Looping

Examples of While Loops & For Loops

What's a Loop?

"A loop is a program construct that repeatedly executes the loop's statements (known as the loop body)" -- 6.1

- Do the above while the loop's expression is true
- When the expression is false, skip the loop.

Summing a bunch of numbers

```
cumulativeSum <-- 0;
while there are numbers remaining...
    add the next number to the cumulative sum;
    move on to the next number;
```

Summing a bunch of numbers

```
cumulativeSum <-- 0;
while there are numbers remaining...
    add the next number to the cumulative sum;
    move on to the next number;
```

```
cumulativeSum: 9
numbers: [9, 1, 5, 17]
          ⬆
```

Summing a bunch of numbers

```
cumulativeSum <-- 0;
while there are numbers remaining...
    add the next number to the cumulative sum;
    move on to the next number;
```

```
cumulativeSum: 10
numbers: [9, 1, 5, 17]
             ⬆
```

Summing a bunch of numbers

```
cumulativeSum <-- 0;
while there are numbers remaining...
    add the next number to the cumulative sum;
    move on to the next number;
```

```
cumulativeSum: 15
numbers: [9, 1, 5, 17]
                ⬆
```

Summing a bunch of numbers

```
cumulativeSum <-- 0;
while there are numbers remaining...
    add the next number to the cumulative sum;
    move on to the next number;
```

```
cumulativeSum: 32
numbers: [9, 1, 5, 17]
                   ⬆
```

Summing a bunch of numbers

```
cumulativeSum <-- 0;
while there are numbers remaining...
    add the next number to the cumulative sum;
    move on to the next number;
```

```
cumulativeSum: 32
numbers: [9, 1, 5, 17]
                         ⬆️
```

🎉 🎉 🎉 🎉 🎉

Activity: Averaging a bunch of numbers

```
cumulativeSum <-- 0;
????????      <-- 0 // what other value should we track?
while there are numbers remaining...
    add the next number to the cumulative sum;
    ??????; // we'll need to update our other value, too.
    move on to the next number;
```

Solution: Averaging a bunch of numbers

```
cumulativeSum <-- 0;
numbersSeen   <-- 0 // need to count how many data points we have
while there are numbers remaining...
    add the next number to the cumulative sum;
    increment numbersSeen;
    move on to the next number;
print (cumulativeSum / numbersSeen)
```

```
cumulativeSum: 0
numbersSeen: 0
numbers: [9, 1, 5, 17]
```

Solution: Averaging a bunch of numbers

```
cumulativeSum <-- 0;
numbersSeen   <-- 0 // need to count how many data points we have
while there are numbers remaining...
    add the next number to the cumulative sum;
    increment numbersSeen;
    move on to the next number;
print (cumulativeSum / numbersSeen)
```

```
cumulativeSum: 9
numbersSeen: 1
numbers: [9, 1, 5, 17]
          ⬆
```

Solution: Averaging a bunch of numbers

```
cumulativeSum <-- 0;
numbersSeen   <-- 0 // need to count how many data points we have
while there are numbers remaining...
    add the next number to the cumulative sum;
    increment numbersSeen;
    move on to the next number;
print (cumulativeSum / numbersSeen)
```

```
cumulativeSum: 10
numbersSeen: 2
numbers: [9, 1, 5, 17]
             ⬆
```

Solution: Averaging a bunch of numbers

```
cumulativeSum <-- 0;
numbersSeen   <-- 0 // need to count how many data points we have
while there are numbers remaining...
    add the next number to the cumulative sum;
    increment numbersSeen;
    move on to the next number;
print (cumulativeSum / numbersSeen)
```

```
cumulativeSum: 15
numbersSeen: 3
numbers: [9, 1, 5, 17]
                ⬆
```

Solution: Averaging a bunch of numbers

```
cumulativeSum <-- 0;
numbersSeen   <-- 0 // need to count how many data points we have
while there are numbers remaining...
    add the next number to the cumulative sum;
    increment numbersSeen;
    move on to the next number;
print (cumulativeSum / numbersSeen)
```

```
cumulativeSum: 32
numbersSeen: 4
numbers: [9, 1, 5, 17]
```

Solution: Averaging a bunch of numbers

```
cumulativeSum <-- 0;
numbersSeen   <-- 0 // need to count how many data points we have
while there are numbers remaining...
    add the next number to the cumulative sum;
    increment numbersSeen;
    move on to the next number;
print (cumulativeSum / numbersSeen)
```

```
cumulativeSum: 32
numbersSeen: 4
numbers: [9, 1, 5, 17]
```

🎉 🎉 🎉 `print --> 8` 🎉 🎉 🎉

While Loops (6.2)

Definition:

A while loop is a program construct that repeatedly executes a list of sub-statements (known as the **loop body**) while the loop's expression evaluates to true.

- Each execution of the loop body is called an iteration.
- Once entering the loop body, execution continues to the body's end, *even if the expression would become false midway through.*

Syntax

```
while (expression) { // Loop expression
    // Loop body: Executes if expression evaluated to true
    // After body, execution jumps back to the "while"
}
// Statements that execute after the expression evaluates to false
```

Worked Example: CountUp.java

- Read user input using a Scanner as an int
- Print out every number `0` ➡️ that input; then, print `"all done!"`

```java
public class CountUp {
    public static void main(String[] args) {
        // read user input
        // create variable to track progress towards upper limit

        // boolean expression that's true while we have work to do
        while () {
            // print the current number
            // update our control variable
        }
        // Afterwards, print "all done!"
    }
}
```

Solution: CountUp.java

```
import java.util.Scanner;

public class CountUp {
    public static void main(String[] args) {
        Scanner scnr = new Scanner(System.in);
        int upperLimit = scnr.nextInt();
        int currentInt = 0;
        while (currentInt <= 0) {
            System.out.println(currentInt);
            currentInt++;
        }
        System.out.println("all done!")'
    }
}
```

Writing Expressions for While Loops

```
while (____) {
    // do something
}
```

Iterate while...	Solution
x is greater than or equal to 0	
c is not equal to `"stop"`	

Writing Expressions for While Loops

```
while (____) {
    // do something
}
```

Iterate while...	Solution
x is greater than or equal to 0	`x >= 0`
c is not equal to `"stop"`	`!c.equals("stop")`

Common Mistakes: Wrong Loop Expression

Remember that the loop expression tells when the loop *should* iterate, not when it should stop!

```java
int x = 20;
while (x < 10) {
    System.out.println(x);
    x -= 2;
}
```

```java
int x = 20;
while (x >= 10) {
    System.out.println(x);
    x -= 2;
}
```

Common Mistakes: Looping Infinitely

It should always be possible for our loop expression to evaluate to `false` at some point.

```
// What's the problem here?
Scanner scnr = new Scanner(System.in);
Gradebook gb = new Gradebook("cit591_grades.csv");
String pennkey = scnr.next();
while (!pennkey.equals("STOP")) {
    int grade = gb.checkGrade(pennkey);
    System.out.println(pennkey + " has grade " + grade);
}
```

Common Mistakes: Looping Infinitely

It should always be possible for our loop expression to evaluate to `false` at some point.

```java
// What's the problem here?
Scanner scnr = new Scanner(System.in);
Gradebook gb = new Gradebook("cit591_grades.csv");
String pennkey = scnr.next();
while (!pennkey.equals("STOP")) {
    int grade = gb.checkGrade(pennkey);
    System.out.println(pennkey + " has grade " + grade);
    pennkey = scnr.next(); // this was missing!
}
```

Common Mistakes: Looping Infinitely

Even when you update the loop control variable, you can get subtle errors...

```
// Get userVal from input

while (userVal != 0) {
 // Put userVal to output
 // userVal = userVal - 2;
}
```

What happens when we start at `6`?

Common Mistakes: Looping Infinitely

Even when you update the loop control variable, you can get subtle errors...

```
// Get userVal from input

while (userVal != 0) {
 // Put userVal to output
 // userVal = userVal - 2;
}
```

What happens when we start at `6`?

`6 -> 4 -> 2 -> 0` 🎉

Common Mistakes: Looping Infinitely

Even when you update the loop control variable, you can get subtle errors...

```
// Get userVal from input

while (userVal != 0) {
 // Put userVal to output
 // userVal = userVal - 2;
}
```

What happens when we start at `3`?

Common Mistakes: Looping Infinitely

Even when you update the loop control variable, you can get subtle errors...

```
// Get userVal from input

while (userVal != 0) {
 // Put userVal to output
 // userVal = userVal - 2;
}
```

What happens when we start at `3`?

```
3 -> 1 -> -1 -> -3 -> -5 --> ...
```
😭

Try Some Examples!

What's printed?

```
x = 0;

while (x > 0) {
   System.out.print(x + " ");
   x = x - 1;
}
System.out.print("Bye");
```

Try Some Examples!

What's printed?

```
x = 10;

while (x != 3) {
   System.out.print(x + " ");
   x = x / 2;
}
```

Worked Example: ReverseDigits.java

We'll use iteration, modulo, and division to print all the digits of an integer (useful for homework!)

For Loops (6.4)

Definition:

A for loop is a loop with three parts at the top that makes it easy to iterate a specific number of times. The parts are:

- Loop variable initialization
- Loop expression
- Loop variable update

Note that these parts are all actually present in a while loop already.

Coming from While Loops

```
int i = 0;
while (i < 5) {
    // loop body
    i = i + 1;
}
```

⬇️ ⬇️ ⬇️ ⬇️

```
for (int i = 0; i < 5; i = i + 1) {
    // loop body
}
```

Exercise: What gets printed?

```java
for (int i = 0; i < 6; i++) {
    System.out.println(i);
}
```

Exercise: What gets printed?

```java
for (int i = 0; i < 6; i++) {
    System.out.println(i);
}
```

```
0, 1, 2, 3, 4, 5
```

Exercise: How do we get 20 iterations?

```
for (int i = 0; _____ ; i++) {
    // ...
}
```

Exercise: How do we get 20 iterations?

```
for (int i = 0; i < 20 ; i++) {
    // ...
}
```

Worked Example: Interest.java

For a given initial balance and interest rate, write a program that calculates what the balance will be after ten years.

```java
double initialSavings = 10000.0;
double interestRate = 0.05; // 5%
double currentSavings = ??? // what should this start as?

// define a for loop that runs 10 times
for (????; ????; ????) {
    // update the current savings based on the interest rate
    // i.e. add the interest on the current amount
    // TO the current amount.
}
System.out.print(initialSavings + " becomes ");
System.out.println(currentSavings + " after 10 years.");
```

Worked Example: Interest.java

For a given initial balance and interest rate, write a program that calculates what the balance will be after ten years.

```java
double initialSavings = 10000.0;
double interestRate = 0.05; // 5%
double currentSavings = initialSavings;

// define a for loop that runs 10 times
for (int i = 0; i < 10; i++) {
    currentSavings += currentSavings * interestRate;
}
System.out.print(initialSavings + " becomes ");
System.out.println(currentSavings + " after 10 years.");
```

FOR vs. WHILE

loop	when to use
for	number of iterations is known (i.e. some `n`)
while	num. iterations unknown, like looping until user inputs "STOP"

Challenge Example: BiggestOfN.java

Print the largest value in a list of integers. Assume the first integer input is the number of integers to expect.

`4 9 -10 8 1` ➡ `9`

`5 9 -10 8 1 20` ➡ `20`

`2 0 -3` ➡ `0`

`0` ➡ ???