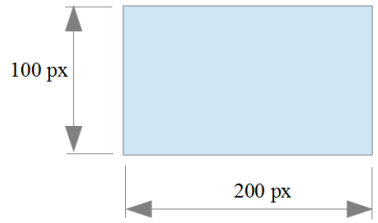
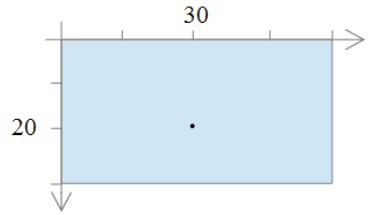
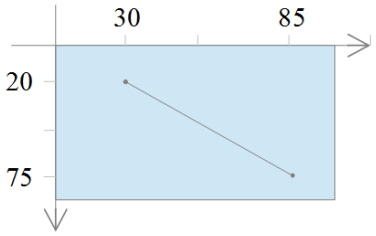
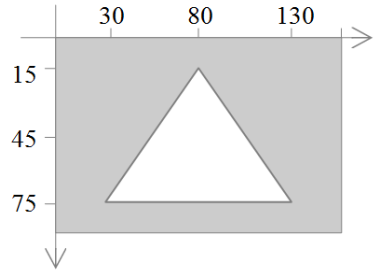
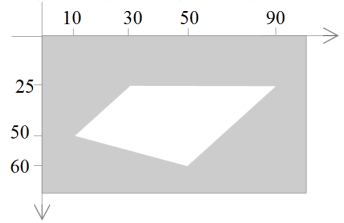
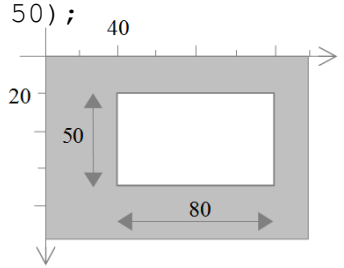
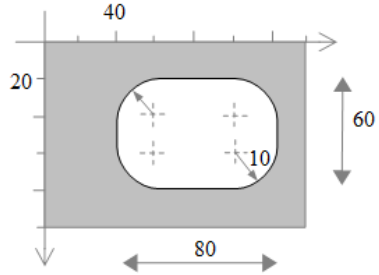
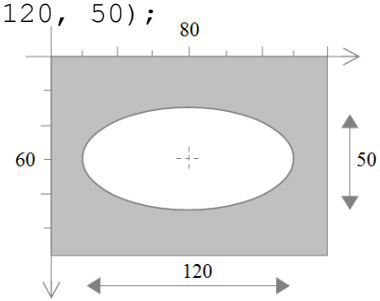
Extraído, traducido y adaptado de la guía de Referencia de Processing en https://processing.org/reference/

Comando	*Uso*	*Ejemplos*
size (ancho, alto);	Define las **dimensiones de la ventana** indicando su **anchura** y **altura** en píxeles. Si no se utiliza esta orden, o no se pone nada entre paréntesis, aparece la ventana por defecto que es de 100x100 Sólo puede usarse **una vez** en un mismo programa	`size(200, 100);` 
point (x, y);	Dibuja un **punto** en las coordenadas indicadas	`point(30, 20);` 
line (x1, y1, x2, y2);	Dibuja una **línea recta** entre los puntos indicados por las coordenadas	`line(30, 20, 85, 75);` 
triangle (x1, y1, x2, y2, x3, y3);	Dibuja un **triángulo** cuyos vértices sean las coordenadas indicadas.	`triangle(30, 75, 80, 15, 130, 75);` 

Comando	*Uso*	*Ejemplos*
quad (x1, y1, x2, y2, x3, y3, x4, y4);	Dibuja un **cuadrilátero** cuyos **vértices** sean los indicados, siguiendo el mismo **orden** en que se presentan	`quad(10, 50, 30, 25, 90, 25, 50, 60);` 
rect (x1, y1, ancho, alto); rect (x1, y1, ancho, alto, radio);	Dibuja un **rectángulo** cuyo **vértice superior izquierdo** esté situado en las coordenadas indicadas (**x1, y1**) y que tenga como dimensiones la **anchura** y **altura** que se indican. Si queremos que tenga todas las **esquinas redondeadas** por igual, incluimos un parámetro más para indicar el **radio** del arco.	`rect(40, 20, 80, 50);`  `rect(40, 20, 80, 60, 10);` 
ellipse (x1, y1, ancho, alto);	Dibuja una **elipse** con **centro** en las coordenadas indicadas (**x1, y1**) y un tamaño definido por la **anchura** y la **altura**.	`ellipse(80, 60, 120, 50);` 

Comando	*Uso*	*Ejemplos*
background (...);	Especifica un **color de fondo para la ventana** de dibujo. Si se pone un sólo número de **0 a 255** se trata de un tono de gris. Es indiferente indicar el tamaño de la ventana antes o después. Cuanto mayor sea el valor, más claro es el color	`background(51);`

stroke (...);	Define el **color de la línea** o el **borde de la figura** que se va a dibujar a continuación. Si se especifica un sólo número de **0 a 255**, se trata de un tono de gris.	`stroke(153);`
		`rect(30, 20, 80, 50);`
		`// rectángulo relleno de color gris`

noStroke ( );	Se usa para indicar que **no se ponga borde** a las figuras que se dibujen después (y tampoco se dibujen líneas o puntos)	`noStroke();` `rect(30, 20, 80, 50);`

fill (...);	Sirve para indicar el **color de fondo de las figuras que se van a dibujar después**. Si se indica un sólo número de **0 a 255**, se trata de un tono de gris.	`fill(153);`
		`rect(30, 20, 80, 50);`
		`triangle (10, 90, 50, 150, 80, 100);`

boolean	Comando que se usa para **crear una variable de tipo lógico** que sólo admite dos valores: **true** o **false** (es decir, verdadero o falso).	`boolean a;`
		`a= false;`
		
		`// también sirve boolean a = false;`
		`// o el contrario boolean a = true;`

float	Se usa para **crear una variable del tipo decimal**. Debido al redondeo, la precisión de estos valores no es muy fiable.	`float a;`
		`a = 1.5387;`
		
		`// también sirve float a = 1.5387;`

int	Sirve para **crear una variable** del tipo **número entero**.	`int n;`
		`n = 4;`
		
		`// también sirve int n = 4;`

String	**Crea una variable del tipo "texto", "frase" o "cadena de caracteres"**. Estos caracteres son tratados como texto y no se puede hacer operaciones aritméticas con ellos, aunque sí de comparación.	`String p;`
		`p = "patata";`
		`println(p);`
		
		`// también sirve String p = "patata";`
		`// RESULTADO EN LA CONSOLA: patata`

Comando	*Uso*	*Ejemplos*

color	Se usa para **crear una variable del tipo "color"**. Si se pone un sólo valor entre 0 y 255 se entiende que es un tono de gris.	`color c = color(125);`
		`fill(c);`
		`rect(30, 20, 80, 50);`

delay (...);	Detiene la ejecución del programa durante los milisegundos que se indiquen entre paréntesis. El efecto producido es un retardo.	`delay (1000);` `/* detiene el programa durante 1 segundo (1000 milisegundos) */`

print (...);	**Escribe** en el área llamada "**consola**" el texto o el número que se indica entre paréntesis. Si escribimos otro texto después, aparecerá en la **misma línea**.	`print ("Hola a todos: ");`
		`String frase = "hoy es día ";`
		`print (frase);`
		`int a = 25;`
		`print (a);`
		`// RESULTADO EN LA CONSOLA:`
		`// Hola a todos: hoy es día 25`

println (...);	**Escribe** en la **consola** el valor o texto indicado entre paréntesis y **después** hace un **salto de línea**. Es decir, si escribimos otro texto después, aparecerá en la **línea de abajo**.	`println ("Hola.");`
		`String frase = "Me llamo Luis";`
		`print (frase);`
		`// RESULTADO EN LA CONSOLA:`
		`// Hola.`
		`// Me llamo Luis`

PI	**PI** es la constante matemática de valor 3.1415927 (representada habitualmente por π)	`int radio = 30;` `float longitud_circ =  2*PI*radio;`

for (...) {...}	**Bucle**: Crea una secuencia de repeticiones controladas por el valor de una variable que cambia de forma ordenada. **Entre paréntesis ( )** hay que indicar: el *valor inicial* de la variable, la *condición para continuar ejecutando el bucle* y la forma en que debe *incrementarse* la variable en cada ocasión. **Entre llaves { }** se situarán las órdenes que se van a repetir en cada pasada.
	Su estructura es:
	for (valor_inicial ; condición_para_seguir ; incremento)
	{ instrucciones a repetir ; ..... ; ..... ; }
		`size(800, 800);`
		`for (int x = 0; x < 800; x = x + 20){`
		`  rect(x, 0, 10, 10);`
		`}`
		--- 	
		`/* Repite el dibujo de un cuadrado cambiando la coordenada x */`

/ / …. *o bien* **/*……*/**	Escribir comentarios en un renglón //... o en varios /*....*/	`// Este texto no se ejecutará`

<table>
<tr><th>Comando</th><th>Uso</th><th>Ejemplos</th></tr>
<tr><td>for (...) {...}

uso con arrays, o sea con cadenas o matrices de datos</td><td>También sirve para seleccionar por orden los valores de una matriz de datos (array):
En el caso del uso con array sería:

<pre>for (elemento : array) {
  comandos;
}</pre>
Se puede anidar bucles for unos dentro de otros</td><td><pre>int[] nums = { 5, 4, 3, 2, 1 };

for (int i : nums) {
  println(i);
}

for (int i = 30; i < 80; i = i+5) {
  for (int j = 0; j < 80; j = j+5) {
    point(i, j);
  }
}</pre></td></tr>
<tr><td>sin (...);</td><td>Calcula el seno de un ángulo</td><td><pre>float a = 0.0;
float inc = 2*PI/25.0;

for (int i = 0; i < 100; i=i+4) {
  line(i, 50, i, 50+sin(a)*40.0);
  a = a + inc;
}</pre></td></tr>
<tr><td>void setup ( ) {...}</td><td>Ejecuta comandos que sólo se usarán una vez. Sirve para definir las condiciones iniciales como el tamaño de la ventana y cargar elementos como imágenes y tipos de letra.
Si se usa la orden size, debe ponerse en primer lugar.
Si se usa void setup hay que usar también void draw justo después.</td><td rowspan="2"><pre>int x = 0;

void setup() {
  size(200, 200);
  background(0);
  noStroke();
  fill(102);
}

void draw() {
  rect(x, 10, 2, 80);
  x = x + 1;
}</pre></td></tr>
<tr><td>void draw ( ) {...}</td><td>Se usa inmediatamente después de void setup( ), la función void draw( ) ejecuta continuamente las líneas de código que aparecen a continuación entre llaves {...}

La visualización se actualiza cada vez que se ejecutan todos los comandos entre llaves {...}, nunca antes.</td></tr>
</table>

Comando	*Uso*	*Ejemplos*

void draw ( ) {...}
notas interesantes
	Para detener el código entre llaves se puede usar varios comandos:
	noLoop( ), detiene el código en **void_draw( ){...}**
	
	redraw( ), hace que el código entre **{...}** se ejecute una sóla vez
	
	loop( ), hace que el código entre **{...}** vuelva a repetirse continuamente de nuevo.
	
	El número de veces que **void draw( )** se ejecuta por segundo se puede controlar con la función **frameRate( );**
	
	Es común utilizar **background( )** cerca del inicio de **draw( )** para limpiar el contenido de la ventana. Como los pixels dibujados en la ventana son acumulativos, omitir **background( )** puede dar resultados inesperados.
	
	void draw() sólo se puede usar una vez en el programa, y es necesario usarlo para procesar órdenes que necesitan a la fuerza que el código se este ejecutando continuamente, como eventos de ratón y teclado tales como **mousePressed()**.
	A veces es necesario ponerlo si se va a utilizar otros comandos que a la fuerza exigen que se incluyan **void setup** y **void draw**
	En ese caso se pondría vacío: **void draw ( ) { }**
		`float yPos = 0.0;`
		
		`void setup() {  // setup() runs once`
		`  size(200, 200);`
		`  frameRate(30);`
		`}`
		
		`void draw() {  // draw() loops forever, until stopped`
		`  background(204);`
		`  yPos = yPos - 1.0;`
		`  if (yPos < 0) {`
		`    yPos = height;`
		`  }`
		`  line(0, yPos, width, yPos);`
		`}`
		
		--- 	
		
		`void setup() {`
		`  size(200, 200);`
		`}`
		
		`// Although empty here, draw() is needed so`
		`// the sketch can process user input events`
		`// (mouse presses in this case).`
		`void draw() { }`
		
		`void mousePressed() {`
		`  line(mouseX, 10, mouseX, 90);`
		`}`

Comando	*Uso*	*Ejemplos*

if (...) {...}	Permite al programa tomar una **decisión** según cierta **condición**.
	
	La estructura es la siguiente:
	
	if (condición) {comandos a ejecutar si se cumple}
	
	Si no se cumple la condición se salta los comandos entre llaves {...}
		`for (int i = 5; i < 100; i = i+5) {`
		`  stroke(255);  // trazo de color blanco`
		`  if (i < 35) {  // Si i es menor de 35..`
		`  stroke(0);  //... trazo color negro`
		`  }`
		`  line(30, i, 80, i);`
		`}`

else {...}	Si se utiliza tiene que ser **combinada con la instrucción if (...) {...}**
	
	Sirve para extender la condición planteada en **if (...) {...}** añadiendo una serie de órdenes que serán las que se ejecuten **si la condición no se cumple**.
	
	La estructura debe ser así:
	
	if (condición) {comandos a ejecutar si se cumple}
	
	else {comandos a ejecutar si no se cumple}
	
	Se puede **anidar** comandos **if...else** unos dentro de otros como se ve en el ejemplo:
	
	if (condición_1) {comandos_1}
	
	else if (condición_2) {comandos_2}
	
	else {comandos_3 si no se cumple ninguna}
		`for (int i = 5; i < 95; i += 5) {`
		`  if (i < 35) {`
		`  line(30, i, 80, i);`
		`  }`
		`  else {`
		`  line(20, i, 90, i);`
		`  }`
		`}`
		
		--- 	
		
		`for (int i = 5; i < 95; i += 5) {`
		`  if (i < 35) {`
		`  line(30, i, 80, i);`
		`  }`
		`  else if (i < 65) {`
		`  line(20, i, 90, i);`
		`  }`
		`  else {`
		`  line(0, i, 100, i);`
		`  }`
		`}`