compiler_def.py (left editor)

```python
if "." in __name__:
    from .PythonParser import PythonParser
    from .PythonParserVisitor import PythonParserVisitor
else:
    from PythonParser import PythonParser
    from PythonParserVisitor import PythonParserVisitor

class Compiler(PythonParserVisitor):

    def __init__(self):
        super(PythonParserVisitor, self).__init__()
        self.vars = {}
        return None

    # Paste here all methods in PythonParserVisitor.py file

    # Visit a parse tree produced by PythonParser#code.          Alt+K to
    def visitCode(self, ctx:PythonParser.CodeContext):
        return self.visitChildren(ctx)


    # Visit a parse tree produced by PythonParser#expr.
    def visitExpr(self, ctx:PythonParser.ExprContext):
        return self.visitChildren(ctx)

del (PythonParser, PythonParserVisitor)
```

PythonParserVisitor.py (right editor)

```python
# Generated from PythonParser.g4 by ANTLR 4.13.1
from antlr4 import *
if "." in __name__:
    from .PythonParser import PythonParser
else:
    from PythonParser import PythonParser

# This class defines a complete generic visitor for a parse tree p

class PythonParserVisitor(ParseTreeVisitor):

    # Visit a parse tree produced by PythonParser#code.
    def visitCode(self, ctx:PythonParser.CodeContext):
        return self.visitChildren(ctx)


    # Visit a parse tree produced by PythonParser#expr.
    def visitExpr(self, ctx:PythonParser.ExprContext):
        return self.visitChildren(ctx)    Alt+K to Edit, Alt+L to Ch


    # Visit a parse tree produced by PythonParser#comment.
    def visitComment(self, ctx:PythonParser.CommentContext):
        return self.visitChildren(ctx)


    # Visit a parse tree produced by PythonParser#assignment.
    def visitAssignment(self, ctx:PythonParser.AssignmentContext):
        return self.visitChildren(ctx)
```