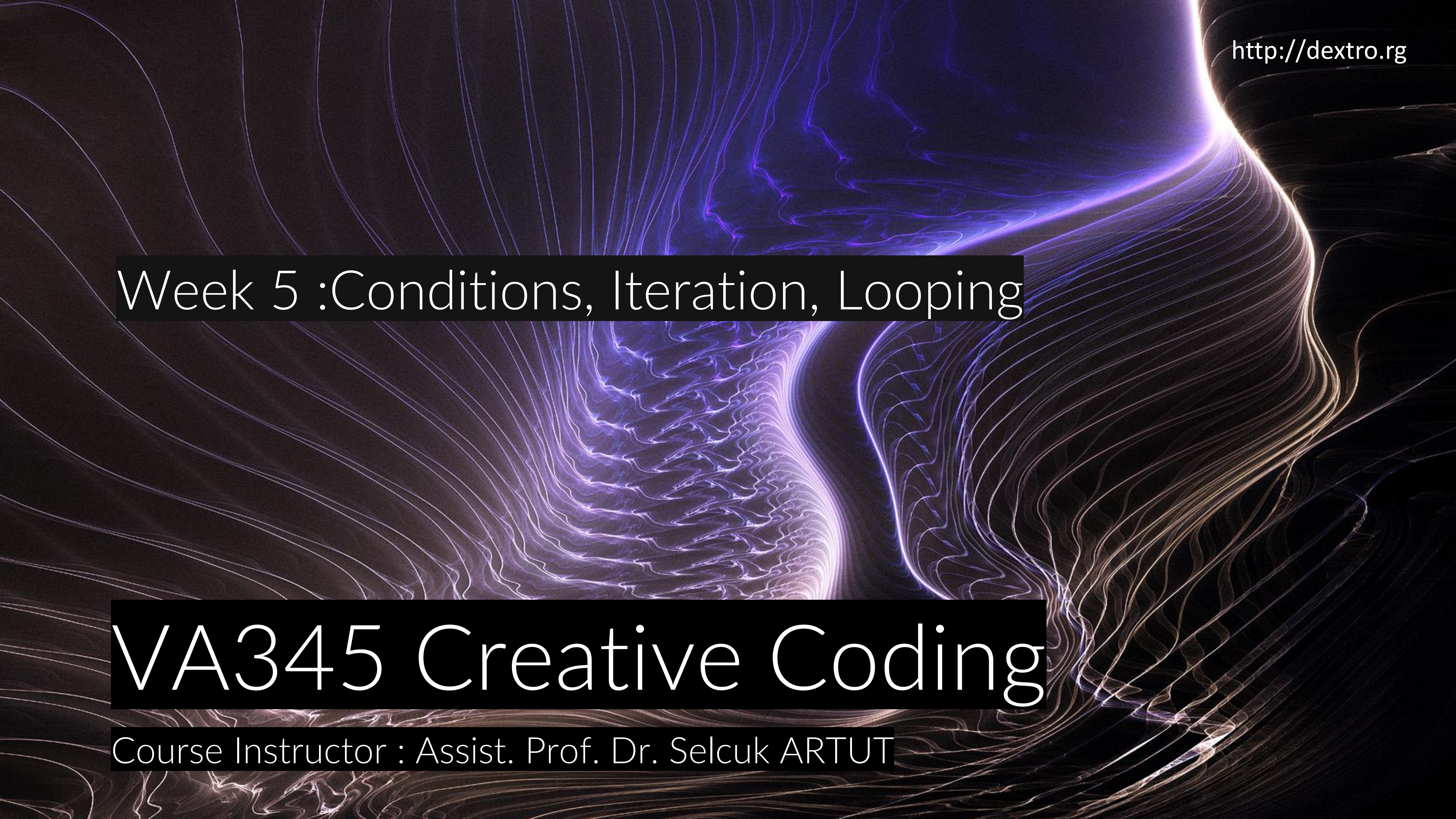


http://dextro.rg

Week 5 :Conditions, Iteration, Looping

VA345 Creative Coding

Course Instructor : Assist. Prof. Dr. Selcuk ARTUT

Sougwen Chung

Sougwen Chung (愫君) is a Chinese-born, Canadian-raised artist based in New York

http://sougwen.com/

Whitespace

In many programming languages, including Processing, there can be an arbitrary amount of space between the elements of a program. Unlike the rigorous syntax of statement terminators, spacing does not matter. The following two lines of code are a standard way of writing a program:

```
size(200, 200);
background(102);
```

However, the whitespace between the code elements can be set to any arbitrary amount and the program will run exactly the same way:

```
size
( 200,
200) ;
background ( 102)
;
```

Console

When software runs, the computer performs operations at a rate too fast to perceive with human eyes.

Because it is important to understand what is happening inside the machine, the functions print() and println() can be used to display data while a program is running.

The console can be used to display a variable, confirm an event, or check incoming data from an external device.

```
println("Hello World");
```

Console

```
// To print the value of a variable, rather than its name,
// don’t put the name of the variable in quotes
int x = 20;
println(x); // Prints "20" to the console
// While println() moves to the next line after the text
// is output, print() does not
print("10");
println("20"); // Prints "1020" to the console
println("30"); // Prints "30" to the console
// The "+" operator can be used for combining multiple text
// elements into one line
int x2 = 20;
int y2 = 80;
println(x2 + " : " + y2); // Prints "20 : 80" to the message window
```

Variable Scope

The rule for variable scope is stated simply: variables declared inside any block can be accessed only inside their own block and inside any blocks enclosed within.

Variables declared at the base level of the program—the same level as setup() and draw()—can be accessed everywhere within the program.

Variables declared within setup() can be accessed only within the setup() block.

Variables declared within draw() can be accessed only within the draw() block.

The scope of a variable declared within a block, called a local variable, extends only to the end of the block

Week 5 :Conditions, Iteration, Looping

Conditionals

Conditionals allow a program to make decisions about which lines of code run and which do not. They let actions take place only when a specific condition is met.

Conditionals allow a program to behave differently depending on the values of their variables. For example, the program may draw a line or an ellipse depending on the value of a variable. The if structure is used in Processing to make these decisions:

```
if (test) {
statements
}
```

Week 5 :Conditions, Iteration, Looping

Conditions

Example:

```
if(aNumber == 3){
        fill(255,0,0);
        ellipse(50,50,80,80);
}
```

The two lines of code inside the curly brackets are executed only when the condition is met –i.e. when the value of the variable **aNumber** is **3**.

Conditions

Example:

```
if(aNumber == 3){
        fill(255,0,0);
        ellipse(50,50,80,80);
}else{
        fill(0,255,0);
        ellipse(50,50,80,80);
}
```

With else, the if condition is expanded by one code snippet that is executed when the condition is not met.

Conditions

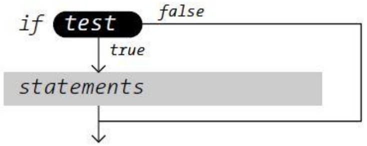
Example:

```
if(aNumber == 3) fill(255,0,0);
else fill(0,255,0);
```

If a code snippet consists of only one line, the curly brackets can be eliminated

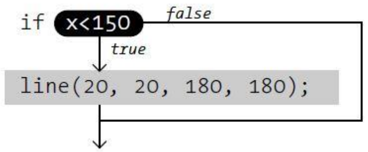
General case if structure

```
if (test) {
  statements
}
```



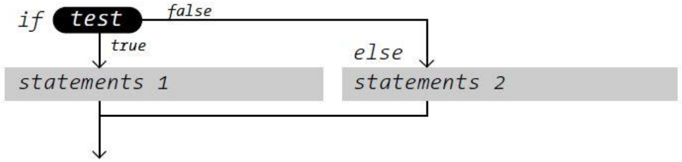
A specific if structure

```
if (x < 150) {
  line(20, 20, 180, 180);
}
```



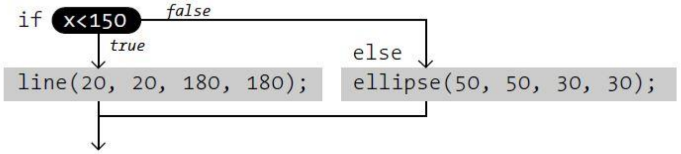
General case if/else structure

```
if (test) {
  statements 1
} else {
  statements 2
}
```



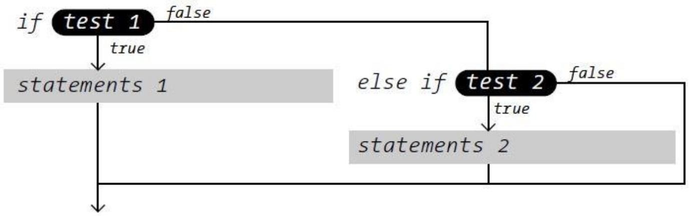
A specific if/else structure

```
if (x < 150) {
  line(20, 20, 180, 180);
} else {
  ellipse(50, 50, 30, 30);
}
```



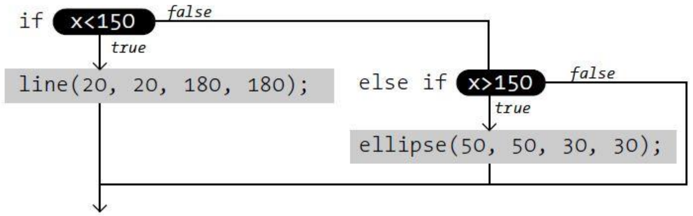
General case if/else if structure

```
if (test 1) {
  statements 1
} else if (test 2) {
  statements 2
}
```



A specific if/else if structure

```
if (x < 150) {
  line(20, 20, 180, 180);
} else if (x > 150) {
  ellipse(50, 50, 30, 30);
}
```



Logical operators

Logical operators are used to combine two or more relational expressions and to invert logical values. They allow for more than one condition to be considered simultaneously.

The logical operators are symbols for the logical concepts of AND, OR, and NOT:

Operator	Meaning
&&	AND
||	OR
!	NOT

The following table outlines all possible combinations and the results.

Expression	Evaluation
true && true	true
true && false	false
false && false	false
true || true	true
true || false	true
false || false	false
!true	false
!false	true

Let’s move things here and there

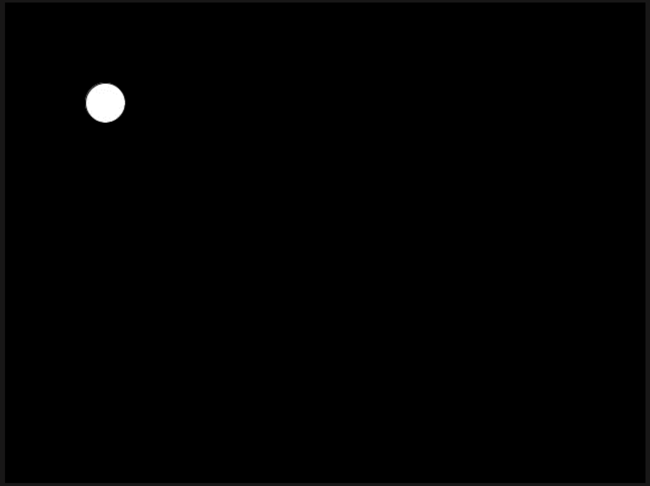
To put a shape into motion, use a variable to change its attributes.

```
int x = 100;
void setup(){
        size(640,480);
}
void draw(){
        ellipse(x,100,40,40);
        x = x + 1;
}
```

Coding Challenge (30 minutes)

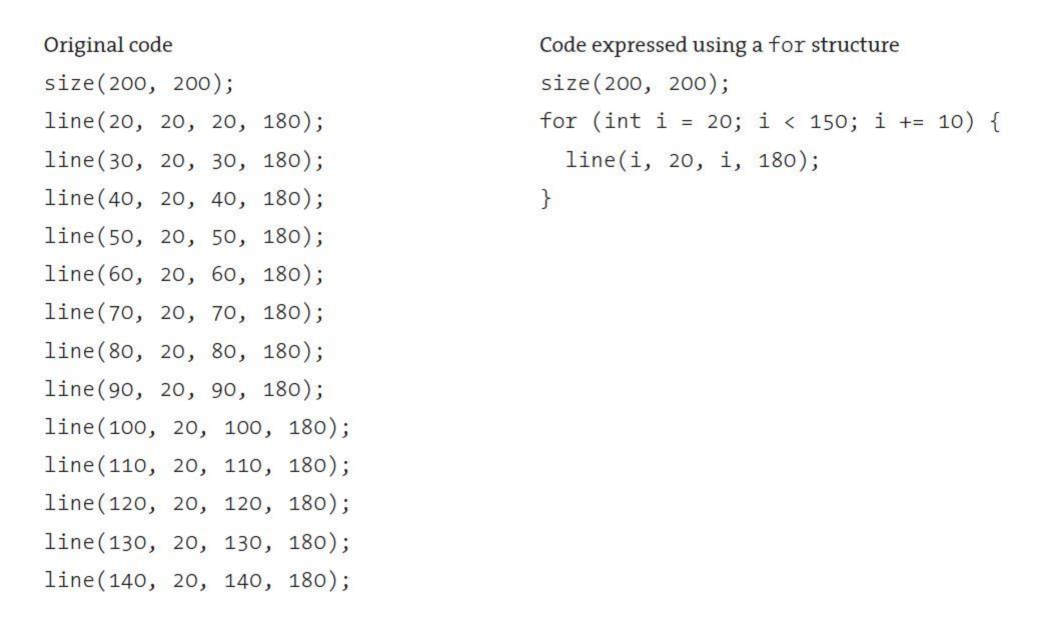
Accomplish this animaton.

Ambition Challenge : Use XY axis



Iteration

Iterative structures are used to compact lengthy lines of repetitive code. Decreasing the length of the code can make programs easier to manage and can also help to reduce errors.



Original code

```
size(200, 200);
line(20, 20, 20, 180);
line(30, 20, 30, 180);
line(40, 20, 40, 180);
line(50, 20, 50, 180);
line(60, 20, 60, 180);
line(70, 20, 70, 180);
line(80, 20, 80, 180);
line(90, 20, 90, 180);
line(100, 20, 100, 180);
line(110, 20, 110, 180);
line(120, 20, 120, 180);
line(130, 20, 130, 180);
line(140, 20, 140, 180);
```

Code expressed using a `for` structure

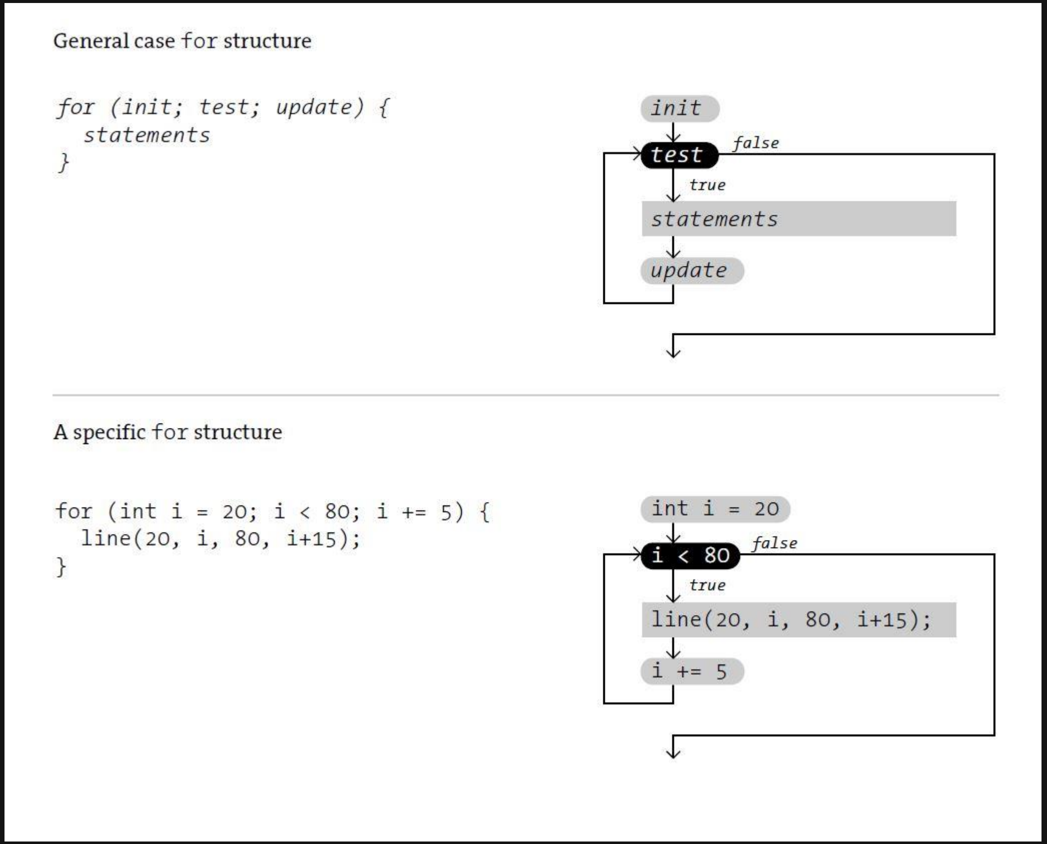
```
size(200, 200);
for (int i = 20; i < 150; i += 10) {
  line(i, 20, i, 180);
}
```

Loops : for loop

Example:

```
for (int i = 0; i<=5; i++){
        line(0,0,i*20,100);
        line(100,0,i*20,100);
}
```

The two lines of code inside the curly brackets are executed exactly six times. First the variable i is set to the value 0, then increased by 1 (i++) after each cycle, as long as the value is 5 or less.



General case for structure

```
for (init; test; update) {
  statements
}
```

A specific for structure

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

1. The *init* statement is run
2. The *test* is evaluated to *true* or *false*
3. If the *test* is *true*, continue to step 4. If the *test* is *false*, jump to step 6
4. Run the statements within the block
5. Run the *update* statement and jump to step 2
6. Exit the structure and continue running the program

Loops : while loop

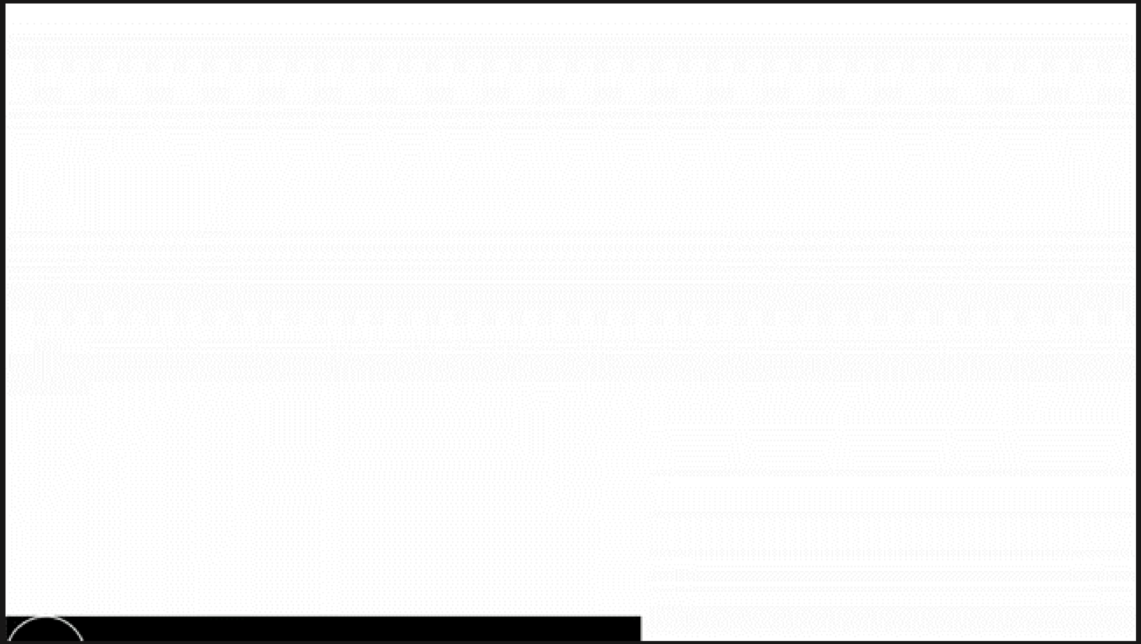
Example:

```
float myValue = 0;
while (myValue < 100){
        myValue = myValue + random(5)
}
```

The two lines of code inside the curly brackets are executed exactly six times. First the variable i is set to the value 0, then increased by 1 (i++) after each cycle, as long as the value is 5 or less.

Let’s make Islamic Patterns

1. Draw a circle 50 px in diameter
2. Draw a square around the circle
3. Draw a circle half the size of the original circle around the bottom corner of the square
4. Repeat steps 1-3 to the immediate right of their original position
5. Repeat step 4 six times
6. Repeat steps 1-3 immediately below their original position
7. Repeat steps 4-5
8. Repeat steps 6-7 six times



Mouse events

The mouse event functions are mousePressed(), mouseReleased(), mouseMoved(), and mouseDragged():

mousePressed() Code inside this block is run one time when a mouse button is pressed

mouseReleased() Code inside this block is run one time when a mouse button is released

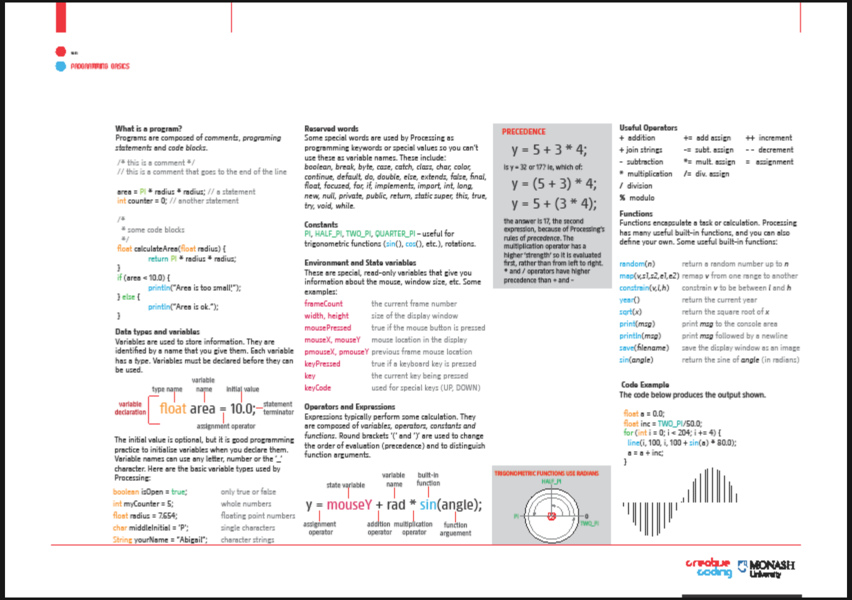
mouseMoved() Code inside this block is run one time when the mouse is moved

mouseDragged() Code inside this block is run one time when the mouse is moved while a mouse button is pressed

mouseX : https://processing.org/reference/mouseX.html

mouseY : https://processing.org/reference/mouseY.html

Refer to foldouts under SUCourse Week 5 about Programming & Interactions



PROGRAMMING BASICS

What is a program?
Programs are composed of *comments, programing statements* and *code blocks.*

```
/* this is a comment */
// this is a comment that goes to the end of the line

area = PI * radius * radius; // a statement
int counter = 0; // another statement

/*
 * some code blocks
 */
float calculateArea(float radius) {
        return PI * radius * radius;
}
if (area < 10.0) {
        println("Area is too small!");
} else {
        println("Area is ok.");
}
```

Data types and variables
Variables are used to store information. They are identified by a name that you give them. Each variable has a *type*. Variables must be declared before they can be used.

```
float area = 10.0;
```

The initial value is optional, but it is good programming practice to initialise variables when you declare them. Variable names can use any letter, number or the '_' character. Here are the basic variable types used by Processing:

boolean isOpen = true;	only true or false
int myCounter = 5;	whole numbers
float radius = 7.654;	floating point numbers
char middleInitial = 'P';	single characters
String yourName = "Abigail";	character strings

Reserved words
Some special words are used by Processing as programming keywords or special values so you can't use these as variable names. These include: *boolean, break, byte, case, catch, class, char, color, continue, default, do, double, else, extends, false, final, float, focused, for, if, implements, import, int, long, new, null, private, public, return, static super, this, true, try, void, while.*

Constants
PI, HALF_PI, TWO_PI, QUARTER_PI – useful for trigonometric functions (sin(), cos(), etc.), rotations.

Environment and State variables
These are special, read-only variables that give you information about the mouse, window size, etc. Some examples:

frameCount	the current frame number
width, height	size of the display window
mousePressed	true if the mouse button is pressed
mouseX, mouseY	mouse location in the display
pmouseX, pmouseY	previous frame mouse location
keyPressed	true if a keyboard key is pressed
key	the current key being pressed
keyCode	used for special keys (UP, DOWN)

Operators and Expressions
Expressions typically perform some calculation. They are composed of *variables, operators, constants and functions.* Round brackets '(' and ')' are used to change the order of evaluation (precedence) and to distinguish function arguments.

```
y = mouseY + rad * sin(angle);
```

PRECEDENCE

```
y = 5 + 3 * 4;
```

Is y = 32 or 17? Ie, which of:

```
y = (5 + 3) * 4;
y = 5 + (3 * 4);
```

the answer is 17, the second expression, because of Processing's rules of *precedence*. The multiplication operator has a higher 'strength' so it is evaluated first, rather than from left to right. * and / operators have higher precedence than + and –

TRIGONOMETRIC FUNCTIONS USE RADIANS

Useful Operators

+ addition	+= add assign	++ increment
+ join strings	-= subt. assign	- - decrement
- subtraction	*= mult. assign	= assignment
* multiplication	/= div. assign	
/ division		
% modulo		

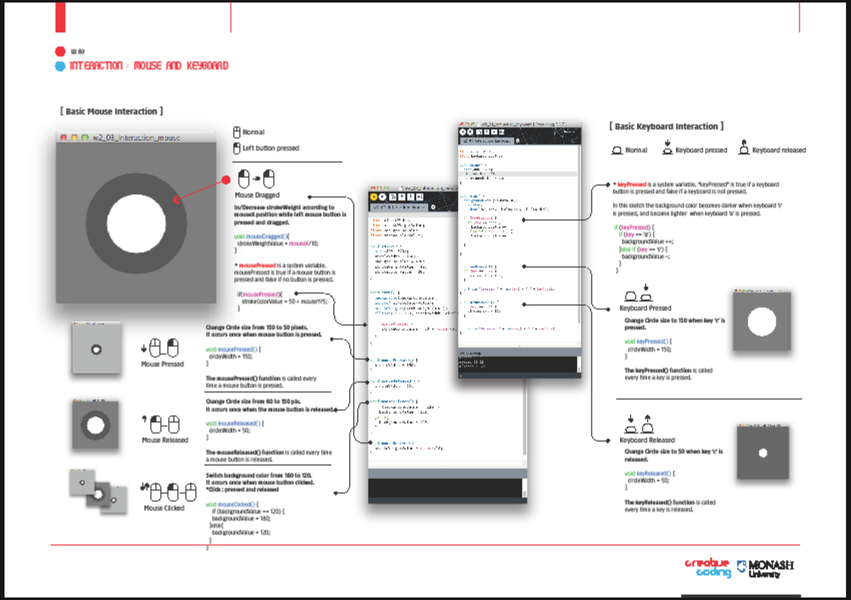
Functions
Functions encapsulate a task or calculation. Processing has many useful built-in functions, and you can also define your own. Some useful built-in functions:

random(*n*)	return a random number up to *n*
map(*v,s1,s2,e1,e2*)	remap *v* from one range to another
constrain(*v,l,h*)	constrain *v* to be between *l* and *h*
year()	return the current year
sqrt(*x*)	return the square root of *x*
print(*msg*)	print *msg* to the console area
println(*msg*)	print *msg* followed by a newline
save(*filename*)	save the display window as an image
sin(*angle*)	return the sine of *angle* (in radians)

Code Example
The code below produces the output shown.

```
float a = 0.0;
float inc = TWO_PI/50.0;
for (int i = 0; i < 204; i += 4) {
  line(i, 100, i, 100 + sin(a) * 80.0);
  a = a + inc;
}
```

creative coding MONASH University



W.02

INTERACTION : MOUSE AND KEYBOARD

[Basic Mouse Interaction]

Mouse Dragged

In/Decrease strokeWeight according to mouseX position while left mouse button is pressed and dragged.

```
void mouseDragged(){
  strokeWeightValue = mouseX/10;
}
```

* mousePressed is a system variable. mousePressed is true if a mouse button is pressed and false if no button is pressed.

```
if(mousePressed){
  strokeColorValue = 50 + mouseY/5;
}
```

Mouse Pressed

Change Circle size from 150 to 50 pixels. It occurs once when mouse button is pressed.

```
void mousePressed() {
  circleWidth = 150;
}
```

The mousePressed() function is called every time a mouse button is pressed.

Mouse Released

Change Circle size from 60 to 150 pix. It occurs once when the mouse button is released.

```
void mouseReleased() {
  circleWidth = 50;
}
```

The mouseReleased() function is called every time a mouse button is released.

Mouse Clicked

Switch background color from 180 to 120. It occurs once when mouse button clicked. *Click : pressed and released

```
void mouseClicked() {
  if (backgroundValue == 120) {
   backgroundValue = 180;
  }else{
   backgroundValue = 120;
  }
}
```

[Basic Keyboard Interaction]

* keyPressed is a system variable. "keyPressed" is true if a keyboard button is pressed and false if a keyboard is not pressed.

In this sketch the background color becomes darker when keyboard 'b' is pressed, and become lighter when keyboard 'a' is pressed.

```
if (keyPressed) {
  if (key == 'a') {
   backgroundValue ++;
  }else if (key == 'b') {
   backgroundValue --;
  }
}
```

Keyboard Pressed

Change Circle size to 150 when key 'c' is pressed.

```
void keyPressed() {
  circleWidth = 150;
}
```

The keyPressed() function is called every time a key is pressed.

Keyboard Released

Change Circle size to 50 when key 'c' is released.

```
void keyReleased() {
  circleWidth = 50;
}
```

The keyReleased() function is called every time a key is released.

creative coding MONASH University

Assignment 003 & Free Coding Time (40 minutes)

Use loop iterations to create complex shape visuals.

Announcement

Midterm1 is arriving October 1st

be prepared!!!

- Multiple choices
- Code challenges
- Theoretical questions