

Perl Lesson 5

L615

Spring 2009

1. `split` — useful for splitting tab-separated or colon-separated data, etc.

- You can use regular expressions to split the string, but the default is to split on whitespace

```
$input_data = "john:mary:jenny:steve";
@my_list = split /:/, $input_data;

while (<>) {
    @my_list = split; # equivalent to: @my_list = split /\s+/ $_;
}
```

- To get each individual character, you can split on nothing ... NB: this is vastly different than splitting on whitespace!

```
@new_list = split//, $input_data;
foreach (@new_list) {
    print "$_\n";
}
```

2. `index` — find a substring

- Format: `index(string, substring)`
- Only returns the location of the *first* match

```
$long = "this is a great discussion";
$location = index($long,"great"); # 10
```

- Has an optional parameter which allows you specify the position to start searching—can loop over this
- When it can't find anything which matches, `index` returns -1

```
$where = 0;
while ($where != -1) {
    $where = index($long,"is", $where+1);
    print "$where\n"; # 2, 5, 17, -1
}
```

3. substr — manipulate a substring

- Allows you to grab a substring of a particular string, if you know the beginning and end of it
- Format: `substr(string, start, length)`

```
$middle = substr($long,5,8);  
print "$middle\n";          # "is a gre"
```

- Can replace the substring, too (even if a different length)
 - Later, we'll see how to use, `s///` (or `s///g`) to replace strings

```
# replace a substring  
substr($long,5,2) = "used to be";  
print "$long\n";          # "this used to be a great discussion"
```

4. Hashes — %

- For arrays, or lists, an integer (the array index) points to a value
- For hashes, a string points to a value—unlike arrays, there is no order to the elements in the hash

```
# store names pointing to ages  
my %this_hash = ("arnold", 8, "willis", 10, "kimberly", 12);  
  
# alternate, "Big Arrow" notation:  
my %this_hash = (  
    "arnold" => 8,  
    "willis" => 10,  
    "kimberly" => 12,    # trailing comma not needed, but uniform  
);
```

5. Access with curly braces

```
# Note that the value is a scalar, necessitating the $  
print "The age of Willis is $this_hash{'willis'}\n";  
  
$housekeeper = 'edna';  
# adding element  
$this_hash{$housekeeper} = 50;  
# kimberly's age changed:  
$this_hash{'kimberly'} = 13;
```

6. each — one way to iterate over the elements in a hash

```
while ( ($key, $value) = each %this_hash ) {  
    print "$key is $value years old";  
}
```