What is a record?

An aggregate data type that consists of labels mapped to arbitrary values.

Give the syntax of records definition.

```
type name =
    { name1 : type1;
      ...
      nameN : typeN
    }
```

Instantiate a record.

3

```
let name =
   { name1 = val1;
     ...
     nameN = valN;
   }
```

How can the members of a record be accessed?

- With projection: `record.label`
- With pattern matching

```
let { name1 = var1; name2 = var2 } = expr
```

Not all fields must be matched.

Functionally "update" a tuple.

```
let name = { someTuple with field1 = val1; ... }
```

The fields must be taken from `someTuple`.

How can mutable records be created?

And updated?

In the declaration prefix one or more of the field names with the `mutable` keyword.

```
mutableRecord.field <- newVal
```

Oddly, the namespace of field labels is ...

... shared between all toplevel records.

Values of records with fields of the same name overwrite values of fields in previous records.

What is the syntax of arrays?

The same as lists, except for the pipe next to the brackets:

```
[| val1 ; ... ; valN |]
```

Access the elements of an array.

Update an array element.

- Use pattern matching.
- Access directly with `arr.(index)`.

```
arr.(index) <- newVal
```

How can arrays be created?

- Using the literal syntax.
- Using `Array.create len initVal`