Team 1: Basic Circuit Solver in Python

Team list:

S.no	NAME
1	
2	
3	K SAMBA SIVA REDDY
4	N SUMANTH
5	
6	

1. Title Page

- Project Name: *Basic Circuit Solver in Python*
- Team Name: *Team1*
- GitHub Repository: GitHub Repository.
- Submission Deadline: 21/10/2024
- Team Members: List all 6 members and their respective roles (mention your role as KVL).

2. Abstract

A brief summary of the project goals, which include creating a basic DC circuit solver using Python. Highlight the importance of solving circuits through Ohm's Law and Kirchhoff's Laws and the collaborative aspect of using GitHub for teamwork.

1. Introduction

This report documents the contribution to the GitHub project **Basic Circuit Solver in Python**, where a team of six developed a Python-based solver for DC circuits using Ohm's Law and Kirchhoff's Laws (KVL, KCL). The project was collaboratively built and version-controlled using GitHub, ensuring smooth integration and teamwork.

As **Person 3**, I was responsible for implementing the solution for **Kirchhoff's Voltage Law (KVL)**. This report covers the steps I took, including cloning the repository, creating a branch, writing the KVL code, and contributing to the final project.

2. Project Setup and GitHub Workflow

1. **Cloning the Repository**: Everyone cloned the main repository created by **Person 6** using the following command:

git clone https://github.com/vaibruce/Basic_Circuit_Solver_in_Python.git

Cd Basic_Circuit_Solver_in_Python.git

2. **Creating a Branch**: created a separate branch for my work to avoid conflicts with other members:

git checkout -b <branch name>

Cd <branch name>

3. **Adding Python File**: After creating the Python file, Every one added it to the their branch:

git add <python file>

git add .

4. **Commit the Changes:** Once the code is ready:

git commit -m "Added KCL implementation"

5. **Push Changes to the Remote Repository:**

git push origin <branch name>

6. **Create a Pull Request:** Once the changes are pushed, they should visit the link provided by GitHub to create a pull request for review and merging.

3. person 1

4. Person 2

5. K Samba Siva Reddy (Kirchhoff's Voltage Law - KVL)

5.1. Understanding KVL

Kirchhoff's Voltage Law (KVL) states that the sum of the electrical potential differences (voltages) around any closed loop or mesh is zero. This is mathematically represented as:

∑i=1nVi=0\sum_{i=1}^{n} V_i = 0i=1∑nVi=0

Where ViV_iVi represents the voltage across each element in the circuit.

5.2. KVL Code Implementation

I implemented a Python function to compute the sum of voltages around a loop, ensuring it adheres to KVL.

```
def solve_kvl(loop_voltages):
```

```
    """Validates Kirchhoff's Voltage Law (KVL): Sum of voltages in a loop
should be zero."""
    total_voltage = sum(loop_voltages)
    if total_voltage == 0:
        return True  # KVL is satisfied
    else:
        return False  # KVL is violated

# Example usage of KVL
loop_voltages = [10, -5, -5]  # Voltages in a closed loop
print(f"KVL Valid: {solve_kvl(loop_voltages)}")  # Output: True
```

Python

5.3. Code Explanation

- **kvl_solver**: This function computes the sum of voltages for a given loop and checks if the total is zero, confirming KVL is satisfied.
- **Example**: If the loop contains voltages of 10V, -5V, and -5V, the sum is 0, meaning the law holds.

This implementation was tested with various sets of data to ensure correctness and accuracy.

7. Sumanth (Unit Tests for Ohm's Law and KCL)

6.1. Understanding Unit Testing

Unit testing is essential for verifying that each individual function behaves as expected. It allows us to test the smallest parts of our application independently, ensuring correctness and reliability. Sumanth was responsible for writing unit tests for **Ohm's Law** and **Kirchhoff's Current Law (KCL)** to ensure they performed correctly under various conditions.

6.2. Unit Test Implementation for Ohm's Law & KCL

For Ohm's Law, the unit test checks if the implemented code correctly computes voltage, current, or resistance when given the other two variables. KCL, the unit test validates that the sum of currents entering and exiting a node equals zero, which confirms KCL is satisfied.

Python

```
import unittest

class TestCircuitSolver(unittest.TestCase):

    # Testing Ohm's Law functions
```

```
    def test_calculate_current(self):
        self.assertEqual(calculate_current(10, 5), 2)
        self.assertRaises(ValueError, calculate_current, 10, 0)

    def test_calculate_voltage(self):
        self.assertEqual(calculate_voltage(2, 5), 10)

    # Testing KCL
    def test_solve_kcl(self):
        self.assertTrue(solve_kcl([3, -2, -1]))  # Should satisfy KCL
        self.assertFalse(solve_kcl([3, -2, -2]))  # Should violate KCL

# Running the tests
unittest.main(argv=[''], verbosity=2, exit=False)
```

6.4. Code Explanation

- **test_calculate_voltage / current / resistance**: These functions check if Ohm's Law correctly computes voltage, current, or resistance when given appropriate values.
- **test_kcl_satisfied / test_kcl_violated**: These functions validate that KCL holds by checking if the sum of currents at a node is zero.

By writing these tests, Sumanth ensured that the logic implemented for Ohm's Law and KCL would work under various conditions, improving the overall reliability of the project.

7. Person 5

8. Person 6

9. Project Structure and Overall Collaboration

The repository was structured so that each team member contributed their respective functions. The key divisions were:

- **Person 1 (Ohm’s Law)**: Developed functions for calculating voltage, current, and resistance based on Ohm's Law.
- **Person 2 (KCL)**: Created functions to solve current flow using Kirchhoff’s Current Law.

- **Person 3 (KVL)**: Developed functions to calculate the sum of voltages using Kirchhoff's Voltage Law (KVL).
- **Person 4:(Unit tests for Ohm's Law and KCL)** Created unit tests for Ohm's Law and KCL implementations to ensure their correctness.
- **Person 5**:**(Unit tests for KVL and code refactoring)** Developed unit tests for KVL and also worked on code refactoring to ensure the code was clean, efficient, and free of errors.
- **Person 6 (Repository Manager)**: Managed repository creation, merging all branches, and resolving conflicts.

10. GitHub Collaboration Process

The following steps ensured smooth collaboration and final integration:

- **Branching**: Each team member created their branch to avoid conflicts while coding.
- **Pull Requests**: Once code was complete and tested, each team member submitted a pull request.
- **Merging**: The sixth team member merged all branches, ensuring that the code worked together without conflicts.
- **Final Submission**: After merging all the code, the final repository was submitted.

11 Conclusion

This project successfully implemented key principles of electrical circuit analysis, including **Ohm's Law**, **Kirchhoff's Current Law (KCL)**, and **Kirchhoff's Voltage Law (KVL)**, to solve DC circuits using Python. Each team member contributed to the development of a robust circuit solver that handled various aspects of circuit calculations, testing, and user interaction.

We leveraged GitHub as our version control system, allowing seamless collaboration across the team. Each team member worked in individual branches, and the final code was efficiently merged, ensuring a smooth workflow. The use of unit tests and a command-line interface further enhanced the functionality and reliability of the project.

Through strong teamwork, effective version control, and collaborative problem-solving, we were able to develop a comprehensive solution for analysing DC circuits. This project demonstrates the power of Python for solving real-world electrical engineering problems while showcasing the importance of collaborative development using modern tools like GitHub.