Java Constructs

Operators (relational, logical)
Conditionals
Loops via iteration
Relational Operators

- Java has a `boolean` type that can take the value `true` or `false`.

- Booleans arise naturally when using **relational operators** to compare two values:
  
  3 < 5  
  3 < 2  
  3 > 2  
  5 <= 1  
  5 >= 1  
  5 == 5  
  5 == 6  
  5 != 6
Logical Operators

- Boolean values can be manipulated with the logical operators ! (not), && (and), and || (or)

  ! (3 < 5)
  ! (3 == 5)
  (3 > 5) && (7 < 8)
  (3 < 5) && (7 < 8)
  (3 > 5) || (7 < 8)
  (3 > 5) || (7 > 8)
Predicates

- A **predicate** is any method that returns a **boolean** value

```java
//determine if n is even
public static boolean isEven(int n)
{
    return (n % 2) == 0;
}

//determine if num is divisible by factor
public static boolean isDivisibleBy(int num, int factor) {
}

//determine if n is between lo and hi
public static boolean isBetween(double n, double lo, double hi) {
}
```
Predicates

Write your own predicate to determine if \( n \) is odd. Then, can you write it another way?

```java
public static boolean isOdd(int n) {
    return (n % 2) == 1;
}
```

```java
public static boolean isOdd(int n) {
    return !isEven(n);
}
```
Conditionals

- To choose between two courses of action, to control the program flow, we use **conditional statements** such as **if**, **else if**, and **else**

```java
//returns absolute value of n
public static double abs(double n){
    if (n < 0) {
        return -n;
    } else {
        return n;
    }
}

//returns absolute value of n
public static double abs(double n){
}
```
public static void main(String[] args) {
    int x = 28; String s = "meow";
    if (x < 30 && s.length() < 10) {
        x = x + 5;
        int y = s.length();
        if (x + y > 36) {
            System.out.println("hello " + x);
        } else if (x + y < 33) {
            System.out.println("howdy " + y);
        } else {
            System.out.println("hi!");
        }
    } else {
        x = x - 10;
        int y = s.length() + 5;
        if (x == 15) System.out.println("Salut " + x);
        else System.out.println("Ciao " + y);
    }
}
Iteration - **while** loop

- **Iteration** refers to a sequence of steps that is repeated until some stopping condition is reached.

```
while(boolean_expression){

(1) evaluate boolean expression

(2) if true, execute body of loop and go back to step (1)

statement 1;
statement 2;
...

(3) if false, go to statement after while loop

}
```

```
int i = 1;
while (i < 4) {
    System.out.println("CS230");
    i = i + 1;
}
```
Do vs While vs For loops
Iteration – **for** loop

Execute this statement once before entering loop

```
for ( initialization of index variable ; boolean expn ; update index variable ) {
statements
}

Execute this statement before next test of the boolean expression

If true, execute body of loop

If the boolean expression was true, the body of the loop will be executed

If the boolean expression evaluates to false, drop down to here

for (int i = 1; i < 4; i++) {
    System.out.println("CS230");
}
```

Write a for loop that prints the numbers from 1 to 10.

for (int i = 1; i <= 10; i++) {
    System.out.println(i);
}
Math Class

```java
System.out.println(Math.max(100, 50));
System.out.println(Math.sqrt(25));
System.out.println(Math.log(10));

// Given area of circle, returns the circle's radius.
// Since area=pi*r*r, we have r = squareroot(area/pi).
public static double getCircleRadius(double area) {
    return Math.sqrt(area/Math.PI);
}

System.out.println(getCircleRadius(100));
```
import java.util.Random;

public class RandomExample {

    public static void main(String[] args) {

        Random rand = new Random();
        for (int i = 0; i < 15; i++) {
            System.out.println(rand.nextInt(10));
        }
    }
}

Random Class
Count Vowels

// Returns true if character is lower-case vowel (a, e, i, o, u), false otherwise.
public static boolean isVowel(char ch)

// Returns the number of occurrences of vowels in the String s
public static int countVowels(String s)
String s1 = new String("Grace Hopper");
String s2 = "CU L8R";
String s3 = ":);"

System.out.println(s1.toLowerCase());
System.out.println(s1.length());
System.out.println(s2.length());
System.out.println(s2.equals(s3));
System.out.println(s2.equals("CU L8R"));
System.out.println(s2.charAt(1));
System.out.println(s1.substring(7,11));
System.out.println(s2.substring(0,2).toLowerCase());
Strings in Java

- Strings in Java and Python are quite similar.
  - Like Python, Java strings are immutable.

- The difference is that
  Java uses method calls where Python uses Operators.

<table>
<thead>
<tr>
<th>Python</th>
<th>Java</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>str[3]</td>
<td>str.charAt(3)</td>
<td>Return character in 3rd position</td>
</tr>
<tr>
<td>str[2:5]</td>
<td>str.substring(2,4)</td>
<td>Return substring from 2nd to 4th</td>
</tr>
<tr>
<td>len(str)</td>
<td>str.length()</td>
<td>Return the length of the string</td>
</tr>
<tr>
<td>str.find('x')</td>
<td>str.indexOf('x')</td>
<td>Find the first occurrence of x</td>
</tr>
<tr>
<td>str.split()</td>
<td>str.split('\s')</td>
<td>Split the string on whitespace into a list/array of strings</td>
</tr>
<tr>
<td>str.split(',')</td>
<td>str.split(',')</td>
<td>Split the string at ',' into a list/array of strings</td>
</tr>
<tr>
<td>str + str</td>
<td>str.concat(str)</td>
<td>Concatenate two strings together</td>
</tr>
<tr>
<td>str.strip()</td>
<td>str.trim()</td>
<td>Remove any whitespace at the beginning or end</td>
</tr>
</tbody>
</table>