

Arrays

- A special kind of variable that holds not one, but many data items of a given type.
- Declared like variables, only the type is followed by a pair of square brackets.

```
float ax;           // Can hold one float
float[] ax;         // May hold many floats
```

- Can be initialized using a special syntax involving the `new` keyword, the type, and a size in brackets.

```
float ax = 12.3;           // Initialized to 12.3

float[] ax = new float[3]; // 3 floats, all 0.0
float[] ax = new float[] { 1.2, 2.3, 3.4 }; // Initialized
```

Step 1 Step 2 Step 3

Arrays

- Individual data items are accessed with an index and square brackets.

```
float sum = ax[0] + ax[1]
```

- **Indexes start at 0!**

- The length of an array can be determined using its `length` property.

```
println( ax.length );
```

- The length of an array is one greater than the last valid index.

- Arrays can be passed to, and returned from functions.

- ... just like other data types

```
void setup() {
  float[] a = new float[3];
  //float[] a = new float[] { 1.2, 2.3, 3.4 };

  for (int i=0; i<3; i++) {
    println( a[i] );
  }
}

void draw() {}
```

0	1	2
0.0	0.0	0.0

0	1	2
1.2	2.3	3.4

Built-in Array Functions

`append(array, item)`

- returns a new array expanded by one and add item to end

`expand(array, newSize)`

- returns a new array with size increased to newSize

`shorten(array)`

- returns a new array shortened by one

`concat(array1, array2)`

- returns a new array that is the concatenation of array1 and array2

`subset(array, offset [, length])`

- returns a subset of array starting at offset and proceeding for length (or end)

`splice(array, value/array2, index) or`

- returns a new array with value or array2 inserted at index

`sort(array)`

- returns a new array sorted numerically or alphabetically

`reverse(array)`

- returns a new array with all elements reversed in order

```
// arrays1
String[] names = new String[5];

void setup() {
  size(500, 500);
  background(200);
  names[0] = "Chococat";
  names[1] = "Cinnamoroll";
  names[2] = "Landry";
  names[3] = "Pekkle";
  names[4] = "Purin";
}

void draw() {
  fill(0);
  int n = names.length - 1;
  float x = random(width);
  float y = random(height);
  text( names[n], x, y );
}

void mousePressed() {
  names = shorten(names);
  background(200);
}
```

```
// bounce1

float ay = 0.2; // y acceleration (gravity)
float ax; // x position
float sy; // y position
float vx; // x velocity
float vy; // y velocity

void setup() {
  size(500, 500);
  fill(255, 0, 0);
  smooth();
  ellipseMode(CENTER);

  sx = random(0.0, width);
  sy = random(0.0, 10.0);
  vx = random(-3.0, 3.0);
  vy = random(0.0, 5.0);
}

void draw() {
  background(255);

  // Move ball
  sx += vx;
  sy += vy;
  vy += ay;

  // Bounce off walls and floor
  if (sx <= 10.0 || sx >= (width-10.0)) {
    vx = -vx;
  }

  if (sy >= (height-10.0) && vy > 0.0) {
    vy = -0.9*vy;
  }

  // Draw ball
  ellipse( sx, sy, 20, 20);
}
```

```
// bounce2

float ay = 0.2; // y acceleration (gravity)
float sx; // x position
float sy; // y position
float vx; // x velocity
float vy; // y velocity

float sx2; // x position
float sy2; // y position
float vx2; // x velocity
float vy2; // y velocity

void setup() {
  size(500, 500);
  fill(255, 0, 0);
  smooth();
  ellipseMode(CENTER);

  sx = random(0.0, width);
  sy = random(0.0, height);
  vx = random(-3.0, 3.0);
  vy = random(0.0, 5.0);

  sx2 = random(0.0, width);
  sy2 = random(0.0, height);
  vx2 = random(-3.0, 3.0);
  vy2 = random(0.0, 5.0);
}

void draw() {
  background(255);

  // Move ball
  sx += vx;
  sy += vy;
  vy += ay;

  // Bounce off walls and floor
  if (sx <= 10.0 || sx >= (width-10.0)) {
    vx = -vx;
  }
  if (sy >= (height-10.0) && vy > 0.0) {
    vy = -0.9*vy;
  }

  if (sx2 <= 10.0 || sx2 >= (width-10.0)) {
    vx2 = -vx2;
  }
  if (sy2 >= (height-10.0) && vy2 > 0.0) {
    vy2 = -0.9*vy2;
  }

  // Draw ball
  ellipse(sx, sy, 20, 20);
  ellipse(sx2, sy2, 20, 20);
}
```

```
// bounce3
int nBalls = 200;

float ay = 0.2; // y acceleration (gravity)
float[] sx = new float[nBalls]; // x position
float[] sy = new float[nBalls]; // y position
float[] vx = new float[nBalls]; // x velocity
float[] vy = new float[nBalls]; // y velocity

void setup() {
  size(500, 500);
  fill(255, 0, 0);
  smooth();
  ellipseMode(CENTER);

  for (int i=0; i<nBalls; i++) {
    // Move ball
    sx[i] = random(0.0, width);
    sy[i] = random(0.0, height);
    vx[i] = random(-3.0, 3.0);
    vy[i] = random(0.0, 5.0);

    // Bounce off walls and floor
    if (sx[i] <= 10.0 || sx[i] >= (width-10.0)) {
      vx[i] = -vx[i];
    }
    if (sy[i] >= (height-10.0) && vy[i] > 0.0) {
      vy[i] = -0.9*vy[i];
    }

    // Draw ball
    ellipse(sx[i], sy[i], 20, 20);
  }
}
```

bounce1 vs. bounce3

<pre>// bounce1 float ay = 0.2; // y acceleration float sx; // x position float sy; // y position float vx; // x velocity float vy; // y velocity void setup() { size(500, 500); fill(255, 0, 0); smooth(); ellipseMode(CENTER); sx = random(0.0, width); sy = random(0.0, height); vx = random(-3.0, 3.0); vy = random(0.0, 5.0); }</pre>	<pre>// bounce3 int nBalls = 200; float ay = 0.2; float[] sx = new float[nBalls]; float[] sy = new float[nBalls]; float[] vx = new float[nBalls]; float[] vy = new float[nBalls]; void setup() { size(500, 500); fill(255, 0, 0); smooth(); ellipseMode(CENTER); for (int i=0; i<nBalls; i++) { sx[i] = random(0.0, width); sy[i] = random(0.0, height); vx[i] = random(-3.0, 3.0); vy[i] = random(0.0, 5.0); } }</pre>
---	---

bounce1 vs. bounce3

<pre>// bounce1 void draw() { background(255); // Move ball sx += vx; sy += vy; vy += ay; // Bounce off walls and floor if (sx <= 10.0 sx >= (width-10.0)) { vx = -vx; } if (sy >= (height-10.0) && vy > 0.0) { vy = -0.9*vy; } // Draw ball ellipse(sx, sy, 20, 20); }</pre>	<pre>// bounce3 void draw() { background(255); for (int i=0; i<nBalls; i++) { // Move ball sx[i] += vx[i]; sy[i] += vy[i]; vy[i] += ay; // Bounce off walls and floor if (sx[i] <= 10.0 sx[i] >= (width-10.0)) { vx[i] = -vx[i]; } if (sy[i] >= (height-10.0) && vy[i] > 0.0) { vy[i] = -0.9*vy[i]; } // Draw ball ellipse(sx[i], sy[i], 20, 20); } }</pre>
--	---

Arrays can be passed as arguments to a function, and returned as results, just like scalar variables

```
// Add two arrays element by element
float[] addArrays(float[] a1, float[] a2)
{
  // Create new array sized appropriately
  float[] a3 = new float[ a1.length ];

  // Add array elements and store in new arrays
  for (int i=0; i<a1.length; i++)
  {
    a3[i] = a1[i] + a2[i];
  }

  return a3;
}
```

Using functions with array arguments and results

```
void setup() {
  // Create arrays
  float[] x1 = new float[10];
  float[] x2 = new float[10];

  // Fill arrays
  for (int i=0; i<10; i++) {
    x1[i] = i;
    x2[i] = 2*i;
  }

  // Add arrays
  float[] xAdd;
  xAdd = addArrays( x1, x2 );

  // Print sums
  for (int i=0; i<10; i++)
  {
    println(x1[i] + " + " + x2[i] + " = " + xAdd[i]);
  }
}
```

Arrays – Once again...

- Declared like variables, with type followed by square brackets.

```
float x;           // Can hold one float
float[] ax;        // May hold many floats
```

- Standard variables: (1) declare, (2) initialize.

```
float x;           // declare
x = 12.3;          // initialize (assign)
float x = 12.3;    // or both at the same time...
```

- Array variables: (1) declare, (2) size, (3) initialize.

```
float[] ax = new float[3];           // 3 floats, all 0.0
float[] ax = new float[] { 1.2, 2.3, 3.4 }; // Initialized
```

Step 1 Step 2 Step 3

- Using standard variables vs. arrays.

```
x = 255.0; random( x );
ax[0] = 127.0; ax[1] = 255.0;
random( ax[0], ax[1] );
```

Recall ... Images

loadImage(filename);

- Loads an image from a file in the *data* folder in sketch folder.
- Must be assigned to a variable of type PImage.

image(img, X, Y, [X2, Y2]);

- Draws the image *img* on the canvas at X, Y
- Optionally fits image into box X,Y and X2,Y2

imageMode(CORNER);

- X2 and Y2 define width and height.

imageMode(CORNERS);

- X2 and Y2 define opposite corner.

Image Example

```
imageExample
├── imageExample.pde
├── data
│   └── natura-morta.jpg
```

```
PImage img;

void setup()
{
    size(500, 400);
    img = loadImage("natura-morta.jpg");
    image(img, 50, 40);
}
```

loadImage is a function that reads image data from a file, stores it in a new PImage object, and returns the new PImage object.

The image function takes a variable of type PImage as its first argument and renders it on your sketch.

Object Oriented Programming

- Objects are software bundles that wrap up all semantically related variables and functions.
 - Object variables are called fields
 - Object functions are called methods
- Objects are said to Encapsulate (hide) its detail
 - How an object method is implemented is not important
 - What it does is important
- Objects can be created, named and referenced with variables
 - Very similar to standard data types
- An object's individual fields and methods are accessed using syntax called dot-notation

The PImage Object

- Fields**
 - width *image width*
 - height *image height*
 - pixels[] *1D array holding all image pixels*
- Methods**
 - loadPixels() *fill the pixels[] array with image pixels*
 - updatePixels() *copy pixels in pixels[] array back to image*
 - get(x, y) *reads a pixel at position x, y*
 - set(x, y, color) *set the color at position x, y*
 - save(path) *saves an image to a file*
 - ...
- Related Functions**
 - loadImage(path) *create a new PImage and init with image file*
 - createImage(w, h, form) *create a new empty Pimage object*
 - image(img, x, y) *draw a PImage to a sketch*

<http://processing.googlecode.com/svn/trunk/processing/build/javadoc/core/index.html>

Image Example

```
// imageExample2

PImage img;

void setup()
{
    size(500, 400);
    img = loadImage("natura-morta.jpg");
    image(img, 50, 40);
}

void mousePressed() {
    // Print the size of the PImage
    println(img.width);
    println(img.height);
}

void draw() {}
```

Dot-notation ...
To access the fields and methods within an object, join the object and field/method using a dot.

```
// imageExample3
PImage img;

void setup() {
  size(500, 400);
  img = loadImage("natura-morta.jpg");
  image(img, 50, 40);
}

void mousePressed() {
  // Fade the image to black
  float fade = 0.95;

  // Load pixel colors from image into array
  img.loadPixels();

  // Reduce value of each color component by fade
  for (int i = 0; i < img.pixels.length; i++) {
    color p = img.pixels[i];
    img.pixels[i] = color(fade*red(p), fade*green(p), fade*blue(p));
  }

  // Copy pixel colors back to array
  img.updatePixels();

  // Draw image to sketch
  image(img, 50, 40);
}

void draw() {}
```

Nearly identical to code used in vevents that continuously faded drawing.

The String Object

- Fields
 - ...
- Methods
 - equals(*anotherString*)
 - length()
 - substring()
 - toLowerCase()
 - toUpperCase()

<http://download.oracle.com/javase/1.4.2/docs/api/>

String Method Examples

```
String s;

s = "BrynMawr";
println(s);
println( s.length() );

println( s.substring(4) );
println( s.substring(3,7) );

println( s.toUpperCase() );
println( s.toLowerCase() );
```

```
BrynMawr
8
Mawr
nMaw
BRYNMAWR
brynmawr
```