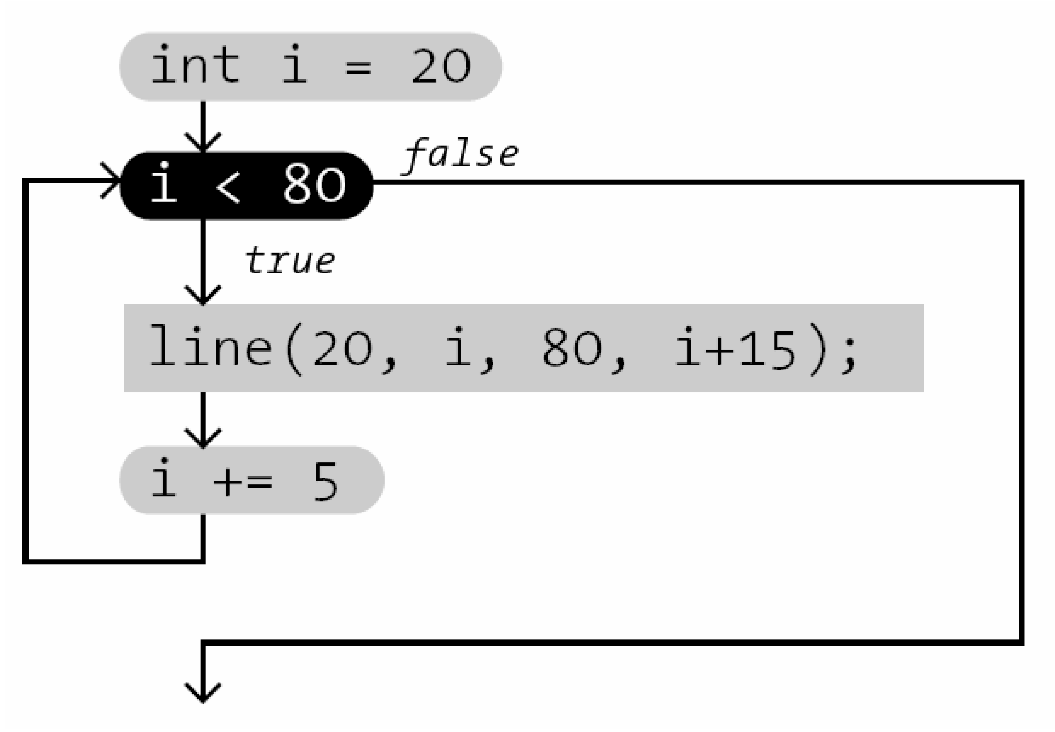
Processing Workshop

Till Nagel, IUAV, 10/2008

for loop

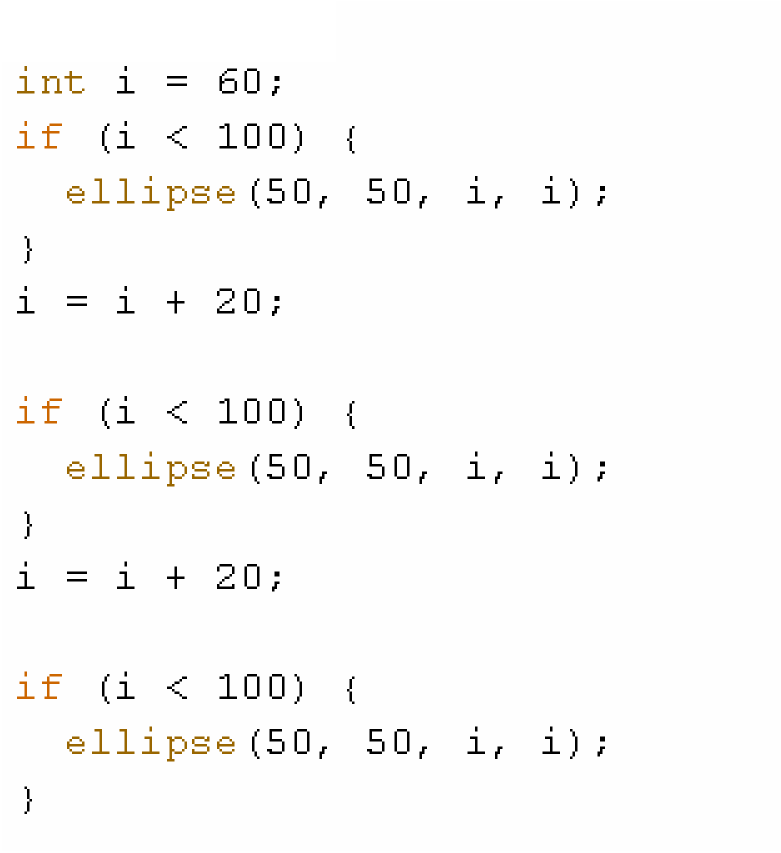
```
for (int i = 20; i < 80; i += 5) {
   line(20, i, 80, i + 15);
}
```



From: Casey Reas, Ben Fry, Processing, MIT Press, 2007

for loop

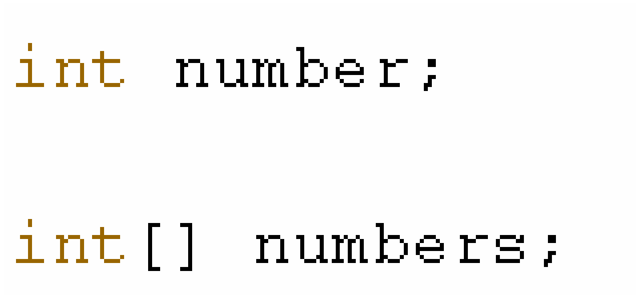
```
for (int i = 60; i < 100; i = i + 20) {
  ellipse(50, 50, i, i);
}
```



```
int i = 60;
if (i < 100) {
  ellipse(50, 50, i, i);
}
i = i + 20;

if (i < 100) {
  ellipse(50, 50, i, i);
}
i = i + 20;

if (i < 100) {
  ellipse(50, 50, i, i);
}
```

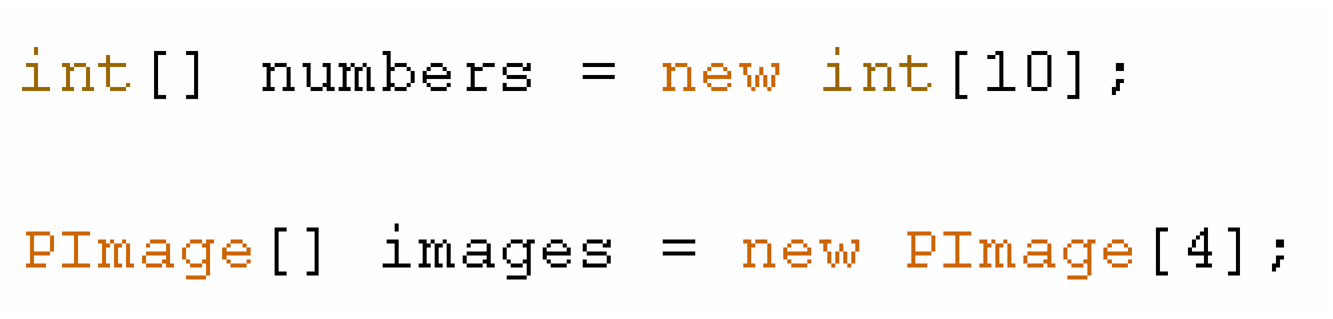
Arrays: Store multiple homogenous values

Arrays



```
int number;

int[] numbers;
```

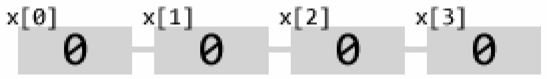
Arrays



```
int[] numbers = new int[10];

PImage[] images = new PImage[4];
```

Arrays: Declaration

```
int[] x = new int[4];
```



Arrays

Read and write access

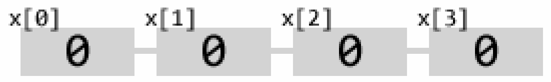
Access via an index

The index is positive and starts with zero

The index itself can be another variable

Arrays: Access

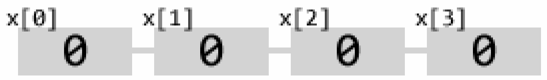
```
int[] x = new int[4];
```



Exercise: Draw an array

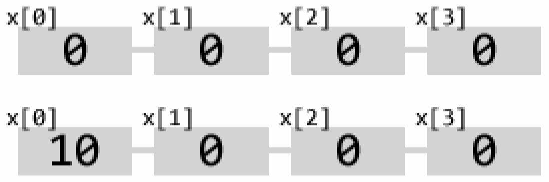
Arrays: Access

```
int[] x = new int[4];

x[0] = 10;

x[2] = -4;
```



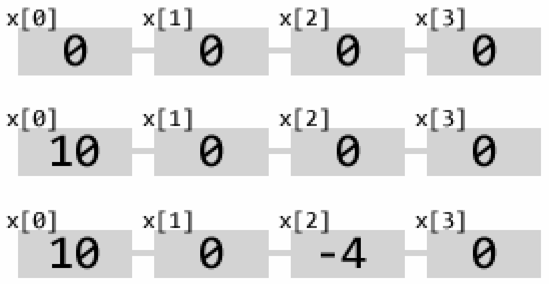
Arrays: Access

```
int[] x = new int[4];

x[0] = 10;

x[2] = -4;
```



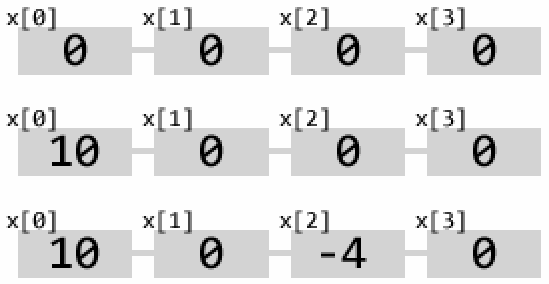
Arrays: Access

```
int[] x = new int[4];

x[0] = 10;

x[2] = -4;
```



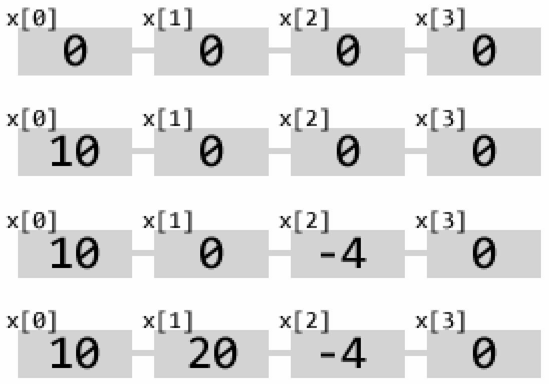
Arrays: Access

```
int[] x = new int[4];

x[0] = 10;

x[2] = -4;

x[1] = 2 * 10;
```



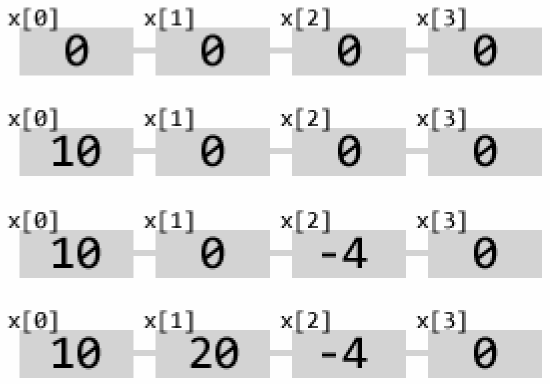
Arrays: Access

```
int[] x = new int[4];

x[0] = 10;

x[2] = -4;

x[1] = 2 * 10;
```



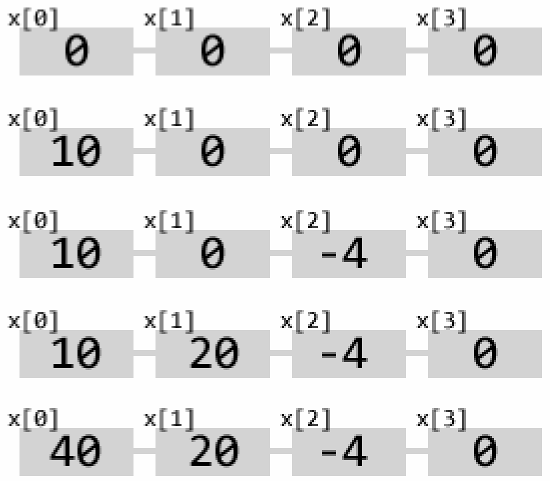
Arrays: Access

```
int[] x = new int[4];

x[0] = 10;

x[2] = -4;

x[1] = 2 * 10;

x[0] = x[1] * 2;
```



Arrays: Access

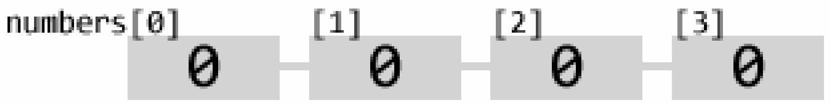
```
int[] x = new int[4];

x[0] = 10;

x[2] = -4;

x[1] = 2 * 10;

x[0] = x[1] * 2;
```


x[0] 0 x[1] 0 x[2] 0 x[3] 0
x[0] 10 x[1] 0 x[2] 0 x[3] 0
x[0] 10 x[1] 0 x[2] -4 x[3] 0
x[0] 10 x[1] 20 x[2] -4 x[3] 0
x[0] 40 x[1] 20 x[2] -4 x[3] 0

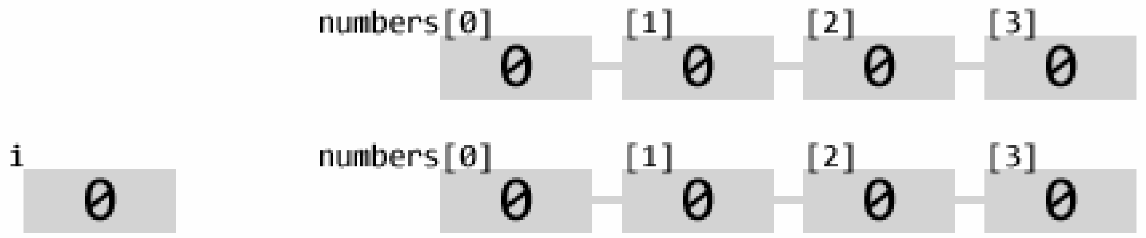
Arrays & Loops

```
int[] numbers = new int[3];
for (int i = 0; i < 3; i++) {
  numbers[i] = i * 10;
}
```

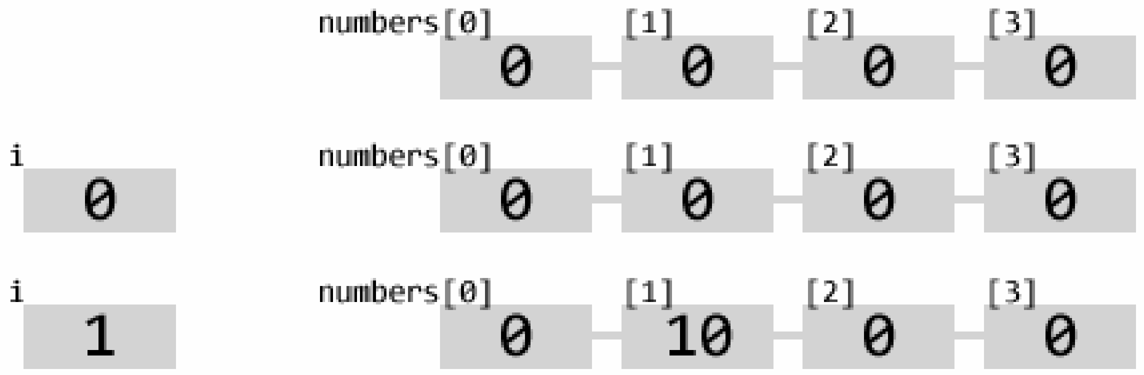
Arrays & Loops



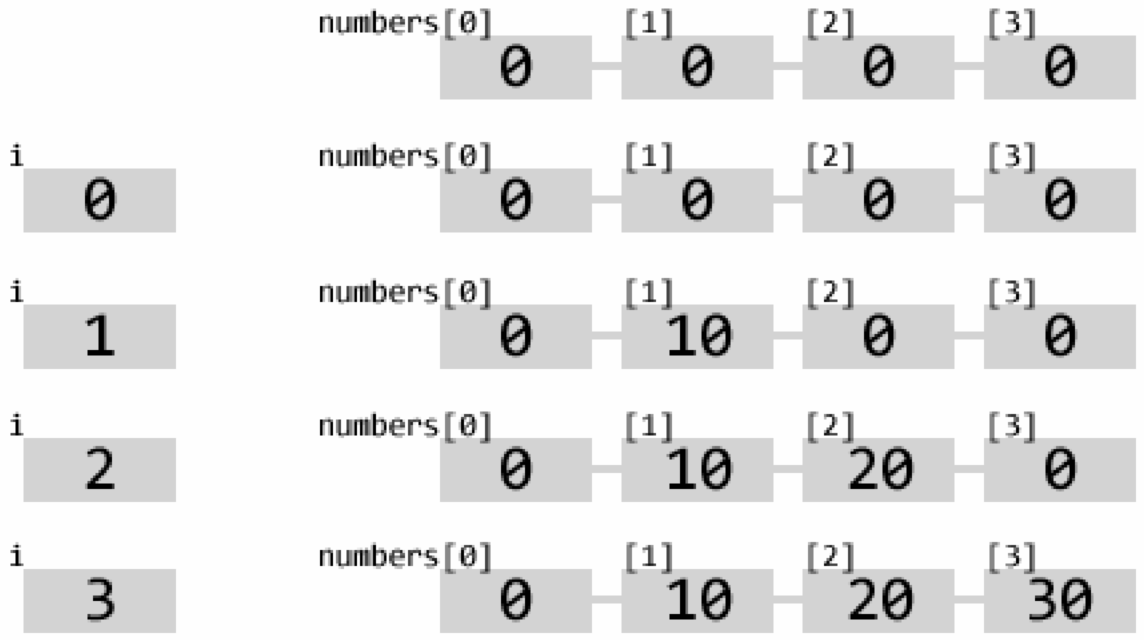
Arrays & Loops



Arrays & Loops



Arrays & Loops


numbers[0] 0 [1] 0 [2] 0 [3] 0
i 0 numbers[0] 0 [1] 0 [2] 0 [3] 0
i 1 numbers[0] 0 [1] 10 [2] 0 [3] 0
i 2 numbers[0] 0 [1] 10 [2] 20 [3] 0
i 3 numbers[0] 0 [1] 10 [2] 20 [3] 30

Arrays instead of copying variables

```
float x;
float y;
float v;

void setup() {
  size(200, 200);
  x = width/2;
  y = 50;
  v = 1;
}
```

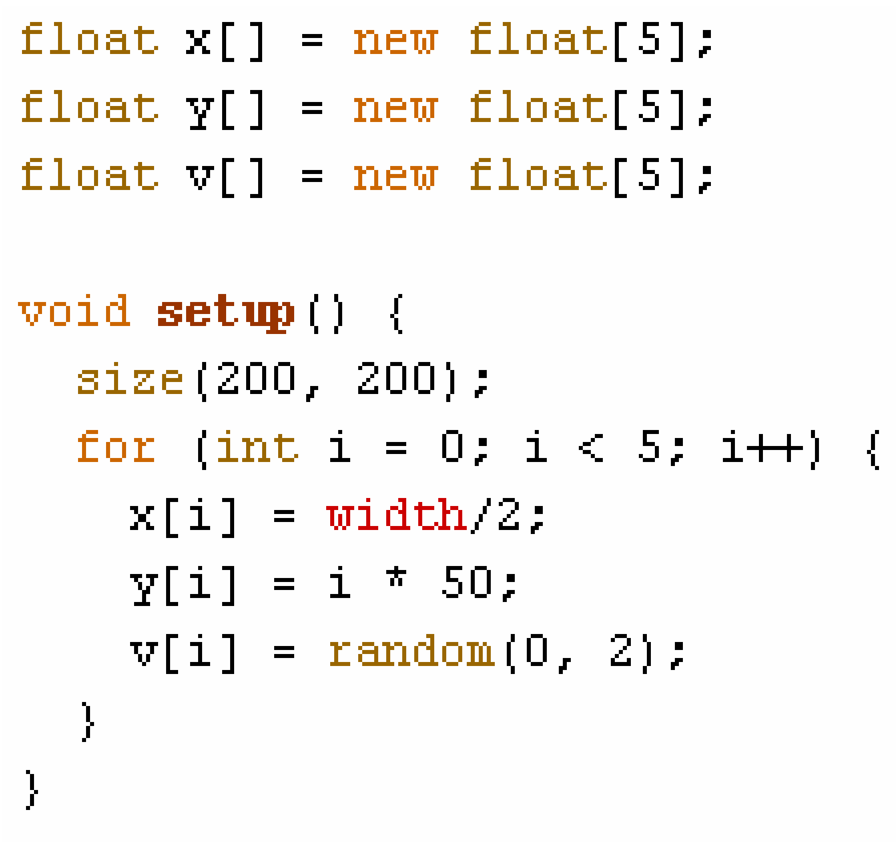
```
void draw() {
  background(0);
  ellipse(x, y, 10, 10);
  x = x + v;
}
```

Arrays instead of copying variables

```
float x;
float y;
float v;
float x2;
float y2;
float v2;

void setup() {
  size(200, 200);
  x = width/2;
  y = 50;
  v = 1;
  x2 = width/2;
  y2 = 150;
  v2 = 1;
}
```

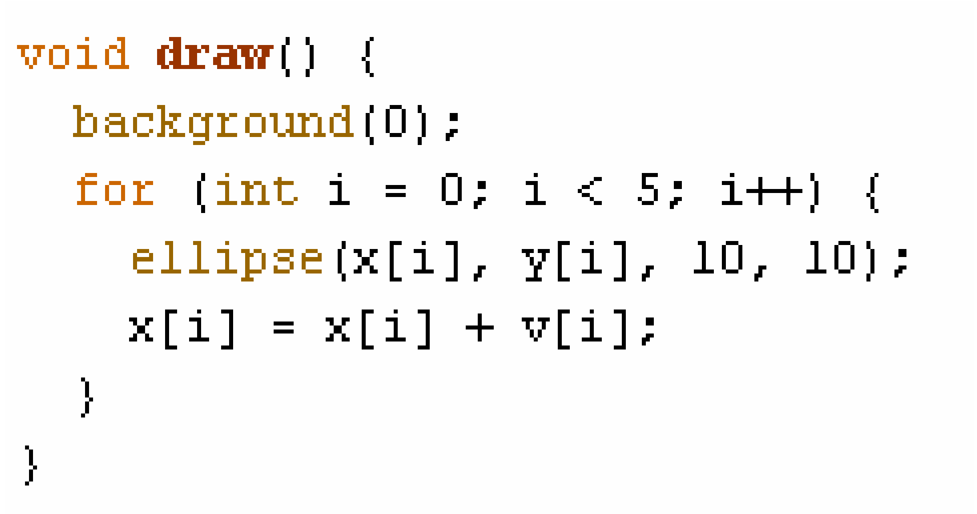
```
void draw() {
  background(0);
  ellipse(x, y, 10, 10);
  x = x + v;

  ellipse(x2, y2, 10, 10);
  x2 = x2 + v2;
}
```

Arrays instead of copying variables



```
float x[] = new float[5];
float y[] = new float[5];
float v[] = new float[5];

void setup() {
  size(200, 200);
  for (int i = 0; i < 5; i++) {
    x[i] = width/2;
    y[i] = i * 50;
    v[i] = random(0, 2);
  }
}
```

Arrays instead of copying variables



```
void draw() {
  background(0);
  for (int i = 0; i < 5; i++) {
    ellipse(x[i], y[i], 10, 10);
    x[i] = x[i] + v[i];
  }
}
```

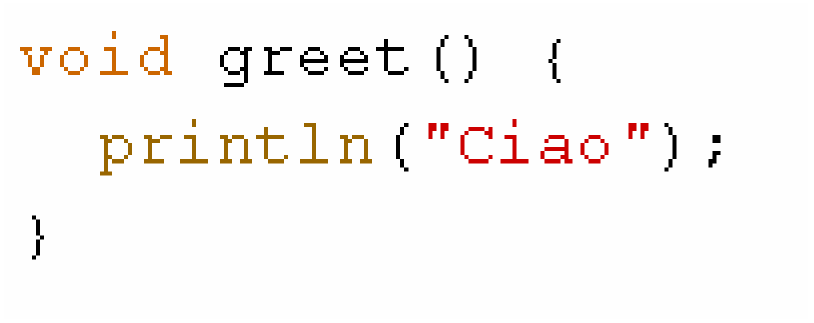
Functions

Functions

```
returnValue functionName( parameter ) {
   statements
}
```

Functions without parameter

```
void functionName() {
   statements
}
```



```
void greet() {
  println("Ciao");
}
```

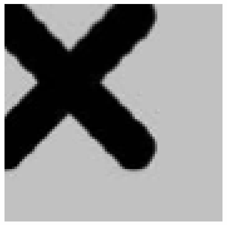
Functions without parameter

```
void setup() {
  size(400, 400);
  noStroke();
  greet();
}

void greet() {
  println("Ciao");
}
```

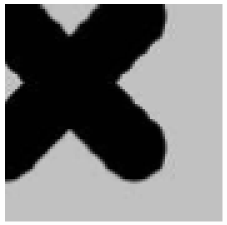
Functions without parameter

```
void functionName( parameter ) {
   statements
}
```

```
void setup() {
  size(100, 100);
  smooth();
  noLoop();
}

void draw() {
  drawX();
}

void drawX() {
  // Draw thick, light gray X
  stroke(160);
  strokeWeight(20);
  line(0, 5, 60, 65);
  line(60, 5, 0, 65);
}
```



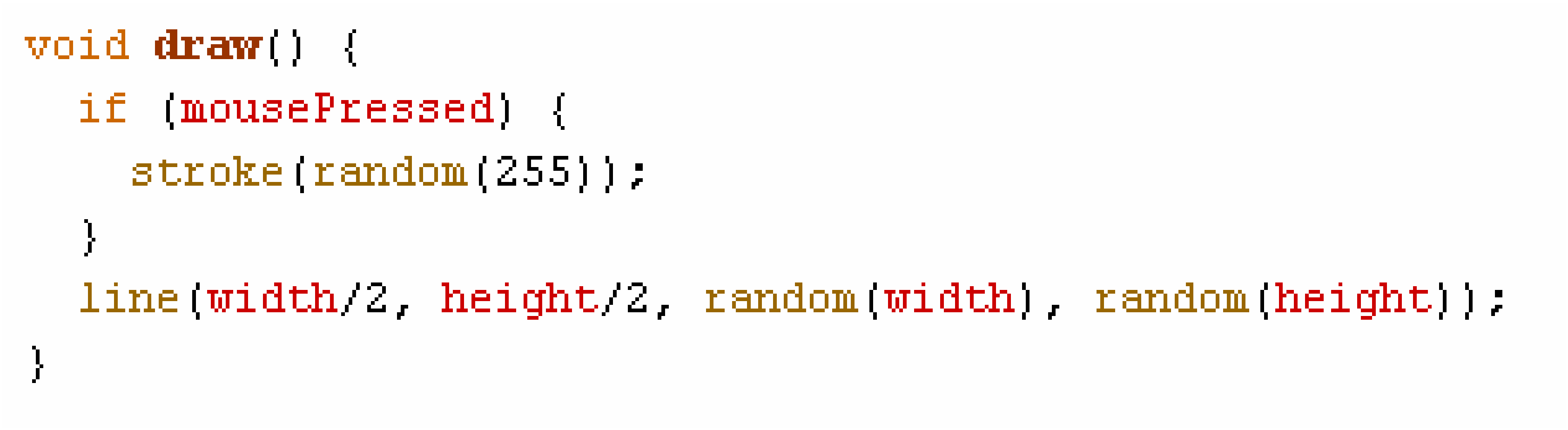
```
void setup() {
  size(100, 100);
  smooth();
  noLoop();
}

void draw() {
  drawX(0);
}

void drawX(int gray) {
  stroke(gray);
  strokeWeight(20);
  line(0, 5, 60, 65);
  line(60, 5, 0, 65);
}
```



```
void setup() {
  size(100, 100);
  smooth();
  noLoop();
}

void draw() {
  drawX(0, 30);
}

void drawX(int gray, int weight) {
  stroke(gray);
  strokeWeight(weight);
  line(0, 5, 60, 65);
  line(60, 5, 0, 65);
}
```

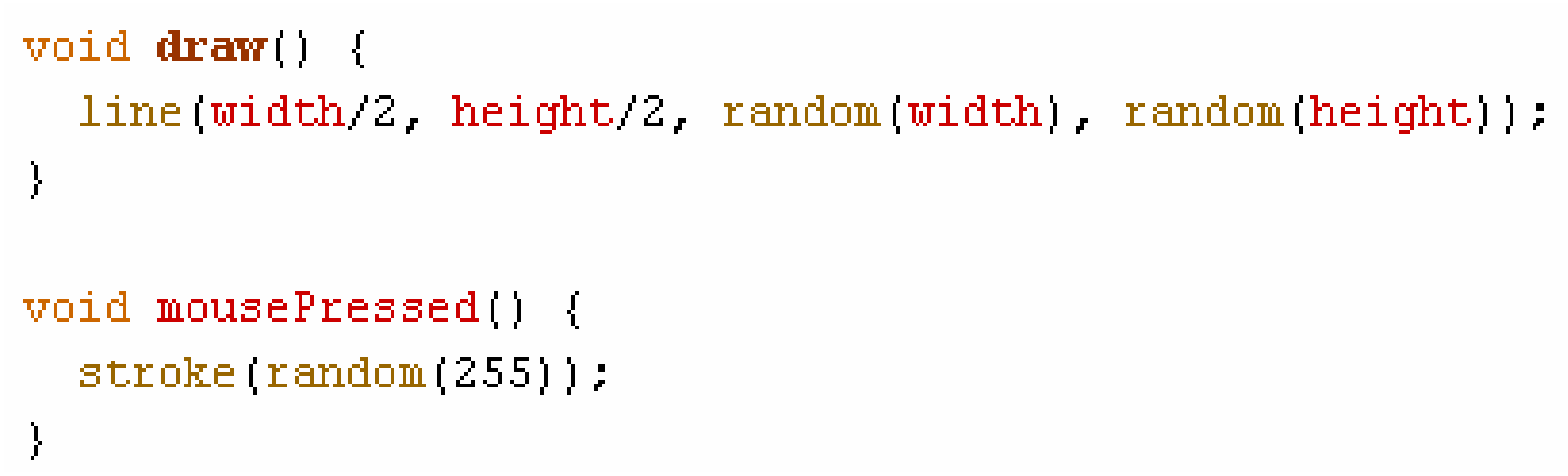
Example: FunctionExample

Mouse events



```
void draw() {
  if (mousePressed) {
    stroke(random(255));
  }
  line(width/2, height/2, random(width), random(height));
}
```

Mouse events: mousePressed()



```
void draw() {
  line(width/2, height/2, random(width), random(height));
}

void mousePressed() {
  stroke(random(255));
}
```

Other events: keyPressed()

Example: SlideSwitcherSimple

Exercises

E13: Create an interactive element, and use three different images or icons for the states normal, mouse-over and mouse-out.

Variation: Create an outer glow which responds to the proximity of the mouse pointer.
Lookup `dist()` in the Processing reference and use it.

Exercises

E15: Extend the program “AnimatedBall” to animate 20 differently coloured balls. (Hint: Look up the data type `color` in the Processing reference.)

Variation: Enhance to animate the balls in both horizontal and vertical directions.

E16: Create a function to draw a simple compound visual element. Draw it in the four corners of the stage, and at the current mouse position.