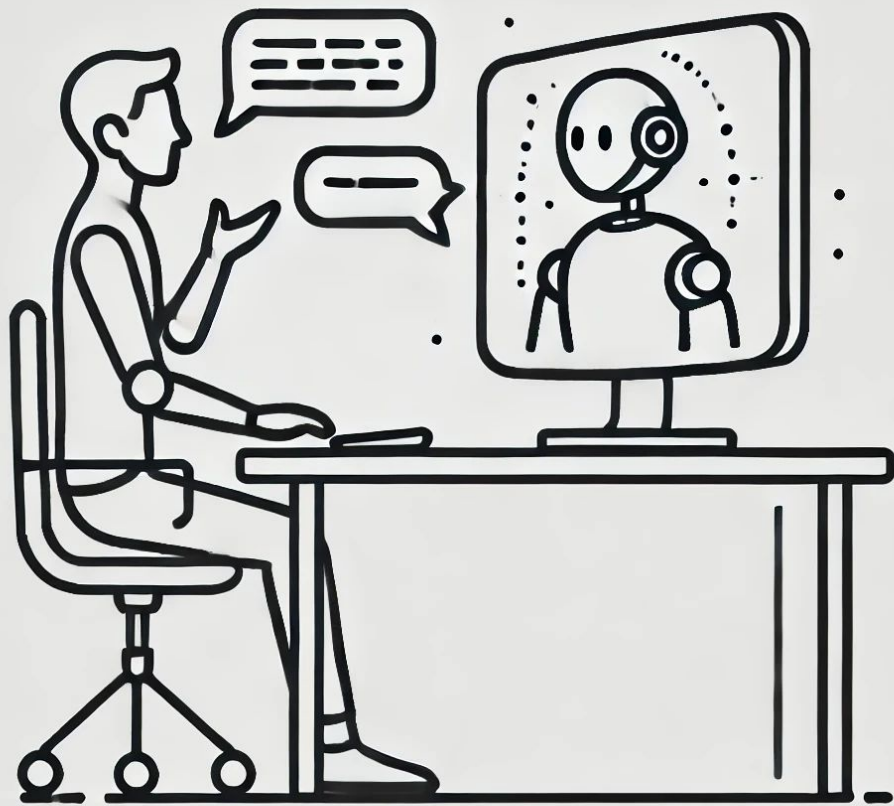
**2025 Spring semester**

# **Interaction Lab (1)**

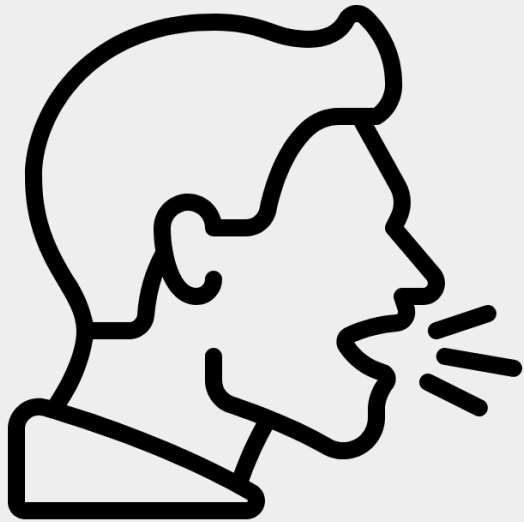
## **week 2**

Keunjung Bae

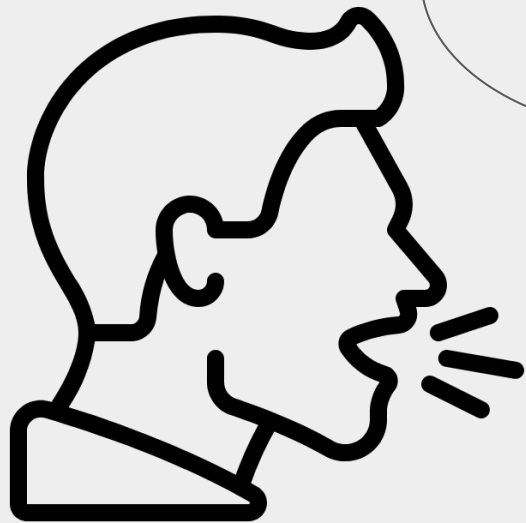
**Programming?**



통역가?



인간의 말과 비슷한 언어 = 프로그래밍 언어

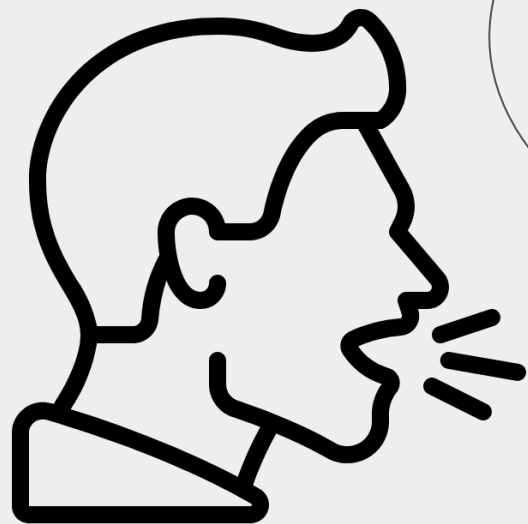


01010101011이  
라고 전해줘.



0/0

컴퓨터의 언어 = 이진수(binary)



안녕 Hello Hola  
Bonjour Hallo こんにちは  
你好



**Javascript, C++,  
C#,  
Java, Python...**

**Javascript, C++, C#,  
Java, Python...**



**통역가  
(번역)**





우리가 사용하는 프로그래밍  
언어는 프로세싱!



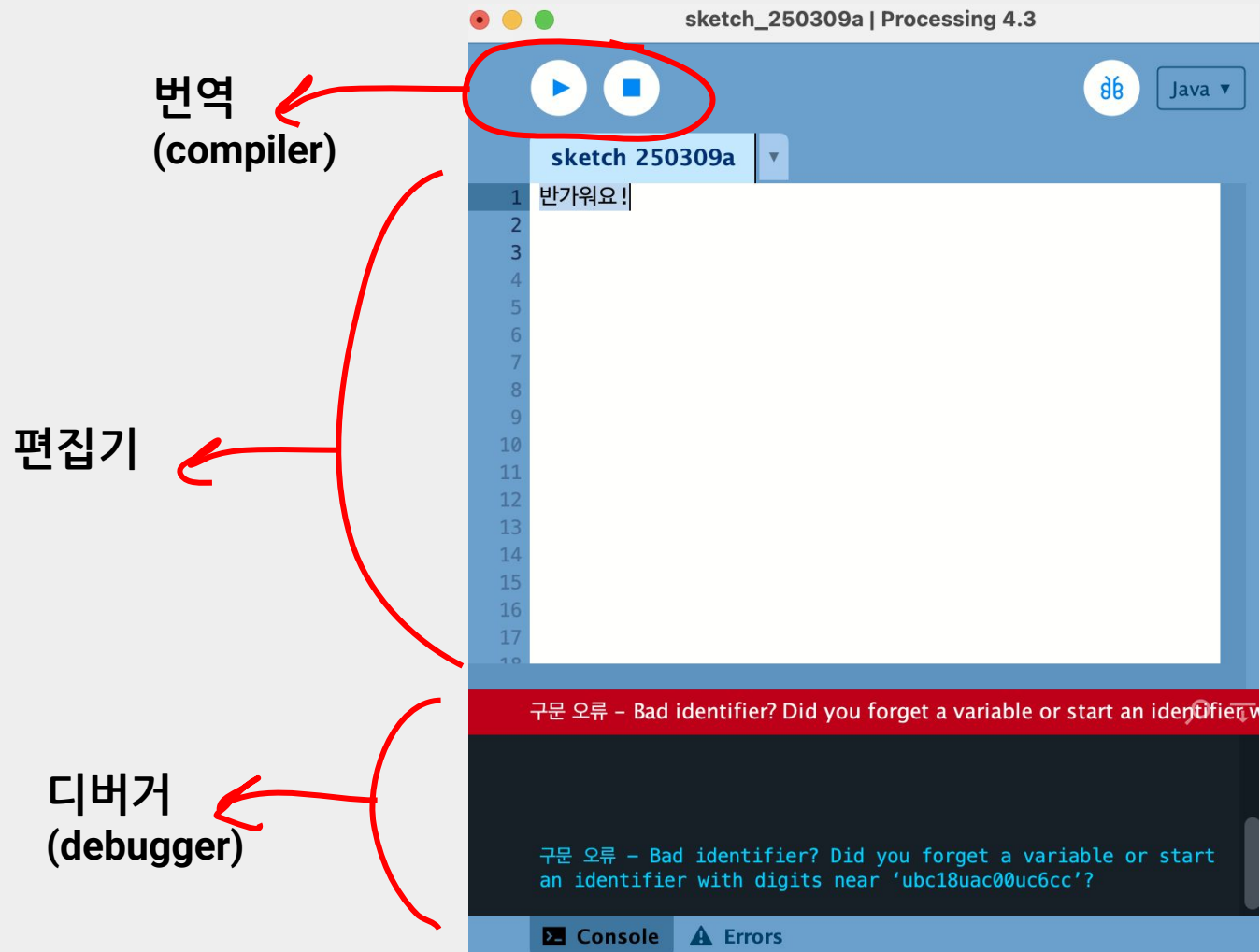
Java



Processing

# **통합개발환경**

## **(Integrated Development Environment)**

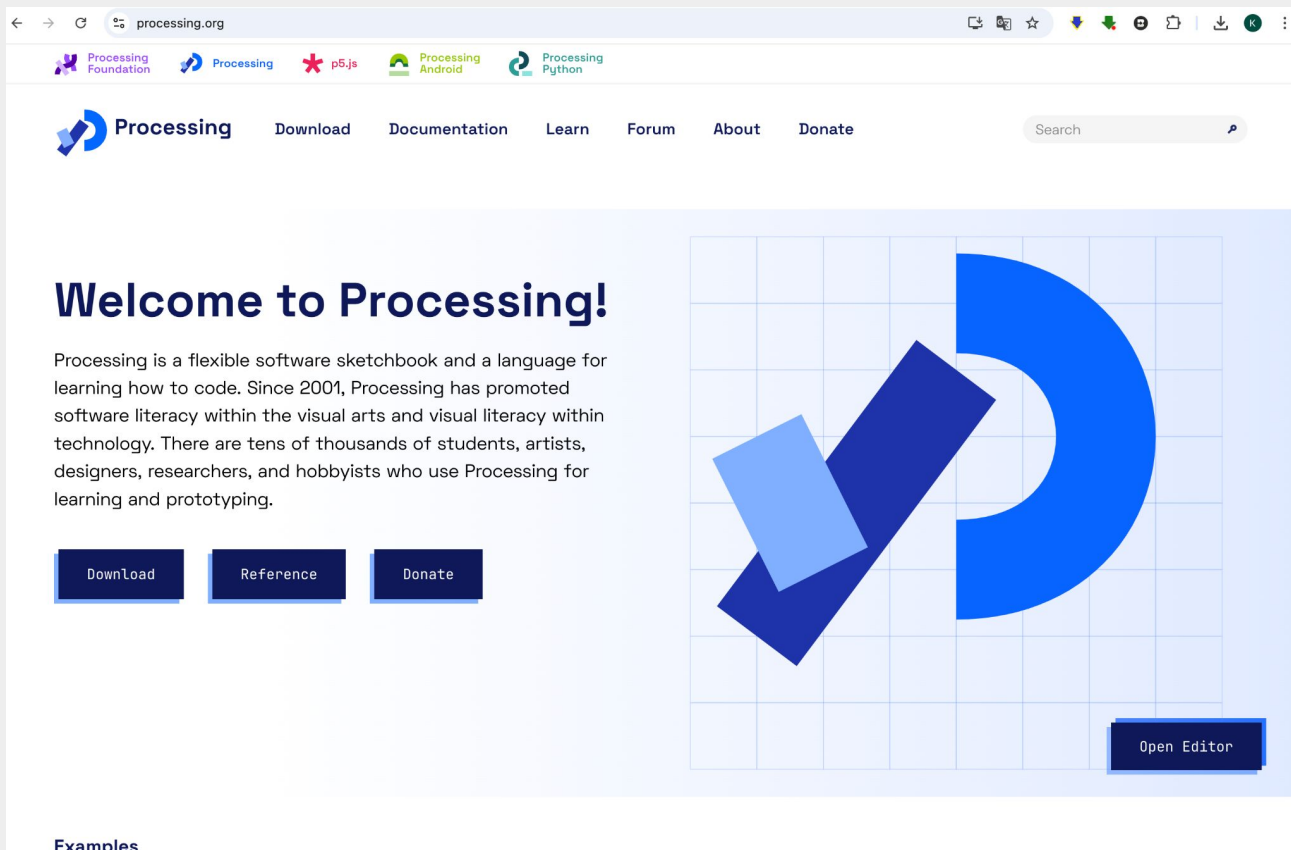


프로세싱 라이브러리

프로세싱 API

프로세싱 = 개발환경 + 라이브러리

[processing.org](https://processing.org) 접속!



The screenshot shows the homepage of the Processing website. At the top, there's a navigation bar with links for Processing Foundation, Processing, p5.js, Processing Android, and Processing Python. Below this is a main navigation bar with links for Download, Documentation, Learn, Forum, About, and Donate, along with a search bar. The main content area features a large "Welcome to Processing!" heading, followed by a paragraph describing Processing as a flexible software sketchbook and a language for learning how to code. Below the text are three buttons: Download, Reference, and Donate. To the right of the text is a large graphic of the Processing logo (a blue 'P' shape) on a grid background. At the bottom right of the grid is an "Open Editor" button. The footer contains the word "Examples".

processing.org 접속!

Processing Foundation Processing p5.js Processing Android Processing Python

Processing Download Documentation Learn Forum About Donate Search

# Welcome to Processing!

Processing is a flexible software sketchbook and a language for learning how to code. 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.

Download Reference Donate

Open Editor

Examples

## Casey Reas + Ben Fry | Processing



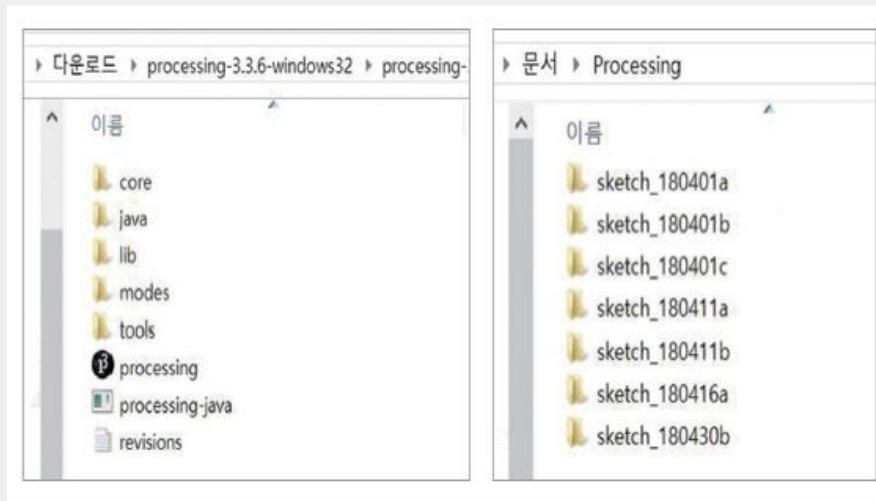
<https://reas.com>

## 윈도우PC 의 경우, <https://processing.org>

- 자신이 사용하고 있는 운영체제에 맞게 프로그램을 다운받는다.
- 다운로드 폴더에 (.exe) 파일이 저장되며, 문서 폴더에도 프로세싱 폴더가 생성된다.
- 만약 프로세싱에서 코드를 작성한 후 저장을 하면, 바로 이 문서 폴더 안에 저장된다.



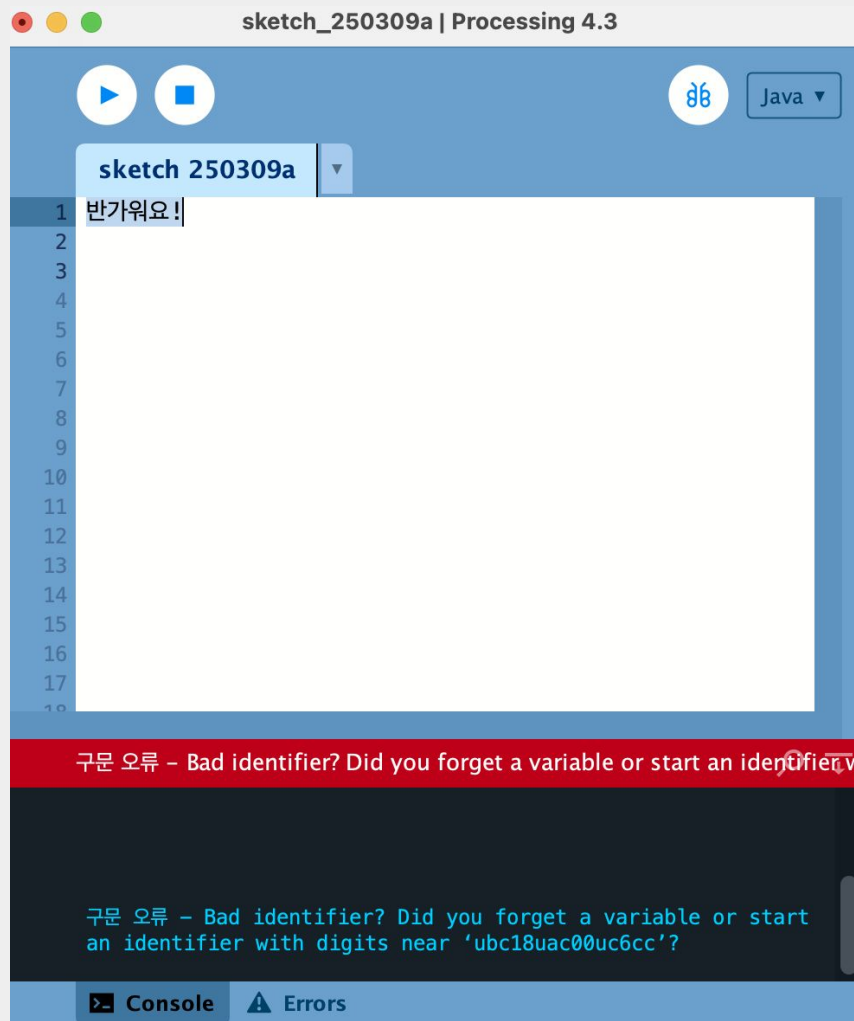
압축해제 폴더



다운로드 폴더

문서 폴더







Java ▼

HelloWorld

newTab



```
1 hello world!
```

```
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18
```

구문 오류 - Error on "hello world!"



구문 오류 - Error on "hello world!"

Console

Errors

스케치 창  
(sketch window)

실행 (run)

멈춤 (stop)

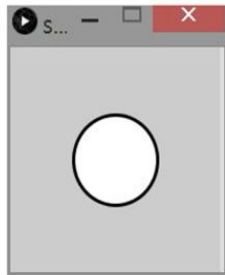
메뉴 (menu)

탭 (tab)

텍스트 에디터  
(text editor)

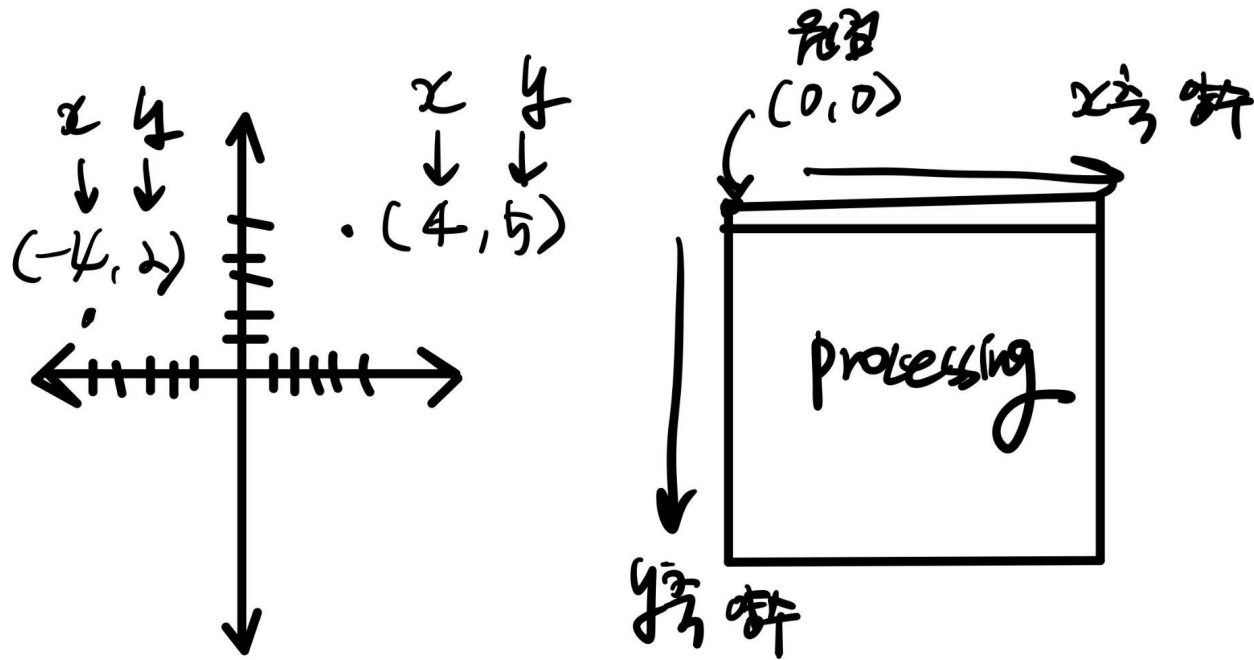
메세지 영역  
(message area)

콘솔 (console)



디스플레이 창  
(display window)





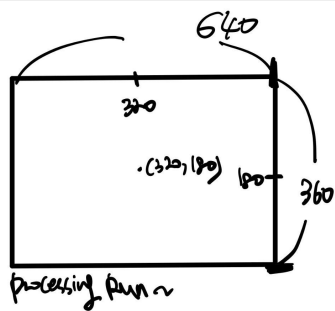
프로세싱화면 기본단위: px(픽셀)

# 도형을 그리기 위한 다양한 함수들

command → `size(800, 300);` → argument  
`ellipse(100, 150, 100, 100);`  
`rect(200, 100, 100, 100);`  
`triangle(400, 100, 350, 200, 450, 200);`  
`line(500, 150, 600, 150);` `arc(700, 150, 100, 100, 0, PI+HALF_PI);`

프로세싱에는 다양한 도형을 그리기 위한 함수들이 있다.

프로그램에서 기본적으로 제공하는 함수를 내장 함수 (built-in function)라 한다. 이러한 함수들을 이용해서 다양한 이미지를 그릴 수 있다.



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

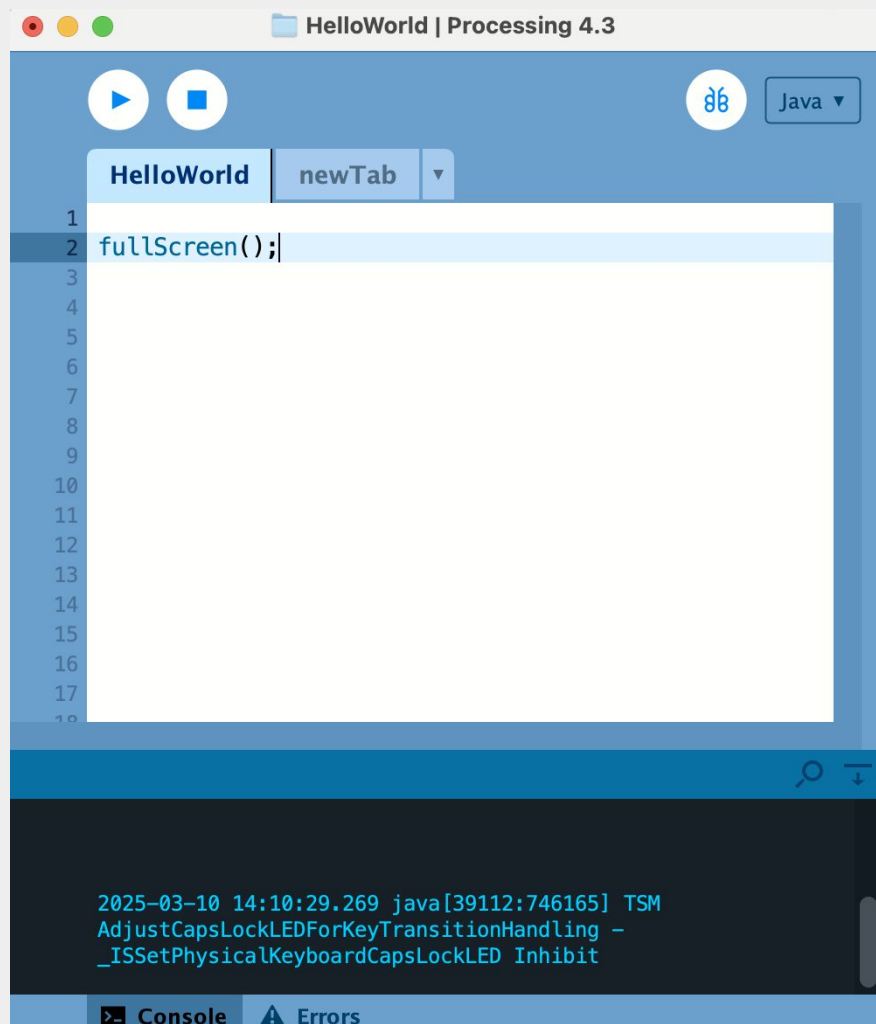
↑  
함수명

↑  
매개변수

함수(function)란 특정한 기능을 수행하는 프로그램의 구성요소 이다.

함수를 사용하기 위해서는 대소문자, 철자, 괄호, 매개변수 콤마, 그리고 세미콜론 등 정해진 문법과 형식을 지켜야 한다.

함수 다음에 오는 괄호 안에는 함수를 실행하는데 필요한 값들이 콤마 (,)로 구분되어 있다. 각 숫자들을 ‘매개변수(parameter)’ 또는 ‘인자(argument)’라 한다.

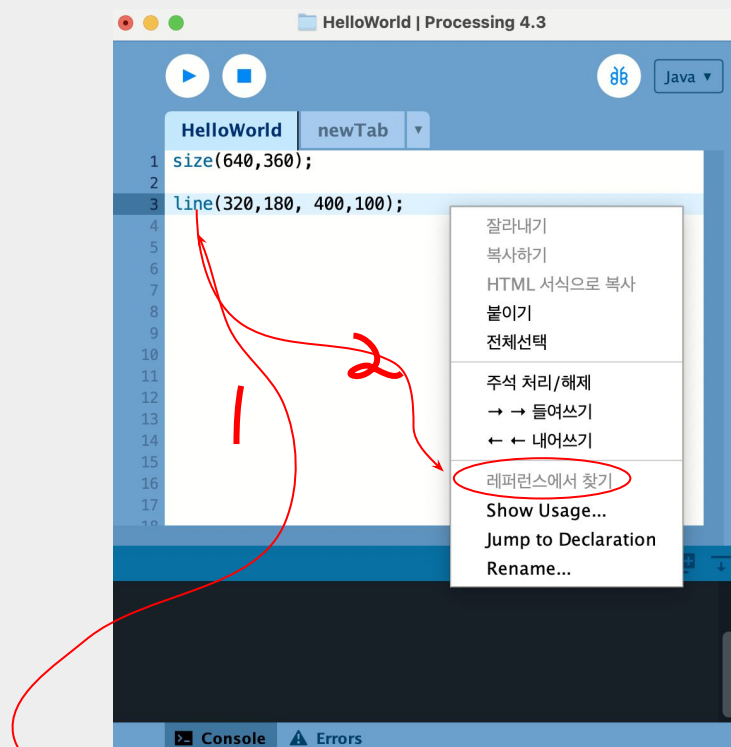
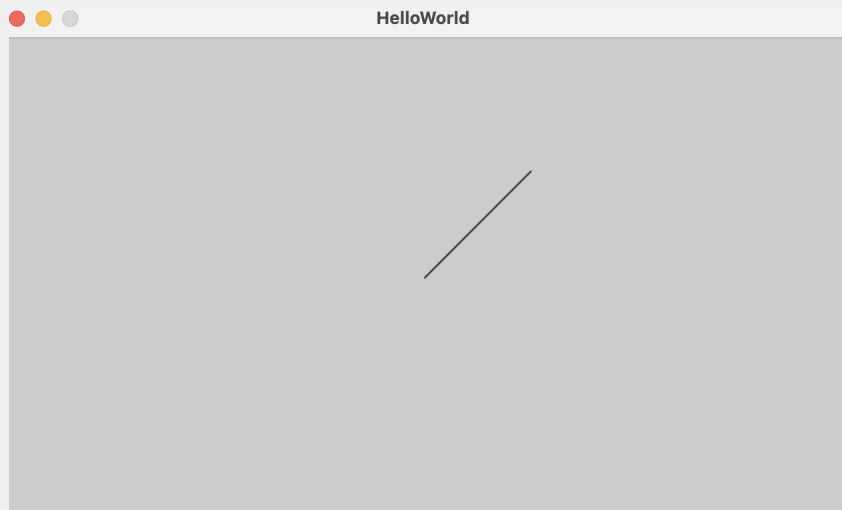


프로세싱 실행창 단축키:

맥= command+R/

윈도우 =ctrl+R

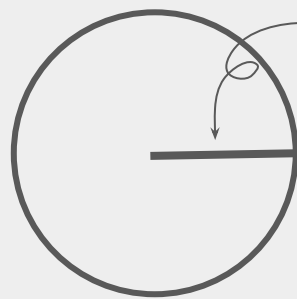
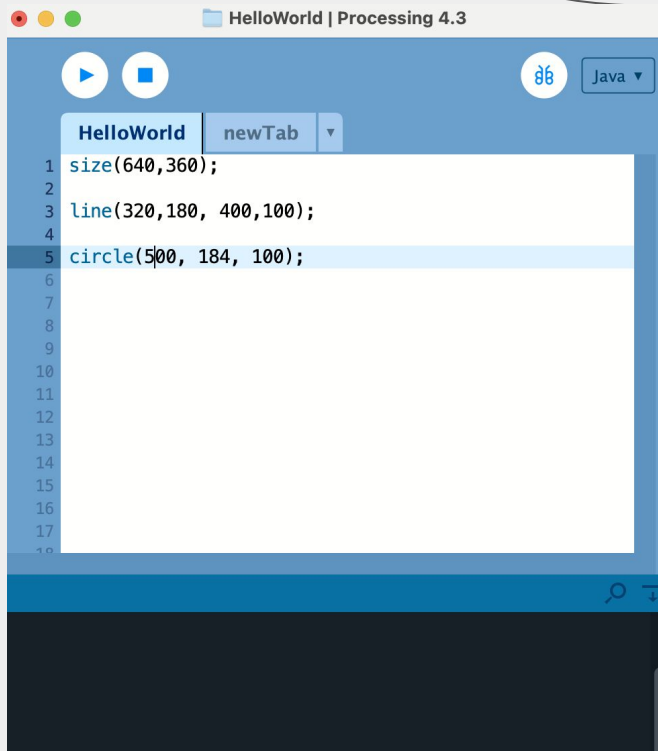
line(x1, y1, x2, y2);



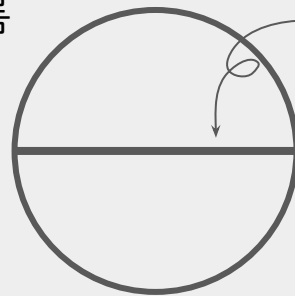
함수에 대고 ctrl+마우스 왼쪽클릭 →  
해당함수의 레퍼런스에서 찾기로 이동!



# circle(x, y, extent= diameter)

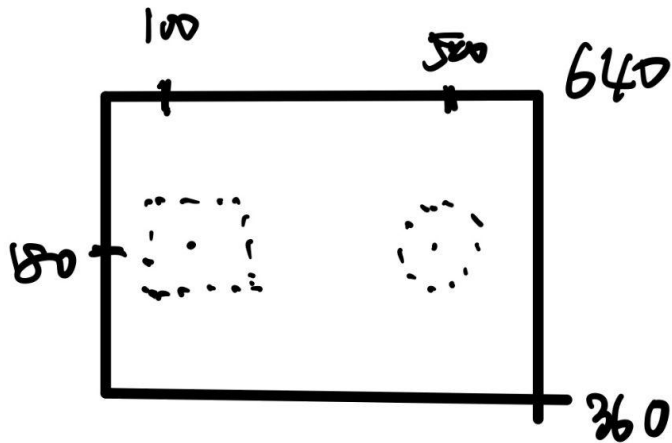


반지름

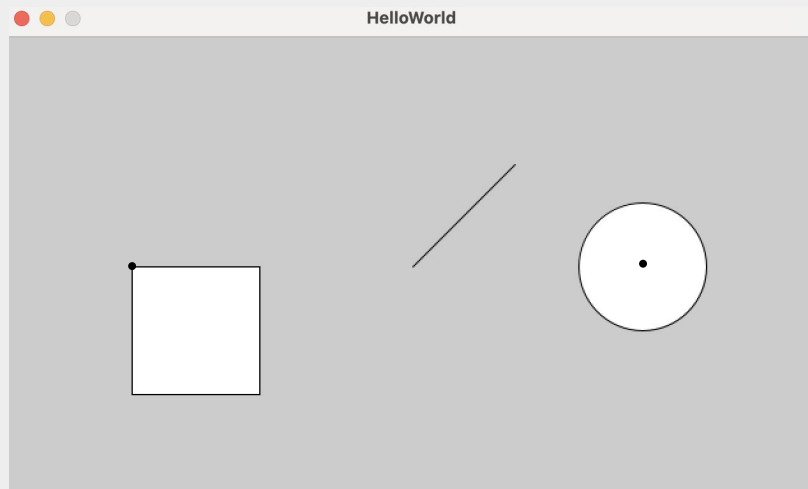


직경(diameter)

```
circle(500, 180, 100);  
square(100, 180, 100);
```



실제 실행시,

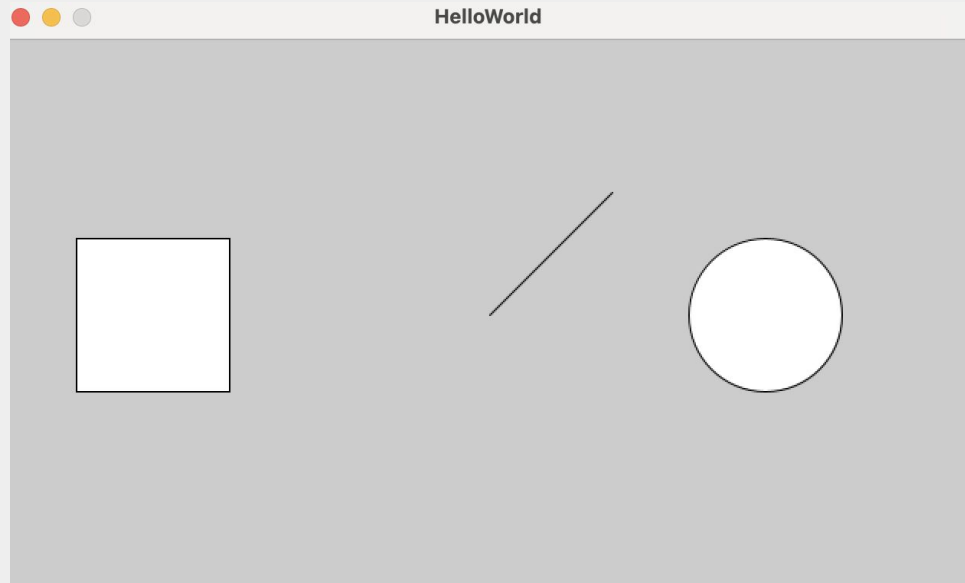
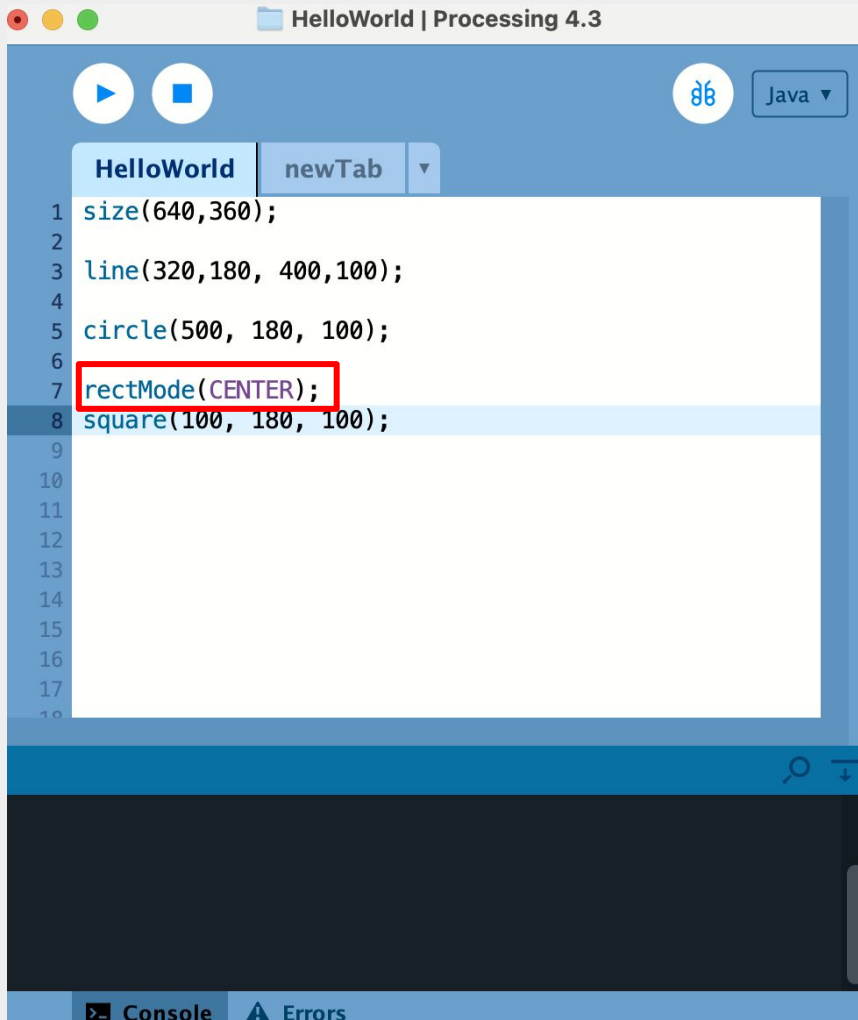


Name

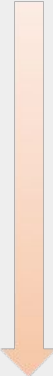
**square()**

Description

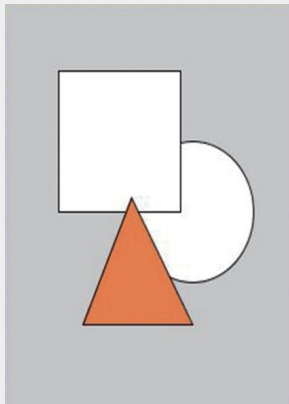
Draws a square to the screen. A square is a four-sided shape with every angle at ninety degrees and each side is the same length. By default, the first two parameters set the location of the upper-left corner, the third sets the width and height. The way these parameters are interpreted, however, may be changed with the `rectMode()` function.



## ❖ 프로그램의 실행순서



```
ellipse(50, 50, 20, 20);           // 원 그리기
rect(30, 30, 50, 50);               // 사각형 그리기
fill(255, 120, 45);                 // 주황색으로 채우기
triangle(100, 100, 50, 50, 30, 30); // 삼각형그리기
```

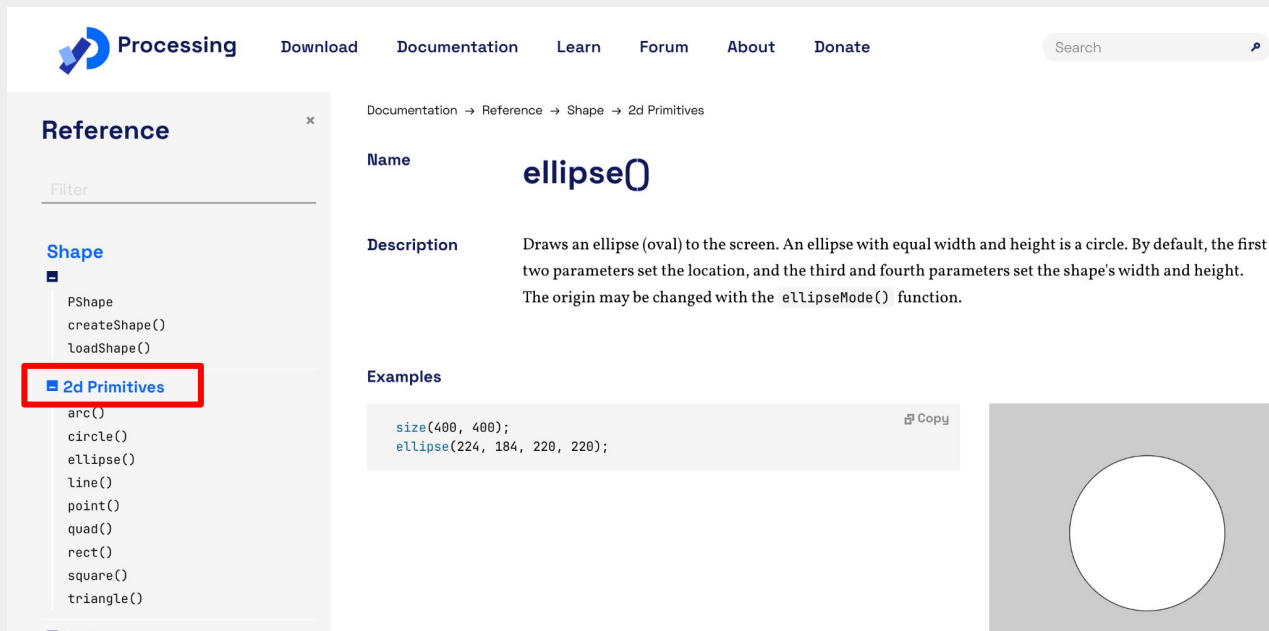


프로그램은 일반적으로 위에서부터 아래로 순차적으로 실행된다.

색 채우기 함수 fill( )에 의한 색상 적용은 그 다음에 오는 삼각형 (triangle)에만 적용된다.

# 실습:

<https://processing.org/reference> 의 2d primitives 메뉴의 ellipse, rect, line, circle, square, quad 등을 이용해서 여러가지 도형들을 프로세싱으로 그려보기!



The screenshot shows the Processing.org website's reference page for the `ellipse()` function. The page is titled "Reference" and has a sidebar on the left with a "Filter" section. Under "Shape", the "2d Primitives" category is highlighted with a red box. The main content area shows the "Name" as `ellipse()` and the "Description" as: "Draws an ellipse (oval) to the screen. An ellipse with equal width and height is a circle. By default, the first two parameters set the location, and the third and fourth parameters set the shape's width and height. The origin may be changed with the `ellipseMode()` function." Below the description, the "Examples" section shows a code snippet: `size(400, 400); ellipse(224, 184, 220, 220);` with a "Copy" button. To the right of the code is a visual example of a white circle on a gray background.

Processing

Download Documentation Learn Forum About Donate

Search

Documentation → Reference → Shape → 2d Primitives

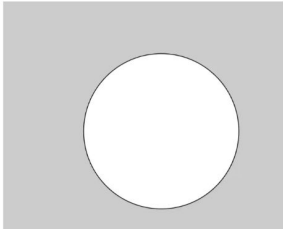
Name **ellipse()**

Description Draws an ellipse (oval) to the screen. An ellipse with equal width and height is a circle. By default, the first two parameters set the location, and the third and fourth parameters set the shape's width and height. The origin may be changed with the `ellipseMode()` function.

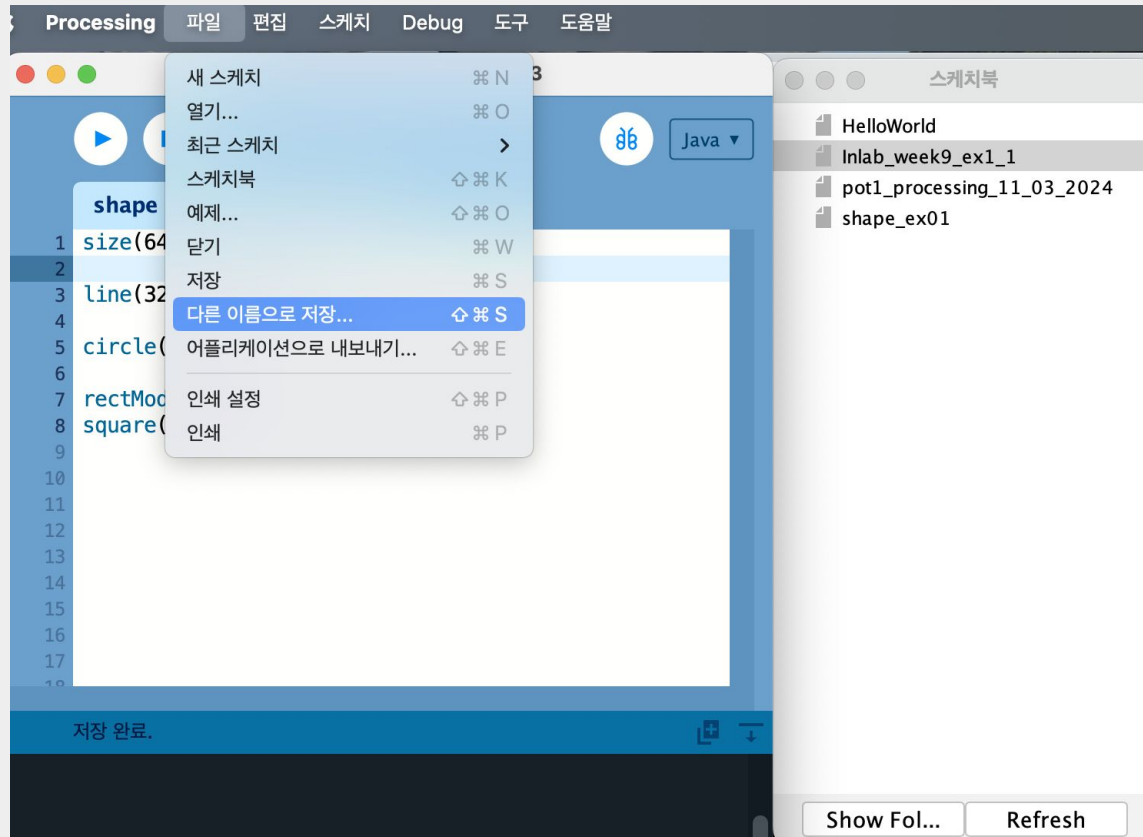
Examples

```
size(400, 400);
ellipse(224, 184, 220, 220);
```

Copy

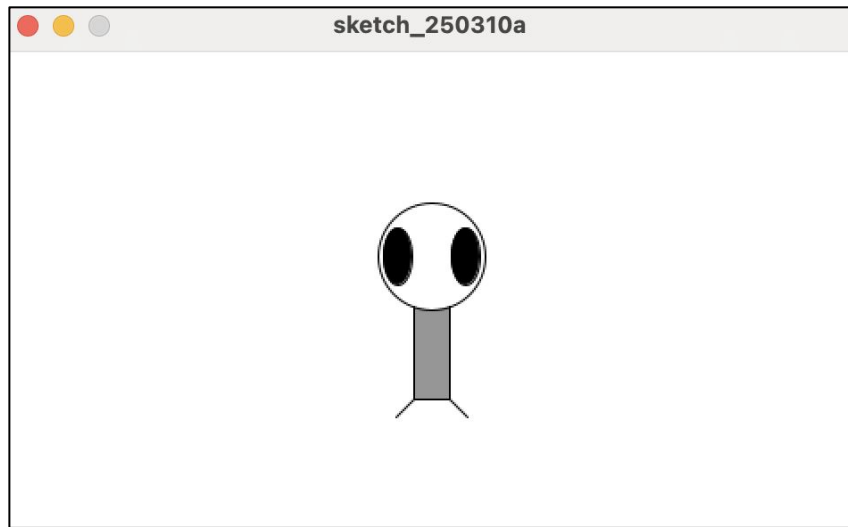


# 코드 저장



```
size(480, 270);  
background(255);  
ellipseMode(CENTER);  
rectMode(CENTER);  
  
// Body  
stroke(0);  
fill(150);  
rect(240, 145, 20, 100);  
  
// Head  
fill(255);  
ellipse(240, 115, 60, 60);  
  
// Eyes  
fill(0);  
ellipse(221, 115, 16, 32);  
ellipse(259, 115, 16, 32);  
  
// Legs  
stroke(0);  
line(230, 195, 220, 205);  
line(250, 195, 260, 205);
```

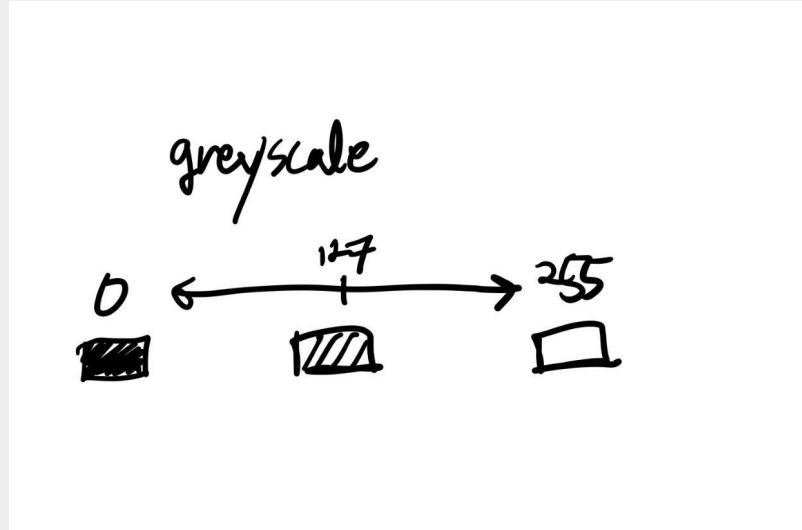
← 옆의 코드 프로세싱에 붙여넣고  
실행!



Color: background()

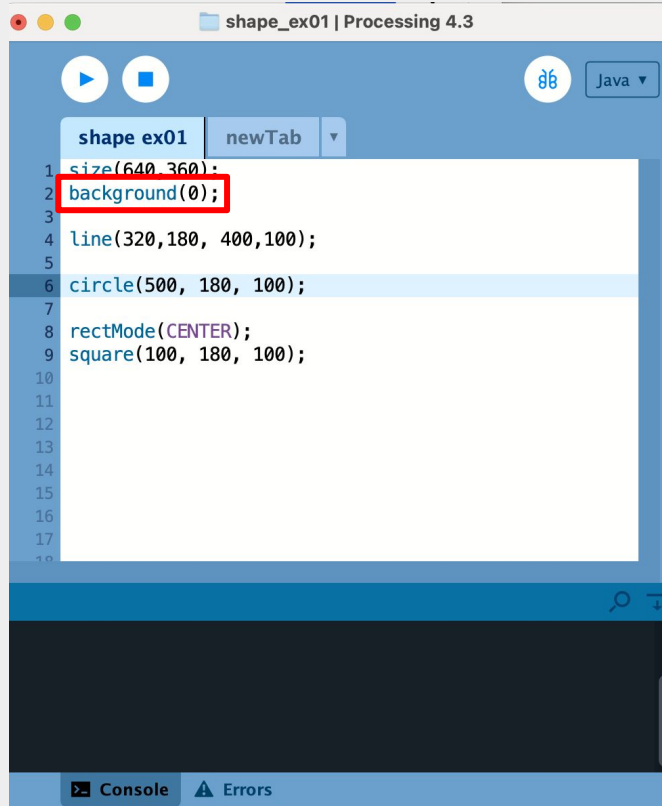
stroke()

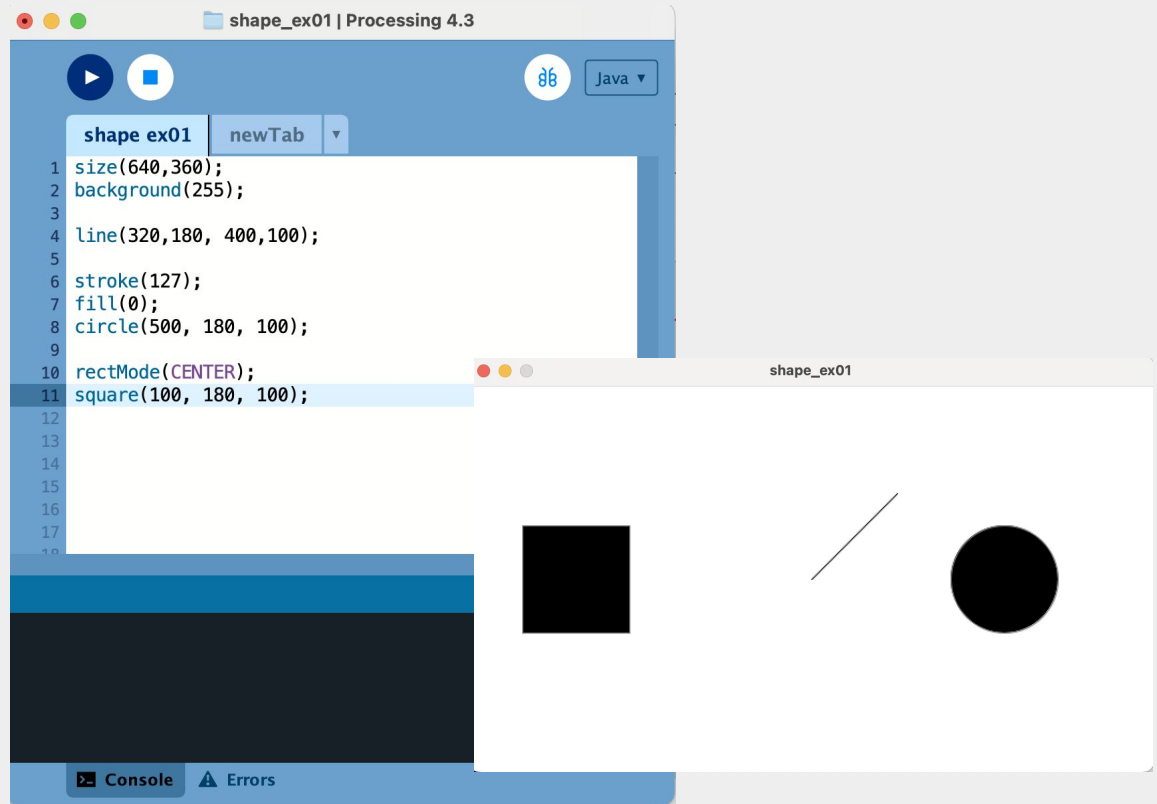
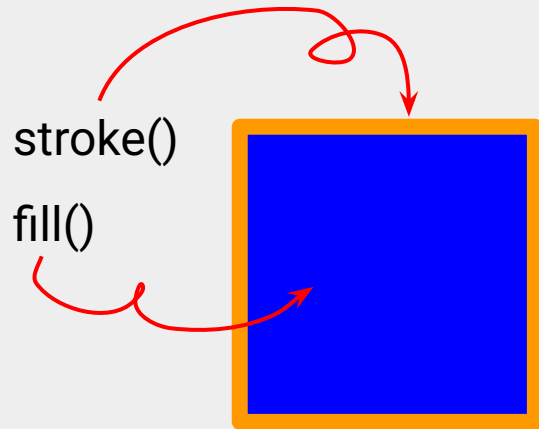
fill()





# background()



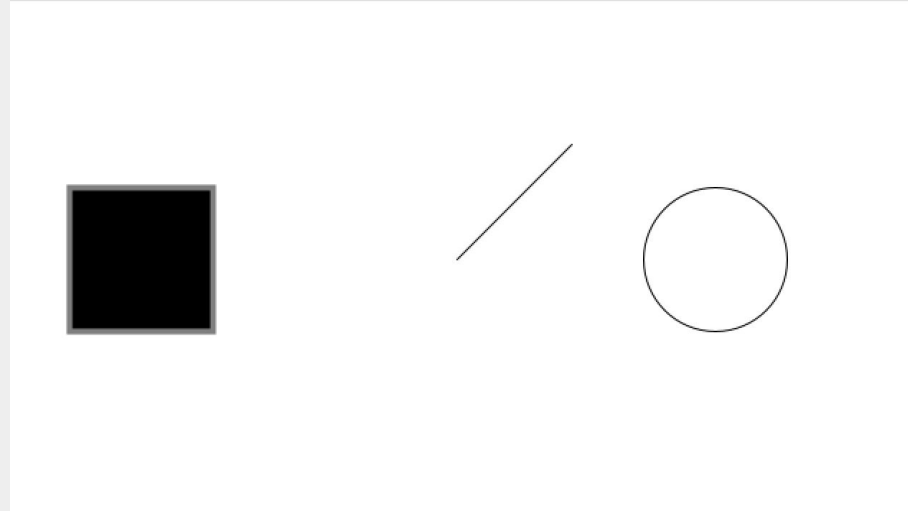
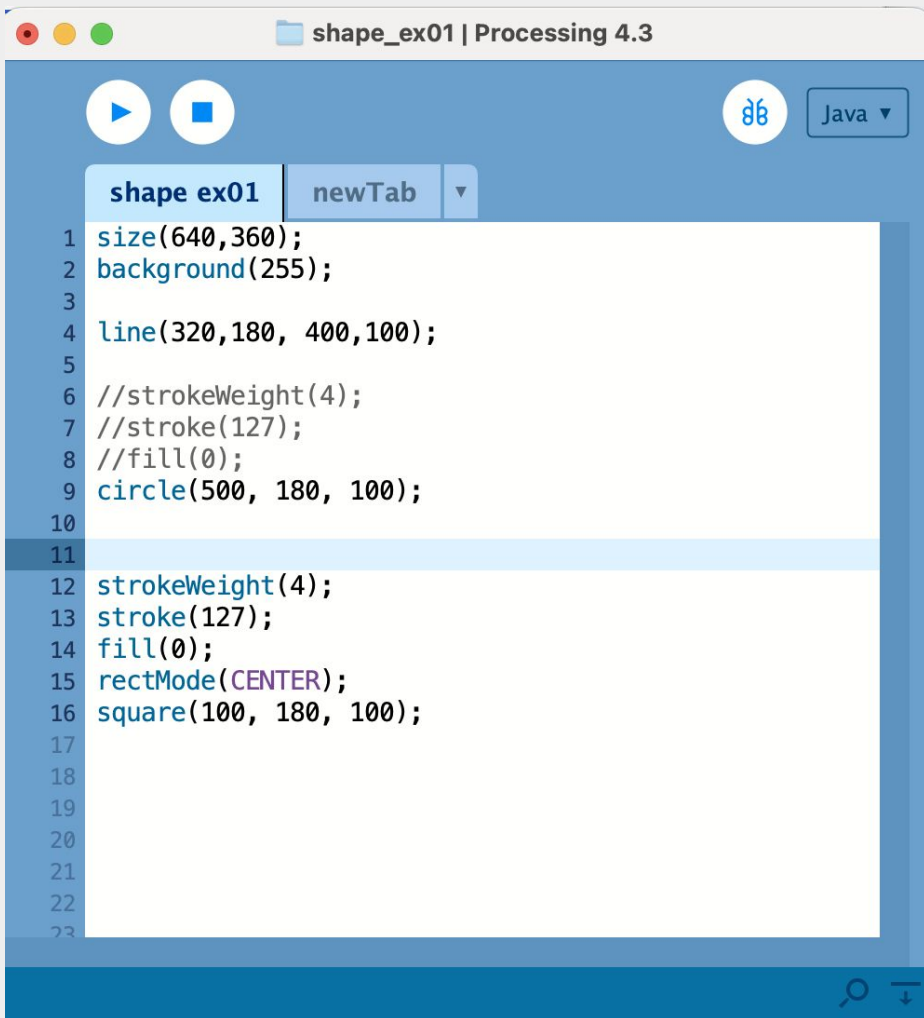


# strokeWeight()

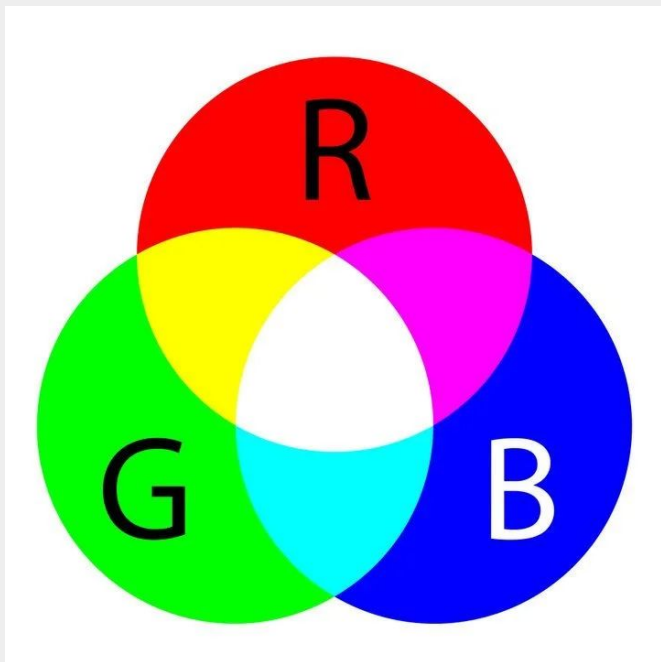
"먼저 도형(circle)을 그린다고  
선언하고,

그에 맞는 스타일(stroke, fill  
등)을 정의한다."





# Color: RGB Color



Red: 0 <-----> 255

Green: 0 <-----> 255

Blue: 0 <-----> 255

2진수(비트)	10진수 값
00000000	0
00000001	1
00000010	2
...	...
11111111	255

$$8\text{비트 (1바이트)} = 2^8 = 256$$

총 색상 조합:  $256 \times 256 \times 256 =$  약 1677만 가지 색 표현 가능

Red: 0 <-----> 255

Green: 0 <-----> 255

Blue: 0 <-----> 255

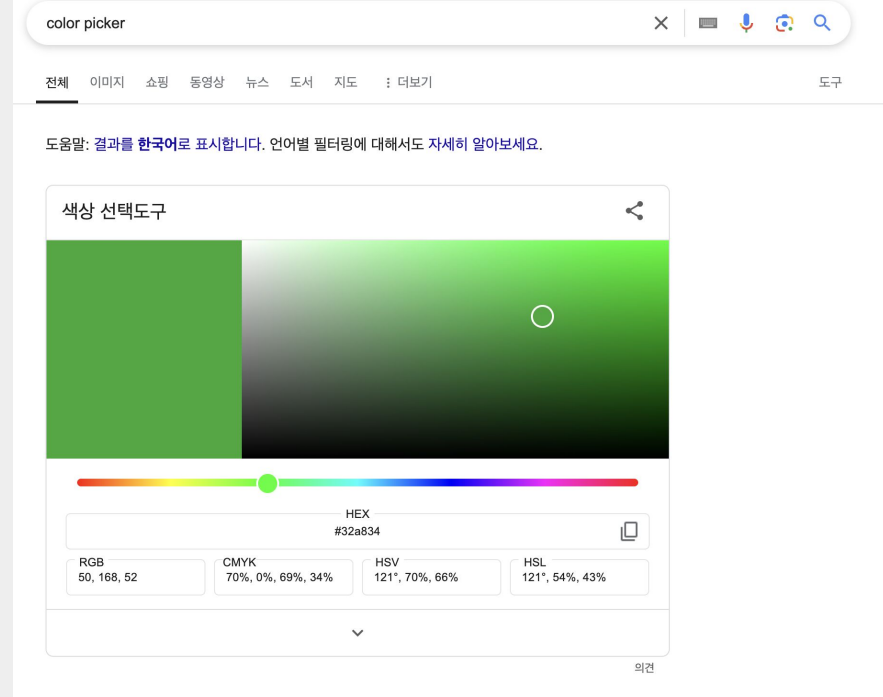
fill ( , , )  
R G B

0~ 255

빨간색: 255, 0, 0

초록색: 0, 255, 0

파란색: 0, 0, 255



background(127, 100, 50);

shape\_ex01 | Processing 4.3

shape ex01 newTab ▾

```
1 size(640,360);
2 background(255, 0, 100);
3
4 line(320,180, 400,100);
5
6 //strokeWeight(4);
7 //stroke(127);
8 //fill(0);
9 circle(500, 180, 100);
10
11
12 strokeWeight(4);
13 stroke(127);
14 fill(0);
15 rectMode(CENTER);
16 square(100, 180, 100);
17
18
19
20
21
22
23
```

shape ex01 newTab ▾

```
1 size(640,360);
2 background(255, 0, 100);
3
4 line(320,180, 400,100);
5
6 //strokeWeight(4);
7 //stroke(127);
8 //fill(0);
9 circle(500, 180, 100);
10
11
12 strokeWeight(4);
13 stroke(127, 10, 255);
14 fill(0, 200, 0);
15 rectMode(CENTER);
16 square(100, 180, 100);
17
18
19
20
21
22
23
```

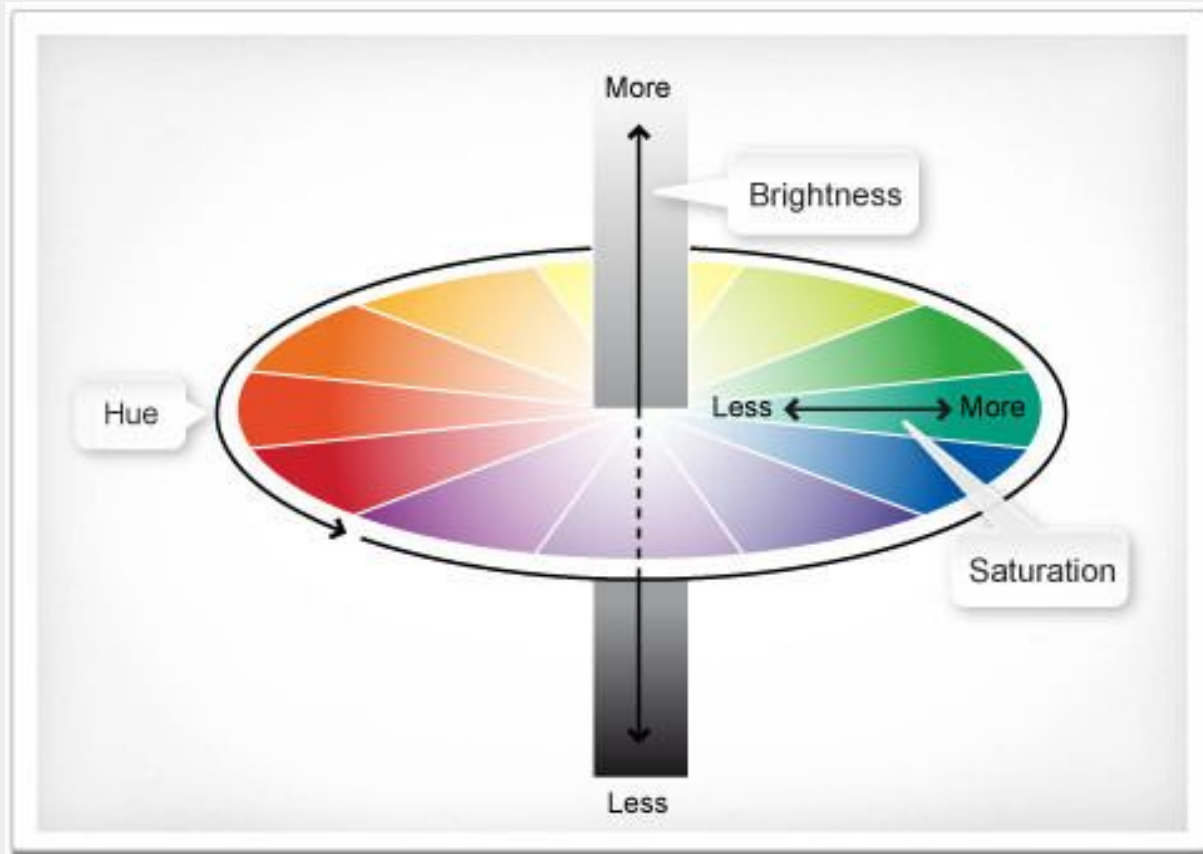


# Color: HSB?

## Processing에서 HSB는.

- 화면에 직접 표시되는 색상이 아님. 색을 쉽게 조절하기 위한 방식.
- 화면에 출력할 때는 내부적으로 RGB로 변환되기 때문에 RGB와 최종적으로는 동일한 색상이 나옴.
- 하지만 색을 조작할 때는 HSB가 훨씬 직관적이고 편리함.

속성	RGB (Red, Green, Blue)	HSB (Hue, Saturation, Brightness)
색 표현 방식	빨강(R), 초록(G), 파랑(B) 조합	색상(H), 채도(S), 밝기(B) 조합
색의 조절	밝기나 색상 변경이 복잡함	색조(H)만 바꾸면 쉽게 색 변경 가능
밝기 변경	R/G/B 값을 조절해야 해서 직관적이지 않음	B(밝기) 값만 조절하면 돼서 직관적



```
colorMode(hsb, 360,100,100);
```

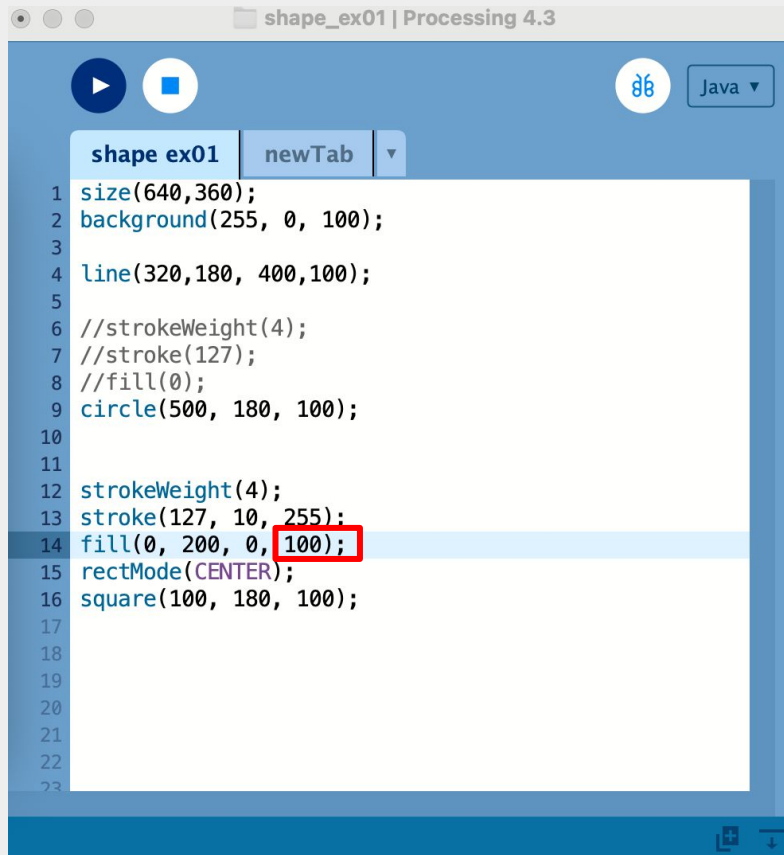
# Transparency: alpha

RGB Color + alpha = color!!

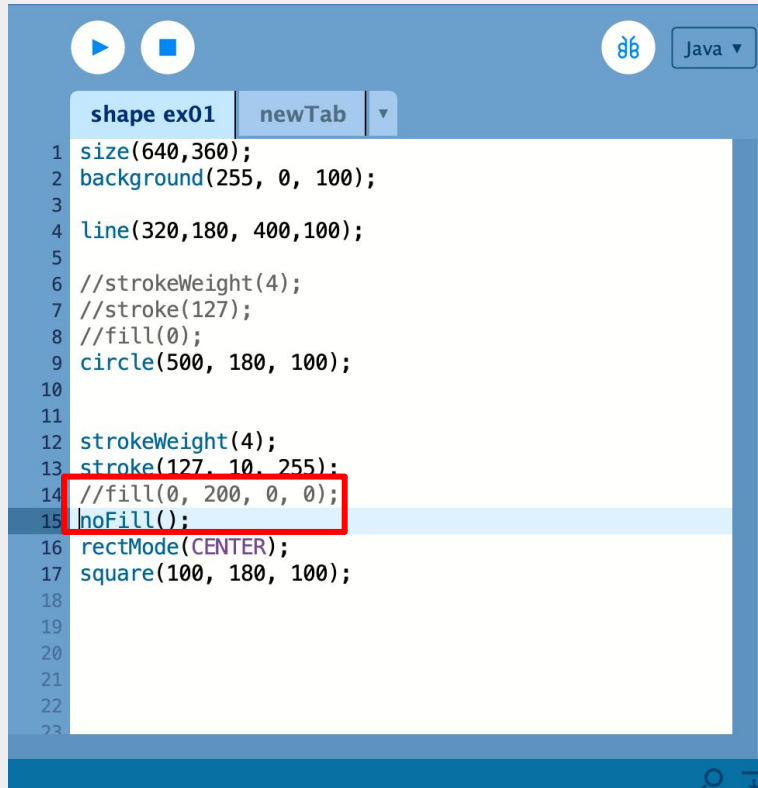
0~ 255

0 → 완전히 투명 (보이지 않음)

255 → 완전히 불투명 (완전히 보임)



# noStroke() noFill()



The screenshot shows a Java IDE window titled "shape ex01" with a "newTab" button. The code editor contains the following lines:

```
1 size(640,360);
2 background(255, 0, 100);
3
4 line(320,180, 400,100);
5
6 //strokeWeight(4);
7 //stroke(127);
8 //fill(0);
9 circle(500, 180, 100);
10
11
12 strokeWeight(4);
13 stroke(127, 10, 255);
14 //fill(0, 200, 0, 0);
15 noFill();
16 rectMode(CENTER);
17 square(100, 180, 100);
18
19
20
21
22
23
```

A red rectangular box highlights lines 14 and 15, which are commented out. The IDE interface includes a play button, a stop button, a line number indicator (86), and a language dropdown menu set to "Java".



The screenshot shows a Java IDE window titled "shape ex01" with a "newTab" button. The code editor contains the following lines:

```
1 size(640,360);
2 background(255, 0, 100);
3
4 line(320,180, 400,100);
5
6 //strokeWeight(4);
7 //stroke(127);
8 //fill(0);
9 circle(500, 180, 100);
10
11
12 //strokeWeight(4);
13 //stroke(127, 10, 255);
14 noStroke();
15 fill(0, 200, 0);
16 //noFill();
17 rectMode(CENTER);
18 square(100, 180, 100);
19
20
21
22
23
```

A red rectangular box highlights lines 14 and 15, which are active. The IDE interface includes a play button, a stop button, a line number indicator (86), and a language dropdown menu set to "Java".

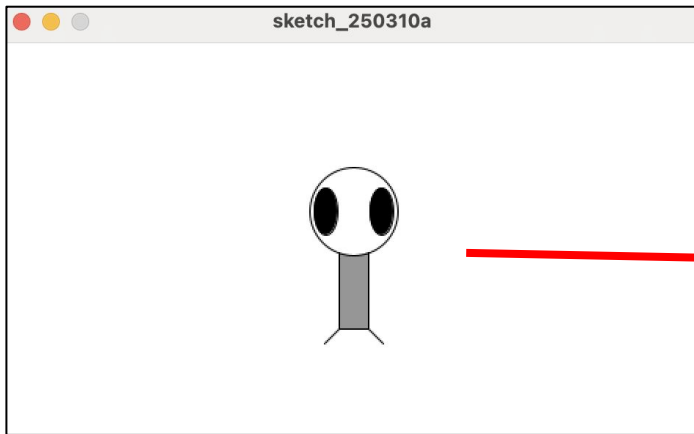
```
size(480, 270);  
background(255);  
ellipseMode(CENTER);  
rectMode(CENTER);  
  
// Body  
stroke(0);  
fill(150);  
rect(240, 145, 20, 100);  
  
// Head  
fill(255);  
ellipse(240, 115, 60, 60);  
  
// Eyes  
fill(0);  
ellipse(221, 115, 16, 32);  
ellipse(259, 115, 16, 32);  
  
// Legs  
stroke(0);  
line(230, 195, 220, 205);  
line(250, 195, 260, 205);
```

## 실습:

옆 코드에서 fill, background, alpha값,

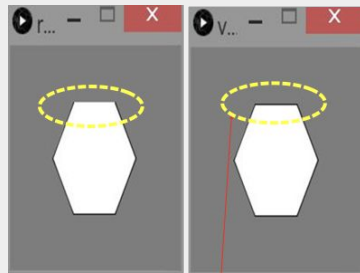
noStroke, noFill, strokeWeight등

색상 다양하게 적용해보기 → 색있는 캐릭터로 바꾸기.



# 복잡한 도형 그리기

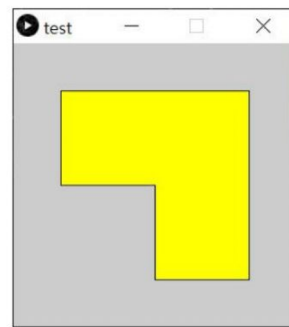
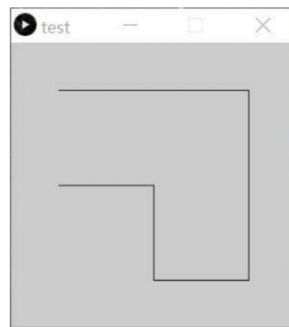
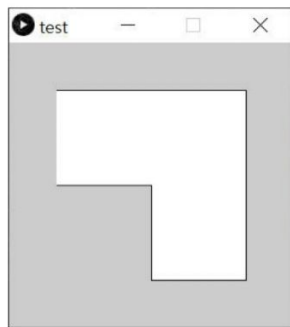
`vertex(x, y);`



- 꼭지점을 뜻하는 `vertex()` 함수를 이용하면 어떠한 복잡한 형태도 자유롭게 그릴 수 있다.
- `vertex()` 함수는 괄호 안에 `x, y` 점의 위치를 매개변수로 갖는다.
- 이 `vertex` 들을 연결하여 다양한 형태를 그릴 수 있다.
- `vertex()` 함수를 사용하기 위해서는 `beginShape()`와 `endShape()` 함수를 써주어야 한다.
- `beginShape()` 함수는 점의 기록을 시작, `endShape()` 함수는 점의 기록을 중지함을 의미한다.
- 외곽선을 끝까지 연결해 주기 위해서는 `endShape(CLOSE)` 모드로 닫아준다.

`endShape(CLOSE);`

```
size(500,500);  
//noFill();  
beginShape();  
  
vertex(50, 50);  
vertex(250, 50);  
vertex(250, 250);  
vertex(150, 250);  
vertex(150, 150);  
vertex(50, 150);  
//endShape();  
endShape(CLOSE);
```



**실습:**

vertex, beginShape(), endShape()등을 이용해서 다양한 형태를 그려보세요.



```
size(480, 270);
background(255);
ellipseMode(CENTER);
rectMode(CENTER);

// Body
stroke(0);
fill(150);
rect(240, 145, 20, 100);

// Head
fill(255);
ellipse(240, 115, 60, 60);

// Eyes
fill(0);
ellipse(221, 115, 16, 32);
ellipse(259, 115, 16, 32);

// Legs
stroke(0);
line(230, 195, 220, 205);
line(250, 195, 260, 205);
```

## setup:

- 프로그램 시작할때 딱 1번만 실행.
- 주로 초기설정(화면 크기, 변수초기화등, size, background..etc)
- 비유: 영화 세트준비과정

## draw:

- 프레임마다 반복실행(초당 60번 기본실행)
- 프레임마다 화면을 업데이트(애니메이션, 인터랙션등)
- 비유: 영화를 실제 프레임으로 찍어 기록하는 과정
- fill, circle.. etc

```
void setup() {  
    size(400, 400); // 창 크기 설정 (이 블록 안에서 실행됨)  
    background(255); // 배경을 흰색으로 설정  
}  
  
void draw() {  
    ellipse(mouseX, mouseY, 50, 50); // 마우스를 따라 원을 그림  
}
```

실행순서: setup → draw1번 → draw2번... → draw무한(종료할때까지)

## 2주차 과제:

오늘 배운 여러가지 함수들로 나만의 얼굴 그리기!(나의 얼굴, 내가 좋아하는 캐릭터 도 좋음!)

- 마감기한:  
다음주 **화요일 오전 9시**까지!(늦으면 감점!)
- 블로그 만들어서 과제 블로그 링크를 **클래스넷-클래스룸-과제 게시판**에 '1주차 과제, 학번, 이름'의 제목으로 업로드하기. (내용: 코드짤거, 실행창 캡처한거).

