F# Syntax

Björn Lisper
School of Innovation, Design, and Engineering
Mälardalen University

`bjorn.lisper@mdh.se`
`http://www.idt.mdh.se/~blr/`

Some F# syntax things that are good to know:

- Indentation-sensitive syntax
- Identifiers
- Operators and functions
- Comments

F# also has other syntactical conveniences, more on this later

A Note on “F# Light” Syntax

We have been careful to indent definitions

F# has an option for “lightweight syntax”, which is on by default

This enables some syntactic simplifications (some keywords kan be dropped)

Also makes the syntax *indentation-sensitive*

This syntax can confuse beginners, so let’s talk about it right away

Basic rule: when starting a new line, if the contents of the new line starts to the *left* of the contents of the old line you start a *new* expression, otherwise you continue the *old* expression

Indentation-sensitive Syntax

Some examples:

```
let f n = match n with
          | 0 -> 1
          | _ -> 2
```

OK! The cases are lined up with the match

```
let f n = match n with
          | 0 -> 1
        | _ -> 2
```

Not OK! The second case starts to the left. Will yield syntax error

```
let f n = match n with
          | 0 -> 1
            | _ -> 2
```

OK! The second case can start to the right of the first.

This syntax can be overruled by using explicit `{...}`-parentheses and “`;`”. But most people find it natural and convenient.

Identifiers

Identifiers are given a meaning by *declarations*

In F#, one can declare own *values* (including functions), *types*, *modules*, and *name spaces*

(We have seen values so far. We'll get back to the other things)

Syntactic rules for F# identifiers are like in most languages

Three examples of valid identifiers: `X`, `x2BlurB`, `no_no`

Entities of different kinds can have the same name. For instance we can have both a function "`foo`" and a type "`foo`"

Reserved keywords in F# (like "`let`") cannot be used as identifiers

Operators, Their Syntax and Types

Operators are just functions!

An operator within parentheses can be used as an ordinary function (prefix notation):

`(+) 2 4` = `2 + 4`

We have

```
(+) : int -> int -> int
```

Declaring own Operators

In F# you an define your own infix operators

Sometimes very useful to increase the readability of the code

A set of “typical operator symbols” (like `+`, `*`, `...`) for operator names

Example (typed into `fsi`):

```
> let (+*) x y = x + 2*y;;

val ( +* ) : int -> int -> int

> 3 +* 4;;
val it : int = 11
```

(Can also declare *prefix* operators, see course book)

Comments

Two ways of making comments in F# source code:

Everything after “`//`” on a line is a comment

```
// This line is a comment
```

Everything between “`(*`” and “`*)`” is a comment

```
(* this is a
multiline comment *)
```

“`(*`” and “`*)`” can be nested