# for-loops

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

WE'LL COVER THE FOLLOWING

- Exercise: blast off

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 lesson carefully. Here's a for-loop in Java:

ForExample.java

```
class ForExample {
  public static void main(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.

```
class WhileExample {
  public static void main(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.

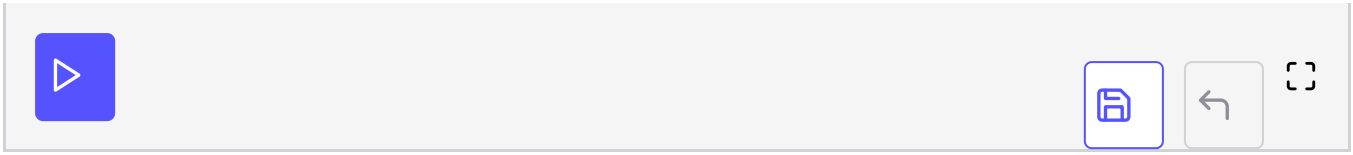
## Exercise: blast off #

Replace the while-loop with a for-loop:

BlastOff.java | Sample solution

```
class BlastOff {
  public static void main(String[] args) {
    int i = 10;
    while(i > 0) {
      System.out.println(i);
      i--;
    }
    System.out.println("Blast off!");
  }
```

```
}
```



There’s a slight difference between the while-loop and the for-loop in the sample solution. The variable `i` 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.