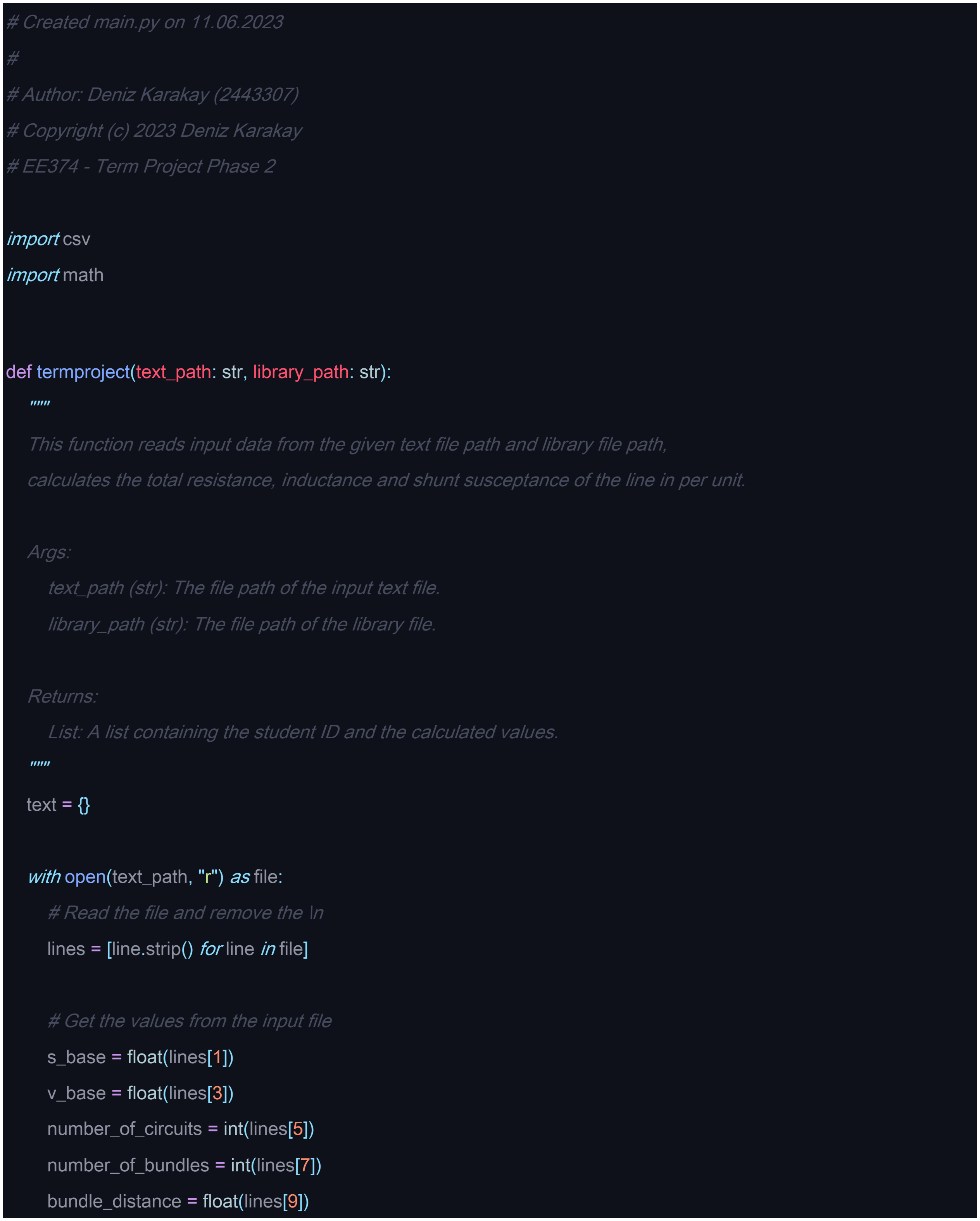
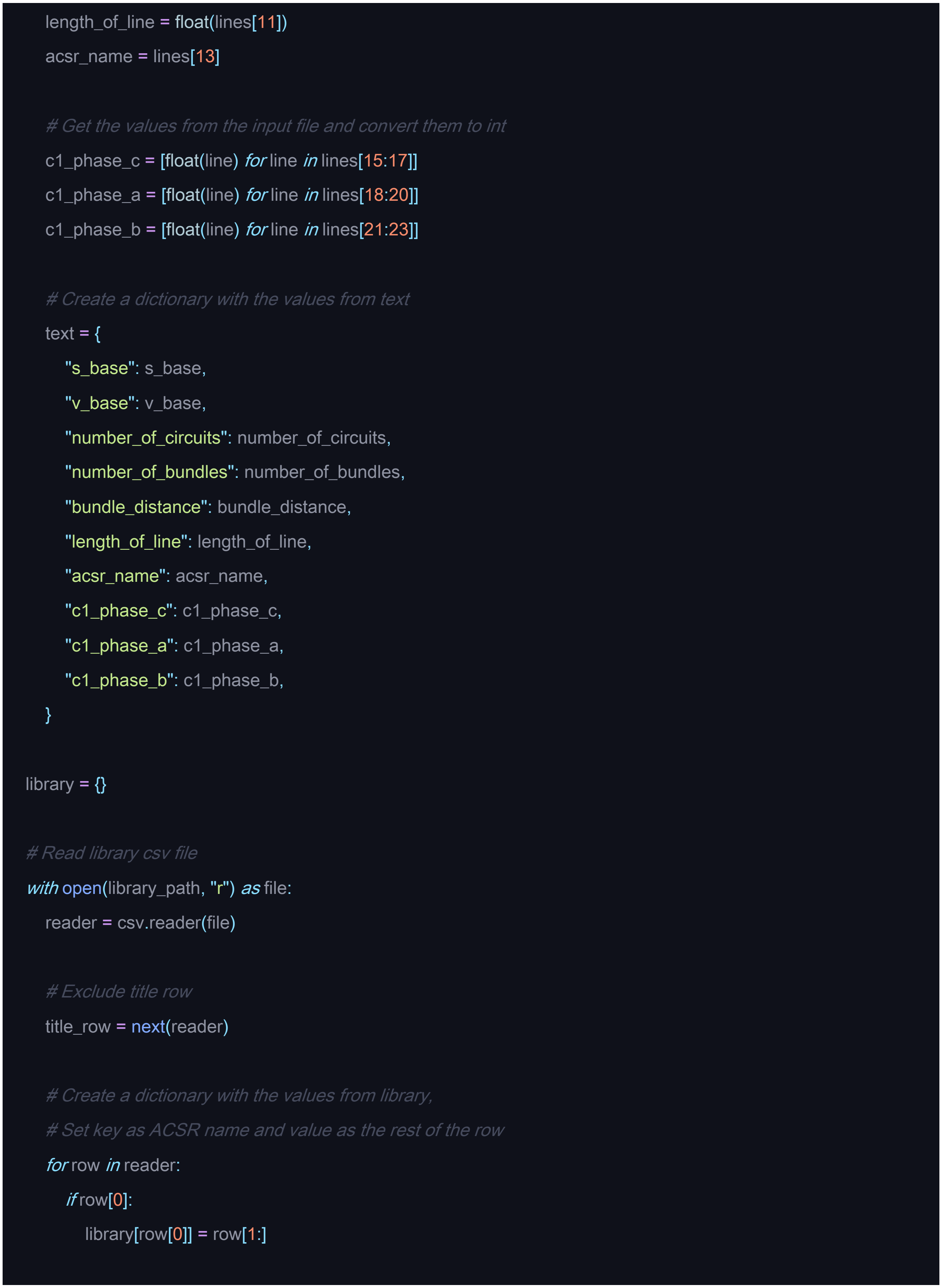
Deniz Karakay 2443307 16.05.2023

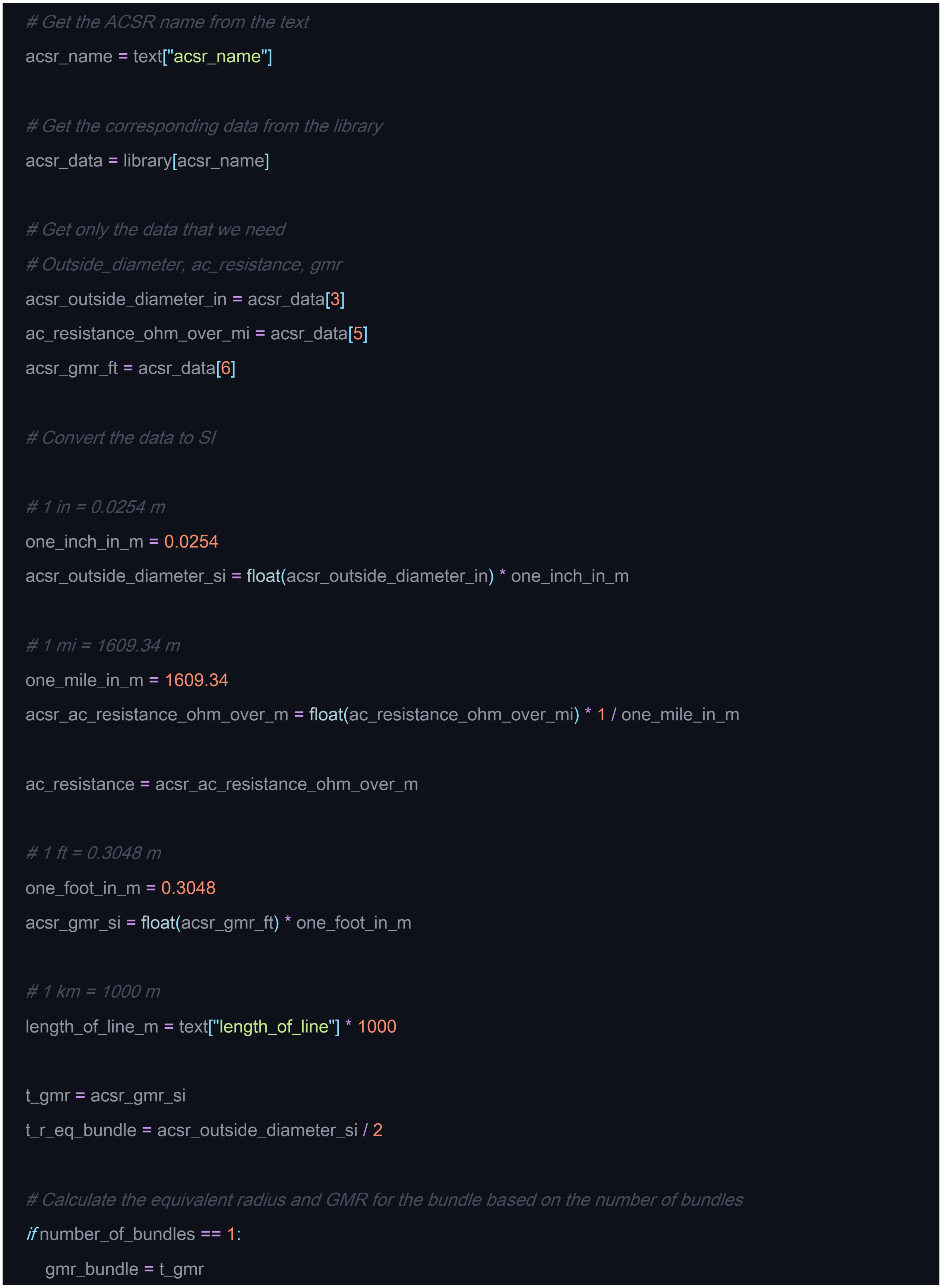
EE374 Term Project Phase 2



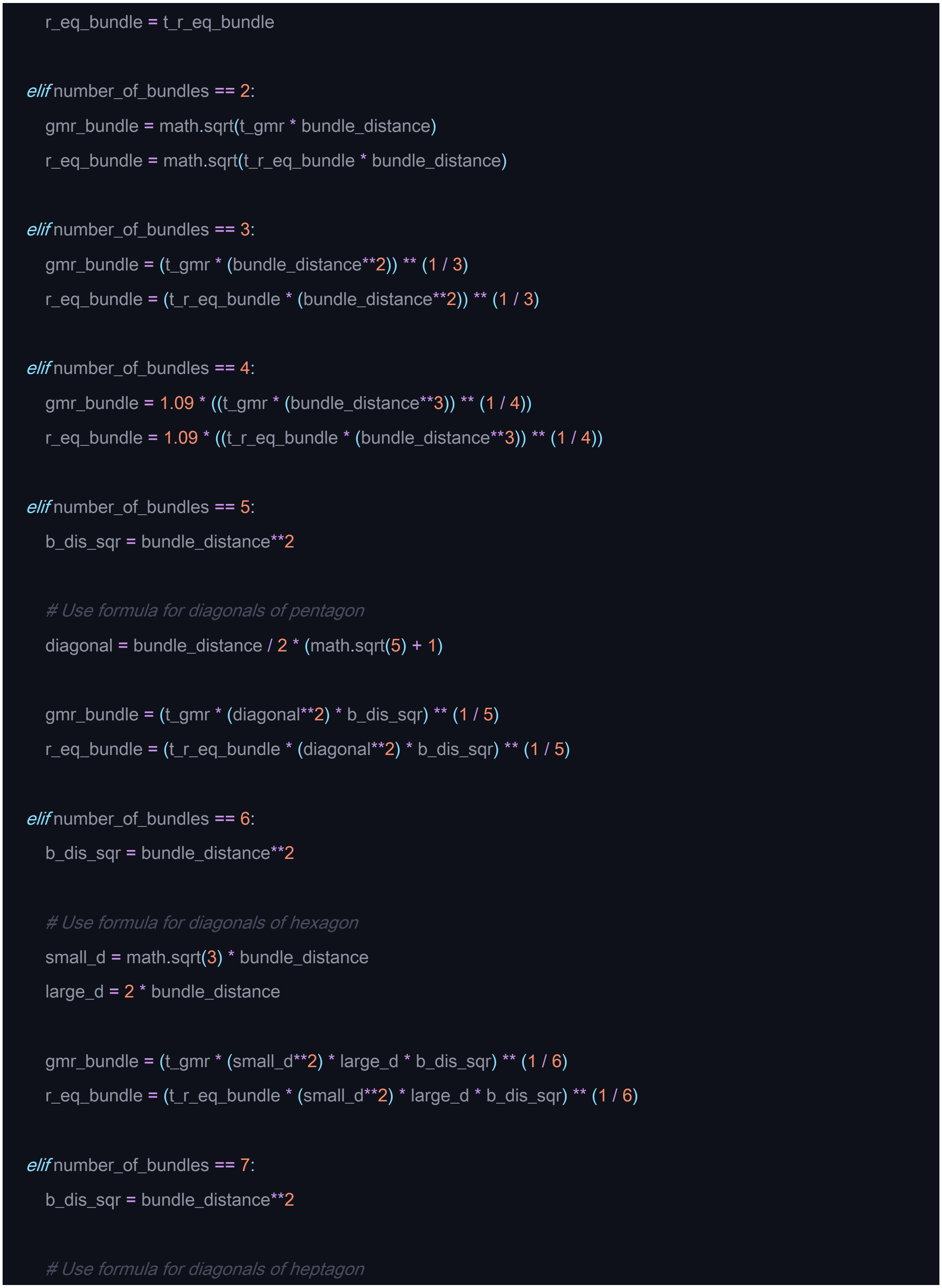
```python
# Created main.py on 11.06.2023
#
# Author: Deniz Karakay (2443307)
# Copyright (c) 2023 Deniz Karakay
# EE374 - Term Project Phase 2

import csv
import math


def termproject(text_path: str, library_path: str):
    """
    This function reads input data from the given text file path and library file path,
    calculates the total resistance, inductance and shunt susceptance of the line in per unit.

    Args:
        text_path (str): The file path of the input text file.
        library_path (str): The file path of the library file.

    Returns:
        List: A list containing the student ID and the calculated values.
    """
    text = {}

    with open(text_path, "r") as file:
        # Read the file and remove the \n
        lines = [line.strip() for line in file]

        # Get the values from the input file
        s_base = float(lines[1])
        v_base = float(lines[3])
        number_of_circuits = int(lines[5])
        number_of_bundles = int(lines[7])
        bundle_distance = float(lines[9])
```



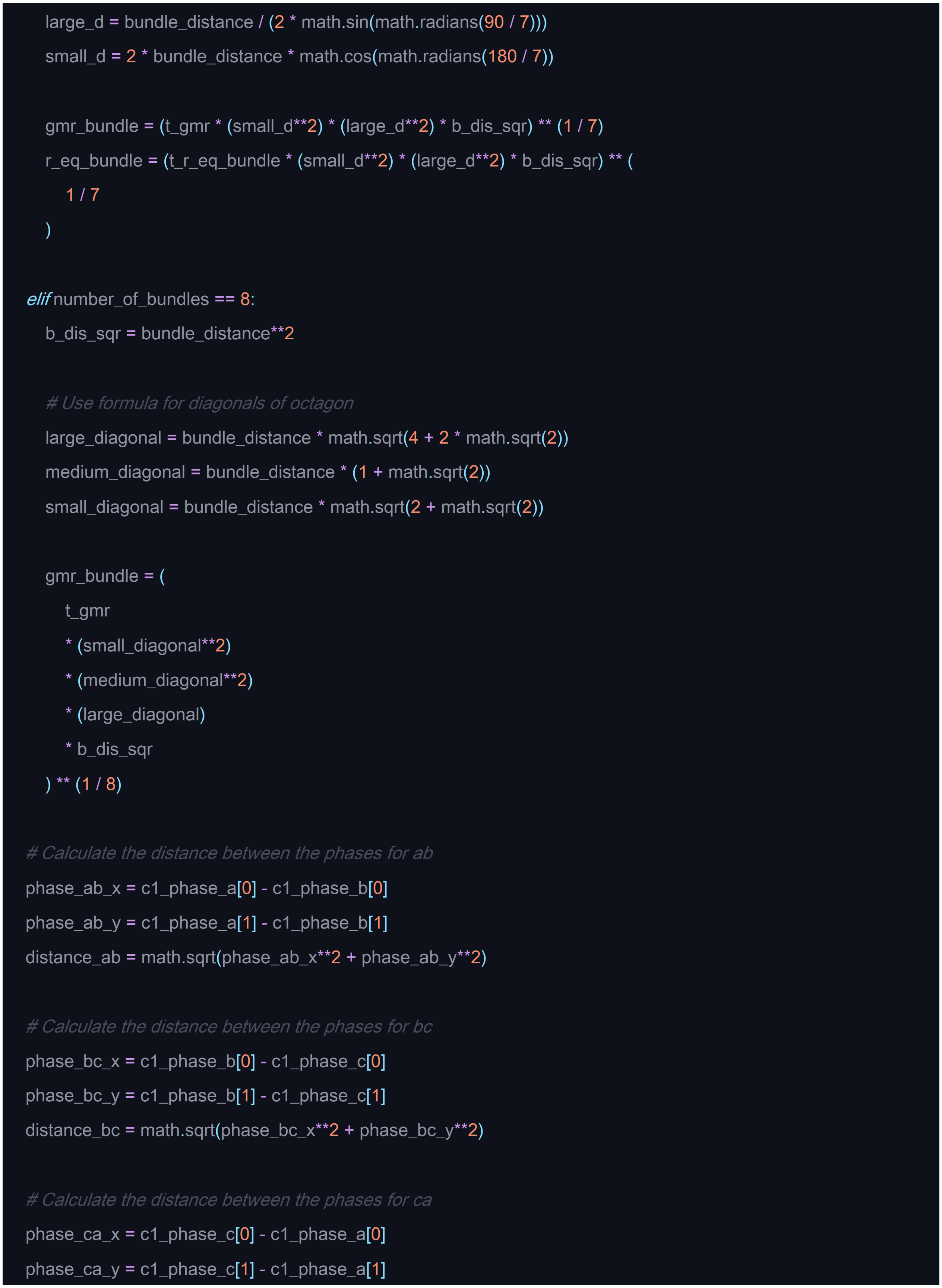
```
    length_of_line = float(lines[11])
    acsr_name = lines[13]

    # Get the values from the input file and convert them to int
    c1_phase_c = [float(line) for line in lines[15:17]]
    c1_phase_a = [float(line) for line in lines[18:20]]
    c1_phase_b = [float(line) for line in lines[21:23]]

    # Create a dictionary with the values from text
    text = {
        "s_base": s_base,
        "v_base": v_base,
        "number_of_circuits": number_of_circuits,
        "number_of_bundles": number_of_bundles,
        "bundle_distance": bundle_distance,
        "length_of_line": length_of_line,
        "acsr_name": acsr_name,
        "c1_phase_c": c1_phase_c,
        "c1_phase_a": c1_phase_a,
        "c1_phase_b": c1_phase_b,
    }

library = {}

# Read library csv file
with open(library_path, "r") as file:
    reader = csv.reader(file)

    # Exclude title row
    title_row = next(reader)

    # Create a dictionary with the values from library,
    # Set key as ACSR name and value as the rest of the row
    for row in reader:
        if row[0]:
            library[row[0]] = row[1:]
```



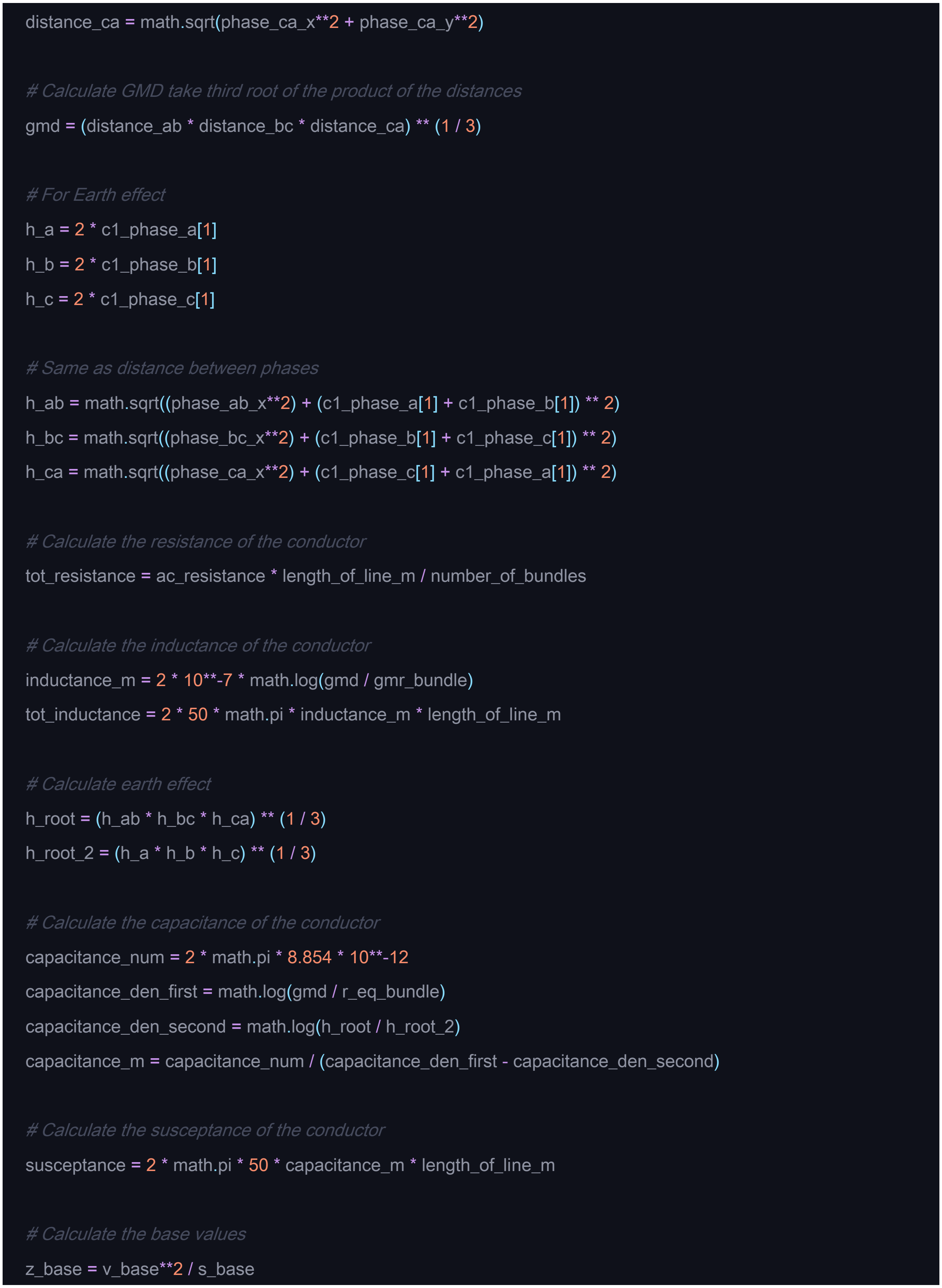
```
# Get the ACSR name from the text
acsr_name = text["acsr_name"]

# Get the corresponding data from the library
acsr_data = library[acsr_name]

# Get only the data that we need
# Outside_diameter, ac_resistance, gmr
acsr_outside_diameter_in = acsr_data[3]
ac_resistance_ohm_over_mi = acsr_data[5]
acsr_gmr_ft = acsr_data[6]

# Convert the data to SI

# 1 in = 0.0254 m
one_inch_in_m = 0.0254
acsr_outside_diameter_si = float(acsr_outside_diameter_in) * one_inch_in_m

# 1 mi = 1609.34 m
one_mile_in_m = 1609.34
acsr_ac_resistance_ohm_over_m = float(ac_resistance_ohm_over_mi) * 1 / one_mile_in_m

ac_resistance = acsr_ac_resistance_ohm_over_m

# 1 ft = 0.3048 m
one_foot_in_m = 0.3048
acsr_gmr_si = float(acsr_gmr_ft) * one_foot_in_m

# 1 km = 1000 m
length_of_line_m = text["length_of_line"] * 1000

t_gmr = acsr_gmr_si
t_r_eq_bundle = acsr_outside_diameter_si / 2

# Calculate the equivalent radius and GMR for the bundle based on the number of bundles
if number_of_bundles == 1:
    gmr_bundle = t_gmr
```



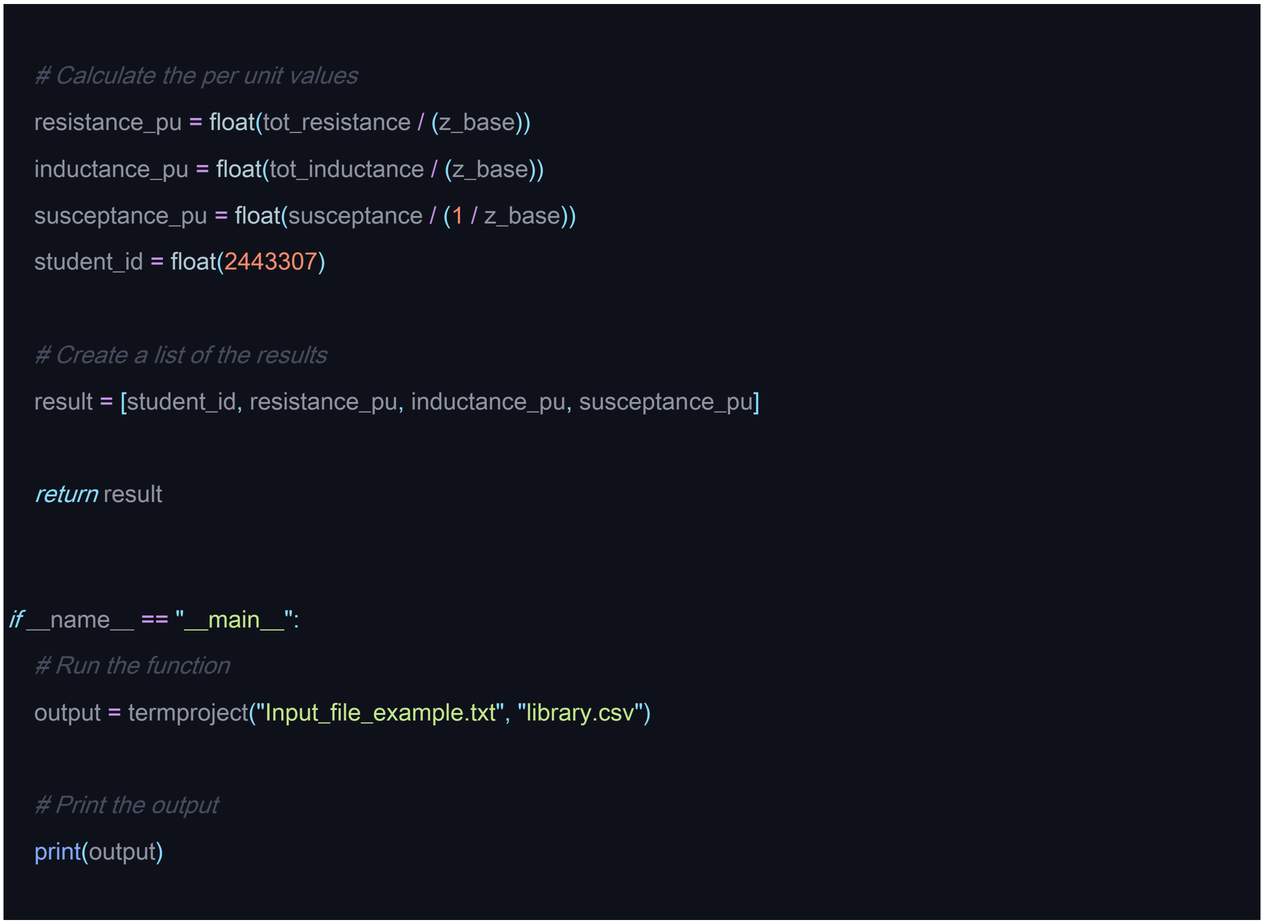
```
    r_eq_bundle = t_r_eq_bundle

elif number_of_bundles == 2:
    gmr_bundle = math.sqrt(t_gmr * bundle_distance)
    r_eq_bundle = math.sqrt(t_r_eq_bundle * bundle_distance)

elif number_of_bundles == 3:
    gmr_bundle = (t_gmr * (bundle_distance**2)) ** (1 / 3)
    r_eq_bundle = (t_r_eq_bundle * (bundle_distance**2)) ** (1 / 3)

elif number_of_bundles == 4:
    gmr_bundle = 1.09 * ((t_gmr * (bundle_distance**3)) ** (1 / 4))
    r_eq_bundle = 1.09 * ((t_r_eq_bundle * (bundle_distance**3)) ** (1 / 4))

elif number_of_bundles == 5:
    b_dis_sqr = bundle_distance**2

    # Use formula for diagonals of pentagon
    diagonal = bundle_distance / 2 * (math.sqrt(5) + 1)

    gmr_bundle = (t_gmr * (diagonal**2) * b_dis_sqr) ** (1 / 5)
    r_eq_bundle = (t_r_eq_bundle * (diagonal**2) * b_dis_sqr) ** (1 / 5)

elif number_of_bundles == 6:
    b_dis_sqr = bundle_distance**2

    # Use formula for diagonals of hexagon
    small_d = math.sqrt(3) * bundle_distance
    large_d = 2 * bundle_distance

    gmr_bundle = (t_gmr * (small_d**2) * large_d * b_dis_sqr) ** (1 / 6)
    r_eq_bundle = (t_r_eq_bundle * (small_d**2) * large_d * b_dis_sqr) ** (1 / 6)

elif number_of_bundles == 7:
    b_dis_sqr = bundle_distance**2

    # Use formula for diagonals of heptagon
```



```
    large_d = bundle_distance / (2 * math.sin(math.radians(90 / 7)))
    small_d = 2 * bundle_distance * math.cos(math.radians(180 / 7))

    gmr_bundle = (t_gmr * (small_d**2) * (large_d**2) * b_dis_sqr) ** (1 / 7)
    r_eq_bundle = (t_r_eq_bundle * (small_d**2) * (large_d**2) * b_dis_sqr) ** (
        1 / 7
    )

elif number_of_bundles == 8:
    b_dis_sqr = bundle_distance**2

    # Use formula for diagonals of octagon
    large_diagonal = bundle_distance * math.sqrt(4 + 2 * math.sqrt(2))
    medium_diagonal = bundle_distance * (1 + math.sqrt(2))
    small_diagonal = bundle_distance * math.sqrt(2 + math.sqrt(2))

    gmr_bundle = (
        t_gmr
        * (small_diagonal**2)
        * (medium_diagonal**2)
        * (large_diagonal)
        * b_dis_sqr
    ) ** (1 / 8)

# Calculate the distance between the phases for ab
phase_ab_x = c1_phase_a[0] - c1_phase_b[0]
phase_ab_y = c1_phase_a[1] - c1_phase_b[1]
distance_ab = math.sqrt(phase_ab_x**2 + phase_ab_y**2)

# Calculate the distance between the phases for bc
phase_bc_x = c1_phase_b[0] - c1_phase_c[0]
phase_bc_y = c1_phase_b[1] - c1_phase_c[1]
distance_bc = math.sqrt(phase_bc_x**2 + phase_bc_y**2)

# Calculate the distance between the phases for ca
phase_ca_x = c1_phase_c[0] - c1_phase_a[0]
phase_ca_y = c1_phase_c[1] - c1_phase_a[1]
```



```
distance_ca = math.sqrt(phase_ca_x**2 + phase_ca_y**2)

# Calculate GMD take third root of the product of the distances
gmd = (distance_ab * distance_bc * distance_ca) ** (1 / 3)

# For Earth effect
h_a = 2 * c1_phase_a[1]
h_b = 2 * c1_phase_b[1]
h_c = 2 * c1_phase_c[1]

# Same as distance between phases
h_ab = math.sqrt((phase_ab_x**2) + (c1_phase_a[1] + c1_phase_b[1]) ** 2)
h_bc = math.sqrt((phase_bc_x**2) + (c1_phase_b[1] + c1_phase_c[1]) ** 2)
h_ca = math.sqrt((phase_ca_x**2) + (c1_phase_c[1] + c1_phase_a[1]) ** 2)

# Calculate the resistance of the conductor
tot_resistance = ac_resistance * length_of_line_m / number_of_bundles

# Calculate the inductance of the conductor
inductance_m = 2 * 10**-7 * math.log(gmd / gmr_bundle)
tot_inductance = 2 * 50 * math.pi * inductance_m * length_of_line_m

# Calculate earth effect
h_root = (h_ab * h_bc * h_ca) ** (1 / 3)
h_root_2 = (h_a * h_b * h_c) ** (1 / 3)

# Calculate the capacitance of the conductor
capacitance_num = 2 * math.pi * 8.854 * 10**-12
capacitance_den_first = math.log(gmd / r_eq_bundle)
capacitance_den_second = math.log(h_root / h_root_2)
capacitance_m = capacitance_num / (capacitance_den_first - capacitance_den_second)

# Calculate the susceptance of the conductor
susceptance = 2 * math.pi * 50 * capacitance_m * length_of_line_m

# Calculate the base values
z_base = v_base**2 / s_base
```



```
    # Calculate the per unit values
    resistance_pu = float(tot_resistance / (z_base))
    inductance_pu = float(tot_inductance / (z_base))
    susceptance_pu = float(susceptance / (1 / z_base))
    student_id = float(2443307)

    # Create a list of the results
    result = [student_id, resistance_pu, inductance_pu, susceptance_pu]

    return result


if __name__ == "__main__":
    # Run the function
    output = termproject("Input_file_example.txt", "library.csv")

    # Print the output
    print(output)
```