Creating "recipes"

- Each programming language provides a set of primitive operations
- Each programming language provides mechanisms for combining primitives to form more complex, but legal, expressions
- Each programming language provides mechanisms for deducing meanings or values associated with computations or expressions

Aspects of languages

- Primitive constructs
  - Programming language – numbers, strings, simple operators
  - English – words
- Syntax – which strings of characters and symbols are well-formed
  - Programming language – we'll get to specifics shortly, but for example `3.2 + 3.2` is a valid Python expression
  - English – "cat dog boy" is not syntactically valid, as not in form of acceptable sentence

Aspects of languages

- Static semantics – which syntactically valid strings have a meaning
  - English – "I are big" has form <noun> <intransitive verb> <noun>, so syntactically valid, but is not valid English because "I" is singular, "are" is plural
  - Programming language – for example, <literal> <operator> <literal> is a valid syntactic form, but `2.3/'abc'` is a static semantic error

Aspects of languages

- Semantics – what is the meaning associated with a syntactically correct string of symbols with no static semantic errors
  - English – can be ambiguous
    - "I cannot praise this student too highly"
  - Programming languages – always has exactly one meaning
    - But meaning (or value) may not be what programmer intended

Where can things go wrong?

- Syntactic errors
  - Common but easily caught by computer
- Static semantic errors
  - Some languages check carefully before running, others check while interpreting the program
  - If not caught, behavior of program unpredictable
- Programs don't have semantic errors, but meaning may not be what was intended
  - Crashes (stops running)
  - Runs forever
  - Produces an answer, but not programmer's intent

Our goal

- Learn the syntax and semantics of a programming language
- Learn how to use those elements to translate "recipes" for solving a problem into a form that the computer can use to do the work for us
- Computational modes of thought enable us to use a suite of methods to solve problems