MusiComputation
Assignment Operators

• Given the following:
  
  \[
  x = 2; \\
  x = x + 1; \\
  \text{println} \ ("x: \ " + x); \\
  \]

• There are actually several ways to rewrite this more concisely.
Short Cut Operator

• One option is to use the += operator
  
x = 2;
x += 1;  // same as x = x + 1;
println ("x: " + x);

• There are similar operators for *, -, /, %
  
  - x = x * 5 is equivalent to x *= 5;
  - x = x - 5; is equivalent to x -= 5;
  - x = x / 5; is equivalent to x /= 5;
  - x = x % 5; is equivalent to x %= 5;

• Good Practice: place a space before and after your short cut operators.
Increment Operator

• A second option is to use an increment operator:
  
  \[ x++ \quad \text{Post-Increment Operator} \]
  
  \[ ++x \quad \text{Pre-Increment Operator} \]

• Both operators will increment \( x \) by 1, but they do have subtle differences.
Pre v. Post Increment

• PostIncrement Operator (x++):
  – use the current value of x in the expression. Then, increment by 1.

• PreIncrement Operator (++x):
  – Increment x by 1. Then, use the new value of x in the expression.
How about a real example?

```java
int c = 5;
println (c);
println (c++);
println (c);
println();
c = 5;
println (c);
println (++c);
println (c);
```

Output:

```
5
5 6
5 6
6
```

* Works for decrement operator too (--)
While Loops

• While Loop: Keep repeating an action while some condition remains true.

• Examples:
  – Every Stairmaster Machine contains a while loop (end condition is based on mode used).
    • while the person is still climbing, keep displaying the status, e.g. number of stairs climbed, calories burned, etc.
  – Keep prompting for book orders until the user is done.
while loop (continued)

• For example (in pseudocode)

while (some Boolean expression is true)
{
    do this (again and again...)
}
Parts of a While Loop

• Every while loop will always contain three main elements:
  1) **Priming**: initialize your variables.
  2) **Testing**: test against some known condition.
  3) **Updating**: updates part (or all) of the expression that is tested.
Simple While Loop

```java
int index = 1;
while (index <= 10) {
    println("Index: "+ index);
    index++;
}
```

1. **Priming**
2. **Test Condition**
3. **Update:** In this case, you can use either the pre or post increment operator.

Index: 1
Index: 2
Index: 3
...
Index: 8
Index: 9
Index: 10
While Loop Flowchart

1. Priming
   Set index=1

2. Test
   index <= 10
   TRUE
   3. Print value of
      index
      Update index++
   FALSE

   1. Priming
      Set index=1

   2. Test
      index <= 10
      TRUE
      3. Print value of
         index
         Update index++
      FALSE
Infinite Loop

• Infinite Loop: A loop that never ends. 
  – Generally, you want to avoid these!
  – There are special cases, however, when you do want to create infinite loops on purpose.
Infinite Loop Example #1

```java
int index = 1;
while (index <= 10)
{
    println ("Index:  " + index);
}
```

Here, I have deleted part 3:

The update statement (index++).
Infinite Loop, Example #2

int index = 1;
while (index >= 0) {
    println ("Index:  " + index);
    index++;  \(\text{Here, I have changed}\)
}
\(\text{Part 2: the test condition.}\)
Parts of a Loop (reminder)

- Every loop will always contain three main elements:
  - **Priming**: initialize your variables.
  - **Testing**: test against some known condition.
  - **Updating**: update the variable that is tested.
Loop types

• Indefinite Loop:
  – You do not know ahead of time how many times your loop will execute.
  – For example, you do not know how many books a person might order.

• Definite Loop:
  – You know exactly how many times you want the loop to execute.
  – not at compile time necessarily
**for loops**

- Another type of loop in Processing is the **for** loop.
- It is very good for definite loops.
- All the parts (priming, testing and updating) are in one place.
- **format:**
  
  ```
  for (prime expression; test expression; update expression)
  ```
- Since the expressions are all in one place, many people prefer **for** to **while** when the number of iterations is known.
Basic For Loop Syntax

- **for** loops are good for creating definite loops.

```java
int counter;
for (counter = 1; counter <= 10; counter++)
    println (counter);
```

1. **Priming:** Set the start value.
2. **Test Condition:** Set the stop value.
3. **Update:** Update the value.

Note that each section is separated by a semicolon.
for Loop Flowchart

1. Priming
   Set counter=1

2. Test
   counter <= 10

   TRUE
   3a. print counter
   3b. Update counter++;

   FALSE
Infinite Loop

• You can still end up with an infinite loop when using for loops

```
for (counter = 0; counter <= 10; counter--)
```
Warnings

• Do not use a float for the counter
  – May result in imprecise counter values and faulty evaluation for loop termination purpose
• Don’t use commas instead of semicolons to separate the components of the for loop
  – (very common error
• As in the if and while, do not put a semicolon ; right after the parentheses – will be an empty loop!
Off-by-one error

• In the first example, shown here, if had written counter < 10
  then loop would execute 9 times, not the desired 10 times

```java
for (counter = 1; counter < 10; counter++)
{
    println (counter);
}
/* end for counter */
```
Help avoid off-by-one errors

• Try to make your conditions in the form $\leq$, not $<$
  – Avoid code like counter $< 11$ or counter $< 10$
void setup() and void draw()

• When you run a program in Processing the setup() function is called.

• Then the draw() function is called repeatedly in a loop.
  – The speed of the loop is the frame rate which can be changed using the frameRate() function.
Nested For Loops

• It is also possible to place a for loop inside another for loop.

```java
int rows, columns;
for (rows = 1; rows <= 5; rows++)
{
    for (columns=1; columns<=10; columns++)
        print ("*");
    println ( );
}
```

Output:
```
********************
********************
********************
********************
********************
```
$int$ rows, columns;

for (rows=1; rows<=5; rows++)
{
    for (columns=1; columns<=rows; columns++)
        print("*");
    println();
}