# Char.IsDigit() matches more than just "0" through "9"

devblogs.microsoft.com/oldnewthing/20040309-00

March 9, 2004

Raymond Chen

Warning: .NET content ahead!

Yesterday, Brad Abrams noted that Char.IsLetter() matches more than just “A” through “Z”.

What people might not realize is that Char.IsDigit() matches more than just “0” through “9”.

> Valid digits are members of the following category in UnicodeCategory:
> **DecimalDigitNumber**.

But what exactly is a **DecimalDigitNumber**?

> **DecimalDigitNumber**
> Indicates that the character is a decimal digit; that is, in the range 0 through 9. Signified by the Unicode designation “Nd” (number, decimal digit). The value is 8.

At this point you have to go to the Unicode Standard Committee to see exactly what qualifies as “Nd”, and then you get lost in a twisty maze of specifications and documents, all different.

So let’s run an experiment.

```
class Program {
  public static void Main(string[] args) {
    System.Console.WriteLine(
      System.Text.RegularExpressions.Regex.Match(
        "\x0661\x0662\x0663", // "١٢٣"
        "^\\d+$").Success);
    System.Console.WriteLine(
      System.Char.IsDigit('\x0661'));
  }
}
```

The characters in the string are Arabic digits, but they are still digits, as evidenced by the program output:

```
True
True
```

Uh-oh. Do you have this bug in your parameter validation? (More examples..) If you use a pattern like `@"^\d$"` to validate that you receive only digits, and then later use `System.Int32.Parse()` to parse it, then I can hand you some Arabic digits and sit back and watch the fireworks. The Arabic digits will pass your validation expression, but when you get around to using it, boom, you throw a `System.FormatException` and die.

Raymond Chen

**Follow**