

`print` in perl doesn't include what?

A newline character.

List the variable type syntax.

\$ scalar

@ array

% hash

& subroutine

* typeglob

What's the difference between double, single quotes, and back ticks?

Double quotes perform variable interpolation and backslash interpretation. Single quotes do not.

Back ticks capture the output from executing a command.

What are the default values?

" " or 0, as appropriate.

How are scalar values interpreted?

As the expected type, depending on context.

What context do double quotes provide?

Assignment to arrays/ hashes?

interpolative context

list context

Create a list using literal syntax.

Unpack that list.

```
@threeprimes = (2, 3, 5);  
($a, $b, $c) = @threeprimes;
```

Index an array.

Update an array.

Either involves a scalar so use \$ not @.

```
$lst[n] = new_el  
print $lst[n]
```


Create a hash using literal syntax.

```
@birthmonths = (  
  "John" => "February",  
  "Mary" => "March",  
);
```

Arrows are just a nicer way of writing more arrows.

Look up an element in a hash.

Update a hash.

```
$hash{"key"}  
$hash{"key"} = val
```

As with arrays, notice the use of `$` when dealing with individual elements.

Describe the Perl noun/verb analogy.

Nouns can be singular (scalars) or plural (arrays and hashes).
Verbs can be procedures or functions.

Run some perl straight from the terminal.

Run some perl stored in a file.

Get interpreter warnings.

```
perl -e 'some perl'  
perl file.pl
```

The `-w` option prints warnings.

What is a filehandle?

Which are built in?

A data type that can represent files, devices, sockets, and pipes.

`STDOUT` and `STDERR` are provided by default.

Get a filehandle.

Use `open`, whose simplest form is:

```
open (HANDLENAME, "filename");
```

What read/write options are available when requesting a new filehandle?

Readonly (default): "<filename"

Write (clobber): ">filename"

Write (append): ">>filename"

What is a common idiom for handling failure
when opening a file?

```
open(FILEHANDLE, "file") or die "Error opening file: $!\n";
```

\$! is the OS's error message.

Read a line from a file.

Read a line from the terminal.

```
$str = <FILEHANDLE>
```

```
$str = <STDIN>
```

Write to a file.

Write to the terminal.

```
print FILEHANDLE 'str'
```

These two are the same:

```
print STDOUT 'str'
```

```
print 'str'
```

What's the difference between `chop` and
`chomp`?

`chop` removes the last character of the string passed to it, and returns it.

`chomp` removes `endl` from the string passed to it, and returns the *number* of characters removed.

What is Perl's string concatenation operator?

Why does Perl have separate operators for string concatenation and arithmetic addition?

Use a period (.) for string concatenation.

Because of weak typing addition of scalars created as strings, but that can be interpreted as numbers, would result in a sum.

How can you achieve Python's "string multiplication"?

Use the repeat operator (\times).

Name three ways to print a formatted string.

- Using the dot operator (like idiomatic Java using +)
- Using interpolation (like idiomatic bash using \$)
- Using list literal syntax

What is the exponentiation operator?

* *

Operators ending with = are treated ...

... specially.

In general, the following two are equivalent:

`lval op= expr`

`lval = lval op expr`

What is the result of assignment?

The value assigned.

The autoincrementing/decrementing operators
are just as in ...

Java

What are the two sets of boolean operators?

What's the difference?

&& - and

|| - or

! - not

The English ones bind less tightly.

What are the standard comparison operators?

What non-standard one is provided?

There are numeric and string versions.

`==` - `eq`

`!=` - `ne`

`<` - `lt`

`>` - `gt`

`<=` - `le`

The standard "greater or equal" is absent.

Nonstandard, but handy:

`<=>` - `cmp`

Name the common file test operators.

- e (exists)
- r (readable)
- w (writable)
- d (directory)
- f (regular file)
- T (text file)

What values are truthy?

Any string except "" and "0".

Any number except 0.

Why are references truthy and undefined values falsy?

References evaluate to non-0 addresses.

Undefined values evaluate to 0 or "".

In idiomatic Perl, simple conditionals are achieved with ...

... booleans operators.

Give the syntax of conditionals.

```
if (cond1) {  
    #block1  
}  
elseif (cond2) {  
    #block2  
}  
...  
else {  
    #blocknN  
}
```

Braces are *always* required.

What's the idiomatic way to accomplish the "if not" control flow?

```
unless (cond) {  
  #block  
}
```

What are the four looping constructions?

- while
- until
- for
- foreach

`while's cousin is ...`

... unless which executes the block as long as the condition is falsy.

Give the canonical `for`-loop iteration construction in Perl.

```
for ($i = 0; $i < num; $i++) {  
    #body  
}
```

Describe the `foreach` construction.

What's the huge surprise?

```
foreach $el (@arr) {  
    #body  
}
```

`$el` is by reference, not value, so you can in fact mutate `@arr` by re-assigning `$el`.

Perl doesn't have `break/continue`, but has
what?

next, last, **and** redo.

next **is just like** continue.

last **is just like** break.

redo **executes the same iteration again.**

`next/last` can optionally specify what?

Block labels, identifying which enclosing looping construction is being referred to.

If an operator can produce either a list or a scalar, which will be returned?

It depends on context. List context can be provided various ways, including by assigning to a list or by an operator that provides the `LIST` context to its arguments.