ReservedKeywords :

Pragma :

Abstract :
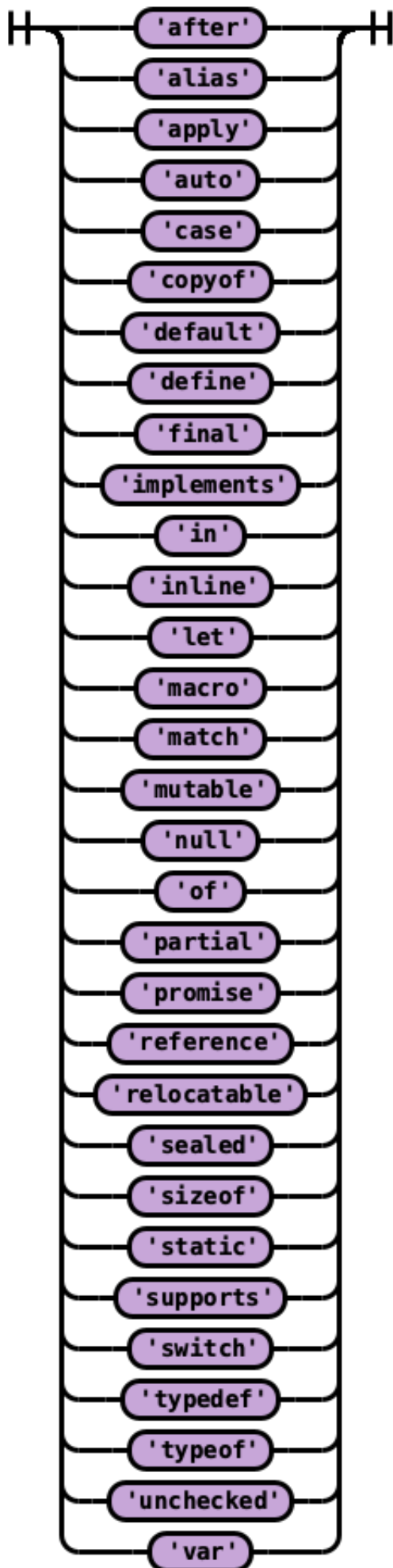

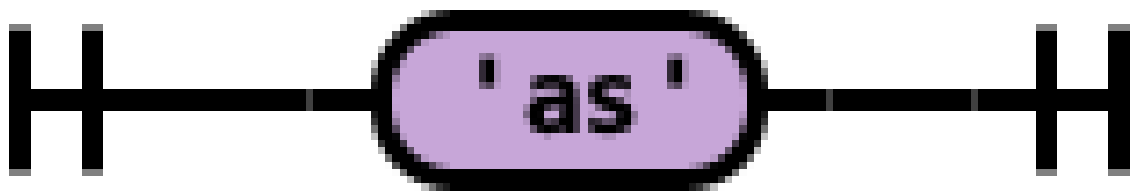Anonymous :

Address :

As :

```
'after'
'alias'
'apply'
'auto'
'case'
'copyof'
'default'
'define'
'final'
'implements'
'in'
'inline'
'let'
'macro'
'match'
'mutable'
'null'
'of'
'partial'
'promise'
'reference'
'relocatable'
'sealed'
'sizeof'
'static'
'supports'
'switch'
'typedef'
'typeof'
'unchecked'
'var'
```

Assembly :

Bool :

Break :

Bytes :
Calldata :

H——('bytes')——H

H——('calldata')——H

Catch :

H——('catch')——H

Constant :

H——('constant')——H

Constructor :
Continue :

'constructor'

'continue'

Contract :

'contract'

Delete :

'delete'

Do :
Else :

Emit :
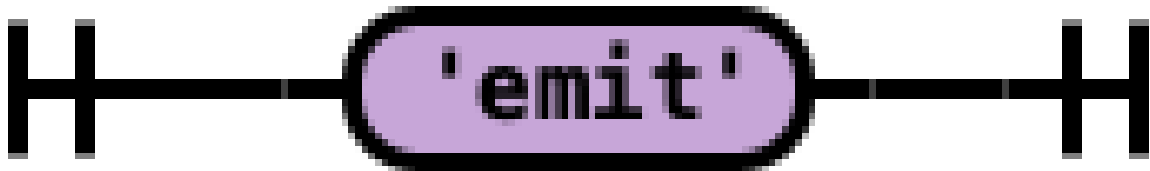
Enum :

Event :
External :

'event'

'external'

Fallback :

'fallback'

False :

'false'

Fixed :

'fixed'

'fixed' [0-9] 'x' [0-9]
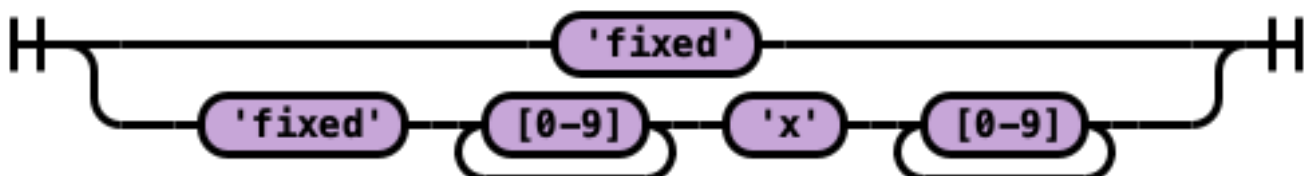
From :

FixedBytes :
For :

Function :

Hex :

If :

'byte'
'bytes1'
'bytes2'
'bytes3'
'bytes4'
'bytes5'
'bytes6'
'bytes7'
'bytes8'
'bytes9'
'bytes10'
'bytes11'
'bytes12'
'bytes13'
'bytes14'
'bytes15'
'bytes16'
'bytes17'
'bytes18'
'bytes19'
'bytes20'
'bytes21'
'bytes22'
'bytes23'
'bytes24'
'bytes25'
'bytes26'
'bytes27'
'bytes28'
'bytes29'
'bytes30'
'bytes31'
'bytes32'
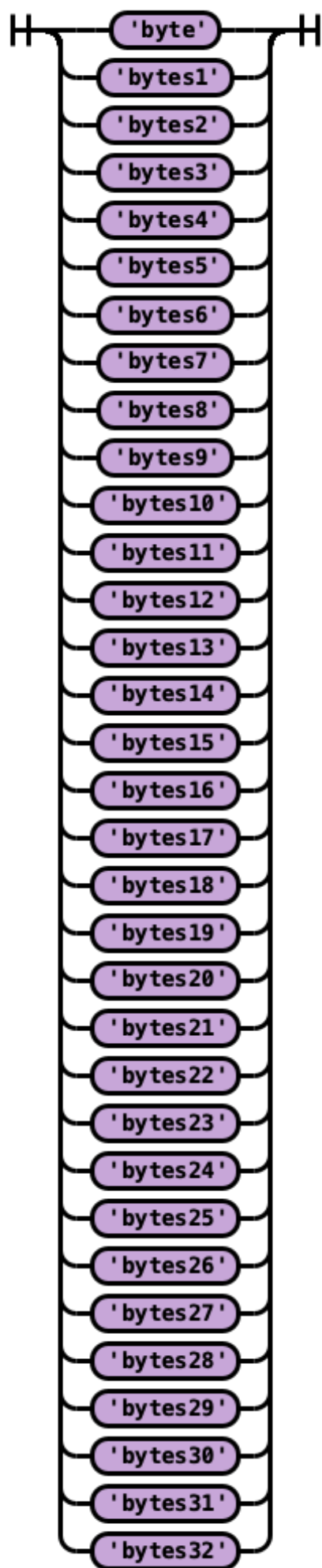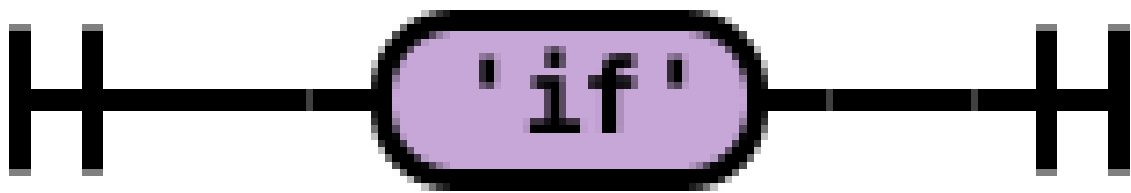
Immutable :

Import :

Indexed :

Interface :
Internal :

```
|—| ——— ('interface') ——— |—|
```

```
|—| ——— ('internal') ——— |—|
```

Is :

```
|—| ——— ('is') ——— |—|
```

Library :

```
|—| ——— ('library') ——— |—|
```

Mapping :
Memory :

'mapping'

'memory'

Modifier :

'modifier'

New :

'new'

NumberUnit :
Override :

'wei'
'gwei'
'ether'
'seconds'
'minutes'
'hours'
'days'
'weeks'
'years'

Payable :

```
|H|————————( 'override' )————————|H|
```

```
|H|————————( 'payable' )————————|H|
```

Private :

```
|H|————————( 'private' )————————|H|
```

Public :

```
|H|————————( 'public' )————————|H|
```
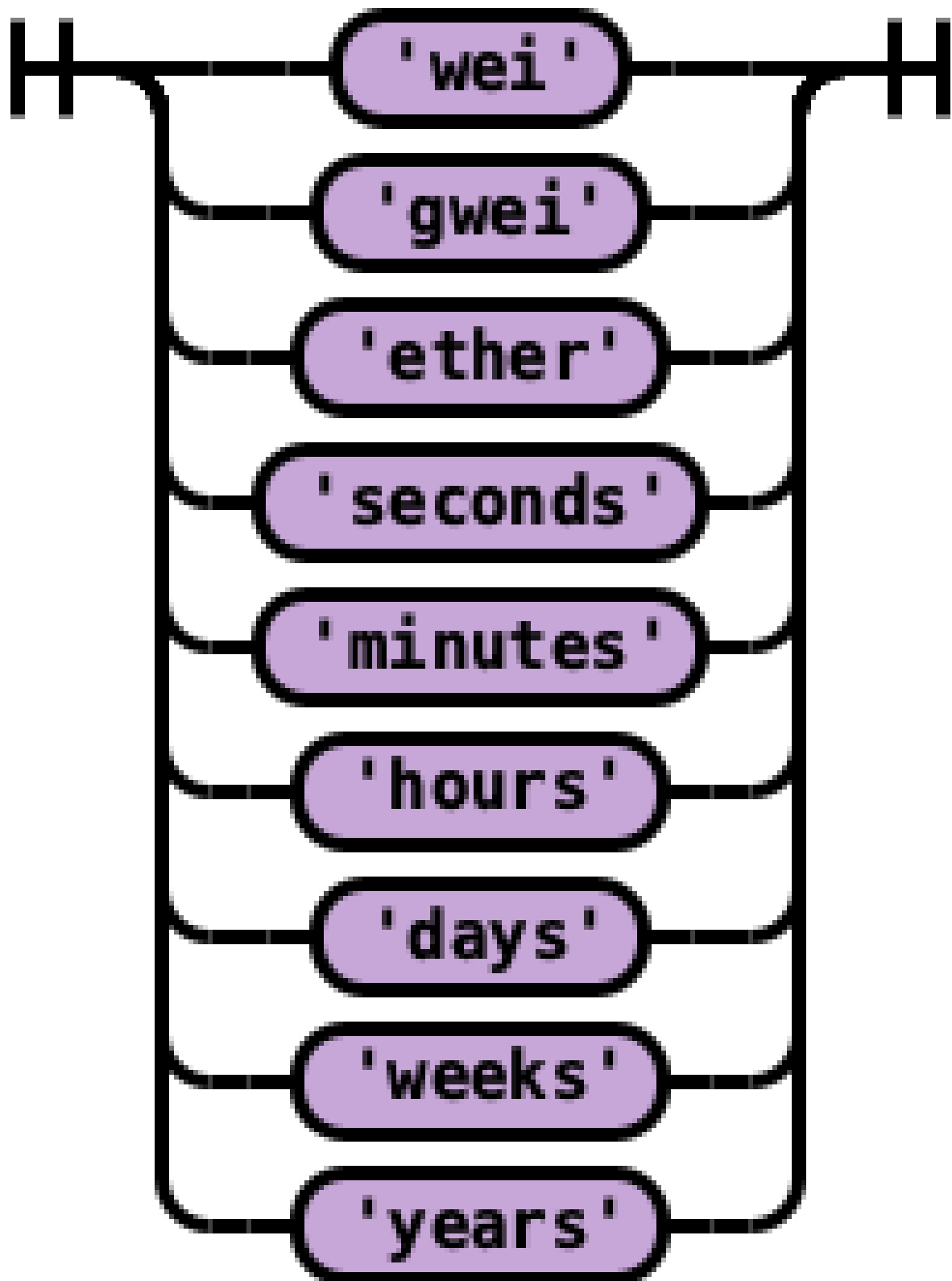
Pure :
Receive :

Return :

Returns :

SignedIntegerType :
Storage :

```
'int'
'int8'
'int16'
'int24'
'int32'
'int40'
'int48'
'int56'
'int64'
'int72'
'int80'
'int88'
'int96'
'int104'
'int112'
'int120'
'int128'
'int136'
'int144'
'int152'
'int160'
'int168'
'int176'
'int184'
'int192'
'int200'
'int208'
'int216'
'int224'
'int232'
'int240'
'int248'
'int256'
```

'storage'
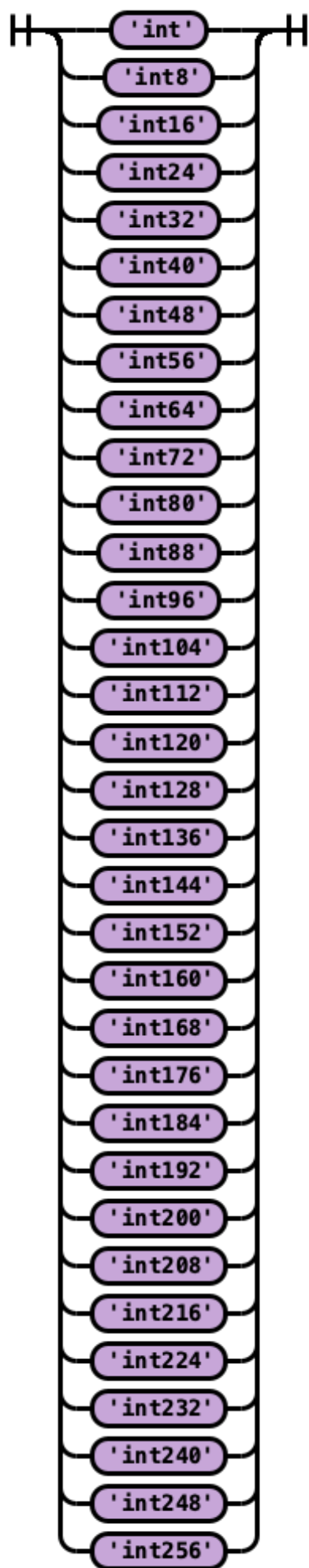
String :

'string'

Struct :

'struct'

True :

'true'
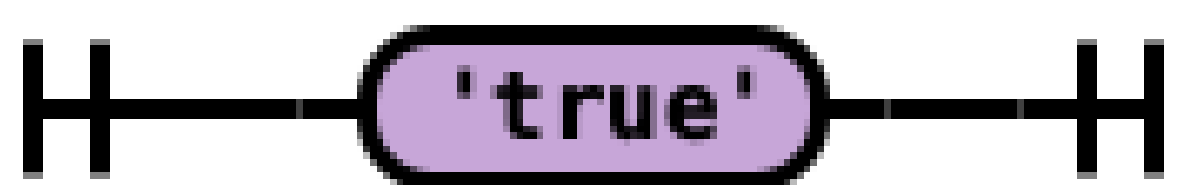
Try :
Type :

Ufixed :
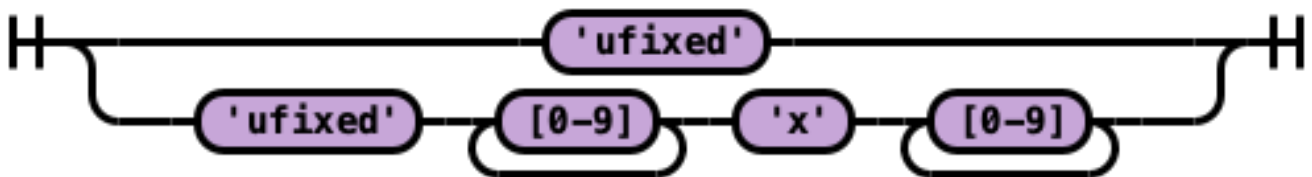
UnsignedIntegerType :
Using :

View :

'uint'
'uint8'
'uint16'
'uint24'
'uint32'
'uint40'
'uint48'
'uint56'
'uint64'
'uint72'
'uint80'
'uint88'
'uint96'
'uint104'
'uint112'
'uint120'
'uint128'
'uint136'
'uint144'
'uint152'
'uint160'
'uint168'
'uint176'
'uint184'
'uint192'
'uint200'
'uint208'
'uint216'
'uint224'
'uint232'
'uint240'
'uint248'
'uint256'
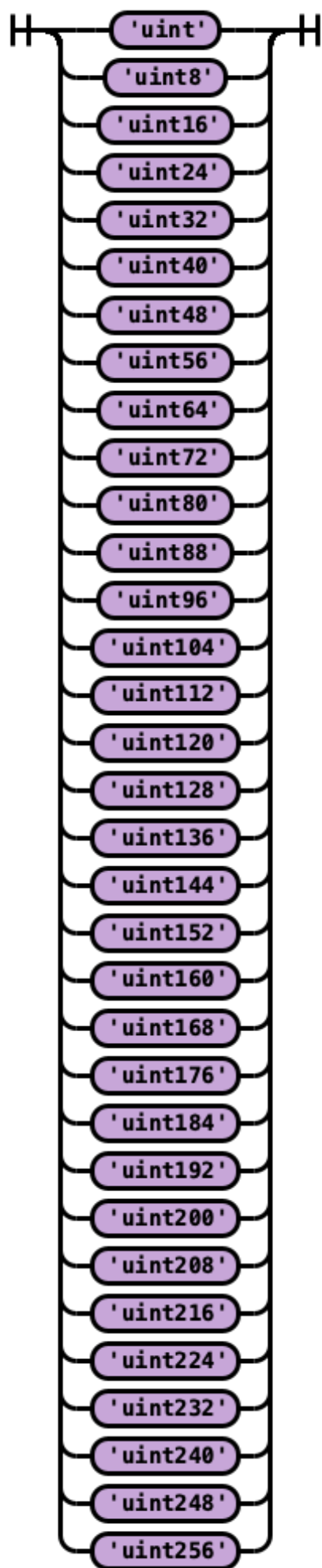
Virtual :

While :
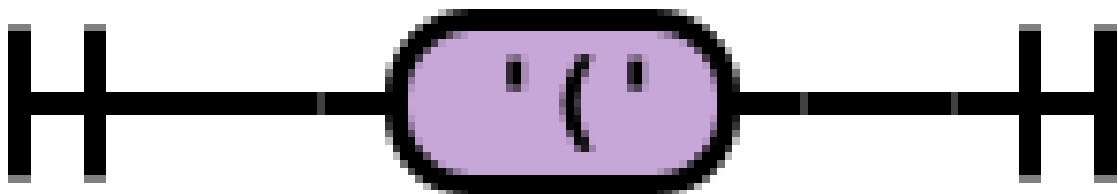
LParen :

RParen :
LBrack :
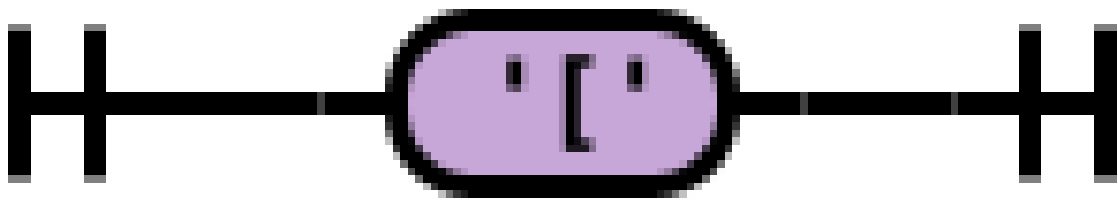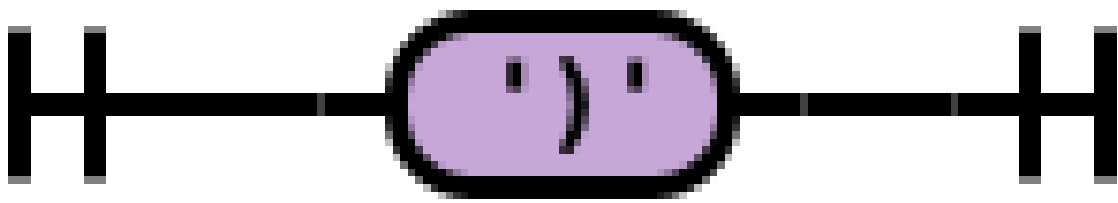
RBrack :
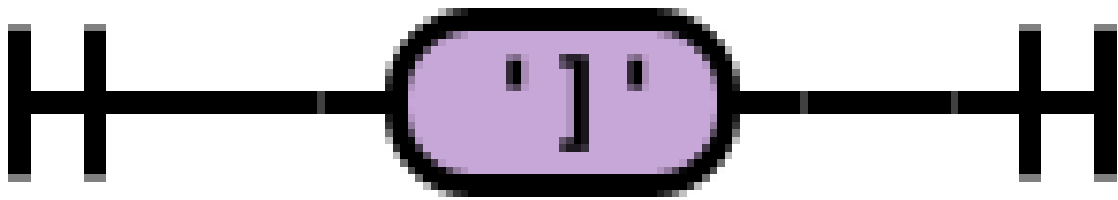
LBrace :
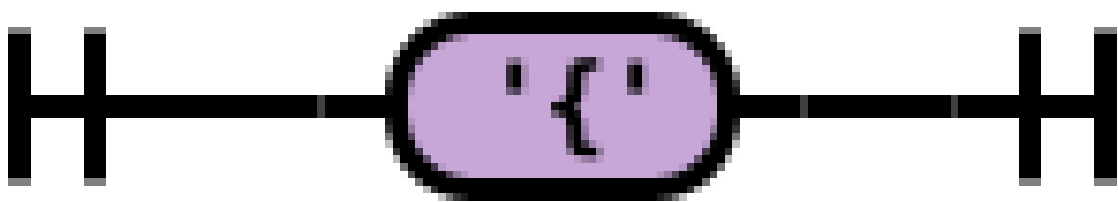
RBrace :
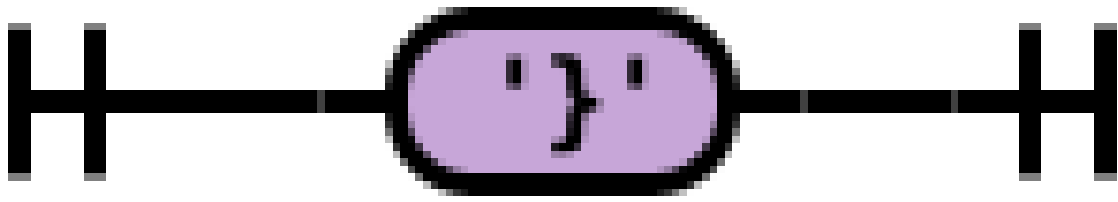
Colon :
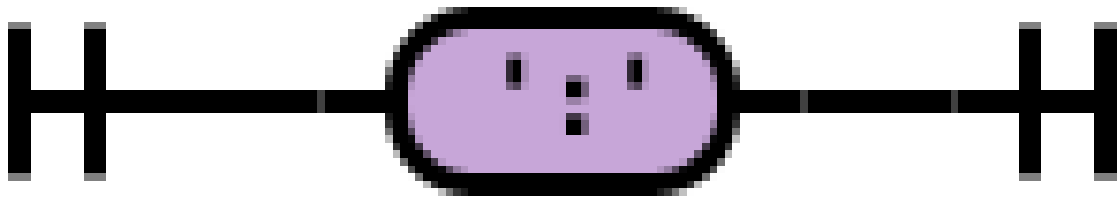
Semicolon :
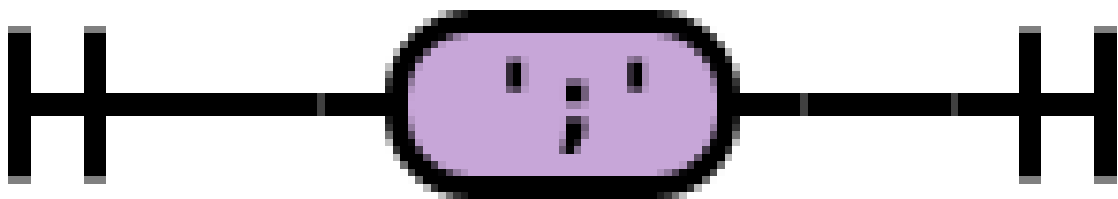
Period :
Conditional :
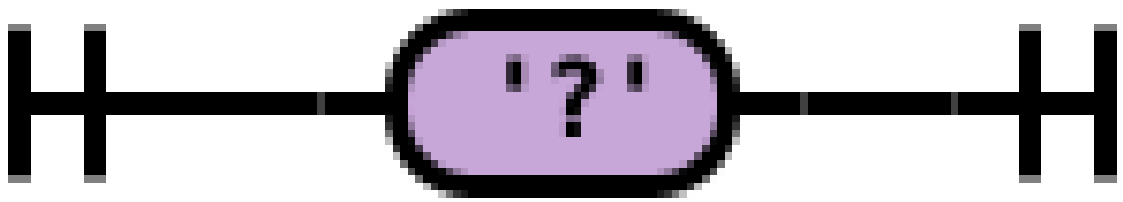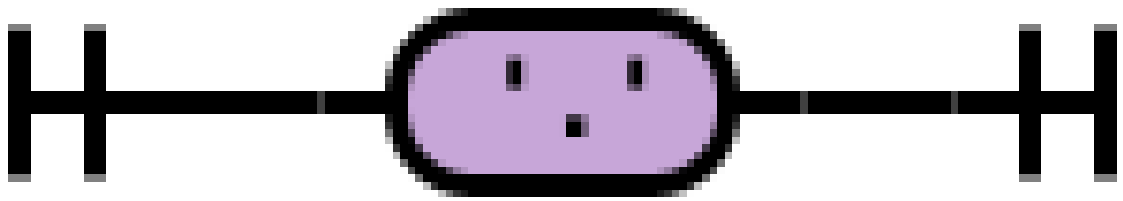
DoubleArrow :
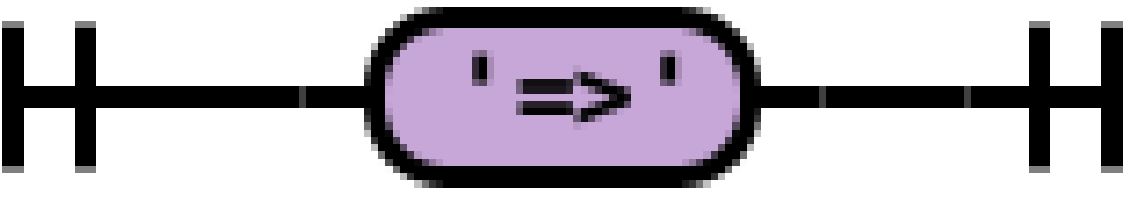
RightArrow :
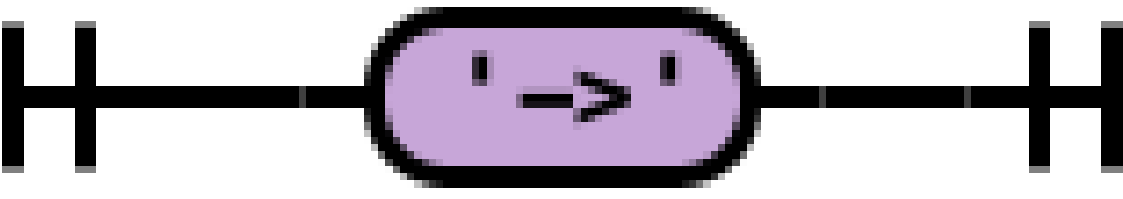
Assign :
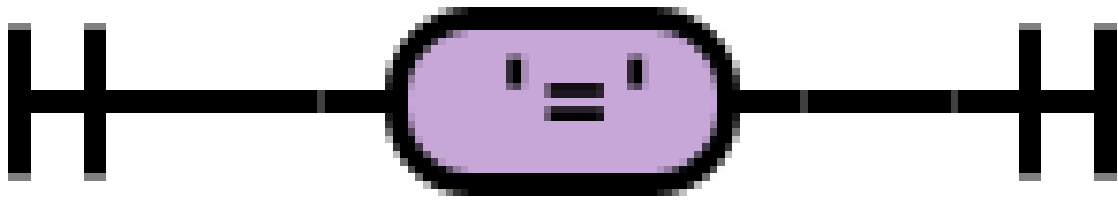
AssignBitOr :
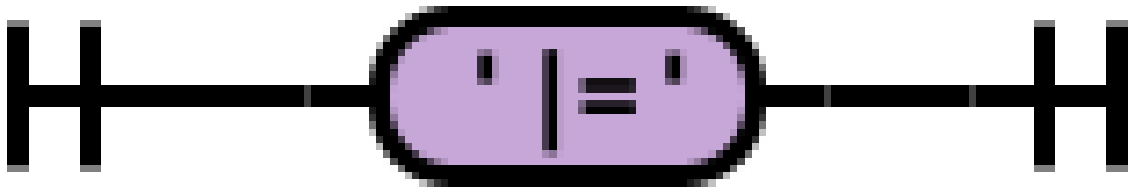
AssignBitXor :
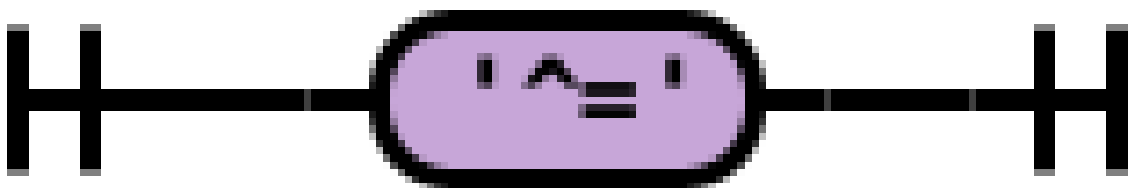
AssignBitAnd :
AssignShl :

'&='

'<<='
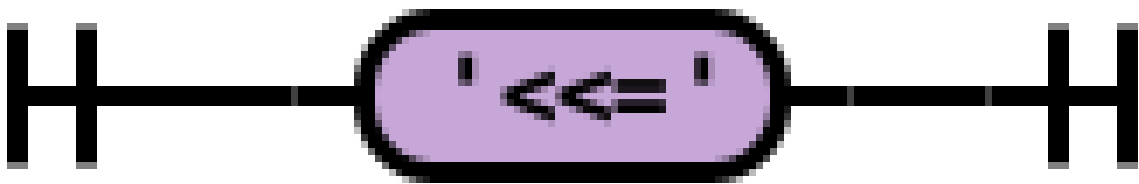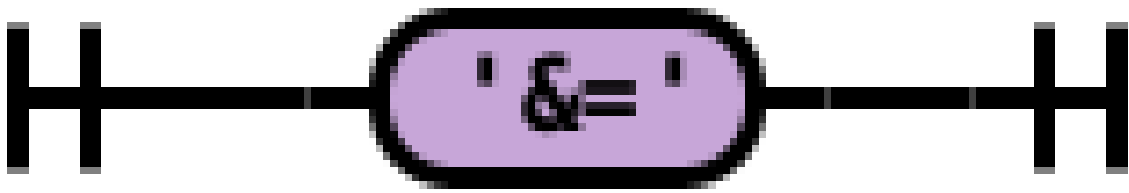
AssignSar :

'>>='

AssignShr :

'>>>='

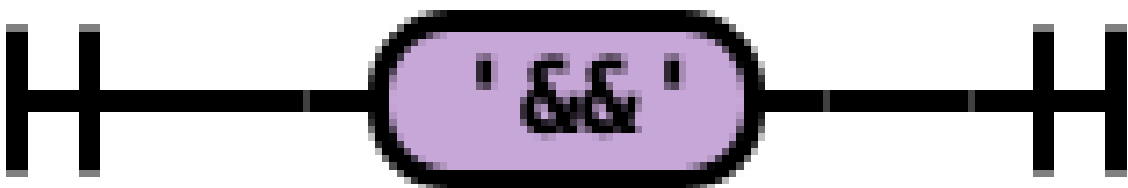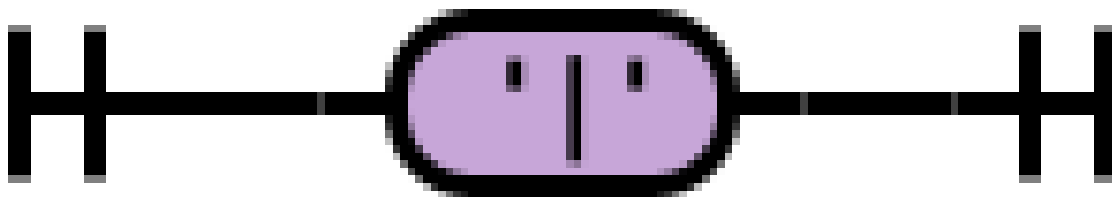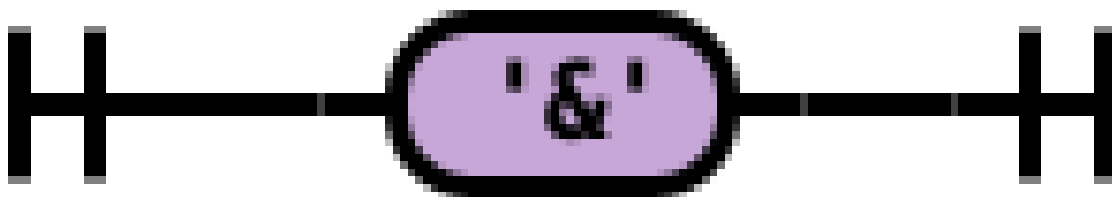AssignAdd :
AssignSub :
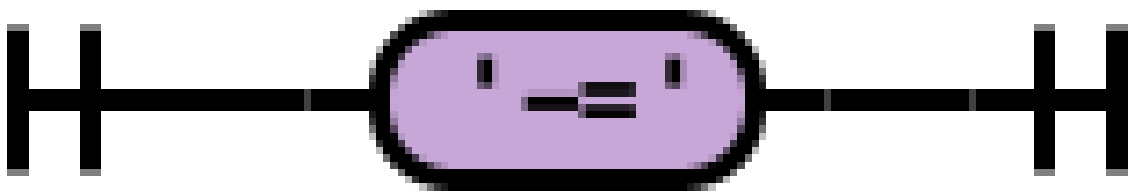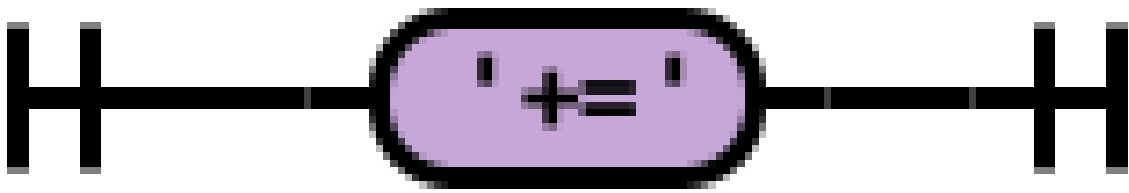
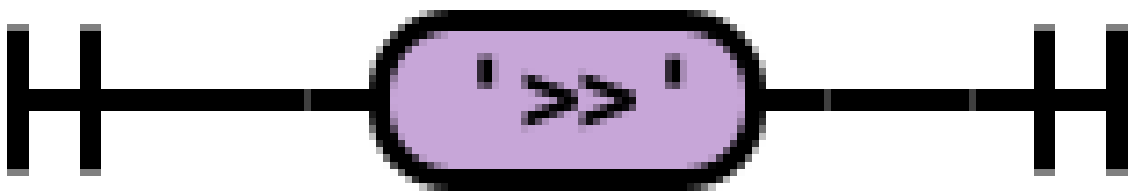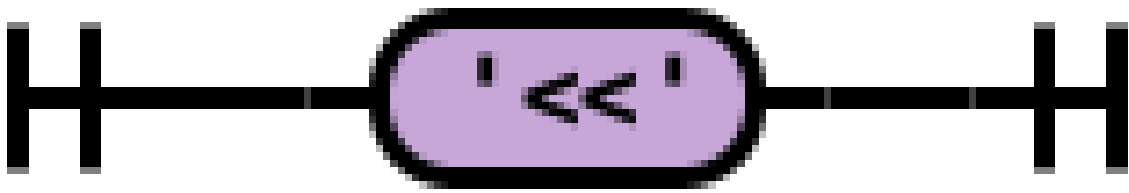AssignMul :
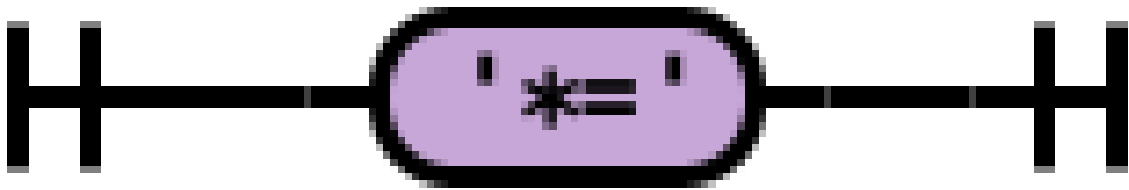
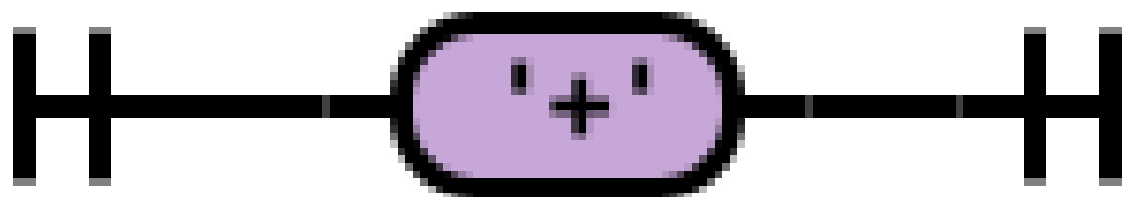AssignDiv :
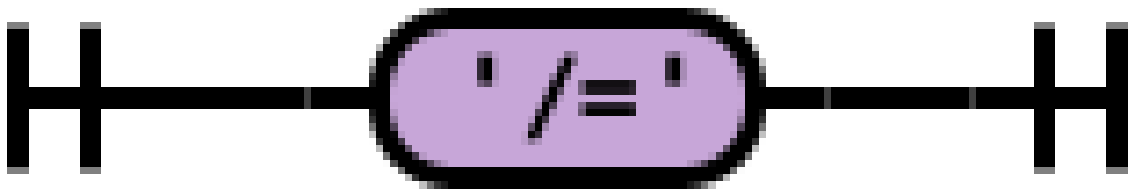
AssignMod :
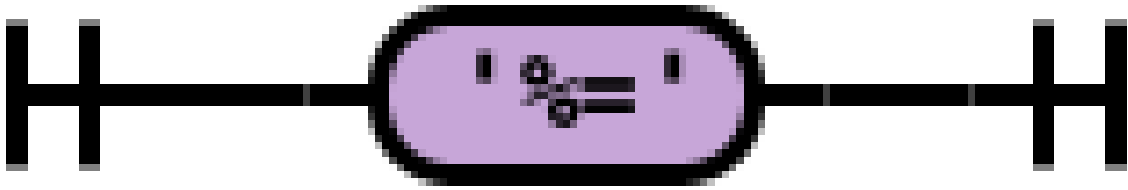
Comma :
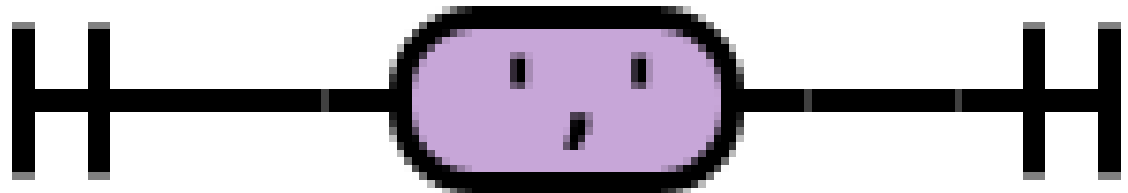
Or :
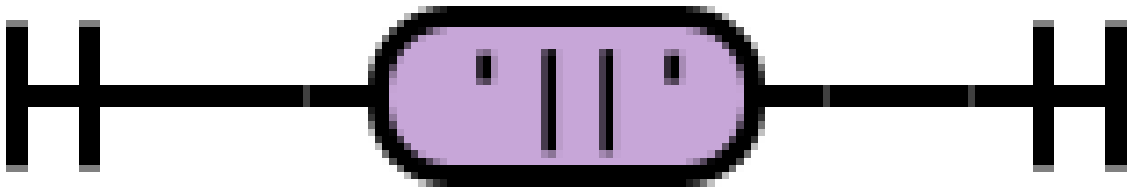
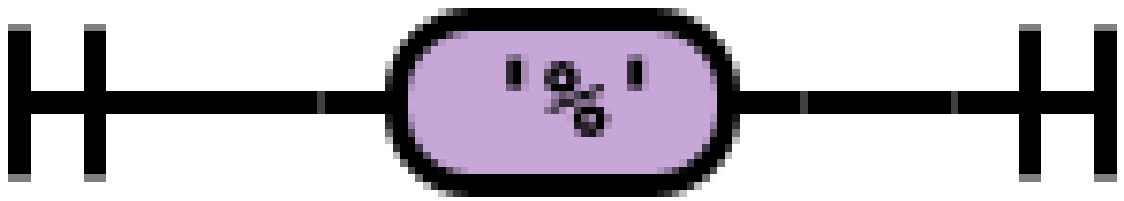And :

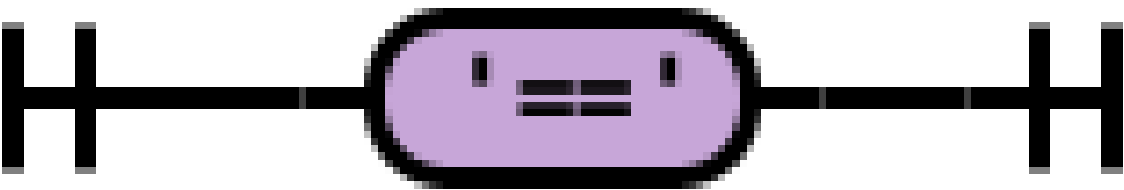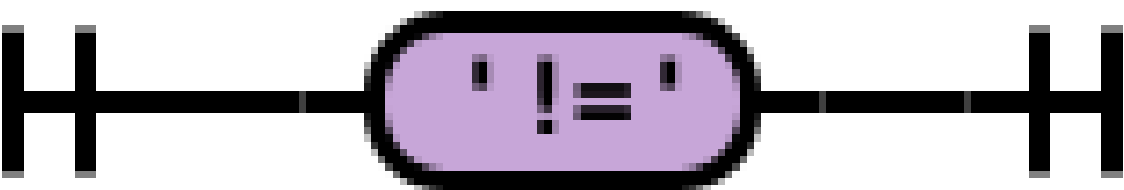BitOr :

BitXor :

BitAnd :

Shl :
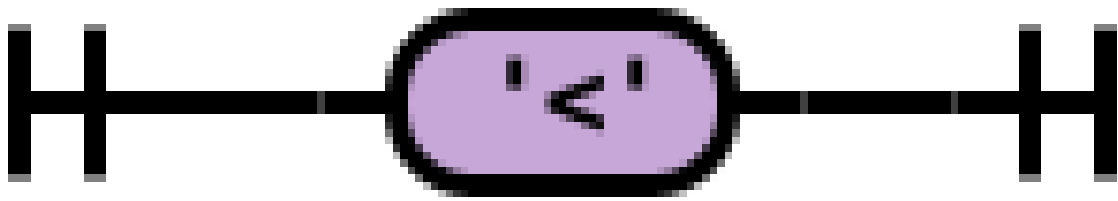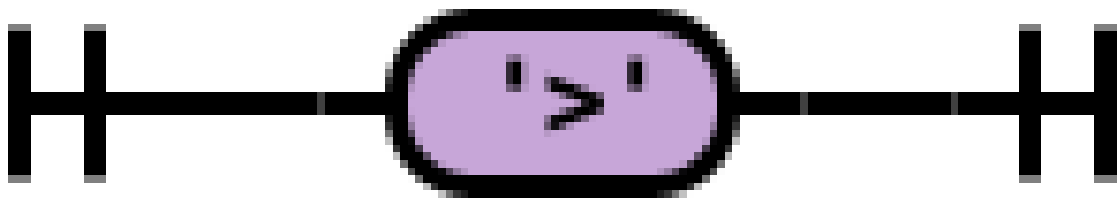Sar :

Shr :
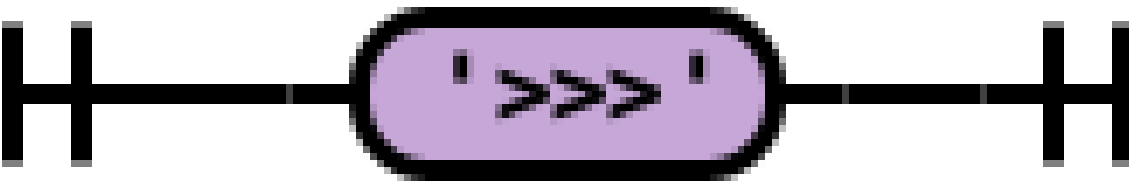
Add :

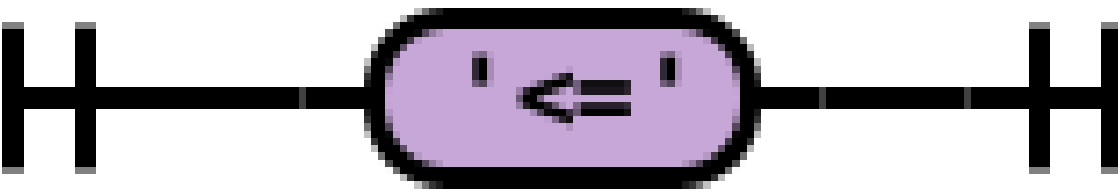Sub :
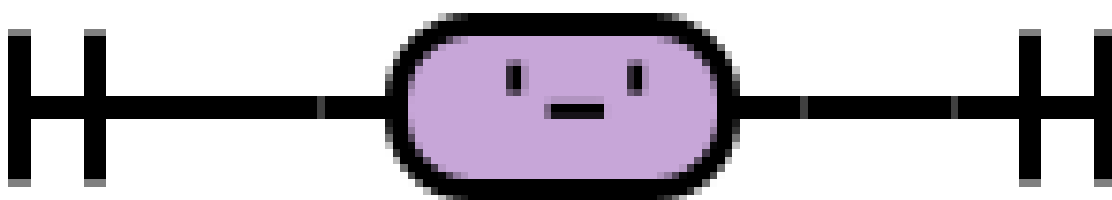
Mul :
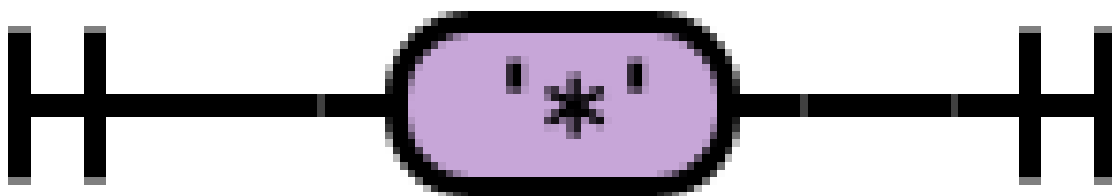
Div :
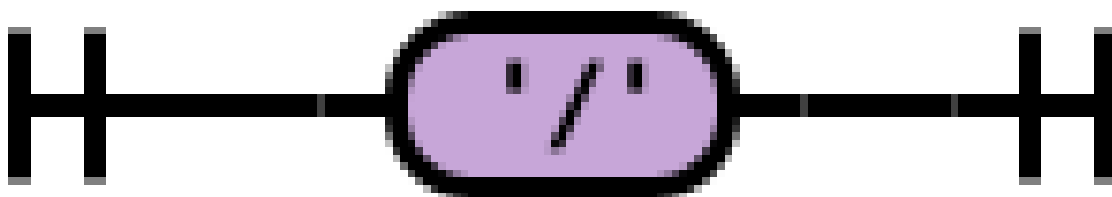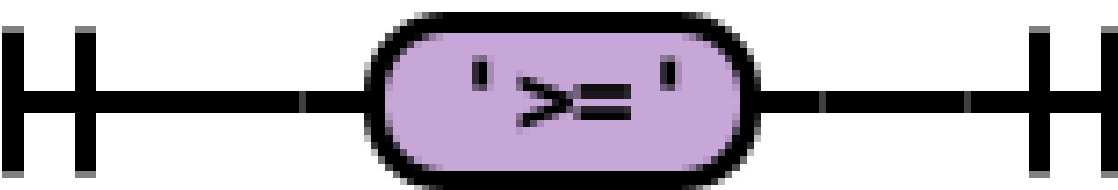

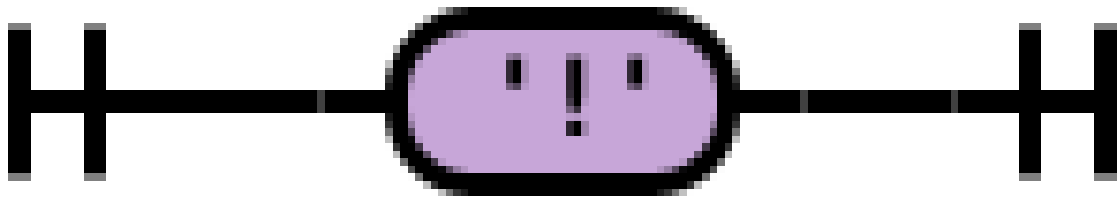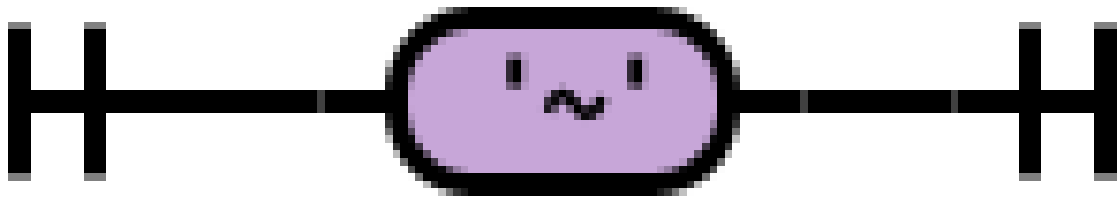Mod :
Exp :
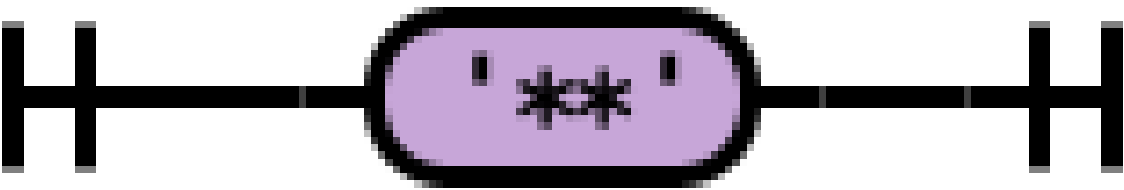
Equal :

NotEqual :

LessThan :

GreaterThan :
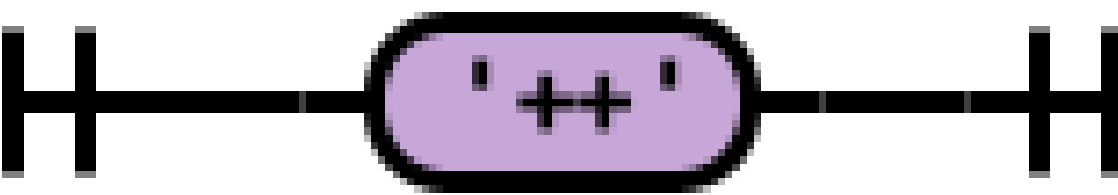

LessThanOrEqual :

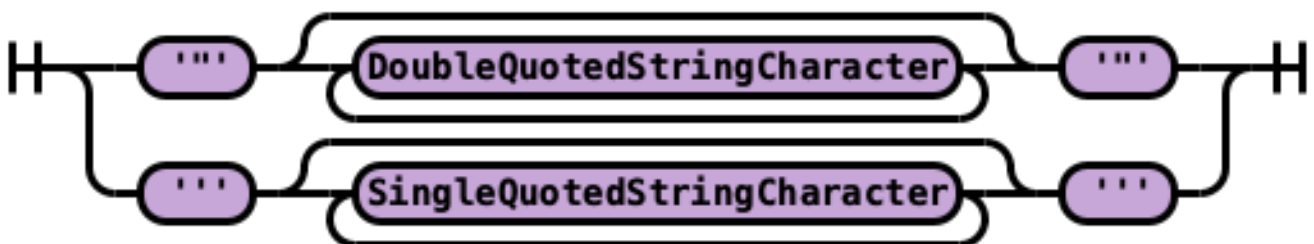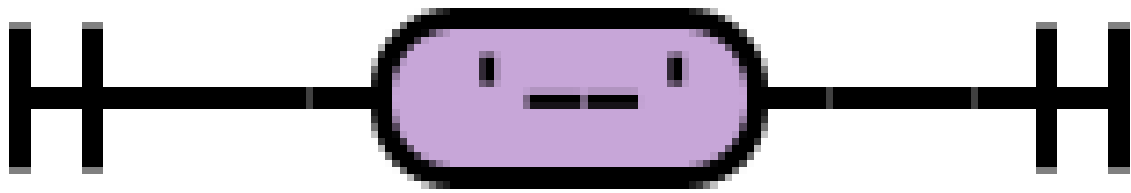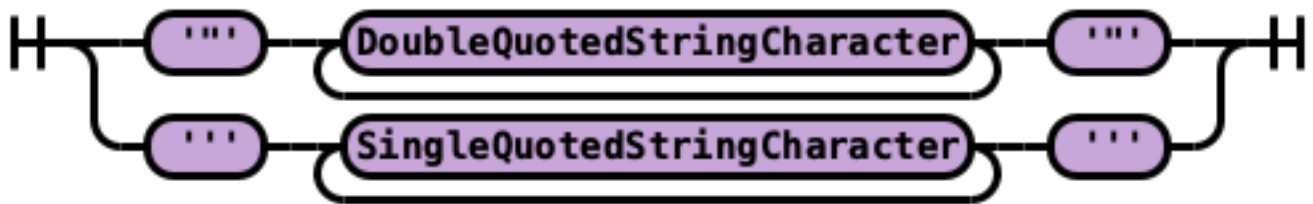GreaterThanOrEqual :

Not :

BitNot :

Inc :

Dec :
StringLiteral :

NonEmptyStringLiteral :

DoubleQuotedStringCharacter :

SingleQuotedStringCharacter :

SingleQuotedPrintable :
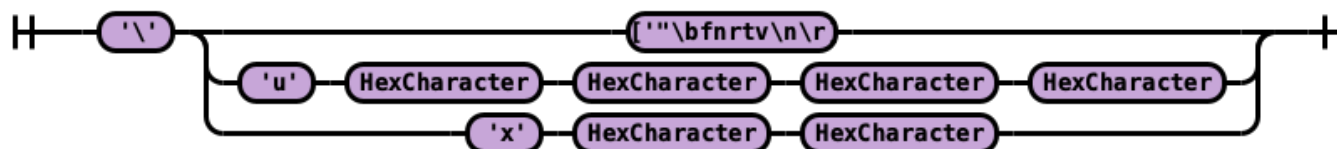
DoubleQuotedPrintable :
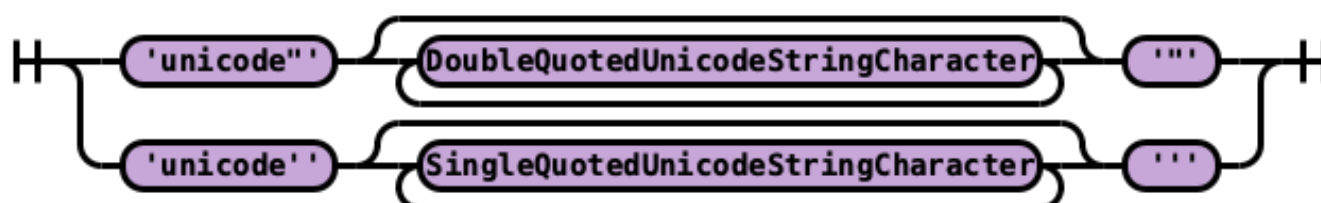
[\u0020-\u0021\u0023-\u005B\u005D-\u007E]

EscapeSequence :

'\'
['"\bfnrtv\n\r]
'u' HexCharacter HexCharacter HexCharacter HexCharacter
'x' HexCharacter HexCharacter

UnicodeStringLiteral :

'unicode"' DoubleQuotedUnicodeStringCharacter '""'
'unicode'' SingleQuotedUnicodeStringCharacter '''

DoubleQuotedUnicodeStringCharacter :

not ["\r\n\]
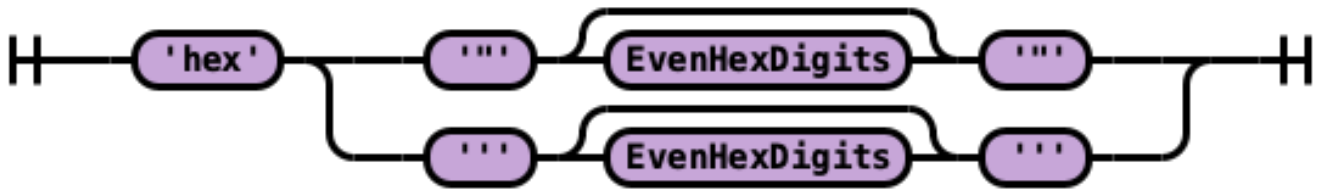EscapeSequence

SingleQuotedUnicodeStringCharacter :

not ['\r\n\]
EscapeSequence
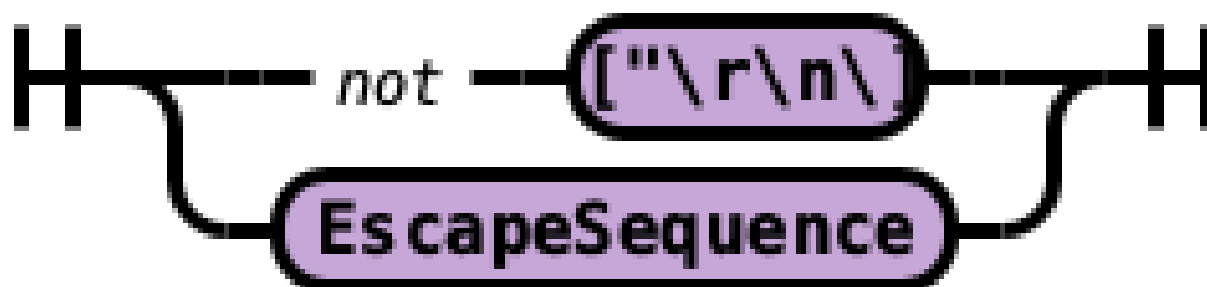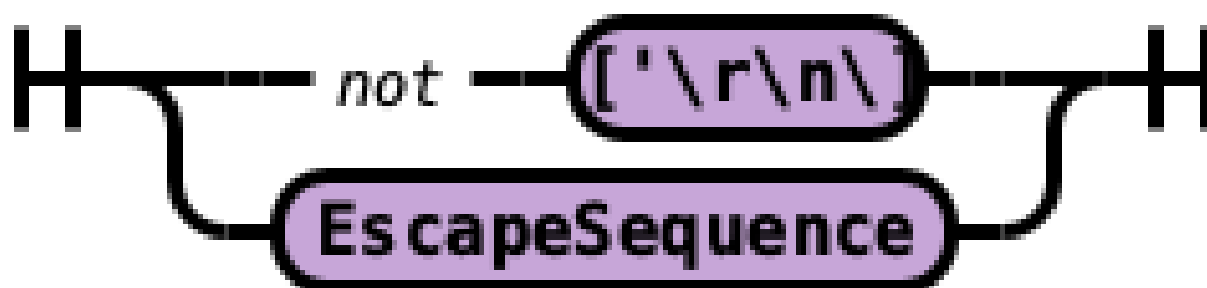
HexString :
HexNumber :
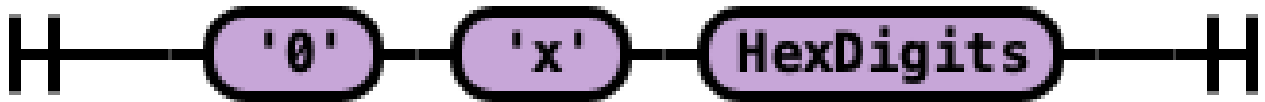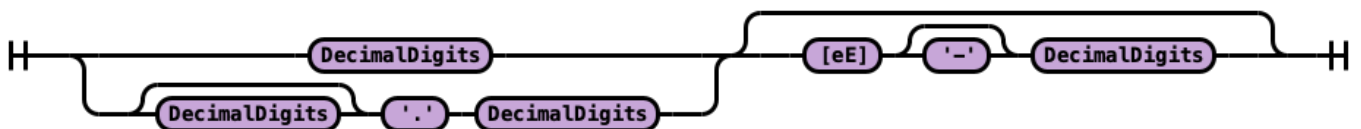
HexDigits :

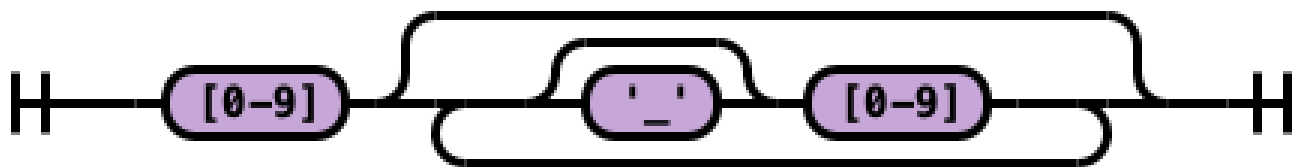EvenHexDigits :

HexCharacter :

DecimalNumber :

DecimalDigits :
Identifier :

IdentifierStart :

IdentifierPart :

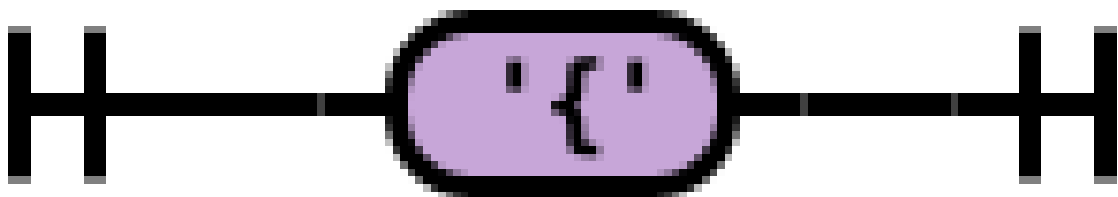WS :

COMMENT :
LINE_COMMENT :

AssemblyDialect :

AssemblyLBrace :

AssemblyBlockWS :

AssemblyBlockCOMMENT :

AssemblyBlockLINE_COMMENT :

YulBreak :

YulCase :

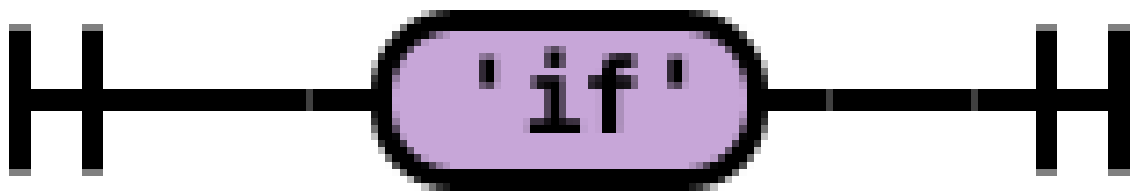YulContinue :

YulDefault :

YulFalse :

YulFor :

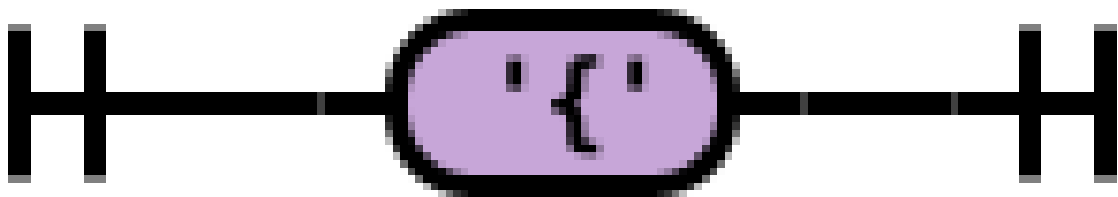YulFunction :

YulIf :
YulLeave :
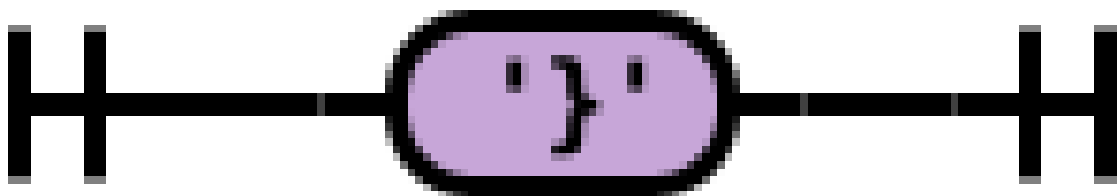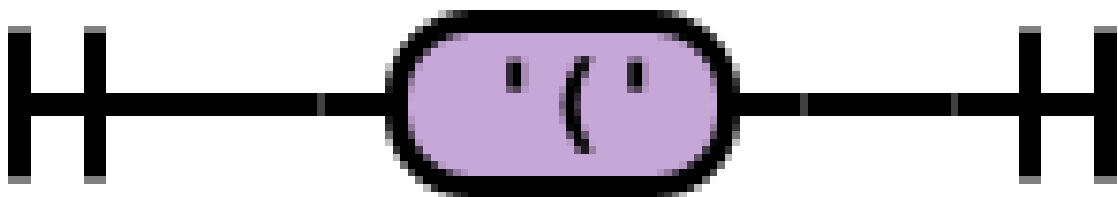
YulLet :
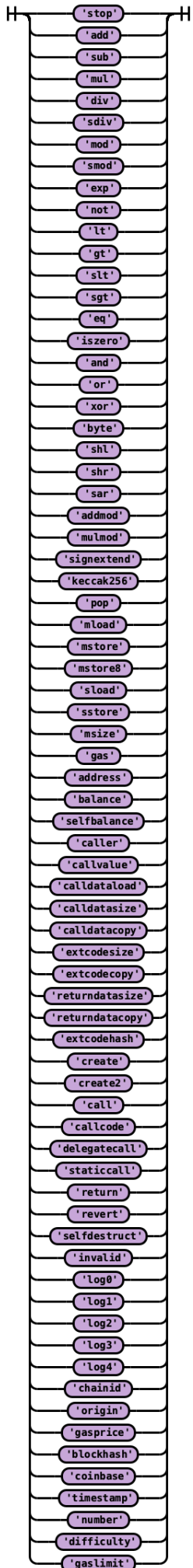

YulSwitch :

YulTrue :
YulEVMBuiltin :

YulLBrace :

YulRBrace :

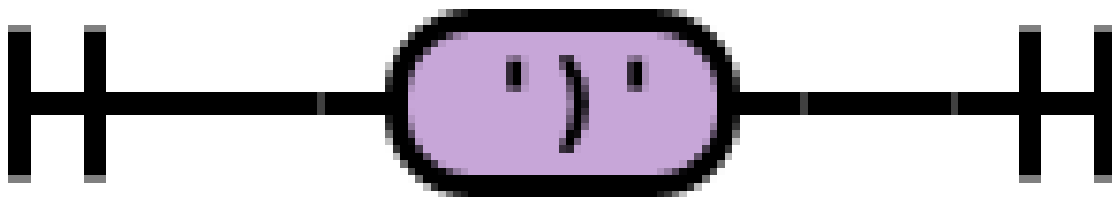YulLParen :

```
'stop'
'add'
'sub'
'mul'
'div'
'sdiv'
'mod'
'smod'
'exp'
'not'
'lt'
'gt'
'slt'
'sgt'
'eq'
'iszero'
'and'
'or'
'xor'
'byte'
'shl'
'shr'
'sar'
'addmod'
'mulmod'
'signextend'
'keccak256'
'pop'
'mload'
'mstore'
'mstore8'
'sload'
'sstore'
'msize'
'gas'
'address'
'balance'
'selfbalance'
'caller'
'callvalue'
'calldataload'
'calldatasize'
'calldatacopy'
'extcodesize'
'extcodecopy'
'returndatasize'
'returndatacopy'
'extcodehash'
'create'
'create2'
'call'
'callcode'
'delegatecall'
'staticcall'
'return'
'revert'
'selfdestruct'
'invalid'
'log0'
'log1'
'log2'
'log3'
'log4'
'chainid'
'origin'
'gasprice'
'blockhash'
'coinbase'
'timestamp'
'number'
'difficulty'
'gaslimit'
```
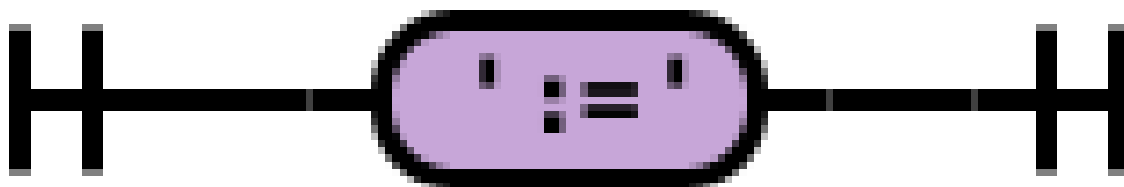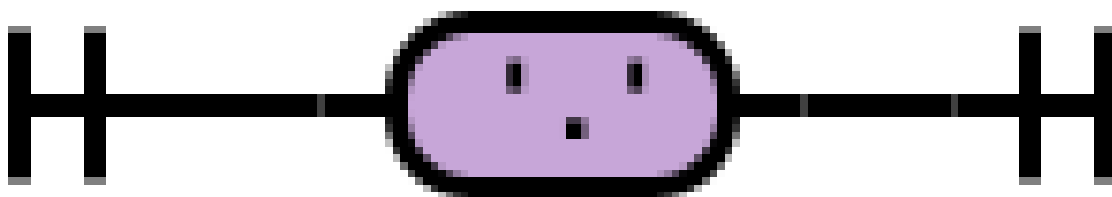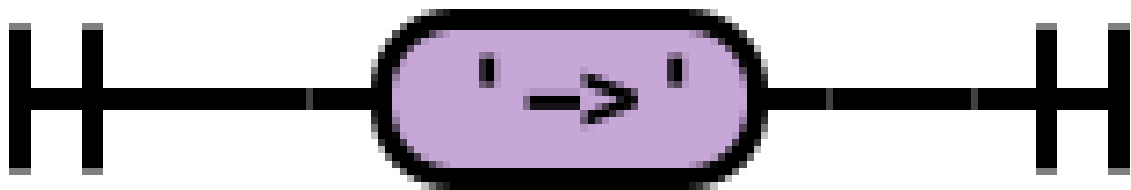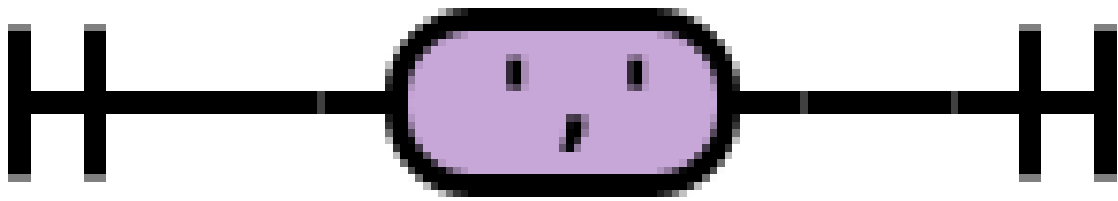
YulRParen :

YulAssign :

YulPeriod :

YulComma :
YulArrow :

YulIdentifier :
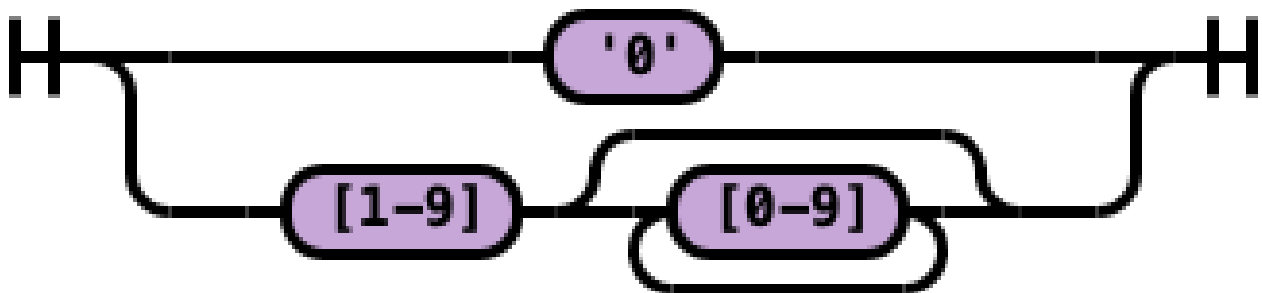

YulIdentifierStart :

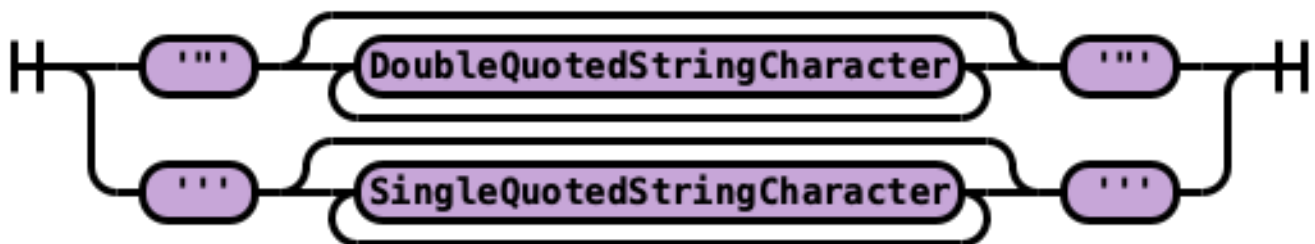YulIdentifierPart :
YulHexNumber :

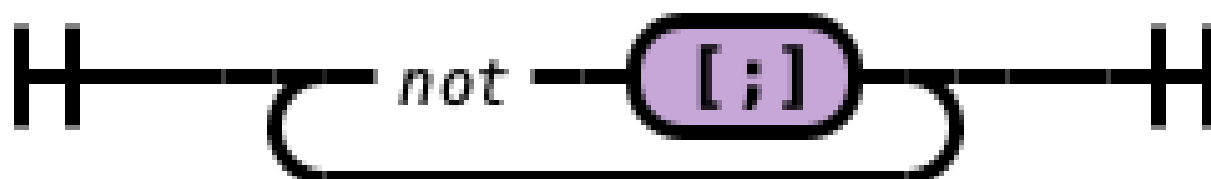YulDecimalNumber :

YulStringLiteral :

YulWS :
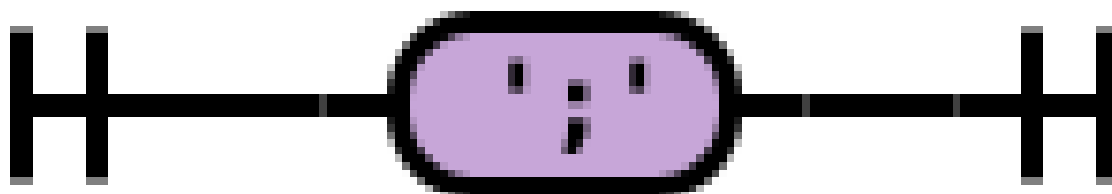

YulCOMMENT :

YulLINE_COMMENT :

PragmaToken :

PragmaSemicolon :

PragmaWS :

PragmaCOMMENT :
PragmaLINE_COMMENT :

```
|—— '/*' —— any char —— '*/' ——|
```

```
|—— '//' ——┬—— not —— [\r\n] ——┬——|
           └─────────────────────┘
```