**Arrays**

- **append(array, item)**: Appends item to the array.
- **arrayCopy(from, to)**: Copies from one array to another.
- **concat(a, b)**: Concatenates array b onto array a.
- **deleteElement(array, begin)**: Deletes elements from the array.
- **expand(array, moreElements)**: Extends the array by moreElements.
- **reverse(a)**: Reverses the elements of the array.
- **shorten(array)**: Shortens the array.

**Color**

- **blendColor(pi1, pi2, mode)**: Blends two colors using the specified mode.
- **color(c)**: Sets the current color to c.
- **color(gray)**: Sets the current color to (gray, gray, gray).
- **color(h, s, b)**: Sets the current color to (h, s, b) in HSB space.
- **color(r, g, b)**: Sets the current color to (r, g, b) in RGB space.
- **lerpColor(c1, c2, t)**: Linearly interpolates two colors.

**Math**

- **abs(f)**: Absolute value of f.
- **acos(c)**: Arc whose cosine is c (in radians).
- **asin(s)**: Arc whose sine is s (in radians).
- **atan(t)**: Arc whose tangent is t (in radians).
- **atan2(y, x)**: Arc whose tangent is y/x (in radians).
- **constrain(klow, high, f)**: Limits f to the range between low and high.
- **cos(r)**: Cosine of r radians.
- **degrees(r)**: Converts radians to degrees.
- **dist(v1, v2)**: Distance between two points.
- **exp(f)**: e^f.
- **float(i)**: Promotes an integer to a float.
- **lerp(low, high, t)**: Linearly interpolates two values.
- **log(f)**: Natural logarithm of f.
- **mag(v)**: Magnitude of a vector.
- **map(input, lowin, highin, lowout, highout)**: Linearly maps an input value to a different range.
- **max(f1, f2)**: Maximum of two numbers.
- **min(f1, f2)**: Minimum of two numbers.
- **pow(f, e)**: f^e.
- **random(low, high)**: Returns a random number between low and high.
- **randomSeed(s)**: Sets the random number sequence to a different pattern.

**Keyboard**

- **keyPressed()**: Gets called when a key is pressed.
- **keyReleased()**: Gets called when a key is released.

**Mouse**

- **mouseDragged()**: Gets called when the mouse has been moved with one or more buttons down.
- **mouseMoved()**: Gets called when the mouse has been moved with all buttons up.
- **mousePressed()**: Gets called when a mouse button is pressed.
- **mouseReleased()**: Gets called when a mouse button is released.

**Printing**

- **println(s)**: Prints a string to the console.
- **status(s)**: Prints a string to the status area.

**Randomness**

- **noise(f)**: Generates a random noise value.
- **noiseDetail(??)**: Sets the frequency of noise.
- **noiseSeed(??)**: Sets the seed for random noise.
- **random(low, high)**: Returns a random number within the given range.
- **randomSeed(s)**: Sets the seed for random numbers.

**Setup**

- **background(gray)**: Sets the background color to gray.
- **background(r, g, b)**: Sets the background color.
- **draw()**: Draws the scene.
- **exit()**: Exits the program.
- **frameRate(fr)**: Sets the refresh rate.
- **framerate(fr)**: Sets the refresh rate.
- **localStorage()**: Starts automatic calling of draw().
- **noLoop()**: Stops automatic calling of draw().
- **noSmooth()**: Sets the antialiasing to off.
- **print(s)**: Prints a string to the console.
- **println(s)**: Prints a string to the console, adding a return.

---

12/10/2012 10:23 AM
### Processing Functions

**Setup**
- `size( w, h )` Set the size of the graphics window to $w \times h$ pixels
- `AVG2D, P2D, 3D, OPENGL, PDF`
- `smooth()` Set the display antialiasing to on

**Shapes**
- `arc( cx, cy, rx, ry, th1, th2 )` Draw an arc with center $(cx, cy)$ with radii $(rx, ry)$, from angle $th1$ to angle $th2$
- `beginShape()` Begin an arbitrary shape
- `box( b )` Draw a 3D box with dimensions $b \times b \times b$ around the origin
- `box( l, w, h )` Draw a 3D box with dimensions $l \times w \times h$ around the origin
- `curve( x0, y0, x1, y1, x2, y2, x3, y3 )` Draw a Catmull-Rom curve
- `ellipse( cx, cy, w, h )` Draw an ellipse in CENTER mode
- `ellipse( cx, cy, x/2., y/2. )` Draw an ellipse in RADIUS mode
- `ellipse( ulx, uly, llx, lly )` Draw an ellipse in CORNERS mode
- `ellipse( ulx, uly, w, h )` Draw an ellipse in CORNER mode
- `ellipseMode( m )` CORNER, CORNERS, CENTER, RADIUS
- `endShape( )` End an arbitrary shape
- `line( x0, y0, x1, y1 )` Draw a line
- `point( x, y )` Put a dot at $(x, y)$
- `quad( x0, y0, x1, y1, x2, y2, x3, y3 )` Draw a quadrilateral
- `rect( cx, cy, w, h )` Draw a rectangle in CENTER mode
- `rect( cx, cy, x/2., y/2. )` Draw a rectangle in RADIUS mode
- `rect( ulx, uly, llx, lly )` Draw a rectangle in CORNERS mode
- `rect( ulx, uly, w, h )` Draw a rectangle in CORNER mode
- `sphere( r )` Draw a 3D sphere with radius $r$ around the origin
- `sphereDetail( slices, stacks )` Set the number of slices and stacks to use when drawing a sphere
- `triangle( x0, y0, x1, y1, x2, y2 )` Draw a triangle
- `vertex( x, y )` Specify a vertex in the outline of the arbitrary shape

**Text**
- `loadFont( font )` Load and assign the given font
- `text( s, x, y )` Draw the text "s" on the screen at $(x, y)$ with the current fill color
- `textFont( theFont )` Set theFont as the current font

**Textures**
- `loadImage( "filename" )` Read and assign an image file to a Pimage variable
- `texture( ?? )`
- `textureMode( ?? )`

**Time**
- `day( )` Current day of the month: 1 - 31
- `delay( ms )` Stops the program for $ms$ milliseconds
- `hour( )` Current hour of the day: 0 - 23
- `millis( )` # milliseconds since the program started
- `minute( )` Current minute of the hour: 0 - 59
- `second( )` Current second of the minute 0 - 59
- `year( )` Current year

**Transformations**
- `rotate( r )` Perform a 2D CW rotation of $r$ radians
- `rotateX( r )` Perform an X rotation of $r$ radians
- `rotateY( r )` Perform a Y rotation of $r$ radians
- `rotateZ( r )` Perform a Z rotation of $r$ radians
- `scale( sx, sy )` Scale by $(sx, sy)$
- `translate( tx, ty )` Translate by $(tx, ty)$

**Variables**
- `frameCount` Current frame number
- `HALF_PI` $\pi / 2$
- `height` Screen height in pixels
- `Infinity` $\infty$
- `key` The keyboard key that is pressed
- `keyCode` Modifier keys
- `keyPressed` true if a keyboard key has been pressed
- `mouseButton` LEFT, CENTER, RIGHT
- `mousePressed` true if a mouse button has been pressed
- `mouseX` The current mouse $x$ position
- `mouseY` The current mouse $y$ position
- `PI` $\pi$
- `QUARTER_PI` $\pi / 4$
- `TWO_PI` $2\pi$
- `width` Screen width in pixels