

Perl Training



Common Practices

<http://perlwizard.org/perl/week16>

Agenda

- Class Stuff

Useful use statements

use warnings;

Gives “helpful” warnings

use strict;

Forces you to use good programming

techniques

Common Blunders

- Putting a comma after the filehandle in a print statement

```
print STDOUT, "hello";           # Wrong
print STDOUT "hello";           # Right
```
- Using `==` instead of `eq` and `!=` instead of `ne`
 - `==` and `!=` are numeric tests, use them for numbers
 - `eq` and `ne` are string tests, use them for strings

```
"123" == "123.00"           # True
"123" eq "123.00"           # False
```
- Forgetting the trailing semicolon
 - All Perl statements are terminated by a `;`
 - A here document is `"print <<'EOF';"` not `"print <<'EOF'"`

Points to Remember

- Some operations behave differently based on context
 - `$line = <FILE>;#` Grabs one line
 - `@line = <FILE>;` # Grabs all lines
- `<FH>` is for reading from files
 - `$line = <FH>;` # Good
 - `print <FH> "Hello";` # Bad
- Using `$_`
 - `while (<FH>) { }` # Data assigned to `$_`
 - `<FH>;` # Data read, but discarded
 - `$line = <FH>;` # Correct way to do it

Even More Points to Remember

- Don't use = when you mean =~
 - `if($x = /foo/) { }` # Searches \$_ for foo, puts result in \$x
 - `if($x =~ /foo/) { }` # Searches \$x for foo, discards result
- Use my as much as possible, local allows you to “hide” a global variable, which can lead to unexpected results
- Use elsif, not else if
 - `if (COND) {} else if (COND2) {}` # Wrong
 - `if (COND) {} elsif (COND2) {}` # Right

Can you remember

- Don't forget to chomp things
 - `chomp($line = <STDIN>);`
 - `chomp($result = qx!date!);`
- Remember that `||` and `&&` are short circuit operators
 - `($tmp == 0) || ($tmp = 7)` # Assigns \$tmp = 7 if it's == 0
- Remember the difference between `==` and `=`
 - `if($tmp = 7) { }` # Always True!!!
 - `if($tmp == 7) { }` # Only true if \$tmp equals 7

More More More!!!

- There are many ways to do things

- `$foo = $a || $b || $c;`

- `if($a) {
 $foo = $a;
} elseif ($b) {
 $foo = $b;
} else {
 $foo = $c;
}`

`# Both do the same thing, but
the first is more compact`

Info Info Info

- Use parentheses to be safe
 - $(8 + (((5 * 6) / 7) - 3))$
- Line the same levels up vertically

```
$tmp = 7;
if ($tmp == 3) {
    print "\$tmp is 3\n";
}
```
- Use the `= ? :` operator when it's more readable
 - `$tmp = ($value == 7) ? "True" : "False";`

Questions?

