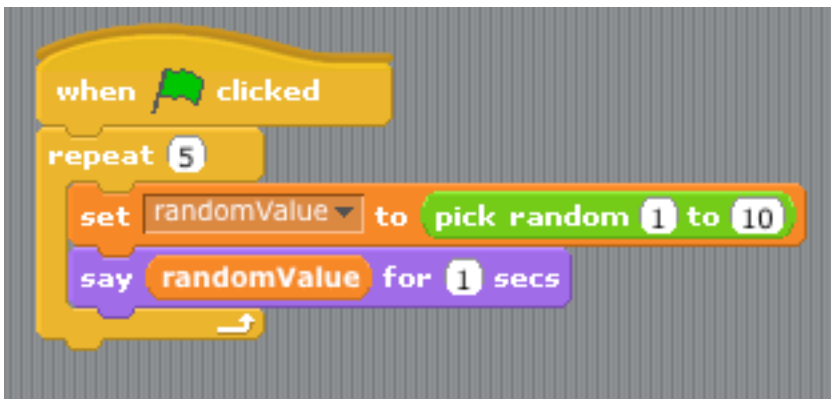
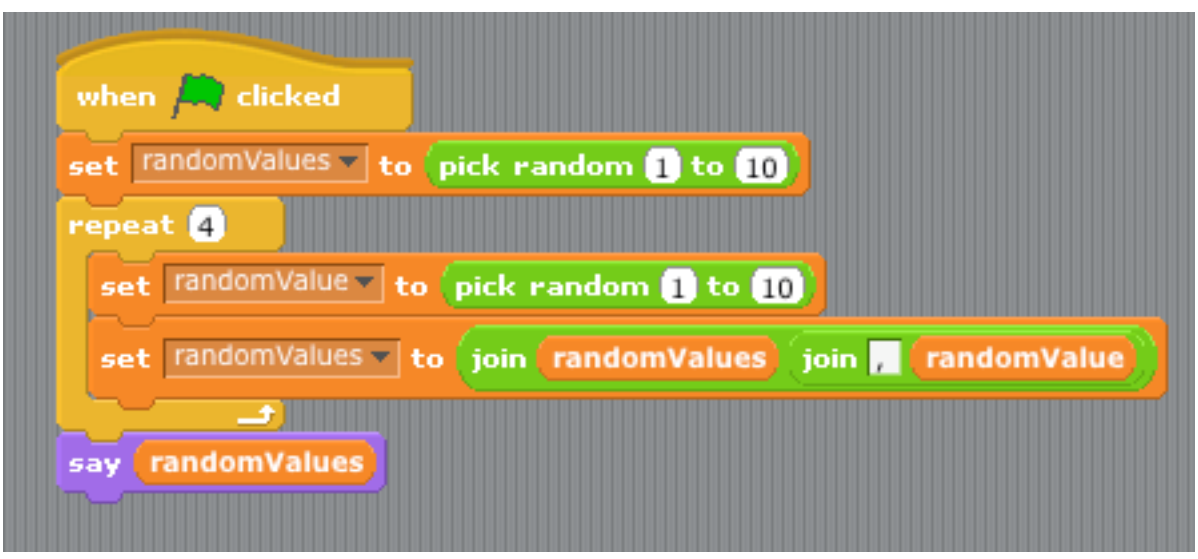
Random Number Generation in Scratch:

Generate 5 random values in Scratch, displaying each value for 1 second...

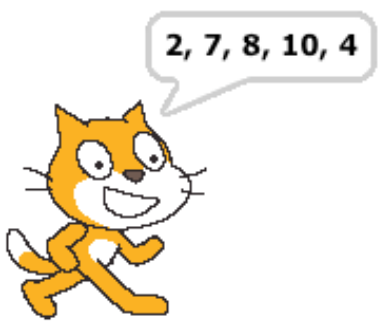


Sample run:

The above code is rather awkward, it would be far better to generate all five random values and print them out together:



Sample Run:



Important Notes:

1. The use of the repeat statement to produce a looping action
    a. In both loops above, two statements are executed multiple times

2. How random values are generated; in these programs between 1 - 10, inclusive

3. Use of the “join” statement to perform string concatenation
    a. The variable randomValues “grows” each time through the loop

More continued on next page...

Random Number Generation in Java:

```
import java.util.Random;

public class tenRands
{
	public static void main(String[] args)
	{
	Random rnd = new Random();

	// Generate ten random values between 0 - 9

	for (int i = 0; i < 10; i++)
		System.out.print(rnd.nextInt(10) + "  ");

	System.out.println();   // put a blank line after the output
	}
}
```

Sample runs:

```
$ java tenRands
0  7  2  9  3  5  4  4  9  7
$ java tenRands
9  7  8  2  5  1  4  0  6  8
$ java tenRands
5  8  9  0  2  8  7  1  9  2
$ java tenRands
4  0  4  1  0  9  3  3  1  3
```

What if we want to generate random values between 1 - 10?

```
import java.util.Random;

public class tenRandsV2
{
	public static void main(String[] args)
	{
	Random rnd = new Random();

	// Generate ten random values between 1 - 10

	for (int i = 0; i < 10; i++)
		System.out.print( (rnd.nextInt(10) + 1) + "  ");

	System.out.println();   // put a blank line after the output
	}
}
```

Sample runs:

```
$ java tenRandsV2
4  3  8  9  2  1  2  6  8  5
$ java tenRandsV2
9  1  10  10  5  4  9  4  8  3
$ java tenRandsV2
2  10  2  10  3  2  5  4  8  3
```

Notes:

- The nextInt(x) will generate random values 0 - (x-1)

- The statement rnd.nextInt(10) in the top-most listing will generate values 0 - 9

- The statement rnd.nextInt(10) + 1 in the second listing will generate values 1 - 10

Listing 2:

```
import java.util.Random;

public class dice
{
      public static void main(String[] args)
      {
      int numberOfTwos    = 0;      // declare and initialize counters
      int numberOfSevens  = 0;
      int numberOfTwelves = 0;

      Random rnd = new Random();

      for (int i = 0; i < 100; i++)
            {
            int die1 = rnd.nextInt(6) + 1;
            int die2 = rnd.nextInt(6) + 1;

            int roll = die1 + die2;

            if (roll == 2)
                  numberOfTwos++;  // numberOfTwos++ => numberOfTwos = NumberOfTwos + 1;
            else if (roll == 7)
                  numberOfSevens++;
            else if (roll == 12)
                  numberOfTwelves++;
            }

      System.out.println("Out of 100 rolls, two, seven, and twelve were rolled:\n");

      System.out.println("  Number of twos:    " + numberOfTwos);
      System.out.println("  Number of Sevens:  " + numberOfSevens);
      System.out.println("  Number of twelves: " + numberOfTwelves);
      }
}
```

Sample run:

% java dice
Out of 100 rolls, two, seven, and twelve were rolled:

Number of twos: 4
Number of Sevens: 16
Number of twelves: 3

Notes:

- The method nextInt(x) returns pseudo-random values between 0 and x-1 which are guaranteed to be uniformly distributed over that range, in the long run

- Why use die1 and die2, instead of just one equation that generates values between 2 and 12?

Continued on next page...

Random number generators generate random values that are uniformly distributed over some range.

Many use an equation similar to (from Texas Instruments):

$X_{n+1} = (a * X_n + c) \% m;$

Where:

X_n = Seed c = 99991
a = 24298 m = 199017

Note that the seed may be set by the user, in Java we can use a value "passed" to Random, or setSeed(long seed)

More Simulations

A six-year-old has an opaque bag that contains 30 green, red, and blue marbles, specifically:

There are 15 green marbles.
There are 10 red marbles.
There are 5 blue marbles.

Accurately simulate picking a marble out of the bag. Which marble, on the average, will the six-year-old select?

To solve this problem, we must make a mapping that reflects the chances of picking a given colored marble.

For example, the chances of picking a blue marble is:

(number of blue marbles)/(total number of marbles) => 5/30 = 1/6.

In other words, for every six marbles taken out of the bag, one (on the average) will be blue.

The following is a program that simulates picking a marble out of the bag:

```
import java.util.Random;

public class pickMarble
{
        public static void main(String[] args)
        {
        Random simulator = new Random();

        int marble = simulator.nextInt(NUMBER_OF_MARBLES) + 1;

        if (marble <= 15)
                System.out.println("You picked a green marble.");
        else if (marble <= 25)
                System.out.println("You picked a red marble.");
        else
                System.out.println("You picked a blue marble.");
        }

        public static final int NUMBER_OF_MARBLES = 30;
}
```

Sample runs:

```
% java pickMarble
You picked a red marble.

% java pickMarble
You picked a green marble.
```