

ML

- ML is a functional programming language
- the ML environment runs in an interactive mode

□\$ sml

Standard ML of New Jersey v110.78 [built:
Tue Sep 8 14:59:55 2015]-

-

← ML System Prompt

- At the prompt the system expects a valid sentence in ML

Read Chap 5

ML – Constant Expressions

The simplest sentence in the ML language is a constant expression

Standard ML... Sentence Delimiter

```
- 1234;
```

↑

Internal Variable

val it = 1234 : int

↑ ↑

Value Type of the Value

Other Constants:

real	123.4
bool	true/false
string	"Susan"
char	#"Q"

ML – Operators and Simple Expressions

Example:

```
- ~ 1 + 2 - 3;  
val it = ~2 : int
```

↑
~ is the unary -,
here -2.

string concatenation;
"abc" ^ "def"

- (4 > 0) andalso (4 mod 2 = 0);
val it = true : bool

Precedence	Operators
High	not ~
	* / div mod
	+ - ^
	< > <= >= = <>
	andalso (logical and)
Low	orelse (logical or)

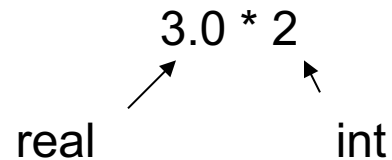
ML – Conditional Expressions

if – then – else
or
if – then

```
-if 1 < 2 then #"x" else #"y";  
val it = #"x" : char
```

ML – Type Conversions

Most programming languages we are used to allow for mixed-type expressions such as



ML does not allow mixed-type expressions.

```
- 3.0 * 2;
```

```
Error: operator and operand don't agree
```

```
operator domain: real * real
```

```
operand:          real * int
```

```
in expression:
```

```
3.0 * 2
```

```
-
```

ML – Type Conversions

However, we can use type conversions to manipulate the types of an expression.

Example:

`real: int → real`
↑
function
name

`int → real`
└──────────┘
signature

conversion function
from integers to reals

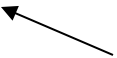
Other conversion functions:

<code>floor: real → int</code>	(round down)
<code>ceil: real → int</code>	(round up)
<code>round: real → int</code>	(round to nearest int)

ML – Type Conversions

We can now rewrite our illegal expression from before:

```
- 3.0 * real(2);
```



Convert 2 into 2.0

```
val it = 6.0 : real
```

or

```
- floor(3.0) * 2;  
val it = 6 : int
```

ML – Variable Definitions

Very simple syntax:

```
- val x = 1 + 2 * floor(3.0);  
val x = 7 : int
```

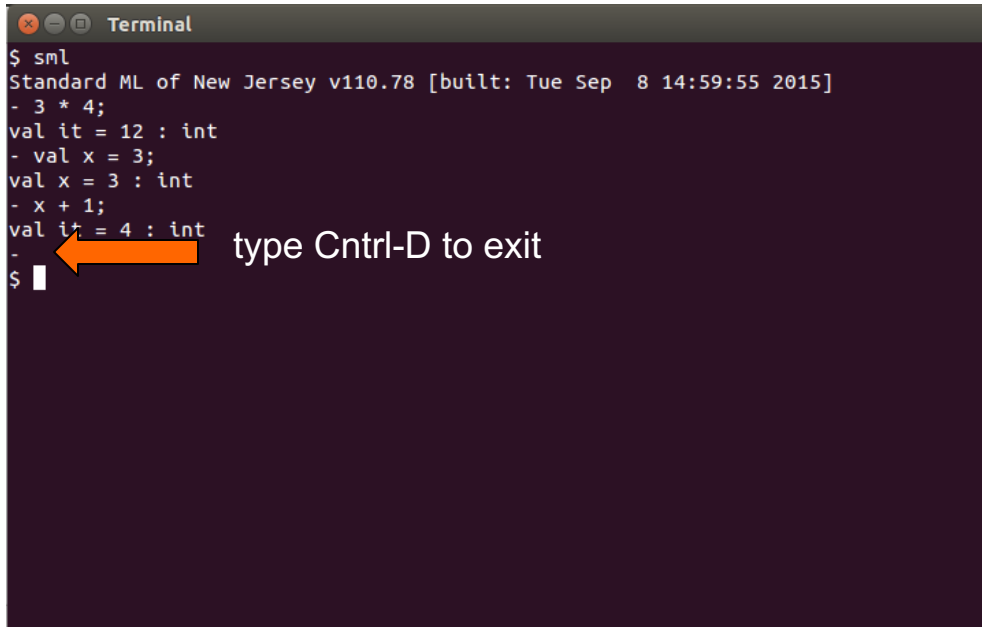
↖
No longer the
default variable

↖ In ML you do not need
to declare the type of a variable;
ML will determine its type through
type inference.

Of course we can use the values of variables:

```
- x + 1;  
val it = 8 : int
```

Try it yourself

A terminal window titled "Terminal" with a dark background. It shows the execution of the SML (Standard ML) interpreter. The prompt is "\$ sml". The output is "Standard ML of New Jersey v110.78 [built: Tue Sep 8 14:59:55 2015]". The user enters a series of commands: "- 3 * 4;", "val it = 12 : int", "- val x = 3;", "val x = 3 : int", "- x + 1;", and "val it = 4 : int". The prompt returns to "\$". An orange arrow points from the text "type Cntrl-D to exit" to the prompt character "\$".

```
$ sml
Standard ML of New Jersey v110.78 [built: Tue Sep 8 14:59:55 2015]
- 3 * 4;
val it = 12 : int
- val x = 3;
val x = 3 : int
- x + 1;
val it = 4 : int
-
$
```

- Log into UbuntuBox
- account: csc301 password: csc301\$is\$fun
- start a terminal
- run sml (see above)