`List.drop(Int)`

`List.drop(Int)` returns a list without the first n elements. `List.dropRight(Int)` returns a list without the last n elements.

```
val source = Source.fromFile(filename)
for(line <- source.getLines)
  println(line)
```

Use `ListBuffer` then call `toList`.

`List.map(func)`

Creates a list composed of the original elements run through a filter.

Pronounced "cons", prepends an element to the list.

`List.exists(func)`

Determines whether an element of the list satisfies the `func` predicate.

```
1::2::3::Nil
```

```
my.Obj.update(i,j)
```

`List.foreach(func)`

Executes `func` on every element of the list.

It's like an immutable list, but can contain objects of multiple types.

`List.last` returns a list without the first element.

`List.tail` returns only the last element.

Operators are methods of the *left* operand unless they end with `:`, in which case they are methods of the *right* operand.

Concatenates two `Lists` into a new list.

`Arrays` are immutable and invariant. `Lists` are mutable and covariant.

Also, arrays aren't really Scala collections as they don't inherit `Traversable`.

`List.mkString(str)`
`List.mkString()` uses no separation.

Makes a `String` with elements of a `List` separated by the argument to `mkString`.

`myTuple._1`

Not `myTuple._0`

`List.forall(func)`

Returns whether all elements of the `List` satisfy the `func` predicate.

`List.filter(func)`

Returns a `List` of all elements satisfying the `func` predicate.

```
myObj.apply(i)
```

`List.head` returns the first element.
`List.init` returns a `List` without the last element.