Haskell's Primitive Types

PoDs

The plain old data types, know more formally as *primitive data types*, in Haskell are `Int`, `Char`, `Bool`, `Double`, `Integer`, and `Float`. These are your most basic building blocks in Haskell. Numbers can be stored as `Int`s or `Integer`s if they are whole, and `Double`s or `Float`s if they are fractional. Likewise, characters will be stored as `Char`s, and Booleans will be stored as `Bool`s.

`Int` vs. `Integer`

There are two main ways that Haskell stores integers: `Int` and `Integer`. `Int`s are bounded and limited in size; `Integer`s are unbounded and have unlimited size. This most evident when working with really BIG numbers. You can give this a go right now by typing a really BIG number into GHCi and telling the compiler that it should have type `Int`:

```haskell
9876543798976543245678 :: Int
```

See how the compiler gives you a wee warning? That is because `Int`s are bounded, it will even have told you the range. But why use `Int`s then? Well `Int`s are faster and easier for the computer to work with. However, you should only use `Int`s when you are sure that the number will stay within the range.