while-loops in Java work exactly the same as in Python, Javascript, and C, although the syntax is different than that of Python.

The examples and exercises are worth working through anyway, to gain comfort with code organization in Java. Here is a simple example of using a while-loop to count in Java:

```
package com.github.akarazhev.jacademy.jprog.basics;

public final class WhileExample {

    public static void main(final String[] args) {
        int number = 0;
        while (number < 100) {
            System.out.println(number);
            number += 2;
        }
    }
}
```

for-loops

Intention: Learn how to use for-loops for counting and indexing.

For-loops in Java are identical to those in C and Javascript, and if you are familiar with those languages, you may skip this section. If you are most familiar with Python, you'll want to work through this chapter carefully. Here's a for-loop in Java:

```
package com.github.akarazhev.jacademy.jprog.basics;

public final class ForExample {

    public static void main(final String[] args) {
        for (int i = 1; i < 11; i++) {
            System.out.println(i);
        }
    }
}
```

Python for-loops iterate over elements of a list. The for loop shown above does not. Let's first look at a while-loop that has a very similar intention as the above for-loop.

```
package com.github.akarazhev.jacademy.jprog.basics;

public final class WhileExample2 {

    public static void main(final String[] args) {
        int i = 1;             // initialize a variable
        while (i < 11) {      // test a condition
            System.out.println(i);
            i++;               // modify the variable value
        }
    }
}
```

The while-loop shows an obvious way to count by incrementing the variable `i`, and the pattern is very standard. Create and initialize a variable `i`, test if `i` satisfies some condition, execute the body, increment `i`, and so forth. Here's the basic structure in pseudocode:

`init while (condition) { // body modify }`

The for-loop looks like this:

`for (init; condition; modify) { // body }`

The for-loop works like this:

1. Execute the instruction **init**;
2. Test the **condition**;
3. Execute the body;
4. Execute the instruction **modify**;
5. Go back to step 2.

Blast off examples:

```
package com.github.akarazhev.jacademy.jprog.basics;

public class BlastOff {

    public static void main(String[] args) {
        int i = 10;
        while (i > 0) {
            System.out.println(i);
            i--;
        }

        System.out.println("Blast off!");

        for (int j = 10; j > 0; j--) {
            System.out.println(j);
        }

        System.out.println("Blast off!");
    }
}
```

There's a slight difference between the while-loop and the for-loop in the example. The variable `j` created in the initialization section of the for-loop is in scope only for the for-loop body, and goes out of scope after the body. On the other hand, the while-loop creates the variable `i` before the loop, so `i` is available after the loop as well.

foreach-loops

Intention: Learn how to use foreach-loops to loop over list-like structures.

Python for-loops loop over sequences; a list is one type of sequence. This can be very convenient, and Java 8 introduced a similar capability, called a foreach-loop. In spite of the name, a foreach-loop uses the keyword `for`. Here is an example:

```
package com.github.akarazhev.jacademy.jprog.basics;

public final class ForEachExample {

    public static void main(final String[] args) {
        final int[] primes = {2, 3, 5, 7, 11, 13};
        for (final int p : primes) {
            System.out.println(p);
        }
    }
}
```

The instruction before the colon creates a variable. That variable then takes on every value in the **collection** after the colon. An array is a particular type of collection; so are ArrayLists and other data structures that we will see later.

Only the value of an array item is copied into `p` above; changing `p` does not change the array.

Java 8 also introduced a method `.forEach()` of collections.