

Unlike Java, F#'s primitive types ...

- Have signed and unsigned versions.
- Include a fixed precision point type.

How can you use an integer as the base of an exponent?

You must use `pown` instead of `**`, or convert to a floating-point type.

What are the conversion functions for primitive types?

They have the same names as the types being converted to.

Why are conversion functions more widely used in F#?

There is no type coercion.



For arbitrary precision integers F# features  
what?

Language support. Literals ending with `ℤ` are  
`System.Numerics.BigIntegers`.

What are the bitwise operators?

`&&&`, `|||`, `^^^`, `<<<`, `>>>`

Index a string.

```
"abcd".[0]
```

How are multi-line strings written?

- One can end each line with a backslash while still in a string. Leading whitespace will be removed from the next line.
- With verbatim strings, but leading whitespace will not be removed.



What is the verbatim string syntax?

An @ before the string begins.

What shortcut is provided for converting a string literals to arrays of bytes?

The suffice `B` after the close quote of a string literal.

How is negation written?

not for unary negation.

How is equality written?

Inequality?

= and <>.



What is the syntax of function definitions?

```
let funcname arg1 arg2 ... argN = body
```

The result of a function is ...

The value of the last expression in a given path. There is no `return` keyword.

How are function types written?

As in Haskell:

`type1 -> type2 -> ... -> typeN`

Unlike in Haskell, functions using operators like `+` will default to ...

... `int` parameters instead of type parameters.



How are type annotations written?

As usual, `name : type`.

How are type parameters written?

With a leading ' , e.g., 'a, 'b.

Unlike in Scala, shadowing is allowed where?

In the same block. E.g., two consecutive lines can `let` the same identifier.

What is the syntax of conditionals?

```
if    cond1 then res1
elif  cond2 then res2
...
else  res
```



Unlike in Scala, each branch of a conditional must have what?

The same type. Upcasting will not be done.

How are tuples written?

How are their types written?

Comma separated values surrounded by round parens.

`type1 * type2 * ... * typeN`

Access the elements of a tuple.

- Use `fst` and `snd`.

- Unpack with `let`

```
let a, b, c = (1, 2, 3)
```

What is the list literal syntax?

Semi-colon separated values surrounded by square brackets.



What is the syntax of list comprehensions?

Basically arbitrary code within square brackets. Elements are selected using the `yield` keyword.

How are list comprehensions evaluated?

Eagerly, in memory.

Loop over a range.

```
for var = start-expr to end-expr do  
    //body
```

to is inclusive, as in Scala.

Loop over a collection.

```
for pattern in expr do  
    //body
```



What shortcut `for` syntax is provided for list comprehensions?

```
for pattern in expr -> expr
```

In the F# API, what's the difference between  
`fold` **and** `reduce`?

`reduce` uses the first element of the list as the initial accumulator value, `fold` takes the initial accumulator value explicitly.

Run a block of code on each element of a list.

`List.iter` is like **Scala's** `foreach`.

What is used in place of `null`?

As in Scala, `Option` can have values `Some('a)` or `None`.



Why is `printfn` more useful than using  
`System.Console's` print methods?

The arguments to the formatting string are type checked, contributing to type inference.

What are the `printfn` format specifiers?

%d, %i - integer

%s - string

%f - floating-point number

%c - character

%b - boolean

%O - object

%A - anything

If no module is specified, you are actually using what?

How can it be accessed?

you are using the `anonymous` module which can nonetheless be accessed using the filename with the first letter capitalized.

For example, `MyFile.someValue`.

Create a module explicitly.

```
module MyModule  
  //whatever
```



Create a nested module.

Equal sign and indentation are mandatory:

```
module Outer
  module Inner =
    //whatever
```

How are namespaces different from modules?

- they can contain declarations but not values.
- they cannot be nested

What are the ideal use-cases for modules and namespaces?

Namespaces: large object-oriented programs

Modules: rapid prototyping

How does an F# program execute?

Straight down the *last* code file.



Create an explicit entry point.

Add the [`<EntryPoint>`] attribute to a method.  
It must be in the last function in the last file of the program.  
It must take a `string array` and return an integer.