What mathematical notation do we introduce for pairs?

The following two are equivalent:

```
pair x y
(x, y)
```

Destruct a pair.

```
destruct p as (n, m).
```

For whatever reason there are no brackets around the pattern.

What notation do we introduce for lists?

List literals:

```
[]
[1, ..., n]
```

Cons:

```
el :: lst
```

What notation do we introduce for appending two lists?

++

What does `hd []` result in?

default

What's an alternative to `if`/`then`/`else` expressions.

```
if X then Y else Z.

match X with
| true  => Y
| false => Z
end.
```

What's another use for `Definition`, apart from creating functions?

Creating type aliases:

```
Definition bag := natlist.
```

Unlike in a Scala `def`, the RHS of a Coq `Definition` can simply be what?

The name of another `Definition`.

```
Definition sum : bag => bag -> bag := app.
```

Demonstrate induction on lists.

```
induction l as [| n l']
Case "l = nil".
  ...
Case "l = cons n l'".
  ...
```

What is the `SearchAbout` command?

```
SearchAbout foo
```

Prints all the theorems Coq knows that involve `foo`.

Describe the `apply` tactic.

It allows you match the current goal to the conclusion of a conditional hypothesis of the current context. Like modus ponens in reverse. Keep in mind a non-conditional statement can be viewed in this case as a degenerate conditional, allowing you to match the current subgoal to a hypothesis and end the proof.