

The results are in! See what nearly 90,000 developers picked as their most loved, dreaded, and desired coding languages and more in the 2019 Developer Survey.

Identify if a string is a number

[Ask Question](#)

If I have these strings:

635

1. "abc" = false
2. "123" = true
3. "ab2" = false



81

Is there a command, like `IsNumeric()` or something else, that can identify if a string is a valid number?

[c#](#)[string](#)[parsing](#)[isnumeric](#)

edited Jan 29 at 18:23



[Alexander Abakumov](#)

4,986 5 48 73

asked May 21 '09 at 18:06



[Gold](#)

1

77 from their examples you can see they meant if the *whole string* represents a number. – [Lucas](#) May 21 '09 at 18:32

36 return str.All(Char.IsDigit); – [Mohsen](#) Oct 23 '13 at 5:45

10 str.All(Char.IsDigit) will declare "3.14" false as well as "-2" and "3E14". Not to speak of: "0x10" – [Harald Coppoolse](#) Oct 20 '14 at 11:52

- 3 It depends on what type of number you are trying to check. For integer numbers without separator (i.e. strings of decimal digits) this check works, and is the same of the accepted answer and the one implied in OP. – [Alex Mazzariol](#) Aug 22 '15 at 10:01

@AustinSalonen, I don't understand your point. #3 has letters (and only one number) so it should be false. If the user enters "ab2" then the assumption is that they don't understand and therefore we can't assume we understand what they intend. – [user34660](#) Apr 24 '17 at 7:13

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

Q&A for work

[Learn More](#)

23 Answers

987

```
int n;  
bool isNumeric = int.TryParse("123", out n);
```

Update As of C# 7:

```
var isNumeric = int.TryParse("123", out int n);
```

The *var* s can be replaced by their respective types!

edited Jan 19 '18 at 1:22



[Academy of Programmer](#)

17.5k 57 238 391

answered May 21 '09 at 18:08



[mquander](#)

55k 13 86 119

- 111 Though, I would use `double.TryParse`, since we want to know if it represents a number at all. – [John Gietzen](#) May 21 '09 at 18:31


- 4 Function will return true if I pass string as "-123" or "+123". I Understand that integer has positive and negative values. But If this string is coming

from user entered textbox then it should return false. – [user2323308](#)

Aug 28 '13 at 13:58 

- 5 This is a good solution, until a user enters a value beyond -2,147,483,648 to 2,147,483,647, and then this silently fails – [BlackTigerX](#) Oct 23 '14 at 22:05

try parsing 0,60 (that is a comma!) it is an invalid number but will be parsed as 60! – [Paul Zahra](#) Dec 2 '16 at 12:12

- 1 I prefer to have extension method for this check: `public static bool IsNumeric(this string text) { double _out; return double.TryParse(text, out _out); }` – [Hamid Naeemi](#) Jan 21 '18 at 19:05 

I've used this function several times:

117

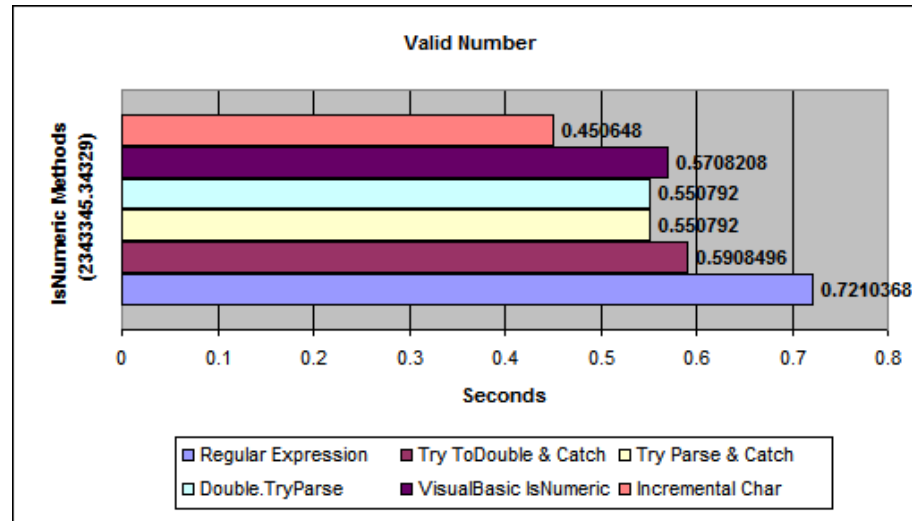
```
public static bool IsNumeric(object Expression)
{
    double retNum;

    bool isNum = Double.TryParse(Convert.ToString(Expression),
        System.Globalization.NumberStyles.Any,
        System.Globalization.NumberFormatInfo.InvariantInfo, out retNum);
    return isNum;
}
```

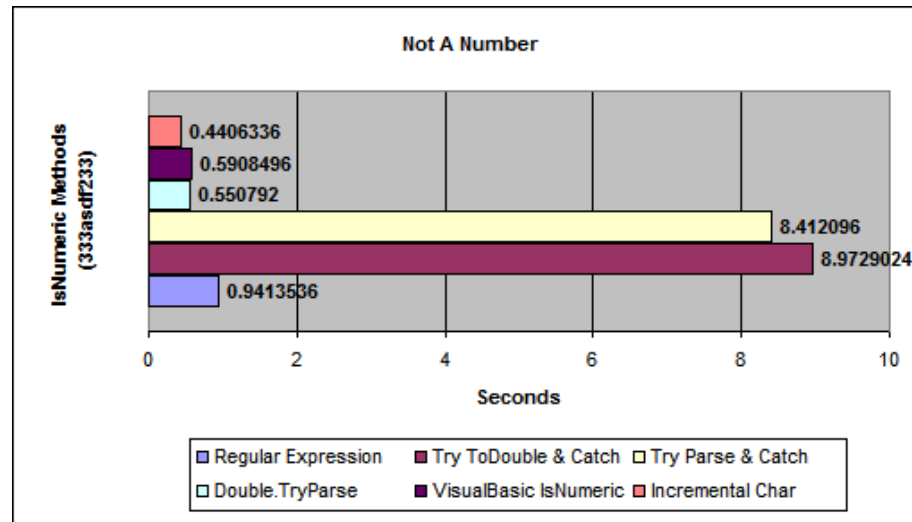
But you can also use;

```
bool b1 = Microsoft.VisualBasic.Information.IsNumeric("1"); //true
bool b2 = Microsoft.VisualBasic.Information.IsNumeric("1aa"); //fal.
```

From [Benchmarking IsNumeric Options](#)



(source: aspalliance.com)



(source: aspalliance.com)

edited Mar 17 at 20:01



Glorfindel

16.7k 11 52 73

answered May 21 '09 at 18:20



Nelson Miranda



4,380 5 27 47

73 referencing Microsoft.VisualBasic.dll from C# app? eww :P – [Lucas](#) May 21 '09 at 18:44

I have no problem to use "IsNumeric" it works good. Also you can see that there's little efficiency difference between TryParse and IsNumeric. Remember that TryParse is new in 2.0 and before then it was better to use IsNumeric than any other strategy. – [Nelson Miranda](#) May 21 '09 at 19:02

8 Well, VB.NET's IsNumeric() internally uses double.TryParse(), after a number of gyrations that are needed (among other things) for VB6 compatibility. If you don't need compatibility, double.TryParse() is just as simple to use, and it saves you from wasting memory by loading Microsoft.VisualBasic.dll in your process. – [Euro Micelli](#) May 21 '09 at 20:06

3 Quick note: using a regular expression will be much faster if you manage to have the underlying finite-state machine built once and for all. Generally, building the state machine takes $O(2^n)$ where n is the length of the regex, whereas reading is $O(k)$ where k is the length of the string being searched. So rebuilding the regex every time introduces a bias. – [Clément](#) Jan 5 '11 at 17:29

2 @Lucas Actually, there's some really nice stuff in there, like a full csv parser. No reason not to use it if it exists in there. – [Nyerguds](#) Apr 4 '16 at 12:04



I guess this answer will just be lost in between all the other ones, but anyway, here goes.

7



I ended up on this question via Google because I wanted to check if a string was numeric so that I could just use `double.Parse("123")` instead of the `TryParse()` method.

Why? Because it's annoying to have to declare an `out` variable and check the result of `TryParse()` before you know if the parse failed or

not. I want to use the ternary operator to check if the string is numerical and then just parse it in the first ternary expression or provide a default value in the second ternary expression.

Like this:

```
var doubleValue = IsNumeric(numberAsString) ? double.Parse(numberAsS
```

It's just a lot cleaner than:

```
var doubleValue = 0;
if (double.TryParse(numberAsString, out doubleValue)) {
    //whatever you want to do with doubleValue
}
```

I made a couple extension methods for these cases:

Extension method one

```
public static bool IsParseableAs<TInput>(this string value) {
    var type = typeof(TInput);

    var tryParseMethod = type.GetMethod("TryParse", BindingFlags.Sta
BindingFlags.Public, Type.DefaultBinder,
    new[] { typeof(string), type.MakeByRefType() }, null);
    if (tryParseMethod == null) return false;

    var arguments = new[] { value, Activator.CreateInstance(type) };
    return (bool) tryParseMethod.Invoke(null, arguments);
}
```

Example:

```
"123".IsParseableAs<double>() ? double.Parse(sNumber) : 0;
```

Because `IsParseableAs()` tries to parse the string as the appropriate type instead of just checking if the string is "numeric" it should be

pretty safe. And you can even use it for non numeric types that have a `TryParse()` method, like `DateTime`.

The method uses reflection and you end up calling the `TryParse()` method twice which, of course, isn't as efficient, but not everything has to be fully optimized, sometimes convenience is just more important.

This method can also be used to easily parse a list of numeric strings into a list of `double` or some other type with a default value without having to catch any exceptions:

```
var sNumbers = new[] { "10", "20", "30" };
var dValues = sNumbers.Select(s => s.TryParseAs<double>()) ? double :
```

Extension method two

```
public static TOutput ParseAs<TOutput>(this string value, TOutput defaultValue)
{
    var type = typeof(TOutput);

    var tryParseMethod = type.GetMethod("TryParse", BindingFlags.Static |
        BindingFlags.Public, Type.DefaultBinder,
        new[] { typeof(string), type.MakeByRefType() }, null);
    if (tryParseMethod == null) return defaultValue;

    var arguments = new object[] { value, null };
    return ((bool) tryParseMethod.Invoke(null, arguments)) ? (TOutput)
        defaultValue;
}
```

This extension method lets you parse a `string` as any `type` that has a `TryParse()` method and it also lets you specify a default value to return if the conversion fails.

This is better than using the ternary operator with the extension method above as it only does the conversion once. It still uses reflection though...

Examples:

```
"123".ParseAs<int>(10);
"abc".ParseAs<int>(25);
"123,78".ParseAs<double>(10);
"abc".ParseAs<double>(107.4);
"2014-10-28".ParseAs<DateTime>(DateTime.MinValue);
"monday".ParseAs<DateTime>(DateTime.MinValue);
```

Outputs:

```
123
25
123,78
107,4
28.10.2014 00:00:00
01.01.0001 00:00:00
```

edited Jan 7 at 23:12



[bubbleking](#)

1,112 1 14 28

answered Mar 7 '14 at 13:07



[Hein Andre Grønnestad](#)

5,511 1 21 36

- 3 I believe you may have invented one of the most inefficient approaches I've seen yet. Not only are you parsing the string twice (in the case that it's parseable), you are also calling *reflection* functions multiple times to do it. And, in the end, you don't even save any keystrokes using the extension method. – [JDB](#) Mar 25 '14 at 16:53

Thank you for just repeating what I wrote myself in the second to last paragraph. Also if you take my last example into account you definitely save keystrokes using this extension method. This answer doesn't claim to be some kind of a magic solution to any problem, it's merely a code example. Use it, or don't use it. I think it's convenient when used right. And it includes examples of both extension methods and reflection, maybe someone can learn from it. – [Hein Andre Grønnestad](#) Mar 25 '14 at 21:40

- 5 Have you tried `var x = double.TryParse("2.2", new double())` ?

`double.Parse("2.2") : 0.0; ?` – JDB Mar 26 '14 at 14:42

2 Yes, and it doesn't work. Argument 2 must be passed with the 'out' keyword and if you specify out as well as new you get A ref or out argument must be an assignable variable . – Hein Andre Grønnestad Oct 22 '15 at 7:50

1 **Performance** TryParse is better than all exposed here. Results: TryParse 8 Regex 20 PHP IsNumeric 30 Reflections TryParse 31 Test code dotnetfiddle.net/x8GjAF – prampte Aug 25 '17 at 7:42



1



The best flexible solution with .net built-in function called- `char.IsDigit` . It works with unlimited long numbers. It will only return true if each character is a numeric number. I used it lot of times with no issues and much easily cleaner solution I ever found. I made a example method.Its ready to use. In addition I added validation for null and empty input. So the method is now totally bulletproof

```
public static bool IsNumeric(string strNumber)
{
    if (string.IsNullOrEmpty(strNumber))
    {
        return false;
    }
    else
    {
        int numberOfChar = strNumber.Count();
        if (numberOfChar > 0)
        {
            bool r = strNumber.All(char.IsDigit);
            return r;
        }
        else
        {
            return false;
        }
    }
}
```

edited Dec 8 '18 at 3:35

answered Dec 8 '18 at 3:17



Liakat Hossain

510 6 14

UPDATE of Kunal Noel Answer

2

```
stringTest.All(char.IsDigit);  
// This returns true if all characters of the string are digits.
```

But, for this case we have that empty strings will pass that test, so, you can:

```
if (!string.IsNullOrEmpty(stringTest) && stringTest.All(char.IsDigit)  
    // Do your logic here  
}
```

answered Oct 18 '18 at 17:55



dayanrr91

104 2 12

2

```
public static bool IsNumeric(this string input)  
{  
    int n;  
    if (!string.IsNullOrEmpty(input)) // .Replace('.', null).Replace('.',  
    {  
        foreach (var i in input)  
        {
```

```

        if (!int.TryParse(i.ToString(), out n))
        {
            return false;
        }
        return true;
    }
    return false;
}

```

edited Aug 13 '18 at 7:25

answered Jul 31 '18 at 7:27



OMANSAK
151 2 12



Use these extension methods to clearly distinguish between a check if the string is *numerical* and if the string *only* contains 0-9 digits

2



```

public static class ExtensionMethods
{
    /// <summary>
    /// Returns true if string could represent a valid number, including
    local culture symbols
    /// </summary>
    public static bool IsNumeric(this string s)
    {
        decimal d;
        return decimal.TryParse(s, System.Globalization.NumberStyles
System.Globalization.CultureInfo.CurrentCulture, out d);
    }

    /// <summary>
    /// Returns true only if string is wholly comprised of numerical characters
    /// </summary>
    public static bool IsNumbersOnly(this string s)
    {
        foreach (char c in s)
        {
            if (!char.IsDigit(c))
            {
                return false;
            }
        }
        return true;
    }
}

```

```
{
    if (s == null || s == string.Empty)
        return false;

    foreach (char c in s)
    {
        if (c < '0' || c > '9') // Avoid using .IsDigit or .IsNu
            return true for other characters
        return false;
    }

    return true;
}
```

answered Jul 3 '18 at 9:59



userSteve

770 9 22

▲ You can also use:

155

stringTest.All(char.IsDigit);

▼ It will return true for all Numeric Digits (not float) and false if input string is any sort of alphanumeric.

Please note: stringTest should not be an empty string as this would pass the test of being numeric.

edited Apr 30 '18 at 9:53



MatSnow



5,396 2 12 26

answered Oct 20 '14 at 12:06



Kunal Goel

1,919 1 9 18

- 13 That's very cool. One thing to be aware of though: an empty string will pass that test as being numeric. – [dan-gph](#) Jun 5 '15 at 6:04
- 2 @dan-gph : I am glad, you like it. Yes, you are correct. I have updated note above. Thanks! – [Kunal Goel](#) Jun 6 '15 at 7:15 
- 1 this also does not work for decimal cases. The right test will be `stringTest.All(l => char.IsDigit(l) || '.' == l || '-' == l)`; – [Salman Hasrat Khan](#) Feb 24 '16 at 14:28
- Thanks for your input Salman, To specifically check decimal out of a string, you can go for - `if (Decimal.TryParse(stringTest2, out value)) { /* Yes, Decimal */ } else { /* No, Not a Decimal */ }` – [Kunal Goel](#) Feb 26 '16 at 5:38 
- 4 Salman, it's not that simple- this would pass `...--` as a valid number. Clearly not. – [Flynn1179](#) Apr 11 '16 at 12:26

With c# 7 it you can inline the out variable:

2

```
if(int.TryParse(str, out int v))
{
}
```

answered Nov 9 '17 at 23:39



[Chad Kuehn](#)

3,057 26 29

If you want to check if a string is a number (I'm assuming it's a string since if it's a number, duh, you know it's one).

9

- Without regex and
- using Microsoft's code as much as possible

you could also do:

```
public static bool IsNumber(this string aNumber)
{
    BigInteger temp_big_int;
    var is_number = BigInteger.TryParse(aNumber, out temp_big_int);
    return is_number;
}
```

This will take care of the usual nasties:

- Minus (-) or Plus (+) in the beginning
- ~~contains decimal character~~ BigIntegers won't parse numbers with decimal points. (So: `BigInteger.Parse("3.3")` will throw an exception, and `TryParse` for the same will return false)
- no funny non-digits
- covers cases where the number is bigger than the usual use of `Double.TryParse`

You'll have to add a reference to `System.Numerics` and have `using System.Numerics;` on top of your class (well, the second is a bonus I guess :)

edited Jun 13 '16 at 18:06



Peter Mortensen

13.9k 19 87 114

answered May 15 '14 at 23:24



Noctis

9,746 3 31 65



```
//To my knowledge I did this in a simple way
static void Main(string[] args)
{
```

-6

```

string a, b;
int f1, f2, x, y;
Console.WriteLine("Enter two inputs");
a = Convert.ToString(Console.ReadLine());
b = Console.ReadLine();
f1 = find(a);
f2 = find(b);

if (f1 == 0 && f2 == 0)
{
    x = Convert.ToInt32(a);
    y = Convert.ToInt32(b);
    Console.WriteLine("Two inputs r number \n so that addition o
" + (x + y).ToString());
}
else
    Console.WriteLine("One or two inputs r string \n so that con
text box is = " + (a + b));
    Console.ReadKey();
}

static int find(string s)
{
    string s1 = "";
    int f;
    for (int i = 0; i < s.Length; i++)
        for (int j = 0; j <= 9; j++)
        {
            string c = j.ToString();
            if (c[0] == s[i])
            {
                s1 += c[0];
            }
        }

    if (s == s1)
        f = 0;
    else
        f = 1;

    return f;
}

```

edited Jun 13 '16 at 18:03



Peter Mortensen

13.9k 19 87 114

answered Apr 7 '13 at 3:15



Vino

1

1 Four downvotes, but nobody has said why? I presume it's because TryParse/Parse would be a better option, but not everybody coming here will know that. – [njplumridge](#) May 26 '17 at 13:07

2 You made it so complicated that even C programmer would say "gosh, there have to be an easier way to write that" – [Ch3shire](#) Oct 20 '17 at 22:31

1. There is no reason to read TWO numbers from console and adding them. Where the string comes from is irrelevant anyways, so there is no reason to read anything from the console at all. – [Algoman](#) Jul 13 '18 at 9:57

2. The variable for f is unnecessary, you could return 0 or 1 directly - if you want a single return, you could use the ternary operator for that. int is also the wrong return type for find, it should be bool and you could return s==s1 – [Algoman](#) Jul 13 '18 at 9:59

3. you copy the digits of s to s1 and then compare s to s1. This is much slower than it needs to be. Also why do you continue the inner loop even if c[0]==s[i] has happened? Do you expect s[i] to be equal to other digits, too? – [Algoman](#) Jul 13 '18 at 10:01



This is probably the best option in C#.

32

If you want to know if the string contains a whole number (integer):



```
string someString;
// ...
int myInt;
bool isNumerical = int.TryParse(someString, out myInt);
```


The TryParse method will try to convert the string to a number (integer) and if it succeeds it will return true and place the corresponding number in myInt. If it can't, it returns false.

Solutions using the `int.Parse(someString)` alternative shown in other responses works, but it is much slower because throwing exceptions is very expensive. `TryParse(...)` was added to the C# language in version 2, and until then you didn't have a choice. Now you do: you should therefore avoid the `Parse()` alternative.

If you want to accept decimal numbers, the decimal class also has a `.TryParse(...)` method. Replace int with decimal in the above discussion, and the same principles apply.

edited May 31 '16 at 20:01



Community ♦

1 1

answered May 21 '09 at 18:16



Euro Micelli

27.8k 6 42 64

14

I know this is an old thread, but none of the answers really did it for me - either inefficient, or not encapsulated for easy reuse. I also wanted to ensure it returned false if the string was empty or null. TryParse returns true in this case (an empty string does not cause an error when parsing as a number). So, here's my string extension method:

```
public static class Extensions
{
    /// <summary>
    /// Returns true if string is numeric and not empty or null or w
    /// Determines if string is numeric by parsing as Double
    /// </summary>
    /// <param name="str"></param>
```

```

    /// <param name="style">Optional style - defaults to NumberStyle:
    and trailing whitespace, leading and trailing sign, decimal point and
    separator) </param>
    /// <param name="culture">Optional CultureInfo - defaults to
    InvariantCulture</param>
    /// <returns></returns>
    public static bool IsNumeric(this string str, NumberStyles style
    NumberStyles.Number,
        CultureInfo culture = null)
    {
        double num;
        if (culture == null) culture = CultureInfo.InvariantCulture;
        return Double.TryParse(str, style, culture, out num) &&
        !String.IsNullOrWhiteSpace(str);
    }
}

```

Simple to use:

```

var mystring = "1234.56789";
var test = mystring.IsNumeric();

```

Or, if you want to test other types of number, you can specify the 'style'. So, to convert a number with an Exponent, you could use:

```

var mystring = "5.2453232E6";
var test = mystring.IsNumeric(style: NumberStyles.AllowExponent);

```

Or to test a potential Hex string, you could use:

```

var mystring = "0xF67AB2";
var test = mystring.IsNumeric(style: NumberStyles.HexNumber)

```

The optional 'culture' parameter can be used in much the same way.

It is limited by not being able to convert strings that are too big to be contained in a double, but that is a limited requirement and I think if you are working with numbers larger than this, then you'll probably need additional specialised number handling functions anyway.

answered Sep 29 '15 at 9:44



cyberspy

707 8 12

-
- 1 Works great, except that `Double.TryParse` doesn't support `NumberStyles.HexNumber`. See MSDN `Double.TryParse`. Any reason why you `TryParse` before checking for `IsNullOrWhiteSpace`? `TryParse` returns false if `IsNullOrWhiteSpace` doesn't it? – Harald Coppoolse Nov 16 '15 at 8:02
-

[Double.TryParse](#)

6

```
bool Double.TryParse(string s, out double result)
```

edited Dec 10 '14 at 19:04

answered May 21 '09 at 18:11



John Pirie

4,487 2 28 45

11

If you want to catch a broader spectrum of numbers, à la PHP's [is_numeric](#), you can use the following:

```
// From PHP documentation for is_numeric
// (http://php.net/manual/en/function.is-numeric.php)

// Finds whether the given variable is numeric.

// Numeric strings consist of optional sign, any number of digits, o
```

```

and optional
// exponential part. Thus +0123.45e6 is a valid numeric value.

// Hexadecimal (e.g. 0xf4c3b00c), Binary (e.g. 0b10100111001), Octal
notation is allowed too but
// only without sign, decimal and exponential part.
static readonly Regex _isNumericRegex =
    new Regex( "^(" +
        /*Hex*/ @"0x[0-9a-f]+" + "|" +
        /*Bin*/ @"0b[01]+" + "|" +
        /*Oct*/ @"0[0-7]+" + "|" +
        /*Dec*/ @"((?!0)|[-+]|(?=0+\.))(\d*\.)?\d+(e\d+)?" +
        ")$" );

static bool IsNumeric( string value )
{
    return _isNumericRegex.IsMatch( value );
}

```

Unit Test:

```

static void IsNumericTest()
{
    string[] l_unitTests = new string[] {
        "123",      /* TRUE */
        "abc",      /* FALSE */
        "12.3",     /* TRUE */
        "+12.3",    /* TRUE */
        "-12.3",    /* TRUE */
        "1.23e2",   /* TRUE */
        "-1e23",    /* TRUE */
        "1.2ef",    /* FALSE */
        "0x0",      /* TRUE */
        "0xfff",    /* TRUE */
        "0xf1f",    /* TRUE */
        "0xf1g",    /* FALSE */
        "0123",     /* TRUE */
        "0999",     /* FALSE (not octal) */
        "+0999",    /* TRUE (forced decimal) */
        "0b0101",   /* TRUE */
        "0b0102"    /* FALSE */
    };

    foreach ( string l_unitTest in l_unitTests )
        Console.WriteLine( l_unitTest + " => " + IsNumeric( l_unitTe

```

```
Console.ReadKey( true );
}
```

Keep in mind that just because a value is numeric doesn't mean it can be converted to a numeric type. For example,

[illegible]

edited Mar 25 '14 at 13:35

answered Apr 30 '13 at 15:04

 **JDB**
17.2k 4 54 96

Not trying to be a smart alec here, but this seems to fail for string "0". My Regex is non-existent. Is there a simple tweak for that? I get "0" and possibly "0.0" and even "-0.0" as possible valid numerics. – [Steve Hibbert](#)
Mar 25 '14 at 10:41

@SteveHibbert - Everyone knows that "0" isn't a number! Seriously though... adjusted the regex to match 0. – JDB Mar 25 '14 at 13:30

Hmmm, is it me, or is "0" still not recognised as numeric? – [Steve Hibbert](#)
Mar 25 '14 at 15:35

- 1 Being lazy, and regex-ignorant, I cut'n'pasted the code above, which looks like it includes the "0.0" type change. I ran a test to check that a string "0" running `.IsNumeric()`, and that returns false. I'm thinking that the Octal test will return true for anything that has two numeric chars where the first is zero (and the second is zero to seven), but will return false for just a big fat lonely zero on it's own. If you test "0", with the code above, do you get false? Apologies, if I knew more regex I'd be able to give better feedback. Must read up. – [Steve Hibbert](#) Mar 26 '14 at 9:21
-
- 1 !Doh! Just re-read your comment above, I had missed the additional asterisk, I only updated the decimal line. With that in place, you're right, "0" `IsNumeric`. Apologies for the faffing about, and thanks very much for the update, hope it helps others out too. Much obliged. – [Steve Hibbert](#) Mar 26 '14 at 13:58



You can always use the built in TryParse methods for many datatypes to see if the string in question will pass.

24



Example.

```
decimal myDec;  
var Result = decimal.TryParse("123", out myDec);
```

Result would then = True

```
decimal myDec;  
var Result = decimal.TryParse("abc", out myDec);
```

Result would then = False

edited Feb 27 '14 at 14:37



Trevor

1,792 15 17

answered May 21 '09 at 18:09



TheTXI

32.9k 9 76 109

I think I may have done that more in VB style syntax than C#, but the same rules apply. – TheTXI May 21 '09 at 18:10



Pull in a reference to Visual Basic in your project and use its Information.IsNumeric method such as shown below and be able to


```
//string is not a number
}
```

answered Jan 2 '13 at 11:32



Arun

1,723

8

27

46



This will return true if `input` is all numbers. Don't know if it's any better than `TryParse`, but it will work.

333



```
Regex.IsMatch(input, @"^\d+$")
```

If you just want to know if it has one or more numbers mixed in with characters, leave off the `^` `+` and `$`.

```
Regex.IsMatch(input, @"\d")
```

Edit: Actually I think it is better than `TryParse` because a very long string could potentially overflow `TryParse`.

answered May 21 '09 at 19:08



John M Gant

11.4k

16

54

81

2 Building the regex once and for all would be much more efficient, though.
– Clément Jan 5 '11 at 17:30

2 @CFP +1...RegEx are always better than usual functions, when applicable! – MAXE Jun 27 '12 at 15:32

19 @MAXE: I would not agree. Regular expression checks are quite slow, so there are often better solutions if performance is under consideration. – Michal B. Dec 18 '12 at 12:49

6 edit: you can add `RegexOptions.Compiled` as a parameter if you're

running thousands of these for a possible speed increase

`Regex.IsMatch(x.BinNumber, @"^\d+$", RegexOptions.Compiled)` –
[Simon_Weaver](#) Nov 22 '13 at 22:39

6 will also fail on negatives and things with . – [Noctis](#) May 15 '14 at 23:13

Here is the C# method. [Int.TryParse Method \(String, Int32\)](#)

edited May 21 '09 at 19:01

0

answered May 21 '09 at 18:35



[Syed Tayyab Ali](#)

2,575 6 25 35

you could at least mention the method name in your answer... – [Lucas](#)
 May 21 '09 at 18:55

method name specified! – [Syed Tayyab Ali](#) May 21 '09 at 19:01

In case you don't want to use `int.Parse` or `double.Parse`, you can roll your own with something like this:

19

```
public static class Extensions
{
    public static bool IsNumeric(this string s)
    {
        foreach (char c in s)
        {
            if (!char.IsDigit(c) && c != '.')
            {
                return false;
            }
        }
    }
}
```

```

    }
    return true;
}
}

```

answered May 21 '09 at 18:13



BFree

86.3k

18

139

193

-
- 3 you might as well return `s.All(c => c.IsDigit(c) || c == '.')`, but... – [Lucas](#) May 21 '09 at 18:42
-
- 5 What if they meant integers only? What about locales where '.' is the group separator, not the comma (e.g. pt-Br)? what about negative numbers? group separators (commas in English)? currency symbols? `TryParse()` can manage all of these as required using `NumberStyles` and `IFormatProvider`. – [Lucas](#) May 21 '09 at 18:43
-
- 11 1.3.3.8.5 is not really a number, though, while 1.23E5 is. – [Clément](#) Jan 5 '11 at 17:31
-
- 3 the logic is flawed. -1 – [Russel Yang](#) Apr 29 '13 at 21:33
-
- 1 @Lucas I agree that `TryParse` handles more, but sometimes that's not needed. I just need to validate my credit card number boxes (which can only have digits). This solution is almost definitely faster than `try parse`. – [Millie Smith](#) Sep 12 '14 at 16:41
-



If you want to know if a string is a number, you could always try parsing it:

5



```

var numberString = "123";
int number;

int.TryParse(numberString, out number);

```

Note that `TryParse` returns a `bool`, which you can use to check if your parsing succeeded.

answered May 21 '09 at 18:11



[Gabriel Florit](#)

2,348 1 16 33



You can use `TryParse` to determine if the string can be parsed into an integer.

8



```
int i;  
bool bNum = int.TryParse(str, out i);
```

The boolean will tell you if it worked or not.

answered May 21 '09 at 18:10



[Craig](#)

7,208 13 38 58

protected by [Matt Fenwick](#) Oct 31 '13 at 18:51

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?