Random numbers

The Tin framework provides two versions of a function called random, that will produce “random numbers.” The first accepts one argument.

```
random(max: Double)
```

The argument max is the ceiling for a range from 0 up to the value of max. Random will return a Double value that is in this range. The value could be 0, but it will always be less than the value of max.

```
random(min: Double, max: Double)
```

The second version accepts two arguments, min and max. The argument min defines the lower bound of the range, while max defines the ceiling.

Random numbers

```
let chanceAngle = random(max: 360.0)
```

This example creates a constant chanceAngle that will be set to a random value in the range 0 to 360.

```
let offset = random(min: -10.0, max: 10.0)
```

This example creates a constant offset that will be set to a random value in the range -10 to 10.

Some Functions Return a Value

```
let luckyNumber = random(max: 360.0)
```

Some Functions Return a Value

```
let luckyNumber = random(max: 360.0)
```

A function is a set of instructions.
Some functions return a value.

Some Functions Return a Value

```
let luckyNumber = 117.539
```

A function is a set of instructions.
Some functions return a value.
The value returned in this example is assigned to a new constant named luckNumber.

Random numbers

The random function generates a random Double value. Each time random is called, it produces a new number that appears to be “random”. In other words, if you recorded a sequence of values produced from random, the numbers produced would not appear to be in any order or pattern. In reality, a pseudorandom sequence is created, based on a finite sequence of numbers that appear to have no order or structure. Eventually, the sequence will repeat.

It is important to note, the probability for any number in the range is uniform for the entire sequence. In other words, there is equal chance to get any number in the range.

There are many different algorithms used to generate random numbers, that make different tradeoffs regarding how much computation is required (eg. speed) vs the quality of the random sequence. This function uses the “Linear Congruential Random Source” in the GameplayKit framework - which prioritizes speed.