What are Clojure's basic numerical types?

Integers, floating point numbers, ratios, and characters.

What are "keywords" in Clojure terminology?

Atoms, as in Prolog or Erlang, not reserved words.

Anonymous functions are also called ...

... lambdas.

What is Clojure's fundamental collection type?

What methods define it?

Sequences are the abstraction above lists, vectors, maps, sets, etc.

They share the methods `first`, `rest`, and `cons`.

Sequences are roughly equivalent to what in Java?

Iterators

What do predicates return in idiomatic Clojure?

Instead of returning the atoms `false`/`true`, they take advantage of the fact that anything other than `false`/`nil` is "true" and return a more meaningful value.

How can you create a list without invoking it as a function?

By quoting:

```
'(a b c)

(quote (a b c))
```

How are special forms different from functions?

What's the catch?

Special forms get arguments unevaluated, controlling if/when to evaluate them.

However, special forms are *not* first-class values.

Give two methods of division.

`/` which returns a `Ratio`.

`quot` performs truncating integer division.

What's the difference between `rem` and `mod`?

They differ in their handling of signs. If the first and second arguments have different signs, the result of `mod` will have the same sign as the second argument, while the result of `rem` will have the same sign as the first argument.

Describe negation.

`not` and `not=` are provided.

What's the difference between = and ==?

`==` only compares arguments that can be case to `java.lang.Number`.

`=` compares arguments in a type-independent manner. For example, vectors can be equal to lists according to `=`.