```java

public class Arrays {

    public static void main(String[] args) {
        /*
         * Java has a concept that Python does not called "arrays". Python's
         * lists are the closest analogue to Java arrays, but there are
         * important differences. First, arrays have a FIXED SIZE, and must be
         * created already knowing that size.
         */
        int[] numbers = new int[5];
        // numbers has room for 5 integers. The 5 integers are by default set
        // to zero.

        // Like Python, arrays are "indexed" with square brackets starting at 0.
        numbers[0] = 5;
        numbers[1] = 6;
        numbers[2] = 2;
        numbers[3] = numbers[2] + numbers[1];

        // The equivalent of len(_) is _.length
        System.out.println(numbers.length);

        // The standard way to print all elements of an array is with a for loop
        for (int i = 0; i < numbers.length; i++) {
            System.out.println(numbers[i]);
        }

        // Find the sum of the numbers in the array.
        int sum = 0;
        for (int i = 0; i < numbers.length; i++) {
            sum += numbers[i];
        }
        System.out.println("Sum: " + sum);


        // Arrays cannot be appended to. We can only assign values to indexes
        // that already exist. Attempting to use an index that does not exist
        // leads to an error.
        System.out.println(numbers[5]);
    }

}

```

```java

public class Methods {
    public static void main(String[] args) {
        // In Java, we call a function in mostly the same way as in Python.
        int m = Math.max(5, 10);

        // Some kinds of functions belong to individual values, in that they
        // do their work using information about the value. We call these
          "methods".
        String s1 = "Hello";
        String s2 = "Goodbye";

        boolean f1 = s1.startsWith("H");
        boolean f2 = s2.startsWith("H");

        // We called the same method "startsWith" twice, but we expect a
        // result for f1 and f2. Why?




        // We don't know how to write these kinds of methods yet. We only know
        // how to write functions that only operate on their parameters. We wrote
        // lots of such functions in Python. We'll write a few more here.

        // The Java equivalent of a Python function is a "static method".
        int a = abs(-10);

        int c = min(5, -6);
    }

    // Returns the absolute value of x.
    public static int abs(int x) {
        // The "public static" will simply be routine for now; JUST DO IT.
        // The "int" is the return type of the function, something that is not
        // listed in Python. It means that this function always gives back an
        // integer value.

        if (x < 0) {
            // return is also used to give back the function's return value.
            // The expression we return must have a type that matches the
            // return type of the method.
            return -x;
        }
        return x;
    }

    // Write these methods:
    // min: returns the smaller of two integer parameters


```

```
52
53
54
55      // average: returns the arithmetic mean of two double parameters
56
57
58
59
60
61      // Note: Java has the same "scope" rules as Python.
62  }
63
64
```

```java

public class ArrayMethods {
    // Arrays can be passed as arguments to methods just like any other value.
    // But you may see some surprising results...


    // This function takes an integer and sets it to 100.
    public static void MutateInt(int x) {
        x = 100;
    }

    public static void main(String[] args) {
        int x = 5;
        System.out.println("x = " + x);
        MutateInt(x);
        System.out.println("x = " + x);
        // What output do we expect? Why?


        // Shortcut syntax for int[] a = new int[5] { ... };
        int[] a = {1, 2, 3, 4, 5};
        System.out.println("a[0] = " + a[0]);

        MutateArray(a);
        // What do you expect the next line will print? Why?
        System.out.println("a[0] = " + a[0]);









        // The difference here in how primitive types like int and complex types
        // like arrays are passed to methods is an example of "pass by value"
        // vs. "pass by reference".









    }

    // We can also write methods that take arrays.
    public static void MutateArray(int[] a) {
        a[0] = 100;
    }


}
```