References

- Analagous (somewhat) to pointers in C/C++
  - Far less messy, and definitely less dangerous
- Assign the memory location of a variable to a scalar variable.
- Use the `\` to create a reference:
  - `@foo = (1, 2, 3);`
  - `$foo_ref = \@foo;`
  - `$foo_ref` now contains a reference to the array `@foo`;
    - Changes to `@foo` will affect array referenced by `$foo_ref`

De-Referencing

- Once you have a reference, de-reference it using the appropriate variable symbol (`@` for array, `%` for hash, etc)
- `$foo_ref = \@foo;`
- `@new_array = @$foo_ref;`
  - `@new_array` is a different array, which contains the same values of members that the array referenced by `$foo_ref` contained.
  - Changes to `@foo` (or even `$foo_ref`) do NOT affect `@new_array`
Referencing Other Types

- You can also reference other kinds of variables:
  - `%hash = ('Paul' => 23, 'Justin' => 22);
  - `$h_ref = %hash;
  - `$bar = "hello world\n";
  - `$bar_ref = \$bar;

Anonymous References

- A value need not be contained in a defined variable to create a reference.
- To create an anonymous array reference:
  - use square brackets, instead of parens
  - `$a_ref = [20, 30, 50, "hi!!"];
  - @a = @a_ref;
- For hash references, use curly brackets, instead of parens:
  - `$h_ref = {"sky" => 'blue', "grass" => 'green'}
  - %h = %h_ref;
- To de-reference specific element of references:

<table>
<thead>
<tr>
<th>Method</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>`$a_ref[2]</td>
<td>&quot;Hello&quot;</td>
</tr>
<tr>
<td>%h-&gt;{$key}</td>
<td>$value</td>
</tr>
</tbody>
</table>

TMTOWTDI

- In fact, there are three...
- `$a_ref = ["Hi", "Hiya", "Hello"];`  
- `$$a_ref[2] = "Hello";
- ` @{$a_ref}[2] = "Hello";
- ` $a_ref->{[2]} = "Hello";
- ` $$h_ref{$key} = $value;`  
- ` @{$h_ref}{[$key]} = $value;
- ` @{$h_ref}{$key} = $value;`  
- These are all valid and acceptable. Form you choose is whatever looks the best to you.
Why bother?

- In the beginning of the semester, we briefly mentioned two-dimensional arrays
  - The mention was “Don’t Do That.”
- To create a two-dimensional array, create an array of array references:
  - `@two_d = ([1, 2], [3, 4], [5, 6]);`
  - `$two_d[1]` is a reference to an array containing 3, 4
  - `@$two_d[1]` is an array containing 3, 4
  - `$two_d[1][0]` is the scalar value 3.

More Complicated

- Using similar methods, you can create arrays of hashes, hashes of arrays, arrays of hashes of arrays, hashes of hashes of arrays, arrays of hashes of arrays, . . .
  - `%letters = {
      ‘lower’ => [‘a’, ‘b’, ‘c’, …],
    }`
  - `$letters{‘lower’}` is an array reference;
  - `@$letters{‘lower’}` is an array;
  - `$letters{‘lower’}[1]` is scalar value ‘b’. 