2. CFG(BNF) for PEMDAS/BODMAS

<Expr> 🡪 <AS> | <MD> | <E> (Leftmost Associative)

<AS> 🡪 <AS> + <MD> | <AS> - <MD> | <MD> (Leftmost Associative)

<MD> 🡪<MD> \* <E> | <MD> /<E> | <MD> %<E> | <E> (Leftmost Associative)

<E> 🡪 <P> | <P> ^ <E> (Rightmost Associative)

<P> 🡪 (<Expr>) | INT\_LIT |ID | FLOAT\_LIT | FUNCTIONCALL. (Leftmost Associative)

3.

<AS> | <MD> | <E> 🡨<Expr>

<AS> + <MD> | <AS> - <MD> | <MD> 🡨 <AS>

<MD> \* <E> | <MD> /<E> | <MD> %<E> | <E> 🡨<MD>

<P> | <P> ^ <E> 🡨<E>

(<Expr>) | INT\_LIT |ID | FLOAT\_LIT | FUNCTIONCALL 🡨 <P>

4. a \* 12 / (x - 7.0)

b) Leftmost Derivation

<expr> => <MD>

* <MD> / <E>
* <MD> \* <E> / <E>
* <E> \* <E> / <E>
* <P> \* <E> / <E>
* ID \* <E> / <E>
* ID \* <P> / <E>
* ID \* INT\_LIT / <E>
* ID \* INT\_LIT / <P>
* ID \* INT\_LIT / (<Expr>)
* ID \* INT\_LIT / (<AS>)
* ID \* INT\_LIT / (<MD>)
* ID \*INT\_LIT / (<AS>)
* ID \*INT\_LIT / (<AS> - <MD>)
* ID \*INT\_LIT / (<MD> - <MD>)
* ID \*INT\_LIT / (<E> - <MD>)
* ID \*INT\_LIT / (<P> - <MD>)
* ID \*INT\_LIT / (ID- <MD>)
* ID \*INT\_LIT / (ID- <E>)
* ID \*INT\_LIT / (ID- <P>)
* ID \*INT\_LIT / (ID- FLOAT\_LIT)

b) Rightmost Derivation

<expr> => <MD>

* <MD> /<E>
* <MD> /<P>
* <MD>/ (<Expr>)
* <MD>/ (<AS>)
* <MD>/ (<AS> - <MD>)
* <MD>/ (<AS> - <E>)
* <MD>/ (<AS> - <P>)
* <MD>/ (<MD> - FLOAT\_LIT)
* <MD>/ (<E> - FLOAT\_LIT)
* <MD>/ (<P> - FLOAT\_LIT)
* <MD>/ (ID - FLOAT\_LIT)
* <MD>\* <E> / (ID - FLOAT\_LIT)
* <MD>\*<E> / (ID - FLOAT\_LIT)
* <MD>\*<P> / (ID - FLOAT\_LIT)
* <MD>\*INT\_LIT / (ID - FLOAT\_LIT)
* <E>\*INT\_LIT / (ID - FLOAT\_LIT)
* <P>\* INT\_LIT /(ID - FLOAT\_LIT)
* ID \*INT\_LIT /(ID - FLOAT\_LIT)

5.

<ForStmt> 🡪`for` [ <[Condition](https://go.dev/ref/spec#Condition)> | <[ForClause](https://go.dev/ref/spec#ForClause)> | <[RangeClause](https://go.dev/ref/spec#RangeClause)> ] <[Block](https://go.dev/ref/spec#Block)>

1.<ForStmt> 🡪`for` <[Block](https://go.dev/ref/spec#Block)>

🡪 `for` {`statmentList `}`

* `for` {`statement `;` `}`
* For {}

2. ForStmt 🡪`for` [Condition](https://go.dev/ref/spec#Condition) [Block](https://go.dev/ref/spec#Block)

🡪 `for` [Condition](https://go.dev/ref/spec#Condition) {}

🡪`for` Expression {}

🡪`for` [UnaryExpr](https://go.dev/ref/spec#UnaryExpr){}

🡪`for` [PrimaryExpr](https://go.dev/ref/spec#PrimaryExpr){``}

🡪`for` [Operand](https://go.dev/ref/spec#Operand) { }

🡪`for` [Literal](https://go.dev/ref/spec#Literal){}

🡪`for` [BasicLit](https://go.dev/ref/spec#BasicLit){ }

3. <ForStmt> 🡪`for` <[ForClause](https://go.dev/ref/spec#ForClause)> <[Block](https://go.dev/ref/spec#Block)>

🡪 `for` <[ForClause](https://go.dev/ref/spec#ForClause)> {}

🡪 `for` [ [InitStmt](https://go.dev/ref/spec#InitStmt) ] ";" [ [Condition](https://go.dev/ref/spec#Condition) ] ";" [ [PostStmt](https://go.dev/ref/spec#PostStmt) ] {}

🡪 `for` `;``;` {}

4.

🡪 `for` <[ForClause](https://go.dev/ref/spec#ForClause)> {}

🡪 `for` [InitStmt](https://go.dev/ref/spec#InitStmt) `;``;` {}

🡪 `for` [SimpleStmt](https://go.dev/ref/spec#SimpleStmt) `;``;` {}

🡪 `for` [EmptyStmt](https://go.dev/ref/spec#EmptyStmt)`;``;` {}

5.

🡪 `for` <[ForClause](https://go.dev/ref/spec#ForClause)> {}

🡪 `for` `;` [ [Condition](https://go.dev/ref/spec#Condition) ] `;` {}

🡪 `for` `;` [Condition](https://go.dev/ref/spec#Condition) `;` {}

🡪 `for` `;` [Expression](https://go.dev/ref/spec#Expression)`;` {}

🡪 `for` `;` [UnaryExpr](https://go.dev/ref/spec#UnaryExpr) `;` {}

6.

🡪 `for` <[ForClause](https://go.dev/ref/spec#ForClause)> {}

🡪 `for` `;` `;` [ [PostStmt](https://go.dev/ref/spec#PostStmt) ] {}

🡪 `for` `;``;`[PostStmt](https://go.dev/ref/spec#PostStmt) {}`;`

🡪 `for` `;``;`[SimpleStmt](https://go.dev/ref/spec#SimpleStmt) {}`;`

7.

<ForStmt> 🡪`for` <[RangeClause](https://go.dev/ref/spec#RangeClause)> <[Block](https://go.dev/ref/spec#Block)>

🡪 `for` <[RangeClause](https://go.dev/ref/spec#RangeClause)> {}

🡪 `for` [ [ExpressionList](https://go.dev/ref/spec#ExpressionList) `=` | [IdentifierList](https://go.dev/ref/spec#IdentifierList) `:=` ] "range" [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for` [ [ExpressionList](https://go.dev/ref/spec#ExpressionList) `=` | [IdentifierList](https://go.dev/ref/spec#IdentifierList) `:=` ] "range" [UnaryExpr](https://go.dev/ref/spec#UnaryExpr) {}

🡪 `for` `range` [UnaryExpr](https://go.dev/ref/spec#UnaryExpr) {}

8.

🡪 `for` <[RangeClause](https://go.dev/ref/spec#RangeClause)> {}

🡪 `for` [ [ExpressionList](https://go.dev/ref/spec#ExpressionList) `=` ] `range` [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for` [ExpressionList](https://go.dev/ref/spec#ExpressionList) `=` `range` [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for` [Expression](https://go.dev/ref/spec#Expression) { "," [Expression](https://go.dev/ref/spec#Expression) } `=` `range` [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for` [Expression](https://go.dev/ref/spec#Expression) `=` `range` [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for` [UnaryExpr](https://go.dev/ref/spec#UnaryExpr)`=` `range` [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for` [UnaryExpr](https://go.dev/ref/spec#UnaryExpr)`=` `range` [UnaryExpr](https://go.dev/ref/spec#UnaryExpr){}

9.

🡪 `for` <[RangeClause](https://go.dev/ref/spec#RangeClause)> {}

🡪 `for` [ [ExpressionList](https://go.dev/ref/spec#ExpressionList) `:=` | [IdentifierList](https://go.dev/ref/spec#IdentifierList) `:=` ] `range` [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for` [[IdentifierList](https://go.dev/ref/spec#IdentifierList) `:=` ] `range` [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for` [IdentifierList](https://go.dev/ref/spec#IdentifierList) `:=` `range` [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for` [identifier](https://go.dev/ref/spec#identifier) { `,` [identifier](https://go.dev/ref/spec#identifier) } `:=` `range` [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for` [identifier](https://go.dev/ref/spec#identifier) `:=` `range` [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for`[letter](https://go.dev/ref/spec#letter) { [letter](https://go.dev/ref/spec#letter) | [unicode\_digit](https://go.dev/ref/spec#unicode_digit) } `:=` `range` [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for`[letter](https://go.dev/ref/spec#letter) `:=` `range` [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for` [unicode\_letter](https://go.dev/ref/spec#unicode_letter) | `\_` `:=` `range` [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for` `\_` `:=` `range` [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for` `\_` `:=` `range` [UnaryExpr](https://go.dev/ref/spec#UnaryExpr) {}

10.

🡪 `for` [ [ExpressionList](https://go.dev/ref/spec#ExpressionList) `:=` | [IdentifierList](https://go.dev/ref/spec#IdentifierList) `:=` ] `range` [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for` [[IdentifierList](https://go.dev/ref/spec#IdentifierList) `:=` ] `range` [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for` [IdentifierList](https://go.dev/ref/spec#IdentifierList) `:=` `range` [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for` [identifier](https://go.dev/ref/spec#identifier) { `,` [identifier](https://go.dev/ref/spec#identifier) } `:=` `range` [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for` [identifier](https://go.dev/ref/spec#identifier) `,` [identifier](https://go.dev/ref/spec#identifier) `:=` `range` [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for` [letter](https://go.dev/ref/spec#letter) { [letter](https://go.dev/ref/spec#letter) | [unicode\_digit](https://go.dev/ref/spec#unicode_digit) } `,` [identifier](https://go.dev/ref/spec#identifier) `:=` `range` [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for` [letter](https://go.dev/ref/spec#letter) `,` [identifier](https://go.dev/ref/spec#identifier) `:=` `range` [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for` [letter](https://go.dev/ref/spec#letter) `,` [letter](https://go.dev/ref/spec#letter) { [letter](https://go.dev/ref/spec#letter) | [unicode\_digit](https://go.dev/ref/spec#unicode_digit) } `:=` `range` [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for` [letter](https://go.dev/ref/spec#letter) `,` [letter](https://go.dev/ref/spec#letter) `:=` `range` [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for` [unicode\_letter](https://go.dev/ref/spec#unicode_letter) | "\_" `,` [letter](https://go.dev/ref/spec#letter) `:=` `range` [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for` "\_" `,` [letter](https://go.dev/ref/spec#letter) `:=` `range` [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for` "\_" `,` [unicode\_letter](https://go.dev/ref/spec#unicode_letter) | "\_" `:=` `range` [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for` "\_" `,` "\_" `:=` `range` [Expression](https://go.dev/ref/spec#Expression) {}

🡪 `for` "\_" `,` "\_" `:=` `range` [UnaryExpr](https://go.dev/ref/spec#UnaryExpr) {}