

Re: Using \$_ in a function if no argument is passed

Source: <http://coding.derkeiler.com/Archive/Perl/perl.beginners/2004-04/0107.html>

From: Beau E. Cox (beau_at_beaucox.com)

Date: 04/02/04

To: beginners@perl.org

Date: Fri, 2 Apr 2004 07:18:25 -1000

On Friday 02 April 2004 07:04 am, Beau E. Cox wrote:

> *On Friday 02 April 2004 06:37 am, JupiterHost.Net wrote:*

> > *Hello List,*

> >

> > *It just occurred to me that many Perl functions use \$_ if not other*

> > *value is supplied. chomp for instance..., which is very handy...*

> >

> > *If one wanted to write a function that used either the given argument or*

> > *\$_ how would you do that?*

> >

> > *myfunc(\$myvalue);*

> > *or*

> > *myfunc; #uses the current value of \$_*

> >

> > *sub myfunc {*

> >

> > *my \$func_arg = shift || '???'; # you wouldn't just do '\$_;' would*

> > *you?*

> >

> > *...*

>

> *Hi –*

>

> *I think what you are trying to do is modify a passed argument – like chomp*

> *does – which really has nothing to do with \$_. Normaly a subroutine gets*

> *its arguments from the @_ array and puts them in a private variable,*

> *does its thing, and returns a value. This is commonly called 'pass by*

> *reference' and is a nice, safe way to do things. If you want to operate*

> *on the passed arguments themselves ('pass by value') – which can be more*

> *error prone (at least in some cases), you can do that – as chomp does.*

>

> *In the sample below, 'to_lower' modifies the incoming argument, and*

> *'to_upper' does not.*

>

> *#!/bin/perl*

>

> *use strict;*

perl.beginners: Re: Using \$_ in a function if no argument is passed

```
> use warnings;
>
> my $string = 'hello';
> print "before to_upper: $string\n";
> to_upper( $string );
> print " after to_upper: $string\n";
> to_lower( $string );
> print " after to_lower: $string\n";
>
> sub to_upper
> {
> # actually modified the incoming argument the –
> # 0th element of @_.
> $_[0] = uc $_[0];
> }
>
> sub to_lower
> {
> # traditional approach
> my $string = shift;
> lc $string;
> return $string;
> }
>
> When run, it returns:
>
> before to_upper: hello
> after to_upper: HELLO
> after to_lower: HELLO
>
```

That's all well and good, but after re-reading your query, here is what you really want:

```
#!/bin/perl

use strict;
use warnings;

my $string = 'hello';
print "before to_upper w/arg: $string\n";
to_upper( $string );
print " after to_upper w/arg: $string\n";

$_ = 'hiya';
print "before to_upper no/arg: $_\n";
&to_upper;
print " after to_upper no/arg: $_\n";

sub to_upper
{
```

perl.beginners: Re: Using \$_ in a function if no argument is passed

```
# point to passed arg or caller's $_  
my $arg = $_[0] ? $_[0] : $_;  
# modify in place  
$$arg = uc $$arg;  
}
```

Aloha => Beau;