Week 7

Readings

Chapter 17
Chapters 4 and 5 from *Intermediate Perl*

Key notes to keep in mind (AKA: how Perl differs)

- The **eval** block will return error status in **$@** which is a scalar with the name @ (page 250).
- Using hash slices can create a hash out of two lists, although the indices must line up to get the desired results (page 259).
- Perl does not allow for multidimensional arrays in the manner of many other languages, instead you must use references (page 23, *Intermediate Perl*).
- "A reference fits wherever a scalar fits." This is what allows them to be placed into arrays (which is really just a list of scalars) (page23, *Intermediate Perl*).
- Notice that when dereferencing an array we can often omit the braces leaving us with two leading sigils like **@$** or **$$** (page 26, *Intermediate Perl*).
- The arrow (**->**) can be used in place of braces when dereferencing an array, and if they would be placed between subscripts then they are not required at all, so they end up looking like multidimensional arrays in other languages (page 29, *Intermediate Perl*).
- Notice the double sigil shortcut when dereferencing hashes as well: **%$**
- And the arrow form works similarly as well and the hash braces count as "subscripty kinds of things" (page 31, *Intermediate Perl*).
- When creating recursive data structures be careful to avoid memory leaks as Perl uses reference counting to determine when variable memory can be freed. Although such structures are rarely need in Perl they can always be freed up manually be setting them to undef (page 39, *Intermediate Perl*).
- Whenever you try to access a variable that doesn't exist, Perl will create it and continue on (page 46, *Intermediate Perl*).

Exercises

Book exercises from Chapter 17 and Chapters 4 and 5 from *Intermediate Perl*.
As the exercises from *Intermediate Perl* are comparatively in-depth from what you've been working on you will not have to do an extra program this week.