Types in Asteroid

- **Primitive types**
  - Asteroid’s type hierarchy, boolean < integer < real < string allows for convenient type coercions.
  - The following is a valid program,

```plaintext
let i:%integer = 1.
let k:%real = 3.1 + i.
```

Type coercion from integer to real.
Consider part of Asteroid’s type hierarchy,
- boolean \(<\) integer \(<\) real \(<\) string
- list \(<\) string
- tuple \(<\) string

Printing values is convenient since everything is a subtype of string,

```fsharp
let l : %list = [1,2,3].
let r : %list = l@reverse().
println ("The reversed list is "+r).
```
Types in Asteroid

- Lists in Asteroid are polymorphic in the sense that they do not enforce any kind of type restrictions on their elements.
- This is similar to Python and very different from languages like C++ where this kind of polymorphism can only be achieved via class inheritance.
- The following is legal in Asteroid,

```plaintext
let l:%list = [1,2.0,"three"]
```
Types in Asteroid

- One way to think about tuples is as “fixed length lists”.
  - Once you have decided on the number of components of a tuple you cannot change it.
  - Tuples with different number of components are incompatible.
- The following program will not succeed,

```plaintext
try
  let (x,y) = (1,2,3).
catch _ do
  println ("error: tuples are incompatible").
end
```
Types in Asteroid

- Asteroid uses name equivalence when computing the compatibility of two constructed types