Syntax

```
var numbers = new List<int> { 6, 2, 8, 3 };
int sum = numbers.Aggregate(func: (result, item)
=> result + item);
// sum: (((6+2)+8)+3) = 19
```

C#: Aggregate (Folding) Syntax

```
var numbers = new List<int> { 6, 2, 8, 3 };
int sum = numbers.Aggregate(func: (result, item)
=> result + item);
// sum: (((6+2)+8)+3) = 19
```

```
import functools
total = functools.reduce(lambda a, b: (0, a[1] +
b[1]), items)[1]
```

Python: Reduce (Fold) Syntax

```
import functools
total = functools.reduce(lambda a, b: (0, a[1] +
b[1]), items)[1]
```

```
let numbers = [1, 2, 3, 4]
let numberSum = numbers.reduce(0, { x, y in
    x + y
})
// numberSum == 10
```

Swift: Reduce (Fold) Syntax

```
let numbers = [1, 2, 3, 4]
let numberSum = numbers.reduce(0, { x, y in
    x + y
})
// numberSum == 10
```

Purpose

Syntax is

- the user interface to the programming language;
- meant for humans
  - a means of communication (a common language between developers);
  - facilitation of a mental model;
    - design
    - organization

Description

Syntax is described using context-free grammars.

Context-Free Grammars

$$V ::= R_1(V) \mid R_2(V) \mid \cdots \mid R_i(V)$$

Variable

Context-Free Grammars

$$V ::= R_1(V) \mid R_2(V) \mid \cdots \mid R_i(V)$$

Variable Rule Rule Rule

Context-Free Grammars

Read as "or" Read as "or" Read as "or"

$$V ::= R_1(V) \mid R_2(V) \mid \cdots \mid R_i(V)$$

Variable Rule Rule Rule

Context-Free Grammars

$$
\begin{aligned}
V ::= \; & R_1(V) \\
| \; & R_2(V) \\
| \; & \cdots \\
| \; & R_i(V)
\end{aligned}
$$

Context-Free Grammars

Arithmetic expressions

$$e ::= 0 \mid 1 \mid e + e \mid e \times e$$

Terminals Terminal Terminal

Context-Free Grammars: Parsing via derivations

Parsing

Given a program p, can we find a derivation using our grammar ending with p?

Context-Free Grammars: Parsing via derivations

$$e ::= 0 \mid 1 \mid e + e \mid e \times e$$

$$\begin{aligned} e &\Rightarrow e + e \\ &\Rightarrow 0 + e \\ &\Rightarrow 0 + e * e \\ &\Rightarrow 0 + 1 * e \\ &\Rightarrow 0 + 1 * 1 \end{aligned}$$

Context-Free Grammars: Parsing via derivations

Valid Program

$$
\begin{aligned}
e &\Rightarrow e + e \\
&\Rightarrow 0 + e \\
&\Rightarrow 0 + e * e \\
&\Rightarrow 0 + 1 * e \\
&\Rightarrow 0 + 1 * 1
\end{aligned}
$$

Invalid Program (Syntax Error)

$$e \Rightarrow e * e$$

?

$$\Rightarrow (0 + 1) * 1$$

Context-Free Grammars

Arithmetic expressions

$$e ::= 0 \mid 1 \mid e + e \mid e \times e \mid (e)$$

Context-Free Grammars: Parsing via derivations

Valid Program

$$
\begin{aligned}
e &\Rightarrow e + e \\
&\Rightarrow 0 + e \\
&\Rightarrow 0 + e \times e \\
&\Rightarrow 0 + 1 \times e \\
&\Rightarrow 0 + 1 \times 1
\end{aligned}
$$

Valid Program

$$
\begin{aligned}
e &\Rightarrow e \times e \\
&\Rightarrow (e) \times e \\
&\Rightarrow (e + e) \times e \\
&\Rightarrow (0 + e) \times e \\
&\Rightarrow (0 + 1) \times e \\
&\Rightarrow (0 + 1) \times 1
\end{aligned}
$$

Context-Free Grammars

Arithmetic expressions in extended form:

$$e ::= 0 \mid 1 \mid e_1 + e_2 \mid e_1 \times e_2$$

subscripts to
point at distinct
expressions

Context-Free Grammars: Parsing via derivations

Valid Program

$$
\begin{aligned}
e &\Rightarrow e_1 \times e_2 \\
&\Rightarrow (e_5) \times e_2 \\
&\Rightarrow (e_3 + e_4) \times e_2 \\
&\Rightarrow (0 + e_4) \times e_2 \\
&\Rightarrow (0 + 1) \times e_2 \\
&\Rightarrow (0 + 1) \times 1
\end{aligned}
$$