Is it a Number?

From: Tom Christiansen <tchrist@perl.com>
Subject: Re: Numeric Test?
Newsgroups: comp.lang.perl.misc
Date: Thu Nov 16 18:25:27 MST 1995

In comp.lang.perl.misc, mlt.software@ortel.org writes:

> *How do I test a string to see if it's a number?*

I hate this question.

> *I'm sure there's an obvious answer to this,*

```
if ($n == 0) { warn "icky number" }
```

> *but I haven't yet found or figured it out. Thanks...*

Why do you want to see whether it's a number? Perl is happy to use strings and number interchangeably. People make way to

Assuming you don't care about whether something's zero, merely do this:

```
do {
    print "Number, please: ";
    $answer = <STDIN>;
    print "Bad number\n" if $answer == 0;
} until $answer;
```

If you do care about getting 0's, then do this:

```
do {
    print "Number, please: ";
    $answer = <STDIN>;
    if ($answer == 0 && $answer ne '0') {
        print "Bad number\n";
    }
} until $answer;
```

If you get bitched at for the warnings that the improper numeric conversion provides, as I'm sure I'm about to, do this:

```
do {
    print "Number, please: ";
    $answer = <STDIN>;
    local $^W = 0;
    if ($answer == 0 && $answer ne '0') {
        print "Bad number\n";
    }
```

```
    } until $answer;
```

If you want to wrap it in a function, do this:

```
    sub bogus_number {
        my $potential_number = shift;
        local $^W = 0;
        my $bogosity = $potential_number == 0
                    && $potential_number ne '';
        return $bogosity;
    }
```

Hm... one of these days we're going to have deal with this problem of maybe getting eof. Remember you can't actually test for eof explicitly, or you'll hose the interactive user.

```
    do {
        print "Number, please: ";
        exit unless defined ($answer = <STDIN>);
        if (bogus_number($answer)) {
            print "Bad number\n";
            $answer = 0;
        }
    } until $answer;
```

You could even be cruel and clobber their input:

```
    sub bogus_number {
        local $^W = 0;
        if ($_[0] == 0 && $_[0] ne '') {
            $_[0] = ''; # squish my caller!
            return 1;
        }
        return 0;
    }

    do {
        print "Number, please: ";
        $answer = <STDIN>;
        print "Bad number\n" if bogus_number($answer);
    } until $answer;
```

Or write it the other way:

```
    do {
        print "Number, please: ";
        $answer = <STDIN>;
    } until nifty_number($answer);

    sub nifty_number {
        my $potential_number = shift;
        local $^W = 0;
        my $bogosity = $potential_number == 0
                    && $potential_number ne '';
        return !$bogosity;
    }
```

Someone is going to ask the question ‘‘Can't I use a regular expression do this?’’ Why, yes, Virginia, you may, and don't say can. :-) Chew on this output from an old paper-tape processing machine:

```
sub nifty_number {
    $_[0] =~ m{    # YANETUT
                ^ ( [+-]? )
                  (?= \d
                    | \.\d
                  )
                  \d*
                  ( \. \d* ) ?
                  (   [Ee] ( [+-]? \d+ ) ) ?
                $
    }x
}
```

Have a nice day. I really don't understand why people get so wigged out about this is-it-a-number thing.

--tom