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.

CIS 554

Clojure

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.

Clojure

5

What's the problem with if?

The nesting of many ifs is difficult to read.

The special form <code>cond</code> allows as many arguments as needed for if ... then ... elseif ... then ... elseif ...

cond provides :else to catch everything as the final branch.

Sequences are roughly equivalent to what in Java?



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.

CIS 554

Clojure

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 ==?

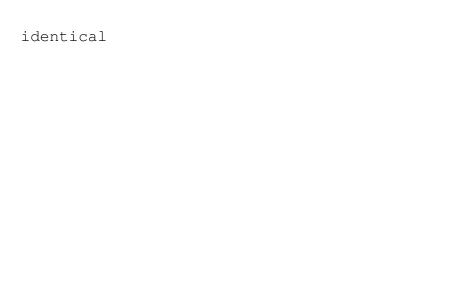
- == only compares arguments that can be case to java.lang.Number.
- = compares arguments in a type-independent manner. For example, vectors can be equal to lists according to =.

Unlike other Lisps, the order of functions ...

... matters. Forward-references are not allowed.

Clojure

Test object identity.

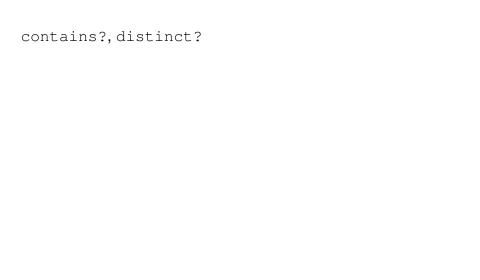


Test for collection type.

coll?, seq?, vector?, list?, map?, set?

Test a collection for membership.

For non-reptition.



In Clojure the standard streams have ...

... special reassignable names.

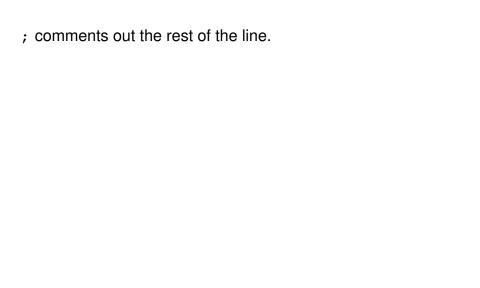
in, *out*, *err*

Get back recent repl values.

*1, *2, *3 are the first, second, and third most recent values.

*e is the most recent exception.

What is the comment syntax?



In place of classes, Clojure has ...

What is their syntax?

```
... structs
```

```
Define with:
```

```
(defstruct name :field1 :field2 ...)
```

Instantiate with:

```
(struct name val1 val2 \dots)
```

Access fields with:

```
(:fieldName structName)
```