# EBNF Grammar condensed



EBNF means Extended Backus-Naur Form.
An EBNF description is an unordered list of EBNF rules.
Each rule has 3 parts:
"left-hand side" (LHS),
"right-hand side" (RHS),
and the "is defined as" ('=') character.

The LHS contains one (possibly hyphenated) word written in lower-case.
It names the EBNF rule.
The RHS supplies the definition associated with this name.
It can include combinations of the four control forms:

| | |
|---|---|
| ***Sequence:*** | ***a) items appear left-to-right*** |
| | ***b) their order is important*** |
| ***Choice:*** | ***a) alternative items are separated by a \| ('pipe')*** |
| | ***b) one item is chosen from this list of alternatives*** |
| | ***c) their order is insignificant*** |
| ***Option:*** | ***a) an optional item is enclosed between [ and ] ('square brackets')*** |
| | ***b) the item can either be included or discared*** |
| ***Repetition*** | ***a) a repeatable item is enclose in { and } ('curly braces')*** |
| | ***b) the item can be repeated zero or more times (possibly infinitely)*** |

EBNF rules can include six characters for special meanings:
=, |, [, ], {, }
Except for these characters and the names of EBNF rules, anything that appears in an RHS stands for itself, such as letters, digits, punctuation marks, and other characters.

=== Advanced EBNF ===

Recursive EBNF Descriptions can contain rules that are either directly recursive or mutually recursive.
Such Rules user their names in a special way.
A directly recursive EBNF rule uses its own name in its definition.
Directly recursive rules must include at least one alternative that is not recursive, otherwise they describe rules of infinite length.
Recursion can always replace repetition, but the inverse is not true, because recursion is more powerful than repetition.

We can also replace any option control form by an equivalent choice control form that contains an empty symbol.

```
sign    = | + | -
digit   = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
digits  = | digit digits
integer = sign digit digits
```

As opposed to:

```
sign    = + | -
digit   = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
integer = [sign]digit{digit}
```