Code of dead code detection and elimination in Python:

```
import ast

class DeadCodeAnalyzer(ast.NodeVisitor):
    def __init__(self):
        self.dead_code = set()
        self.visited_nodes = set()
        self.function_calls = set()
        self.line_numbers = set()

    def visit_FunctionDef(self, node):
        self.function_calls.add(node.name)  # Collect function names
        self.visited_nodes.add(node)
        self.generic_visit(node)

    def visit_Call(self, node):
        if isinstance(node.func, ast.Name):
            self.function_calls.add(node.func.id)
        self.visited_nodes.add(node)
        self.generic_visit(node)

    def visit_Assign(self, node):
        self.visited_nodes.add(node)
        self.line_numbers.add(node.lineno)
        self.generic_visit(node)

    def visit_Name(self, node):
        if isinstance(node.ctx, ast.Store) and node not in self.visited_nodes:
            self.dead_code.add(node.id)
        self.visited_nodes.add(node)
        self.generic_visit(node)

    def visit_Module(self, node):
        for stmt in node.body:
            if isinstance(stmt, ast.FunctionDef):
                self.visit_FunctionDef(stmt)
            elif isinstance(stmt, ast.Expr):
                self.visit_Expr(stmt)
            elif isinstance(stmt, ast.Assign):
                self.visit_Assign(stmt)
            elif isinstance(stmt, ast.Call):
                self.visit_Call(stmt)

    def visit_Expr(self, node):
        self.visited_nodes.add(node)
        self.generic_visit(node)

# The find_dead_code function remains the same
def find_dead_code(source_code):
    tree = ast.parse(source_code)
    analyzer = DeadCodeAnalyzer()
    analyzer.visit(tree)

    # Identify dead functions by subtracting used functions from all defined
functions
    dead_functions = analyzer.function_calls - analyzer.dead_code

    # Identify dead code lines
    dead_lines = analyzer.line_numbers - analyzer.dead_code

    return dead_functions, dead_lines

# Example usage
example_code = """
def example_code():
    a = 5
    b = 10

    def unused_function():
        c = a + b
        print(c)

    def used_function():
        d = 20
        print(d)

    used_function()

def example_code_1():
    x = 5
    y = 10

    def unused_function():
        z = x + y
        print(z)

    used_function()

def used_function():
    print("This function is used.")

def example_code_2():
    a = 5
    b = 10
    c = a + b
    d = a * b
    e = c - d

    print("Result:", e)

def example_code_3():
    x = 5
    y = 10

    def dead_function():
        z = x * y
        print(z)

    print("This is the main function.")

if __name__ == "__main__":
    example_code()
    example_code_1()
    example_code_2()
    example_code_3()
"""

dead_functions, dead_lines = find_dead_code(example_code)
print("\nDead functions:", dead_functions)
print("Dead lines:", dead_lines)
```

Results of this code:

```
Dead functions: {'unused_function', 'example_code', 'example_code_3',
'example_code_2', 'used_function', 'print', 'example_code_1', 'dead_function'}
Dead lines: {32, 33, 34, 3, 4, 7, 39, 40, 11, 43, 17, 18, 21, 30, 31}
```