

# Types

- Haskell assigns a *type* to each expression, which describes the kind of value represented by the expression. Values can be of a *basic type* such as integer, boolean etc. or a *compound type*. That is, made up of values of other types.
- Examples of basic types:

value	type	
-273	Int	integers
3.14	Float	floating-point numbers
True	Bool	truth values
'A'	Char	characters

# Types

- Examples of compound types:

value	type	
<code>(7, 'Z')</code>	<code>(Int, Char)</code>	tuple
<code>[1, 2, 9]</code>	<code>[Int]</code>	list
<code>"Hello"</code>	<code>String</code>	string (list of characters)
<code>square</code>	<code>Int -&gt; Int</code>	function

Types are useful because some functions only make sense on certain types. For example the expression `square 'A'` would give a type error.



# Numbers

- Numbers in Haskell can be integers or floats.
- Floats must have a fractional part (e.g. 3.0) or an exponential part (e.g. 3E7), or both (e.g. 3.0E7).
- The following infix operators for numbers are built-in (the normal precedences apply):
  - + integer or real addition
  - - integer or real subtraction
  - \* integer or real multiplication
  - / real division
  - `div`, `mod` integer division and remainder



# Examples

---

```
Prelude> 3+4*6
```

```
27
```

```
Prelude> (3+4)*6
```

```
42
```

```
Prelude> div 5 2
```

```
2
```

```
Prelude> 5 `mod` 2;
```

```
1
```

```
Prelude> 65/4
```

```
16.25
```

Note the use of `` to use a function in infix position



# Booleans

- Booleans can have the values `True` or `False`.
- The following operators for booleans are built-in (the normal precedences apply):
  - `&&` logical and
  - `||` logical or
  - `not` logical negation
- The relational operators which can be used to give a boolean result are `==`, `/=`, `>`, `>=`, `<`, `<=`
- Equality can be tested for most types (except functions).



# Examples

---

```
Prelude> True
```

```
True
```

```
Prelude> False || True
```

```
True
```

```
Prelude> 4 > 1+2
```

```
True
```

```
Prelude> 3>6 || 3>2
```

```
True
```

```
Prelude> 3>6 && 3>2
```

```
False
```



# Characters

- Values of type Char are individual characters from the ASCII character set.
- Literal characters are written within single quotes: 'a' 'R'  
'4' '\$'
- Some special characters are represented as follows:
  - \t tab
  - \n new line
  - \f form feed
  - \b back space
  - \\ back slash
  - \' single quote