

How do I generate a random int number?

[Ask Question](#)

How do I generate a random integer in C#?

1647

c#

random



161

edited Mar 6 at 10:20



Uwe Keim

27.7k

32

134

216

asked Apr 24 '10 at 23:09



Rella

23.1k

90

311

593

28 Answers

The [Random class](#) is used to create random numbers. (Pseudo-random that is of course.).

2182

Example:



```
Random rnd = new Random();  
int month = rnd.Next(1, 13); // creates a number between 1 and 12  
int dice = rnd.Next(1, 7); // creates a number between 1 and 6  
int card = rnd.Next(52); // creates a number between 0 and 51
```

If you are going to create more than one random number, you should keep the `Random` instance and reuse it. If you create new instances too close in time, they will produce the same series of

[Home](#)[PUBLIC](#)[Stack Overflow](#)[Tags](#)[Users](#)[Jobs](#)[Teams](#)

Q&A for work



[Learn More](#)

random numbers as the random generator is seeded from the system clock.

edited Mar 17 at 11:05



AustinWBryan

2,381 2 14 32

answered Apr 24 '10 at 23:19



Guffa

564k 78 575 885

-
- 101 Just a note for Unity C# users, you must specify "System.Random()" as Unity has a "UnityEngine.Random()" which clashes (note that "UnityEngine.Random()" doesn't have the ability to generate random numbers). As unity doesn't always import System by default this can cause some confusion. – [Joehot200](#) Aug 13 '17 at 17:39
-
- 1 In order to reuse it, you can declare `rnd` as `static` and/or set it just once when initializing the code. – [Junior M](#) Feb 10 '18 at 19:09
-
- 2 @JuniorM: Yes, you can make it static to reuse it, but you have to be careful so that you don't access it from multiple threads as it's not thread safe (as is usual for any class that is not specifically made thread safe). – [Guffa](#) Feb 13 '18 at 17:11
-
- 4 Plus one for the comment about keeping the same 'Random' instance. That had been bothering me for some time. – [AndyUK](#) Apr 12 '18 at 19:13
-
- 3 I think it would be useful to add a disclaimer that this is not cryptographically secure, since this is a popular question, just in case someone blindly tries to do cryptography with `Random` ... – [Daniel García Rubio](#) Jun 27 '18 at 15:33
-



I always have methods that generate random numbers which help for various purposes. I hope this may help you too:

-2

```

public class RandomGenerator
{
    public int RandomNumber(int min, int max)
    {
        var random = new Random();
        return random.Next(min, max);
    }

    public string RandomString(int size, bool lowerCase)
    {
        var builder = new StringBuilder();
        var random = new Random();
        char ch;

        for (int i = 0; i < size; i++)
        {
            ch = Convert.ToChar(Convert.ToInt32(Math.Floor(26 * rand
65)));
            builder.Append(ch);
        }

        if (lowerCase)
            return builder.ToString().ToLower();
        return builder.ToString();
    }
}

```

edited Mar 17 at 11:24



AustinWBryan

2,381 2 14 32

answered May 15 '18 at 7:50



PatsonLeaner

189 3 13

```
int n = new Random().Next();
```

-4



You can also give minimum and maximum value to `Next()` function.
Like:

```
int n = new Random().Next(5, 10);
```

edited Mar 17 at 11:24



AustinWBryan

2,381 2 14 32

answered Aug 8 '16 at 9:31



Muhammad Awais

2,039 1 25 22



Modified answer from [here](#).

5

If you have access to an Intel Secure Key compatible CPU, you can generate real random numbers and strings using these libraries:

<https://github.com/JebteK/RdRand> and <https://www.rdrand.com/>



Just download the latest version from [here](#), include `Jebtek.RdRand` and add a using statement for it. Then, all you need to do is this:

```
bool isAvailable = RdRandom.GeneratorAvailable(); // Check to see if
compatible CPU
string key       = RdRandom.GenerateKey(10);      // Generate 10 random
string apiKey    = RdRandom.GenerateAPIKey();     // Generate 64 random
useful for API keys
byte[] b         = RdRandom.GenerateBytes(10);    // Generate an array of
bytes
uint i           = RdRandom.GenerateUnsignedInt() // Generate a random
```

If you don't have a compatible CPU to execute the code on, just use the RESTful services at [rdrand.com](https://www.rdrand.com/). With the `RdRandom` wrapper library included in your project, you would just need to do this (you get 1000 free calls when you signup):

```
string ret = Randomizer.GenerateKey(<length>, "<key>");
uint ret  = Randomizer.GenerateUInt("<key>");
byte[] ret = Randomizer.GenerateBytes(<length>, "<key>");
```

edited Mar 17 at 11:23



AustinWBryan

2,381 2 14 32

answered Mar 27 '14 at 19:36



JebaDaHut

388 3 8



16



You could use Jon Skeet's [StaticRandom](#) method inside the MiscUtil class library that he built for a pseudo-random number.

```
using MiscUtil;
...

for (int i = 0; i < 100;
    Console.WriteLine(StaticRandom.Next());
```

edited Mar 17 at 11:22



AustinWBryan

2,381 2 14 32

answered Apr 24 '10 at 23:31



mbcrump

763 2 5 15

4 "truly random"? How does that work? – [Andreas Rejbrand](#) Apr 25 '10 at 0:29

25 I just had a look at the source code, and this function uses exactly the same random number engine, the one "included" in C#, but makes sure

that the same "seed"/"mother object" is used for all calls. (I am sorry that I do not know the C# terminology. But my point is that this function does not make any better random numbers than the standard function.) –

[Andreas Rejbrand](#) Apr 25 '10 at 0:41

- 4 It's impossible for anything to be 'truly random' since there is always some limiting factor or prejudice included that is inherent by its very existence. Didn't you listen to the teacher in science classes? ;-) – [Phill Healey](#) Oct 15 '14 at 10:10

Let's say , according to him this is as 'truly random' as it gets –

[The Mitra Boy](#) Aug 21 '15 at 7:42

203

The question looks very simple but the answer is bit complicated. If you see almost everyone has suggested to use the Random class and some have suggested to use the RNG crypto class. But then when to choose what.

For that we need to first understand the term RANDOMNESS and the philosophy behind it.

I would encourage you to watch this video which goes in depth in the philosophy of RANDOMNESS using C#

<https://www.youtube.com/watch?v=tCYxc-2-3fY>

First thing let us understand the philosophy of RANDOMNESS. When we tell a person to choose between RED, GREEN and YELLOW what happens internally. What makes a person choose RED or YELLOW or GREEN?

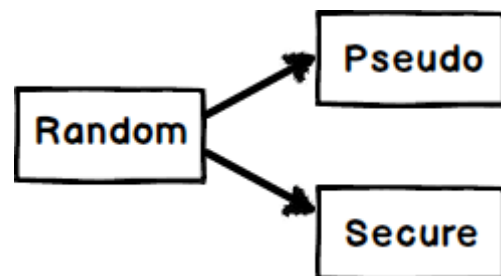


Some initial thought goes into the persons mind which decides his choice, it can be favorite color , lucky color and so on. In other words some initial trigger which we term in RANDOM as SEED.This SEED

is the beginning point, the trigger which instigates him to select the RANDOM value.

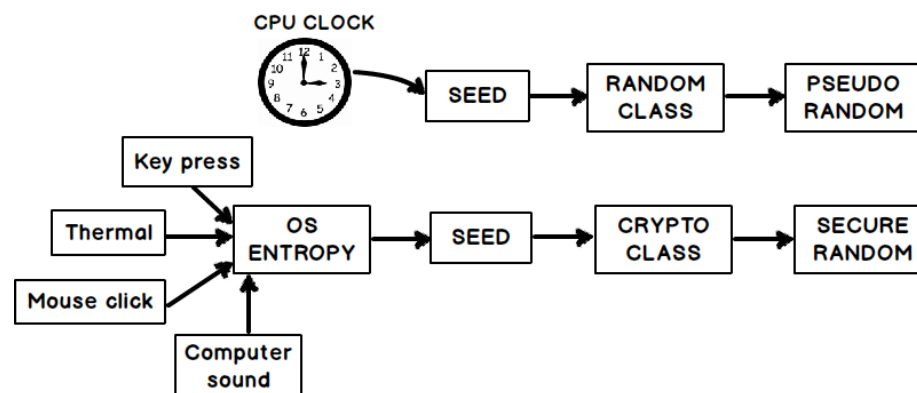
Now if a SEED is easy to guess then those kind of random numbers are termed as **PSEUDO** and when a seed is difficult to guess those random numbers are termed **SECURED** random numbers.

For example a person chooses is color depending on weather and sound combination then it would be difficult to guess the initial seed.



Now let me make an important statement:-

*“Random” class generates only PSEUDO random number and to generate SECURE random number we need to use “RNGCryptoServiceProvider” class.



Random class takes seed values from your CPU clock which is very much predictable. So in other words RANDOM class of C# generates pseudo random numbers , below is the code for the same.

```
var random = new Random();  
int randomnumber = random.Next()
```

While the RNGCryptoServiceProvider class uses OS entropy to generate seeds. OS entropy is a random value which is generated using sound , mouse click and keyboard timings , thermal temp etc. Below goes the code for the same.

```
using (RNGCryptoServiceProvider rg = new RNGCryptoServiceProvider())  
{  
    byte[] rno = new byte[5];  
    rg.GetBytes(rno);  
    int randomvalue = BitConverter.ToInt32(rno, 0);  
}
```

To understand OS entropy see this video from 14:30 <https://www.youtube.com/watch?v=tCYxc-2-3fY> where the logic of OS entropy is explained. So putting in simple words RNG Crypto generates SECURE random numbers.

edited Mar 17 at 11:21



AustinWBryan

2,381 2 14 32

answered Jun 14 '16 at 6:27



Shivprasad Koirala

17.2k 6 61 56

4 shouldn't be your byte[5] only [4] as ToInt32 parses only 4 bytes? – [Bernhard](#) Nov 21 '17 at 11:34

1 It's always helpful to know where these classes live. System.Security.Cryptography – [Elton](#) Dec 18 '18 at 23:49

You can try with random seed value using below:

0

```
var rnd = new Random(11111111); //note: seed value is 11111111
string randomDigits = rnd.Next();
var requestNumber = $"SD-{randomDigits}";
```

edited Feb 23 at 5:36

answered Apr 22 '18 at 10:55



[Rejwanul Reja](#)

371 1 7 12

Thanks @MikaelDúiBolinder to see this snippet – [Rejwanul Reja](#) Feb 23 at 5:38

-1

Sorry, OP indeed requires a random `int` value, but for the simple purpose to share knowledge if you want a random `BigInteger` value you can use following statement:

```
BigInteger randomVal =
BigInteger.Abs(BigInteger.Parse(Guid.NewGuid().ToString().Replace("-",
NumberStyles.AllowHexSpecifier)));
```

edited Dec 10 '18 at 12:19

answered Dec 8 '18 at 14:38



[Ciro Corvino](#)

1,345 4 11 26

▲ Try these simple steps to create random numbers:

-1

Create function:

▼

```
private int randomnumber(int min, int max)
{
    Random rnum = new Random();
    return rnum.Next(min, max);
}
```

Use the above function in a location where you want to use random numbers. Suppose you want to use it in a text box.

```
textBox1.Text = randomnumber(0, 999).ToString();
```

0 is min and 999 is max. You can change the values to whatever you want.

edited Nov 6 '18 at 15:56



glennsl

11.6k 10 31 50

answered Sep 9 '15 at 1:16



Hani Mehdi

92 1 7

This will return the same number when called multiple times closely together as it uses System time as a seed... – [MOnsDaR](#) Apr 12 '16 at 19:04

-
- 1 randomnumber(0, 999) will never return 999. Max Value is not inclusive. This is common misunderstanding (feature) of Random.Next max value. – [Gordon Bell](#) Jul 26 '16 at 17:15
-



15



it's better to seed the Random object with current milliseconds, to ensure true random number, and you almost won't find duplicates using it many times

```
Random rand = new Random(DateTime.Now.Millisecond);
```

Update

I know that `new Random()` uses the current ticks as seed, but seeding with current milliseconds is still good enough as it's good random start

The last remaining point is that you don't have to initialize `new Random()` every time you need a random number, initiate one Random object then use it as many times as you need inside a loop or whatever

edited Aug 28 '18 at 13:16

answered Nov 1 '15 at 12:30



Mohamed Selim

1,772 16 28

4 `new Random()` uses the current ticks as seed. When you instantiate multiple instances within the same millisecond (as opposed to tick), then you will get the same value returned. – [Hans Kesting](#) Apr 7 '17 at 14:09

7 This is actively bad. `DateTime.Now.Millisecond` (unlike `DateTime.Now.Ticks`) is a number between 0 and 999. If you're creating a new one of these for each random number, you'll only have 1000 possibilities. – [Oren Melzer](#) Aug 16 '18 at 22:47

Use one instance of Random repeatedly

-2

```
// Somewhat better code...
Random rng = new Random();
for (int i = 0; i < 100; i++)
{
    Console.WriteLine(GenerateDigit(rng));
}
...
static int GenerateDigit(Random rng)
{
    // Assume there'd be more logic here really
    return rng.Next(10);
}
```

This article takes a look at why randomness causes so many problems, and how to address them.

<http://csharpindepth.com/Articles/Chapter12/Random.aspx>

edited Jun 14 '18 at 12:30

answered Jun 14 '18 at 12:08



Ankit Agarwal

51 4

Random isn't a thread safe class. If you create a single instance, you should restrict access to it behind a locking mechanism. – Brad M Jan 24 at 17:59

4

```
Random random = new Random ();
int randomNumber = random.Next (lowerBound,upperBound);
```

edited Jun 14 '18 at 11:26



Andrius Naruševičius

5,263 6 36 64

answered Mar 17 '17 at 2:59



ReRoute

168 8

While this code may answer the question, it is better to explain how to solve the problem and provide the code as an example or reference. Code-only answers can be confusing and lack context. – [Robert Columbia](#) Aug 22 '18 at 0:30

For strong random seed I always use CryptoRNG and not Time.

2

```
using System;
using System.Security.Cryptography;

public class Program
{
    public static void Main()
    {
        var random = new Random(GetSeed());
        Console.WriteLine(random.Next());
    }

    public static int GetSeed()
    {
        using (var rng = new RNGCryptoServiceProvider())
        {
            var intBytes = new byte[4];
            rng.GetBytes(intBytes);
            return BitConverter.ToInt32(intBytes, 0);
        }
    }
}
```

answered May 25 '18 at 17:53



Davit Tvildiani

1,311 3 13 25

0

I will assume that you want a uniformly distributed random number generator like below. Random number in most of programming language including C# and C++ is not properly shuffled before using them. This means that you will get the same number over and over, which isn't really random. To avoid to draw the same number over and over, you need a seed. Typically, ticks in time is ok for this task. Remember that you will get the same number over and over if you are using the same seed every time. So try to use varying seed always. Time is a good source for seed because they change always.

```
int GetRandomNumber(int min, int max)
{
    Random rand = new Random((int)DateTime.Now.Ticks);
    return rand.Next(min, max);
}
```

if you are looking for random number generator for normal distribution, you might use a Box-Muller transformation. Check the answer by yoyoyoyosef in Random Gaussian Variable Question. Since you want integer, you have to cast double value to integer at the end.

```
Random rand = new Random(); //reuse this if you are generating many
double u1 = 1.0 - rand.NextDouble(); //uniform(0,1] random doubles
double u2 = 1.0 - rand.NextDouble();
double randStdNormal = Math.Sqrt(-2.0 * Math.Log(u1)) *
    Math.Sin(2.0 * Math.PI * u2); //random normal(0,1)
double randNormal =
    mean + stdDev * randStdNormal; //random normal(mean,stdDev^2)
```

[Random Gaussian Variables](#)

edited May 22 '18 at 13:18

answered May 14 '18 at 21:52



auto9817

38 3 9

Quick and easy for inline, use bellow code:

-2

```
new Random().Next(min, max);
```

```
// for example unique name
```

```
strName += "_" + new Random().Next(100, 999);
```

answered May 10 '18 at 7:48



Zolfaghari

735 7 10

Every time you do new Random() it is initialized . This means that in a tight loop you get the same value lots of times. You should keep a single Random instance and keep using Next on the same instance.

205

```
//Function to get random number
```

```
private static readonly Random getrandom = new Random();
```

```
public static int GetRandomNumber(int min, int max)
```

```
{
```

```
    lock(getrandom) // synchronize
```

```
    {
```

```
        return getrandom.Next(min, max);
```

```
    }
```

```
}
```

edited Feb 12 '18 at 21:37



MechMK1

1,499 5 25 43

answered Oct 4 '10 at 11:52



Pankaj Mishra

11k 14 59 100

-
- 2 Is this not what @Guffa said in his answer 6 months ago? "If you create new instances too close in time, they will produce the same series of random numbers" – [Chris](#) Oct 4 '10 at 12:00
-
- 3 @Chris- That's right what you said. In this I have provided the implementation of that. I think it's a good way of doing it. It works better. – [Pankaj Mishra](#) Oct 4 '10 at 14:16
-
- 21 This is an implementation that synchronises the code for use from several threads. That is good for a multi threaded application, but a waste of time for a single threaded application. – [Guffa](#) Feb 26 '13 at 0:05
-
- 5 if you use the code you can get a lock convoy... – [Offler](#) Jun 5 '13 at 13:20
-
- 1 @testing: I agree that Pankaj's method is the correct one to use. What I'm saying is that it can be simplified to this: //Function to get random number
 private static readonly Random getRandom = new Random(); public static int GetRandomNumber(int min, int max) { lock(getrandom) { //
 synchronize return getRandom.Next(min, max); } } – [Sean Worle](#) Sep 14 '17 at 1:51
-

▲ This is the class I use. Works like `RandomNumber.GenerateRandom(1, 666)`

5

**internal static class RandomNumber**

{

private static Random r = **new Random()**; **private static object** l = **new object()**;


```

private static Random globalRandom = new Random();
[ThreadStatic]
private static Random localRandom;
public static int GenerateNewRandom(int min, int max)
{
    return new Random().Next(min, max);
}
public static int GenerateLockedRandom(int min, int max)
{
    int result;
    lock (RandomNumber.l)
    {
        result = RandomNumber.r.Next(min, max);
    }
    return result;
}
public static int GenerateRandom(int min, int max)
{
    Random random = RandomNumber.localRandom;
    if (random == null)
    {
        int seed;
        lock (RandomNumber.globalRandom)
        {
            seed = RandomNumber.globalRandom.Next();
        }
        random = (RandomNumber.localRandom = new Random(seed));
    }
    return random.Next(min, max);
}
}

```

edited Feb 11 '18 at 17:10



MechMK1

1,499 5 25 43

answered Dec 7 '14 at 8:15



AskethZ

59 1 1

Your GenerateRandom class will never return the number 666, only 665. This is common misunderstanding (feature) of Random.Next max value. – [Gordon Bell](#) Jul 26 '16 at 16:59



Beware that `new Random()` is seeded on current timestamp.

84

If you want to generate **just one number** you can use:



```
new Random().Next( int.MinValue, int.MaxValue )
```

For more information, look at the [Random](#) class, though please note:

However, because the clock has finite resolution, using the parameterless constructor to create different `Random` objects in close succession creates random number generators that produce identical sequences of random numbers

So do not use this code to generate a series of random number.

edited Dec 22 '17 at 22:03

answered Apr 24 '10 at 23:10



[Fyodor Soikin](#)

43.7k 5 69 102

34 -1: The default seed is based on time; do this in a loop and you'll get very non-random results. You should create **one** generator and use it for all your numbers, not a separate generator each time. – [Bevan](#) Apr 24 '10 at 23:39

14 Hey, that's unfair. The question was how to generate random int number. No loops or series were mentioned. – [Fyodor Soikin](#) Apr 25 '10 at 1:45

22 Ok, fair point. Rescinded. Though, I still think that not using `new Random()` in a loop is an important point. – [Bevan](#) Apr 25 '10 at 9:08

12

I've tried all of these solutions excluding the COBOL answer... lol

None of these solutions were good enough. I needed randoms in a fast for int loop and I was getting tons of duplicate values even in very wide ranges. After settling for kind of random results far too long I decided to finally tackle this problem once and for all.

It's all about the seed.

I create a random integer by parsing out the non-digits from Guid, then I use that to instantiate my Random class.

```
public int GenerateRandom(int min, int max)
{
    var seed = Convert.ToInt32(Regex.Match(Guid.NewGuid().ToString(),
    return new Random(seed).Next(min, max);
}
```

Update: Seeding isn't necessary if you instantiate the Random class once. So it'd be best to create a static class and call a method off that.

```
public static class IntUtil
{
    private static Random random;

    private static void Init()
    {
        if (random == null) random = new Random();
    }

    public static int Random(int min, int max)
    {
        Init();
        return random.Next(min, max);
    }
}
```

Then you can use the static class like so..

```
for(var i = 0; i < 1000; i++)
{
    int randomNumber = IntUtil.Random(1,100);
    Console.WriteLine(randomNumber);
}
```

I admit I like this approach better.

edited Aug 20 '16 at 4:50

answered May 19 '15 at 21:19



Proximo

2,001 8 29 47

-
- 4 Guid is not random, it is not a good seed. A GUID doesn't make guarantees about randomness, it makes guarantees around uniqueness. stackoverflow.com/questions/2621563/... – Magu Oct 11 '15 at 19:02
-
- 1 Guid is a good seed. I am only using the numbers in the Guid. Try the method for yourself. Put it in a long for loop and look at the results for yourself. – Proximo Oct 11 '15 at 23:29
-
- 1 Hmm, on second thought.. seed isn't necessary at all. Updating answer – Proximo Oct 11 '15 at 23:49
-
- 1 Great point on the update. I didn't even think to make it a static field, this way works much better and is cleaner. – JCisar Aug 19 '16 at 18:53
-
- 1 There are some problems with this answer. First of all GUID is not a great seed source - just because it looks random doesn't mean it is. It *may* be good enough for your personal needs. Secondly the Random class isn't thread safe. You need to instantiate it once per thread. – Hector Oct 10 '17 at 7:43
-

```
Random rand = new Random();
```

2

```
int name = rand.Next();
```

Put whatever values you want in the second parentheses make sure you have set a name by writing prop and double tab to generate the code

edited Jul 27 '16 at 9:33



shA.t

13.1k 4 38 71

answered Jun 25 '14 at 2:28



user3773367

61 1

1 Prop and double tab? – [DCShannon](#) Sep 23 '14 at 4:32

2 added value compared to the already existing answers? – [chiccodoro](#) Mar 6 '15 at 13:20

Why *prop and double tab*? Are you saying without that keyboard shortcut, which is *short-hand* for property, your code won't work? – [Shadowfax](#) Mar 1 '18 at 14:57

I use below code to have a random number:

7

```
var random = new Random((int)DateTime.Now.Ticks);
var randomValue = random.Next(startValue, endValue + 1);
```

edited Jul 26 '16 at 17:12



Gordon Bell

11.1k 2 38 61

answered Oct 14 '15 at 14:38



shA.t



13.1k 4 38 71

While this is okay:

3

```
Random random = new Random();  
int randomNumber = random.Next();
```

You'd want to control the limit (min and max numbers) most of the time. So you need to specify where the random number starts and ends.

The `Next()` method accepts two parameters, min and max.

So if i want my random number to be between say 5 and 15, I'd just do

```
int randomNumber = random.Next(5, 16)
```

edited Jul 26 '16 at 16:57



Gordon Bell

11.1k 2 38 61

answered Aug 22 '14 at 10:24




ojonugwa ochalifu

16.3k 22 84 111

I wanted to demonstrate what happens when a new random generator is used every time. Suppose you have two methods or two classes each requiring a random number. And naively you code them like:

2



```
public class A
{
    public A()
    {
        var rnd=new Random();
        ID=rnd.Next();
    }
    public int ID { get; private set; }
}
public class B
{
    public B()
    {
        var rnd=new Random();
        ID=rnd.Next();
    }
    public int ID { get; private set; }
}
```

Do you think you will get two different IDs? **NOPE**

```
class Program
{
    static void Main(string[] args)
    {
        A a=new A();
        B b=new B();

        int ida=a.ID, idb=b.ID;
        // ida = 1452879101
        // idb = 1452879101
    }
}
```

The solution is to *a/ways* use a single static random generator. Like this:

```
public static class Utils
{
    public static readonly Random random=new Random();
}

public class A
```

```
{
    public A()
    {
        ID=Utils.random.Next();
    }
    public int ID { get; private set; }
}
public class B
{
    public B()
    {
        ID=Utils.random.Next();
    }
    public int ID { get; private set; }
}
```

answered Apr 8 '16 at 19:24



ja72

18.6k 3 51 106

or to seed the generator.. – [Millie Smith](#) Apr 11 '16 at 0:37

How do you safely choose a seed then? – [ja72](#) Apr 11 '16 at 2:28

Well, first, if your objects are created even 10 ms apart, the random numbers generated are different with the default seed. Second, you can mash whatever random environmental or process data you have around to get a seed. Then there's the tradeoff of whether you want one long sequence of numbers that will most likely start repeating itself, or multiple streams even if two streams end up being identical. And if you're concerned about security, `RNGCryptoServiceProvider` is a better call anyway. – [Millie Smith](#) Apr 11 '16 at 2:59

It would be cool to have a randomizer chip with a small alpha particle radioactive source and a detector (how smoke detector works) in order to randomize numbers based on radioactive decay (which is very random). – [ja72](#) Apr 11 '16 at 13:01

Why not use `int randomNumber = Random.Range(start_range,`

end_range) ?

-1

▼

answered Mar 19 '16 at 16:57



[Recomer](#)

123 1 10

Actually `int randomNumber = Random.Range(start_range, end_range + 1)` – [Gordon Bell](#) Jul 26 '16 at 17:09

2 Does `Random.Range()` exist? I couldn't find it in the MSDN documentation. – [Phillip Ngan](#) Sep 1 '16 at 10:17

I couldn't find `Random.Range()` in the MSDN documentation too – [user2455111](#) Jan 20 '17 at 10:16 ✎

▲

Numbers calculated by a computer through a deterministic process, cannot, by definition, be random.

2

▼

If you want a genuine random numbers, the randomness comes from atmospheric noise or radioactive decay.

You can try for example `RANDOM.ORG` (it reduces performance)

answered Mar 14 '16 at 13:58



[Stefano Lonati](#)

399 2 13

▲

I wanted to add a cryptographically secure version:

27

`RNGCryptoServiceProvider` Class ([MSDN](#) or [dotnetperls](#))

It implements `IDisposable`.

▼

```
using (RNGCryptoServiceProvider rng = new RNGCryptoServiceProvider())
{
    byte[] randomNumber = new byte[4]; // 4 for int32
}
```

```
rng.GetBytes(randomNumber);
int value = BitConverter.ToInt32(randomNumber, 0);
}
```

answered Dec 18 '15 at 21:02



joordan831

591 4 6



The numbers generated by the inbuilt `Random` class (`System.Random`) generates pseudo random numbers.

8



If you want true random numbers, the closest we can get is "secure Pseudo Random Generator" which can be generated by using the Cryptographic classes in C# such as `RNGCryptoServiceProvider`.

Even so, if you still need *true* random numbers you will need to use an external source such as devices accounting for radioactive decay as a seed for a random number generator. Since, by definition, any number generated by purely algorithmic means cannot be truly random.

answered Jul 22 '15 at 15:40



Anshita Arya

91 1 4



```
Random r = new Random();
int n = r.Next();
```

46



answered Apr 24 '10 at 23:10



Joren

12.2k 2 42 51

protected by [cimmanon](#) Oct 19 '15 at 15:09

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 [reputation](#) on this site (the [association bonus](#) does not count).

Would you like to answer one of these [unanswered questions](#) instead?