Opening Discussion

- Minute essay comments:
  - What does asInstanceOf[] do?
  - Parser usage.
- Strings match themselves.
- RegEx and tokens give strings.
- $P \sim Q$ gives back $\sim(p,q)$, where $p$ and $q$ are the matches of $P$ and $Q$.
- $P \mid Q$ gives either $p$ or $q$.
- $\text{rep}(P)$ or $\text{repsep}(P, \text{separator})$ give a list of $p$ values.
- $\text{opt}(P)$ gives an Option, either Some$(p)$ or None.
Specifying Output

- You can override the default of P by using P \( ^\wedge \) f. The f is a function (or partial function) that takes the normal output of P.

- The output you get is f(p).

- Example uses:
  - floatingPointNumber \( ^\wedge \) (_.toDouble)
  - "true" \( ^\wedge \) (x=>true)
  - "("~ident~","~ident~")" \( ^\wedge \) { case "("~i1~","~i2~")" => (i1,i2) }
In something like the last example shown, there are strings that are part of the parse that really don't impact the result.

When you have this type of situation you can use `~>` or `<~` instead of just `~`. The parse result will only include what the arrow points to.

```
``“(~>ident~”,”~ident<~”)” ^^ { case i1~”,”~i2 => (i1,i2) }
```
Let's work on putting this type of functionality in our formula code.

We had the parser, but we want to parse to a tree similar to what we produced with the recursive parser we wrote ourselves.

With that we can make this alternate code functional.
What questions do you have about parsers, regex, or grammars?

Next class we do spatial trees.

IcP #7 is next class.

Spring classes and Web Apps.