(* The start of a nix expression *)
Expression →

(*ExpressionNoAssignment* | *Assignment*)

ExpressionNoAssignment →

(*Attrset* | *Function* | *Arithmetic*)

(* A let .. in or with *)
AssignmentPreamble →
{ (*let-in* | *with*) }+,

(* Let in *)
(* (* `let a = 5; in ...` *) *)
let-in →
`let` { *AssignmentNoPreamble* } `in`

with →
`with`

{ ExpressionNoAssignment }+ `;`

Inherit →
`inherit` ({ ? identifier ? }+) `;`

(* Expression of an attrset *)
Attrset →
[*AssignmentPreamble*] `{` { (*Assignment* | *Inherit*) }+ `}`

(* An assignment *)
(* (* `let x = 5; in a = x;` *) *)
Assignment →
[*AssignmentPreamble*] *AssignmentNoPreamble*

(* An assignment without the preamble *)
(* (* `a = 5;` *) *)
AssignmentNoPreamble →
? someIdentifier ? `=` *ExpressionNoAssignment* `;`

(* A string, integer or float *)
Primary →
(? string ? | ? integer ? | ? float ? | `true` | `false` | *List*)

List →
`[` *ExpressionNoAssignment* `]`

(* A function *)
(* (* `input: output` *), *)
Function →
[AssignmentPreamble] ? InputIdentifier ? `:` [let-in] *ExpressionNoAssignment* `;`

Arithmetic →
(`(` *ArithmeticOrPrimary* { (`-` | `+` | `*` | `/`) *ArithmeticOrPrimary* } `)` |
ArithmeticOrPrimary { (`-` | `+` | `*` | `/`) *ArithmeticOrPrimary* })

ArithmeticOrPrimary →
(*Primary* | *Arithmetic*)