Perl was originally developed for text manipulation, and now used for tasks including system administration, web development, network programming, GUI development, and more.

### Basics

Include libraries with the `use` keyword.

```perl
use strict;
use warning;
use utf8;
```

Perl modules are available from [http://cpan.org](http://cpan.org)

### Variables

A **scalar** is a single item.

```perl
my $var=2;
my $var="foo";
my $var=3.1415;
```

An **array** is a list of items.

```perl
my @var="foo", "bar", 2, 3.1415;
```

A **hash** is a collection of key and value pairs.

```perl
my %var=
  "apple" => "red",
  "lime" => "green",
);
```

Use **scalar** notation to reference a scalar, even when it's contained in an array or hash.

```perl
print $var[0];
```

### Functions (subroutines)

Create a subroutine with the keyword `sub`, and place code between braces.

```perl
my $temp=celsius(70);
print "$temp\n";
```

A **return** statement is optional.

```perl
sub celsius {
  my $f=shift;
  my $cel=($f - 32)*(5/9);
  return($cel);
}
```

Parse parameters with `shift`. 
### Perl Cheat Sheet

#### Comparison

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>==</code></td>
<td>Equal / not equal</td>
<td><code>eq</code></td>
<td>Returns true if both operands are equal, otherwise false</td>
</tr>
<tr>
<td><code>!=</code></td>
<td>Not equal</td>
<td><code>ne</code></td>
<td>Returns true if both operands are not equal, otherwise false</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>Greater / less than</td>
<td><code>gt</code></td>
<td>Returns true if the left operand is greater than the right operand</td>
</tr>
<tr>
<td><code>&lt;</code></td>
<td>Less than</td>
<td><code>lt</code></td>
<td>Returns true if the left operand is less than the right operand</td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>Greater or equal</td>
<td><code>ge</code></td>
<td>Returns true if the left operand is greater than or equal to the right operand</td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>Less than or equal</td>
<td><code>le</code></td>
<td>Returns true if the left operand is less than or equal to the right operand</td>
</tr>
<tr>
<td><code>=~</code></td>
<td>String contains / does not contain</td>
<td></td>
<td>Returns true if the left operand contains the string captured by the right operand</td>
</tr>
<tr>
<td><code>!~</code></td>
<td>String does not contain</td>
<td></td>
<td>Returns true if the left operand does not contain the string captured by the right operand</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>Greater / less than / compares</td>
<td><code>gt</code></td>
<td>Returns <code>-1</code>, <code>0</code>, or <code>1</code> for less, equal, or greater</td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>Greater or less than / compares</td>
<td><code>ge</code></td>
<td>Returns <code>-1</code>, <code>0</code>, or <code>1</code> for less, equal, or greater</td>
</tr>
</tbody>
</table>

#### If statement

- `if ( condition ) {`  
  - `# code here`  
- `elsif ( other condition ) {`  
  - `# code here`  
- `else {`  
  - `# code here`  

- `print "True" if 1 == 1;`  

#### Unless (negated) statement

- `unless ( condition ) {`  
  - `# code here`  

#### While statement

- `while ( condition ) {`  
  - `# code here`  

- `print "Infinite" while 1;`  

#### Until (negated) statement

- `until ( condition ) {`  
  - `# code here`  

#### For each item in array

- `foreach (@array) {`  
  - `print "Item is \$_\n";`  

#### For each item in hash

- `foreach my $key (keys %hash) {`  
  - `print "Value is $hash{$key}\n";`  

---

Seth Kenlon  CC BY-SA 4.0  Opensource.com