1. A word about $a++ and ++$a
   1. Show 7-_-1.pl
2. Multiassignments (review):
   1. ($a,$b) = @list  # Grab first two elts from list
   2. ($a,$b,$c) = @list  # grab first two elts from list and put rest in @c
   3. lists are greedy!
3. Subroutines are declared like this:
   sub function{
     stuff
   }
4. Subroutines are called like this:
   &function(arg1, arg2, ...);
   or like this:
   function(arg1, arg2, ...);
5. Arguments are accessed using @_
6. shift works too
   i. shift works on @ARGV outside of subroutines
   ii. shift works on @_ inside of subroutines
7. pop works too, but from the right side.
8. Values can be returned explicitly (using return()) or implicitly (value of last line executed)
9. Show example 7-2.pl
10. Variable scope
   1. my is lexical (only understood within the block)
       i. A block is a region of code surrounded by fancy braces.
   2. local is dynamic (understood from block down)
   3. 7-3.pl
11. Now what about subroutines and arrays?
   1. 7-4.pl, 7.5.pl
12. PERL 5.* supports references (kinda like pointers)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Instantiating the scalar</th>
<th>Instantiating a reference to it</th>
<th>Referencing it</th>
<th>Dereferencing it</th>
<th>Accessing an element</th>
</tr>
</thead>
<tbody>
<tr>
<td>$scalar</td>
<td>$scalar = &quot;steve&quot;;</td>
<td>$ref = &quot;$steve&quot;;</td>
<td>$ref = $scalar</td>
<td>$$ref or ${$ref}</td>
<td>N/A</td>
</tr>
<tr>
<td>@list</td>
<td>@list = (&quot;steve&quot;, &quot;fred&quot;);</td>
<td>$ref = [&quot;steve&quot;, &quot;fred&quot;];</td>
<td>$ref = @list</td>
<td>@{$ref}</td>
<td>${$ref}[3] ${$ref}-&gt;[3]</td>
</tr>
<tr>
<td>%hash</td>
<td>%hash = (&quot;name&quot; =&gt; &quot;steve&quot;, &quot;job&quot; =&gt; &quot;Troubleshooter&quot;);</td>
<td>$hash = {&quot;name&quot; =&gt; &quot;steve&quot;, &quot;job&quot; =&gt; &quot;Troubleshooter&quot;};</td>
<td>$ref = %$hash</td>
<td>%{$ref}</td>
<td>${$ref}[&quot;president&quot;] ${$ref}-&gt;[&quot;president&quot;]</td>
</tr>
<tr>
<td>FILE</td>
<td></td>
<td></td>
<td>S$ref = *$FILE</td>
<td>$$\ref or scalar ${$ref}</td>
<td>${$ref}[3] ${$ref}-&gt;[3]</td>
</tr>
</tbody>
</table>

13. We use references to pass arrays and hashes as variables (see 7-6.pl)
14. See the problem with using a hash in 7-7. (same as a list)

15. perl has anonymous variables
   i. $arrayref=[1,2,[1,'b','c']]; reference to anonymous array (7-7-5.pl)
      1. Why are there only 3 elts in the array?
   ii. This is how we make arrays with multiple dimensions. (7-8.pl)
   iii. A reference to an anonymous hash can be created using curly brackets:
         
         $hashref = {
            'Adam'  => 'Eve',
            'Clyde' => 'Bonnie',
         };
   iv. Taking a reference to an enumerated list is not the same as using square brackets--instead
      it’s the same as creating a list of references!
      @list = (\$a, \@b, \%c);
      @list = (\$a, @b, %c);    # same thing!
   v. anonymous subroutine:
      $coderef=sub {print “Boink!\n” };

16. A Perl module is a discrete component of software for the Perl programming language—there are also perl libraries (not quite sure what the difference is—modules end in .pm, libraries end in .pl)

17. PERL looks in the @INC variable (sort of like classpath for java) for directories to search
   1. perl -e '$,="\n";print @INC,"\n"
   2. use <module>; loads the module, if it is a bareword ‘pm’ is appended
   3. require <module> works too, but will fail if module is not found
   4. require <num> requires that version of PERL or later to run:
      i. require 5.004; (for example)
      ii. $J contains the version of perl running the script
   5. Loaded modules are stored in %INC;
   6. make domino.pl
      @games=(“Tiddly-wink”,“Matador”, “Bergen”, Sebastopol”);
   7. make dominos
      require ‘domino.pl’;
      srand(time ^ ($$ = ($$<<15)));
      print “A random game: $games[rand(@games)]\n”;

18. It’s common to use this mechanism to affect how perl works:
   1. use strict; (for example)
      i. Doesn’t allow symbolic references, Can’t use ‘globals’, can’t use barewords
      ii. This is an example of a pragma

19. Example of module set using ‘Math::Trig’
   1. Show directory
   2. Show 7-10.pl

20. http://www.cpan.org/ has most modules

22. Get the module and download it in class (TERM::READKEY)
   1. Get address from web
   2. use ‘wget’ to download
   3. gunzip it
   4. tar –xvf it
   5. ‘less’ the README
   6. Follow these directions! (we’ll stop at the requirement for ‘root’)

23. Math!

24. Basic probability
   1. Discrete: good/total
   2. Everything is just a trick to make **that** easier to calculate

25. Probability of drawing a vowel from the alphabet?

26. Probability of NOT drawing a vowel?
   1. Principle of negation P(!A) = 1-P(A)

27. Conditional probability and law of total probability: Probability of getting a vowel in two draws (with replacement):
   1. P(getting a vowel first) + P(getting a consonant first and then getting a vowel)
   2. 5/26*1 + 21/26*5/26

28. Same... but this time WITHOUT replacement:
   1. P(getting a vowel first) + P(getting a consonant first and then getting a vowel)
   2. 5/26*1 + 21/26*5/25

29. Permutation (order MATTERS)... Once upon a time HorUS the SUn god created a race of beings call the Ur-men to do to his will and all was well in the Universe, but even HorUS could not forsee all consequences, and it came to pass that some Ur-men were born with MUTATIONs that made them nasty, and they rebelled, and all of heaven and earth were aflame with the fires of rebellion—the cosmic ORDER was shattered... and it sUcked! the end. (moral of the story—order matters for U!)

30. How many permutations of 3 objects?
   1. 1,2,3
   2. 1,3,2
   3. 2,1,3
   4. 2,3,1
   5. 3,1,2
   6. 3,2,1

31. How many permutations of n objects?

32. In **combinations** order does NOT matter.... after all , if you order the COMBO deal at burger king does it matter if they hand you the coke first or the fries? Nope!

33. How many ways can I CHOOSE 3 objects from five?
   1. CHOOSE means order doesn’t matter
   2. 5 choices for first, 4 for second, 3 for third
   3. BUT some give same thing:
i. 1,2,3 and 1,3,2 and 2,1,3 and 2,3,1 and 3,1,2 and 3,2,1
ii. the amount of overspecifying is ALWAYS the same
iii. So... If order matters, The number of ways I can pull 3 items from 5 is:
   1. 5*4*3
   2. When order doesn’t matter i have to ERASE the multi-counting:
   3. 5*4*3/6
4. in general nCr (read this as n items are in the bag, choose r of them)
   1. nCr = n!/[((n-r)!r!]

34. We use the CHOOSING method to count the number of ways that we can make a team:
   1. How many 5 person teams are possible from a pool of 10 people?

35. We’d like to be able to estimate how likely it is that two different DNA sequences are related.
   1. Sequences are copied from generation to generation
   2. Mutations arise and become fixed in the population
   3. Two rather different sequences may still have derived from a common ancestor
   4. Consider these two sequences from two different species:
      i. ggagactgtgtagacagctaatgtctata
      ii. gaacgcctagcagcgagcccttatc
   5. Do they show significantly more similarity than we would expect from two arbitrary segments of DNA from the two species?

36. How much similarity would we expect if they were generated randomly?
   1. In that case, they should agree in about ¼ of the locations... here they agree in 11 out of 26—that’s more than 40% agreement. How unlikely is that to occur by pure chance?
   2. This is where statistics comes in: Differences are distilled to a single number and then different models are used to interpret that number: This provides evidence for (or against) the model. The models usually involve some numeric parameter—for example: The probability any specific nucleotide is produced randomly is ¼. Perhaps a better model would be: A and T are produced with a probability of 1/3 and C and G are produced with a probability of 1/6 and 1/6 resp.
   3. Do we want to look for the number of shared locations? Or would we rather examine the length of the longest matching sequence? (here it’s 3) Or should we look at the length of the longest SHARED subsequence:
      i. gagctagcagctat
   4. No easy answers... every model depends upon certain assumptions! if those assumptions aren’t met, then the model is inappropriate (though sometimes if they’re almost met it’s still worth considering

37. A discrete random variable is a numerical quantity that
   1. takes one value from some discrete set of possible values
   2. involves some degree of randomness in the experiment
   3. Example,
      i. experiment: roll two six-sided die,
      ii. random variable is sum of the two numbers rolled.
iii. Set of possible values is \{2,3,4,5,6,7,8,9,10,11,12\}

iv. NOTICE that not all results are equally likely.

4. Notation:
   i. let X denote the random variable in this example
   ii. <draw table and figure out probabilities> (a few examples given below)
   iii. \( P(X=2) = 1/36 \)
   iv. \( P(X=3) = 2/36 \)
   v. \( P(X=4) = 3/36 \)
   vi. \( P(X=5) = 4/36 \)
   vii. \( P(X=6) = 5/36 \)

38. The Probability Distribution of a random variable is the list of possible values AND the associated probabilities:
   1. example: Toss a fair coin twice let Y denote the total number of heads. (Table)
      
      \[
      \begin{array}{ccc}
      Y & 0 & 1 & 2 \\
      P(Y) & .25 & .5 & .25 \\
      \end{array}
      \]
   2. barplot(dbinom(0:2,2,.5),names.arg=0:2) (graph)
   3. \( P(Y=0) = .25 \), \( P(Y=1) = .5 \), \( P(Y=2) = .25 \)

39. Third approach allows us more mathematical flexibility

40. Let X be a random variable:
    \( P(X=y) = y^2/14, \ y = 1, 2, 3 \)

41. Sometimes abbreviated like \( P(y) \), if \( x \) is understood.

42. A few rules
   1. \( \sum_{i=1}^n P(X=x_i) = 1 \)
   2. \( P(X=x_i) \geq 0 \)

43. Cumulative Distribution Function:
   1. \( P(X\leq y) = \sum_{y' < y} P(x=y') \)
   2. Often abbreviated like \( F(y) \)