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]-

At the prompt the system expects a valid sentence in ML.
The simplest sentence in the ML language is a constant expression.

Standard ML...
- 1234;
val it = 1234 : int

Sentence Delimiter

Internal Variable
Value
Type of the Value

Other Constants:
real 123.4
bool true/false
string “Susan”
char #”Q”
**ML – Operators and Simple Expressions**

Example:

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

~ is the unary -, here -2.

<table>
<thead>
<tr>
<th>Precedence</th>
<th>Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>not ~</td>
</tr>
<tr>
<td></td>
<td>* / div</td>
</tr>
<tr>
<td></td>
<td>mod</td>
</tr>
<tr>
<td></td>
<td>+ - ^</td>
</tr>
<tr>
<td>Low</td>
<td>&lt; &gt; &lt;= &gt;= = &lt;&gt;</td>
</tr>
<tr>
<td></td>
<td>andalso (logical and)</td>
</tr>
<tr>
<td></td>
<td>orelse (logical or)</td>
</tr>
</tbody>
</table>

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

string concatenation; “abc” ^ “def”
if – then – else
or
if – then

-if 1 < 2 then "x" else "y";
val it = "x" : char
Most programming languages we are used to allow for mixed-type expressions such as

\[
3.0 \times 2
\]

\[
\text{real} \quad \text{int}
\]

ML does not allow mixed-type expressions.

```ml
- 3.0 * 2;
Error: operator and operand don't agree
  operator domain: real * real
  operand:       real * int
  in expression:
    3.0 * 2
- 
```
However, we can use type conversions to manipulate the types of an expression.

Example:

```
real: int → real
```

conversion function from integers to reals

Other conversion functions:

- `floor: real → int` (round down)
- `ceil: real → int` (round up)
- `round: real → int` (round to nearest int)
We can now rewrite our illegal expression from before:

\[-3.0 \times \text{real}(2)\]

\[
\text{val it = 6.0 : real}
\]

or

\[-\text{floor}(3.0) \times 2;\]

\[
\text{val it = 6 : int}
\]
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

- Log into UbuntuBox
- account: user301 password: fun$fun
- start a terminal
- run sml (see above)