Ctrl+G, q, Enter

```
base#num
```

For example, `16#b` is 11.

No, variables are restricted to the scope in which they were defined, typically a clause of a function. There's no wider scope that will accept variable definitions.

Pattern matching, not assignment.

The right side is evaluated first and then matched against the left side.

Not only do you need to find a variable with the wrong value, you also need to figure out when that unexpected value was assigned. Without reassignment the second part of that process is eliminated.

Integers are exact and arbitrary-size values. Integer division results in floating-point results without truncation.

They begin with a lower case letter, and continue with letters, underscores, or at signs.
Atom interpretation can also be forced on other sequences by surrounding them with single quotes.

Global constants defined in a header using `#define`.

- Tuples don't define a new type.
- The members of a tuple don't have names. That can be accomplished by nesting tuples, however.

They have global scope, unlike variables.

In Erlang it's an error, while Prolog would backtrack.

An anonymous variable.

You can prepend/match multiple elements at once without creating a wrapper list.

```
A = [1, 2 | [3, 4]]

[B, C, D, | T] = A.
```

`$X` where `X` is some character.

This is **not** a "conversion", just syntactic sugar for an integer.