Perl Lesson 3
L415 / L615
Fall 2015

1. Arrays/Lists — allow us to keep lists of items

   - Array variables indicated by @ sign
   - To access a particular item, use $ and indicate which item in brackets

```
@names = ('jack', 'janet', 'chrissy');
# alternate notation for strings:
@names = qw( jack janet chrissy );

$names[2] = 'cindy';
$names[4] = 'terry'; # also creates an undefined item at position 3
```

- To add items to the right of a list, use push; to remove them, use pop
- To add items to the left of a list, use shift; to remove them, use unshift

```
@names = qw( jack janet chrissy );
$gone = pop(@names);
push(@names,'cindy');
```

- Scalars can be converted to lists and vice versa, so be careful what you’re doing

```
@list = @names; # a list of names
$n = @names; # the length of the list @names (e.g., 3)
@short = 3*5; # is a one-element list (15)
```
2. for (use 1) — loop over the items in a list

```perl
foreach $person (@names) {
    print "My favorite cast member is $person";
}
```

# Or, using the special default variable:
```perl```
``` perl
foreach (@names) {
    print "My favorite cast member is \$_";
}
```

3. for (use 2)

- The condition has the general form: for (initialization; test; increment)

```perl
for ($i = 1; $i <= 5; $i++) {
    print "The current value is \$i\n";
}
```

# same as:
```perl```
``` perl
$i = 1;
while ($i <= 5) {
    print "The current value is \$i\n";
    $i++;
}
```

4. elsif — check a second condition (cf. earlier if statements)

```perl
if ($a < 1) {
    $b = 'below';
}
elsif ($a > 1) {
    $b = 'above';
}
else {
    # note that we assume $a is an number
    $b = 'one';
}
```

5. <STDIN> (redux) — earlier, we had something like $line = <STDIN>

- But now we can read all the lines at once

```perl```
``` perl
@lines = <STDIN>;
chomp(@lines);
```