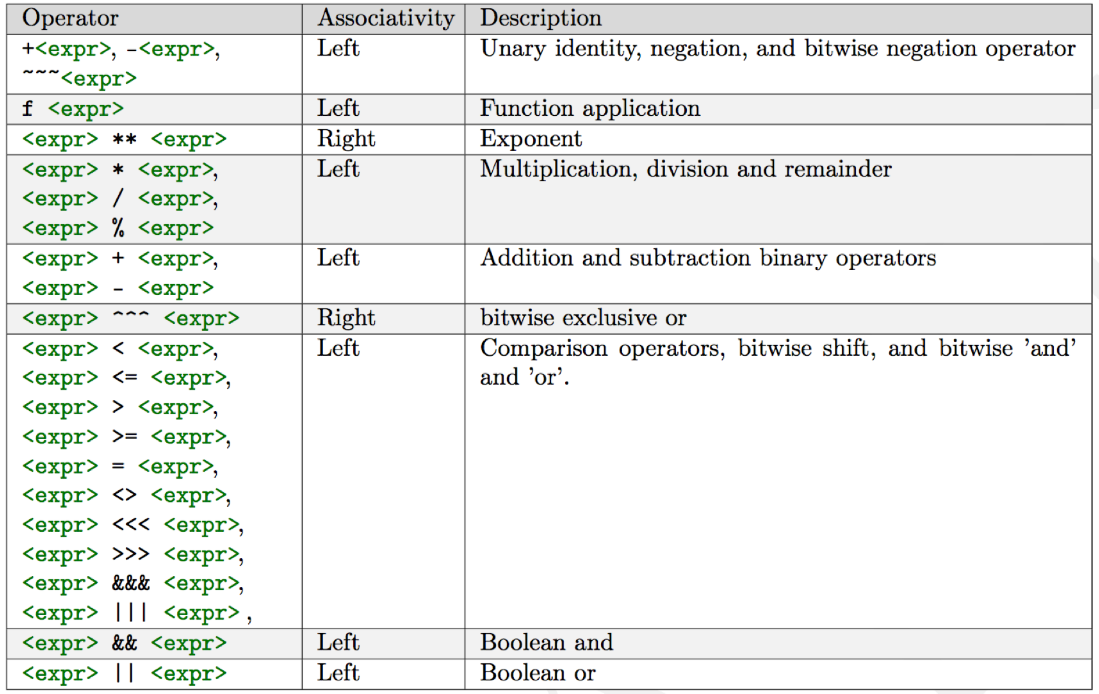
Programmering og Problemløsning

3.1: Funktioner, dokumentation og løkker

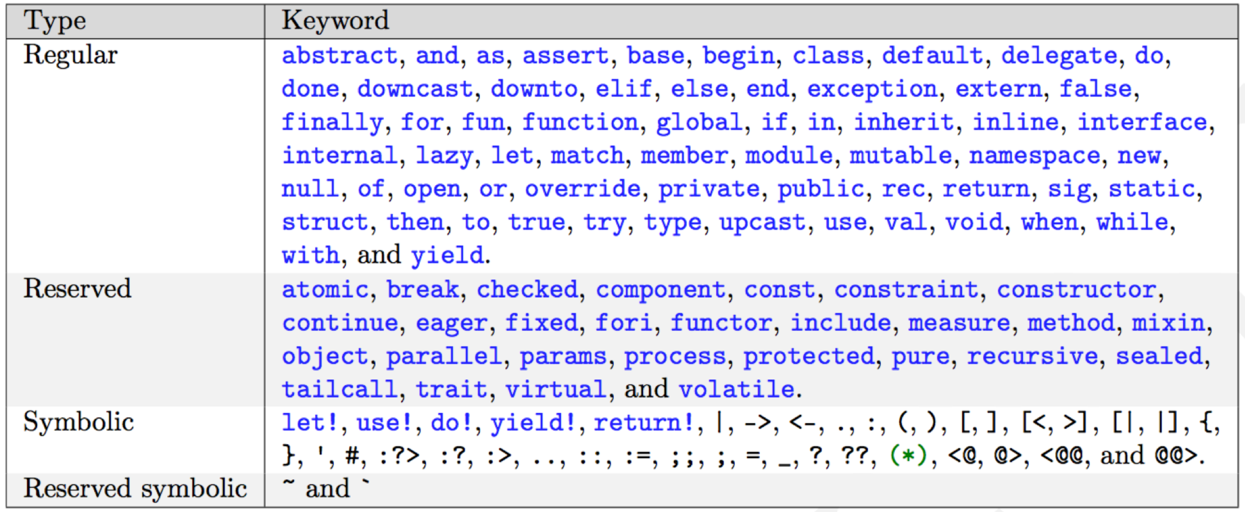
Repetition af Nøglekoncepter

- Heltal, flydende tal, tegn, strenge
- Typer og operatorer

- Præcedens og association
- Verbose og letvægtssyntaks
- Virkefelter
- Nøgleord



Operator	Associativity	Description
+<expr>, -<expr>, ~~~<expr>	Left	Unary identity, negation, and bitwise negation operator
f <expr>	Left	Function application
<expr> ** <expr>	Right	Exponent
<expr> * <expr>, <expr> / <expr>, <expr> % <expr>	Left	Multiplication, division and remainder
<expr> + <expr>, <expr> - <expr>	Left	Addition and subtraction binary operators
<expr> ^^^ <expr>	Right	bitwise exclusive or
<expr> < <expr>, <expr> <= <expr>, <expr> > <expr>, <expr> >= <expr>, <expr> = <expr>, <expr> <> <expr>, <expr> <<< <expr>, <expr> >>> <expr>, <expr> &&& <expr>, <expr> ||| <expr>,	Left	Comparison operators, bitwise shift, and bitwise 'and' and 'or'.
<expr> && <expr>	Left	Boolean and
<expr> || <expr>	Left	Boolean or

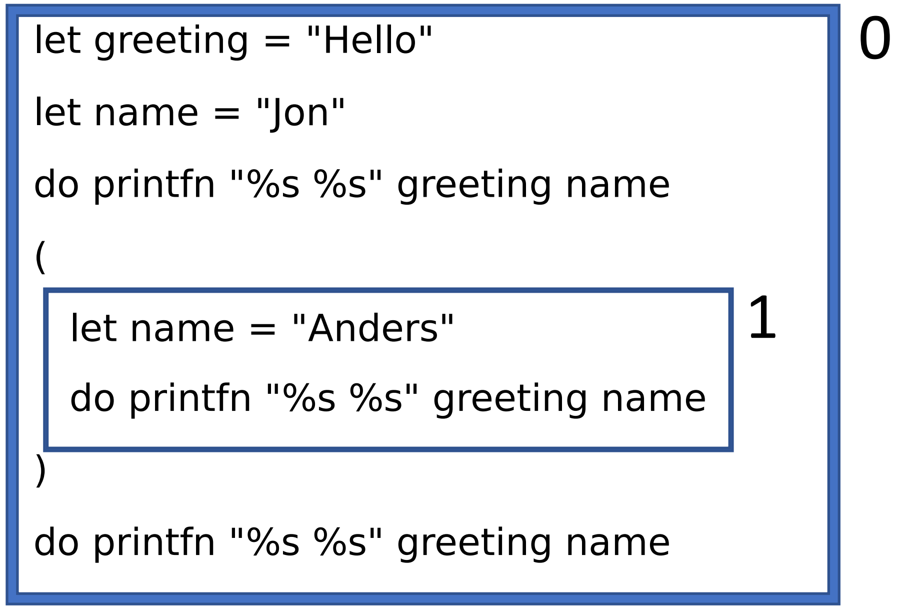


Type	Keyword
Regular	abstract, and, as, assert, base, begin, class, default, delegate, do, done, downcast, downto, elif, else, end, exception, extern, false, finally, for, fun, function, global, if, in, inherit, inline, interface, internal, lazy, let, match, member, module, mutable, namespace, new, null, of, open, or, override, private, public, rec, return, sig, static, struct, then, to, true, try, type, upcast, use, val, void, when, while, with, and yield.
Reserved	atomic, break, checked, component, const, constraint, constructor, continue, eager, fixed, fori, functor, include, measure, method, mixin, object, parallel, params, process, protected, pure, recursive, sealed, tailcall, trait, virtual, and volatile.
Symbolic	let!, use!, do!, yield!, return!, |, ->, <-, ., :, (,), [,], [<, >], [|, |], {, }, ', #, :?>, :?, :>, .., ::, :=, ;;, ;, =, _, ?, ??, (*), <@, @>, <@@, and @@>.
Reserved symbolic	~ and `

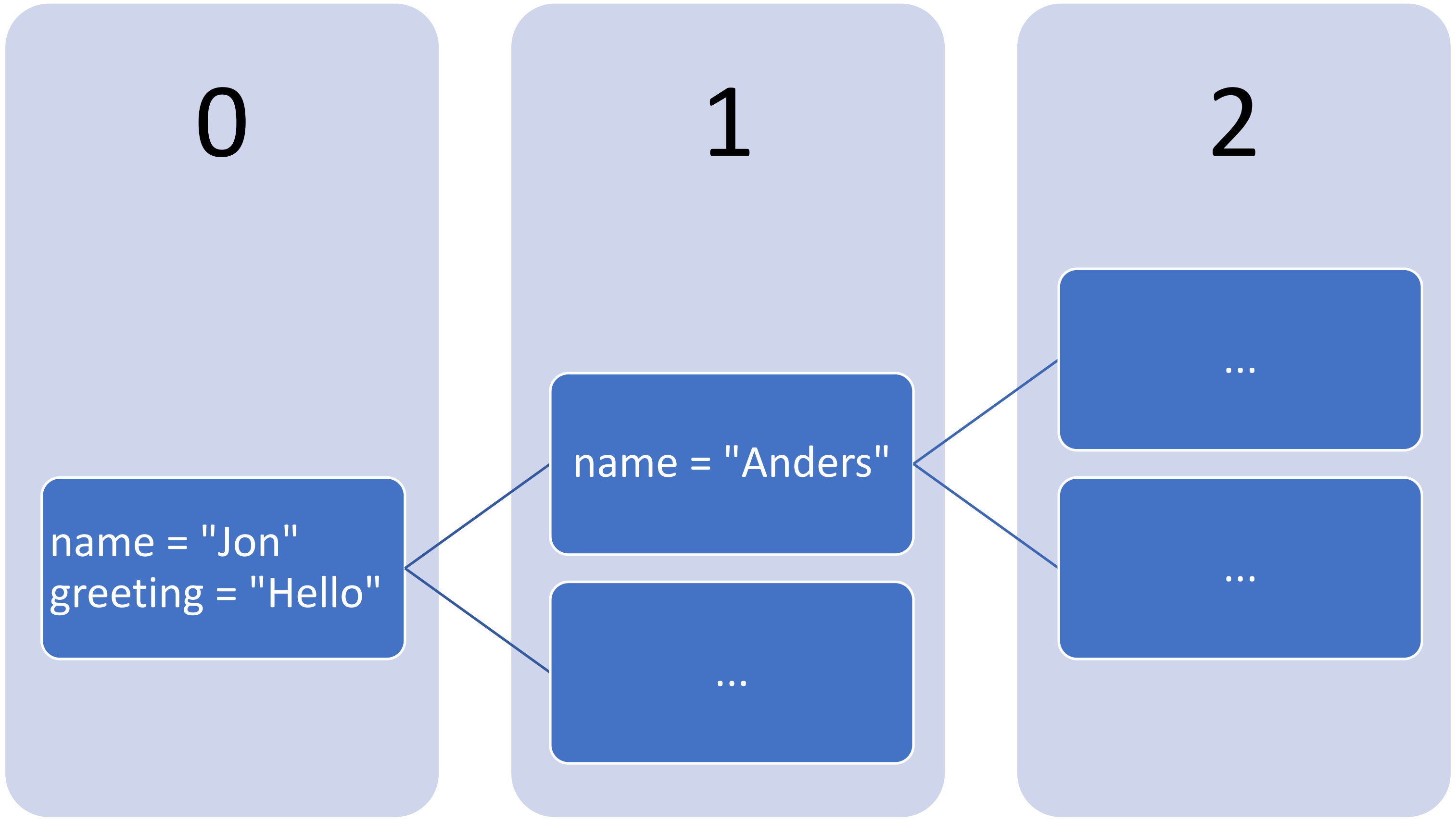
https://tinyurl.com/yc3or9fh

Virkefelter (scope)

Virkefelter via parenteser



```
let greeting = "Hello"
let name = "Jon"
do printfn "%s %s" greeting name
(
  let name = "Anders"
  do printfn "%s %s" greeting name
)
do printfn "%s %s" greeting name
```



Funktioner

Organisering = nemmere at forstå og vedligeholde

```
let greetings (name : string) : string =
  "Hello " + name

let str = greetings "Jon"
do printfn "%s" str
do printfn "%s" (greetings "World")
```

```
let greetings name =
  "Hello " + name

let str = greetings "Jon"
do printfn "%s" str
do printfn "%s" (greetings "World")
```

Løs en andengradsligning (baglæns!)

```
let discriminant a b c =
  b ** 2.0 - 4.0 * a * c


let solution a b c sgn =
  let d = discriminant a b c
  (-b + sgn * sqrt d) / (2.0 * a)


let a = 1.0
let b = 0.0
let c = -1.0
let xp = (solution a b c +1.0)
printfn "0 = %.1fx^2 + %.1fx + %.1f => x_+ = %.1f" a b c xp
```

$$ax^2 + bx + c = 0$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

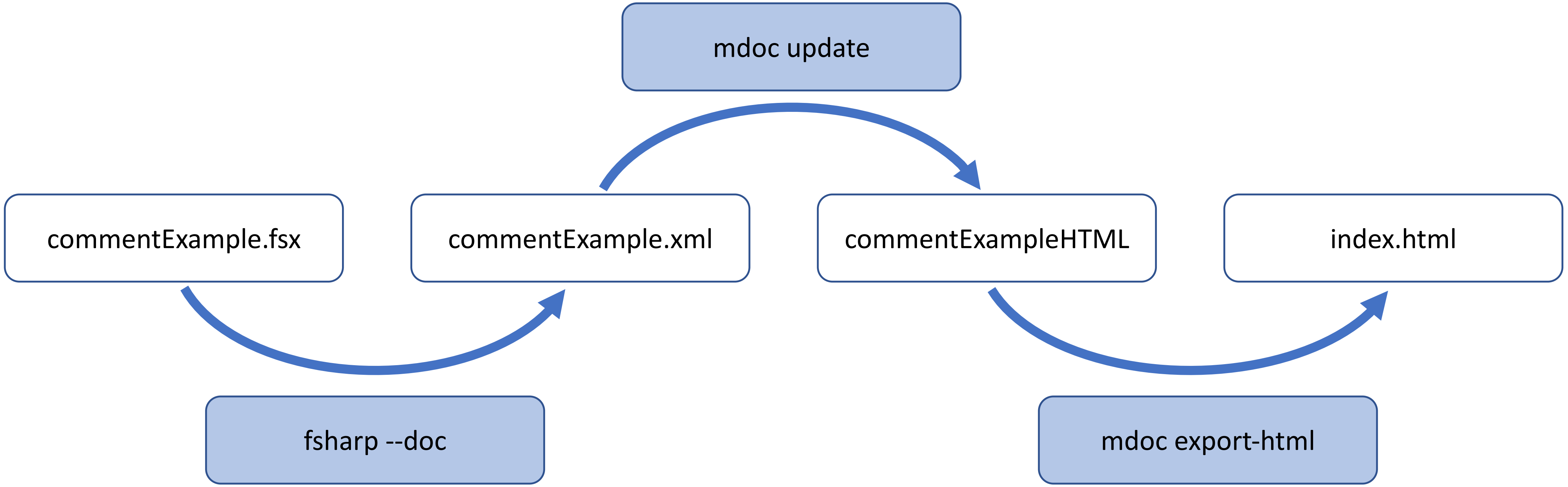
Dokumentation - simpel

```
/// The discriminant of a quadratic equation with parameters a, b, and c
let discriminant a b c = b ** 2.0 - 4.0 * a * c
```

Dokumentation - grundig

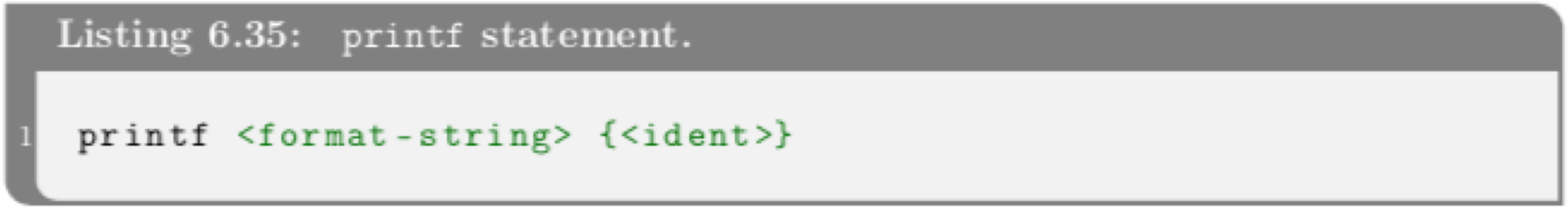
```
/// <summary>Find x when 0 = ax^2+bx+c.</summary>
/// <remarks>Negative discriminants are not checked.</remarks>
/// <example>
///   The following code:
///   <code>
///     let a = 1.0
///     let b = 0.0
///     let c = -1.0
///     let xp = (solution a b c +1.0)
///     printfn "0 = %.1fx^2 + %.1fx + %.1f => x_+ = %.1f" a b c xp
///   </code>
///   prints <c>0 = 1.0x^2 + 0.0x + -1.0 => x_+ = 0.7</c> to the console.
/// </example>
/// <param name="a">Quadratic coefficient.</param>
/// <param name="b">Linear coefficient.</param>
/// <param name="c">Constant coefficient.</param>
/// <param name="sgn">+1 or -1 determines the solution.</param>
/// <returns>The solution to x.</returns>
let solution a b c sgn =
  let d = discriminant a b c
  (-b + sgn * sqrt d) / (2.0 * a)
```

XML dokumentationspipeline



```
fsharpc --doc:commentExample.xml commentExample.fsx
mdoc update -o commentExample -i commentExample.xml commentExample.exe
mdoc export-html -out commentExampleHTML commentExample
```

printf familien

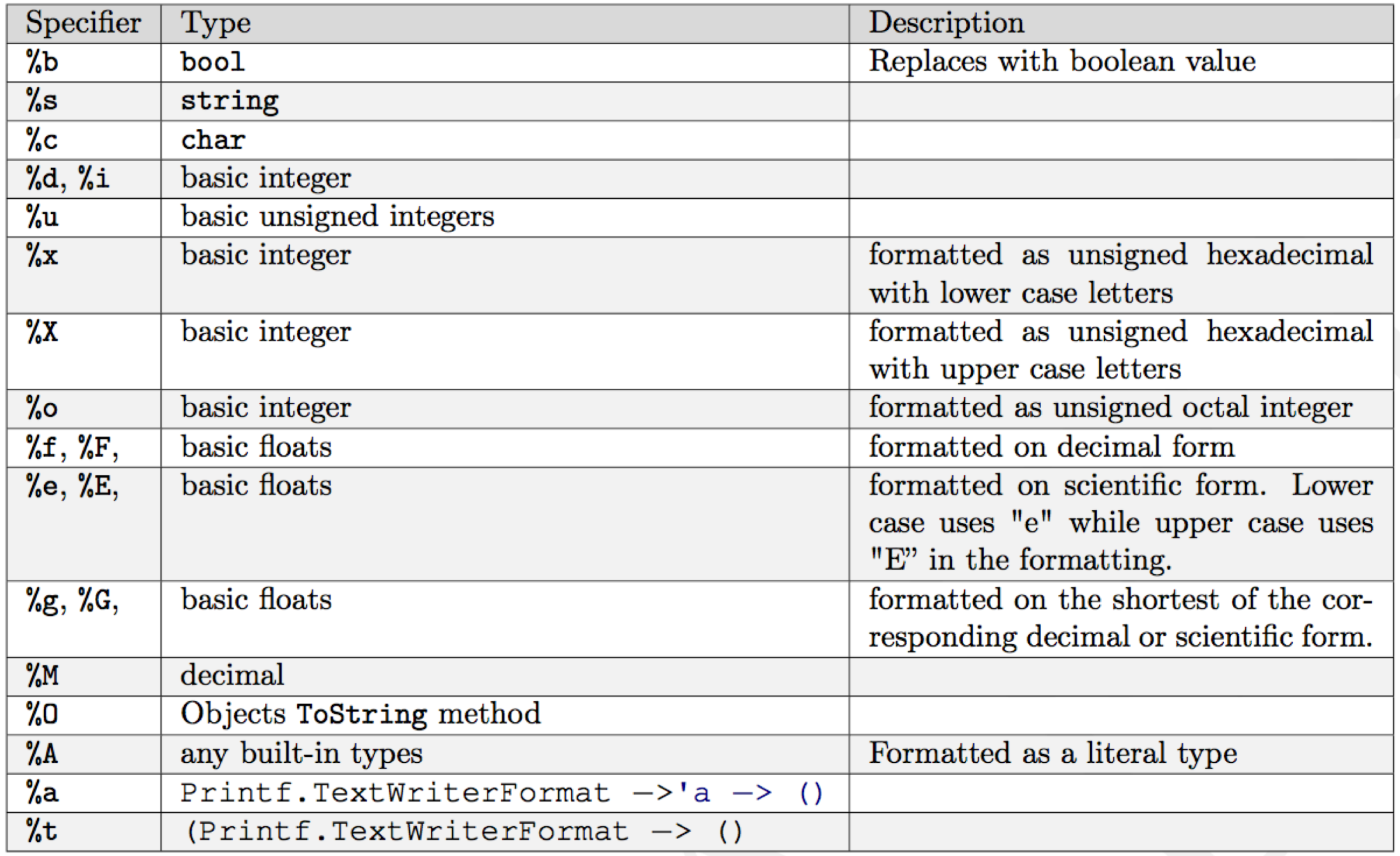

Listing 6.35: printf statement.

```
printf <format-string> {<ident>}
```

printf "The number is %d" 3

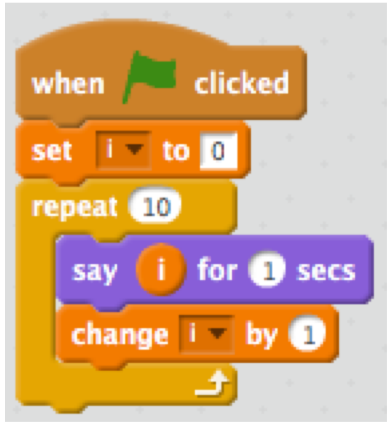
printfn "The number is %d" 3

sprintf "The number is %d" 3



Specifier	Type	Description
%b	bool	Replaces with boolean value
%s	string	
%c	char	
%d, %i	basic integer	
%u	basic unsigned integers	
%x	basic integer	formatted as unsigned hexadecimal with lower case letters
%X	basic integer	formatted as unsigned hexadecimal with upper case letters
%o	basic integer	formatted as unsigned octal integer
%f, %F,	basic floats	formatted on decimal form
%e, %E,	basic floats	formatted on scientific form. Lower case uses "e" while upper case uses "E" in the formatting.
%g, %G,	basic floats	formatted on the shortest of the corresponding decimal or scientific form.
%M	decimal	
%O	Objects ToString method	
%A	any built-in types	Formatted as a literal type
%a	Printf.TextWriterFormat ->'a -> ()	
%t	(Printf.TextWriterFormat -> ()	

Muterbare værdier og løkker



```
when 🏁 clicked
set i to 0
repeat 10
  say i for 1 secs
  change i by 1
```

```
let mutable x = 5
printfn "%d" x
x <- -3
printfn "%d" x
```

```
for i = 1 to 10 do
  printf "%d " i
printfn ""
```

```
let mutable i = 1
while i <= 10 do
  printf "%d " i
  i <- i + 1
printf "\n"
```