PROCESSING *CHEAT SHEET*

DATA TYPES

Primitive
boolean
byte
char
color
double
float
int
long

Composite
Array
ArrayList
HashMap
Object
String
XMLElement

Conversion
binary()
boolean()
byte()
char()
float()
hex()
int()
str()
unbinary()
unhex()

String Functions
join()
match()
matchAll()
nf()
nfc()
nfp()
nfs()
split()
splitTokens()
trim()

Array Functions
append()
arrayCopy()
concat()
expand()
reverse()
shorten()
sort()
splice()
subset()

Constants
HALF_PI
PI
QUARTER_PI
TWO_PI

Assign variables

=	assign value to a variable
;	statement terminator
,	separates parameters in function separates variables in declarations separates variables in array

```
/*** Assign variables ***/
//Format is in variable_type variable_name;
int total;
//Then you can assign a value to it later
total = 0;
//Or, assign a value to it at the same time
int total = 0;
//Note: use one of the primitive data types
on the left
```

Structure: program structure

setup()	defines initial enviroment properties, screen size, background before the draw()
draw()	called after setup() & executes code continuously inside its block until program is stopped or noLoop() is called.
size()	size() *must* be first line in setup() defines dimension of display in units of pixels
noLoop()	Stops Processing from executing code within draw() continuously

```
/*** Example ***/
void setup() {
  size(200, 200);
  background(0);
  fill(102);
}
void draw() {
  //Draw code here
 }
```

2D Primitives

Texto

point()	draws a point point(x, y) point(x, y, z)//3D
line()	draws a line line(x1, y1, x2, y2) line(x1, y1, z1, x2, y2, z2)//3D
rect()	draws a rectangle rect(x, y, width, height)
elipse()	Draws an elipse ellipse(x, y, width, height)
arc()	draws an arc arc(x, y, width, height, start, stop)

```
/*** Arc (portion of circle) ***/
//x & y = coords, width & height = size
//start + stop = starting and end points
(think angle in radians) of circle in π pie
LINK
arc(x, y, width, height, start, stop)
arc(100, 100, 50, 50, PI, 2*PI);//Sad Face
arc(100, 100, 50, 50, 0, PI);//Happy Face
//Note:  Play around with start and stop. Use
PIE constants or math operators PI/3 , .5*PI
```

Relational

==	equality
>	greater than
>=	greater than or equal to
!=	inequality
<=	less than or equal to

```
/*** Example ***/
if(total == 100){
  //Then do this
}
```

Iteration

while	executes statements while the expression is true
for	loop continues until the test evaluates to false

```
/*** while Example ***/
while(total < 100){
  total++; //adds 1 to total
}

/*** for Example ***/
for(int i=0; i<100; i++; ){
  //Do something here
}
```

Conditionals

if	if statement evaluates to true then execute code
else	extension of if statement executes if equals false
else if	extension of if statement executes if equals true

```
/*** if  / else / else if  ***/
if(total == 100){
  //total is equal to 100
}
else
if(total < 100){
  //total is smaller then 100
}
else{
  //total is bigger then 100
}
```

Coloring stuff

background()	sets background color in RGB or hexadecimal color
	background(value1, value2, value3)
	background(hexadecimal_value)

fill()	sets color for shape
	fill(value1, value2, value3)
	fill(hexadecimal_value)

stroke()	sets color for shape
	stroke(value1, value2, value3)
	stroke(hexadecimal_value)

```
/*** Example  ***/
//Note call fill or stroke before every shape you
are planning on using different colors on each
stroke(#CCCFFF);
fill(#FFFCCC);
rect(100,100,50,50);
```

CONTROL

Relational Operators
== (equality)
> (greater than)
>= (greater than or equal to)
!= (inequality)
< (less than)
<= (less than or equal to)

Iteration
for
while

Conditionals
break
case
?: (conditional)
continue
default
else
if
switch()

Logical Operators
&& (logical AND)
! (logical NOT)
|| (logical OR)