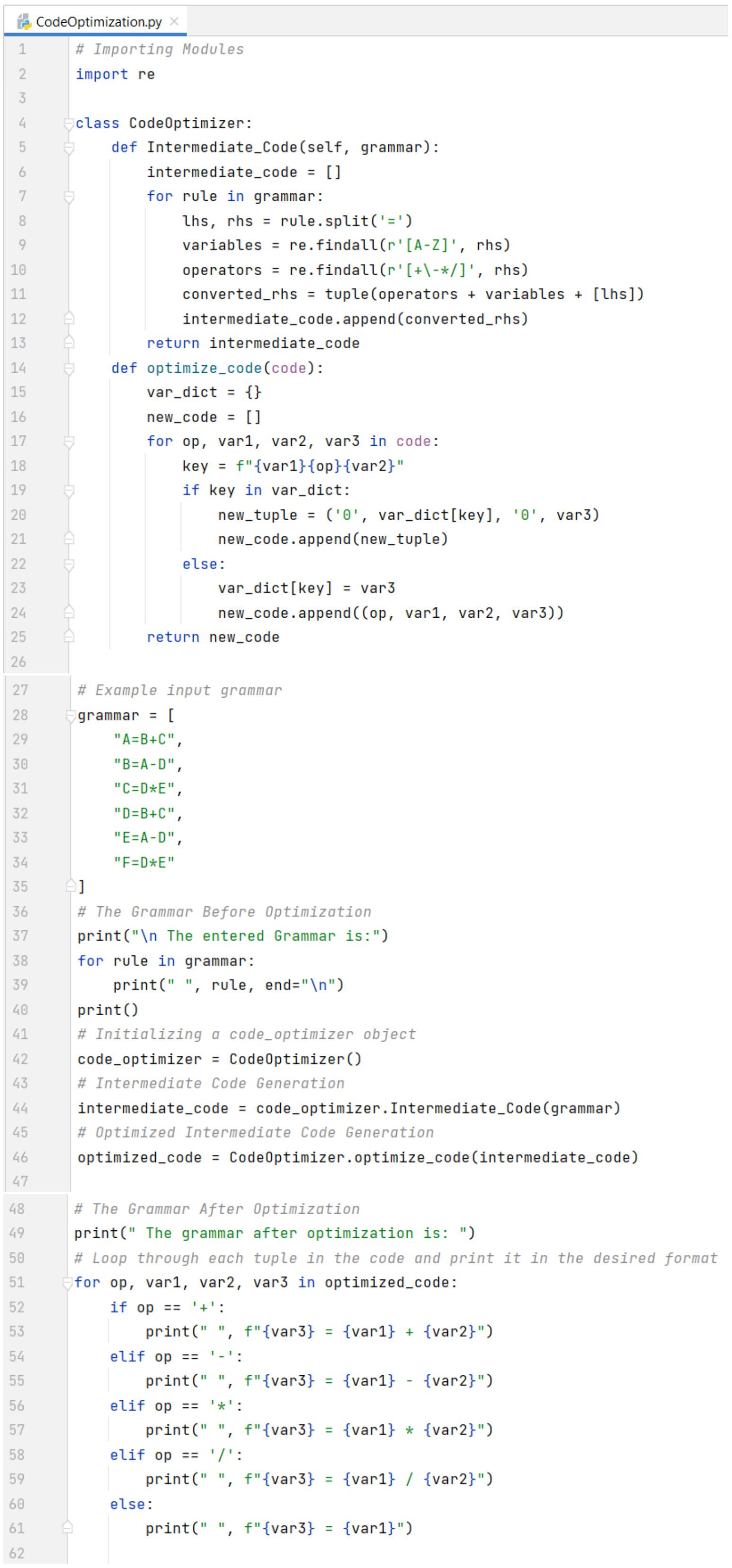
Input:



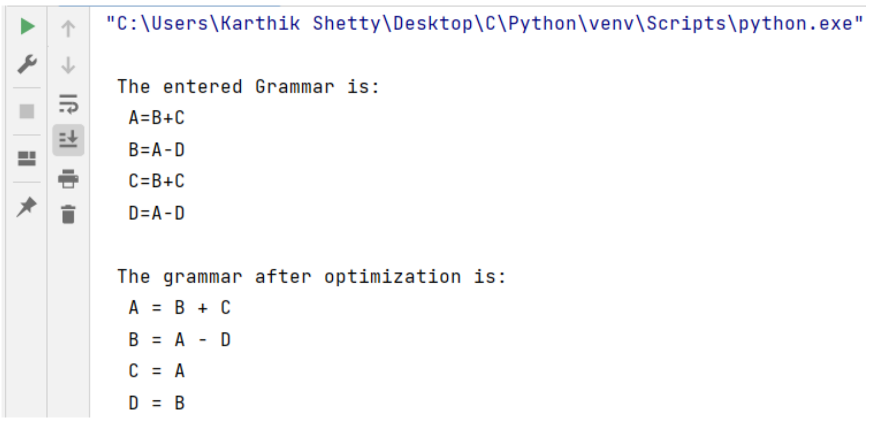
```python
# Importing Modules
import re

class CodeOptimizer:
    def Intermediate_Code(self, grammar):
        intermediate_code = []
        for rule in grammar:
            lhs, rhs = rule.split('=')
            variables = re.findall(r'[A-Z]', rhs)
            operators = re.findall(r'[+\-*/]', rhs)
            converted_rhs = tuple(operators + variables + [lhs])
            intermediate_code.append(converted_rhs)
        return intermediate_code
    def optimize_code(code):
        var_dict = {}
        new_code = []
        for op, var1, var2, var3 in code:
            key = f"{var1}{op}{var2}"
            if key in var_dict:
                new_tuple = ('0', var_dict[key], '0', var3)
                new_code.append(new_tuple)
            else:
                var_dict[key] = var3
                new_code.append((op, var1, var2, var3))
        return new_code

# Example input grammar
grammar = [
    "A=B+C",
    "B=A-D",
    "C=D*E",
    "D=B+C",
    "E=A-D",
    "F=D*E"
]
# The Grammar Before Optimization
print("\n The entered Grammar is:")
for rule in grammar:
    print(" ", rule, end="\n")
print()
# Initializing a code_optimizer object
code_optimizer = CodeOptimizer()
# Intermediate Code Generation
intermediate_code = code_optimizer.Intermediate_Code(grammar)
# Optimized Intermediate Code Generation
optimized_code = CodeOptimizer.optimize_code(intermediate_code)

# The Grammar After Optimization
print(" The grammar after optimization is: ")
# Loop through each tuple in the code and print it in the desired format
for op, var1, var2, var3 in optimized_code:
    if op == '+':
        print(" ", f"{var3} = {var1} + {var2}")
    elif op == '-':
        print(" ", f"{var3} = {var1} - {var2}")
    elif op == '*':
        print(" ", f"{var3} = {var1} * {var2}")
    elif op == '/':
        print(" ", f"{var3} = {var1} / {var2}")
    else:
        print(" ", f"{var3} = {var1}")
```

Output:



```
"C:\Users\Karthik Shetty\Desktop\C\Python\venv\Scripts\python.exe"

 The entered Grammar is:
  A=B+C
  B=A-D
  C=B+C
  D=A-D

 The grammar after optimization is:
  A = B + C
  B = A - D
  C = A
  D = B
```