

PM SHRI KENDRIYA VIDYALAYA

A Project Report

On

Scientific Calculator

For

SESSION ENDING EXAM, 2025

As a part of the Computer Science Course (083)



SUBMITTED BY:

Student Info

Under the Guidance of:

PGT COMPUTER SCIENCE

PROJECT CERTIFICATE



This is to certify that the Project / Dissertation entitled, Scientific Calculator is a bonafide work done by Master / Mister ... of class ... Session 202...-2... in partial fulfillment of Session Ending Exam 202... and has been carried out under my direct supervision and guidance.

This report or a similar report on the topic has not been submitted for any other examination and does not form a part of any other course undergone by the candidate.

Signature of Teacher/ Guide

Signature of Principal

Name:

Designation: PGT Computer Science

Principal



ACKNOWLEDGEMENT

As usual, a large number of people deserve my thanks for the help they provided me for the preparation of this project.

First of all, I would like to thank my teacher ... for his support during the preparation of this project. I am very thankful for his guidance.

...

At last, I would also to show our sincere gratitude to everyone who directly or indirectly, have lent their helping hand in this venture.

...

INDEX

<u>Index</u>	<u>Topic</u>	<u>Page Number</u>
1	PROJECT CERTIFICATE	3
2	ACKNOWLEDGMENT	4
3	OBJECTIVE	6
4	SCOPE	7
5	THEORETICAL BACKGROUND	8
6	SYSTEM PLANNING	10
7	METHODOLOGY ADOPTED AND DETAILS OF HARDWARE/SOFTWARE USED	11
8	SNIPPETS OF SOURCE CODE	12
9	SNIPPETS OF TEST RUN	25
10	USER MANUAL	35
11	BRIEF BACKGROUND OF ORGANIZATION	38
12	REFERENCE	39
13	ANNEXURE - 1	40

OBJECTIVE

The primary objective of this project is to develop a menu-driven Scientific Calculator program using Python.

This calculator integrates various functionalities such as simple arithmetic operations, unit conversions, and trigonometric computations into a single, cohesive program.

It aims to provide users with a seamless experience for performing essential mathematical and scientific calculations.

The project also seeks to demonstrate fundamental programming concepts such as modular programming, and dictionary usage, aligning with the foundational knowledge expected from CBSE computer science curriculum, along with showcasing the power of error handling provided in Python(not a part of the curriculum).

SCOPE OF THE PROJECT

The scope of the Scientific Calculator project encompasses the following key aspects:

1. Functionality:

- The project includes three distinct modules:
SimpleArithmeticModule(for arithmetic), UnitConversionModule (to convert different units of length, mass and time), and TrigFuncModule(to calculate the value of sine, cosine and tangent in degree or radian).
- Users can perform calculations within a single module or switch between modules seamlessly using the main menu-driven program.

2. Modular Design:

- Each module functions independently and can also be invoked through the main Scientific Calculator program.
- This modular approach promotes code reusability and maintainability.

3. User-Friendly Interface:

- The program is designed to handle user inputs effectively, providing clear prompts and feedback for errors.
- It ensures that the program remains accessible to users with basic knowledge of computers and mathematics.

THEORETICAL BACKGROUND

The project leverages key theoretical concepts and principles from computer science and mathematics to build a Scientific Calculator. These include:

1. Modular Programming:

- The calculator is structured into three separate Python modules, each handling a specific type of calculation. This modular design enhances readability, simplifies debugging, and encourages reusability of code.

2. Mathematical Foundations:

- The Simple Arithmetic module performs basic mathematical operations such as addition, subtraction, multiplication, and division, using Python's arithmetic operators.
- The Unit Conversion module incorporates conversion factors for length, mass, and time, allowing accurate transformations between units.
- The Trigonometric Functions module utilizes Python's `math` library to compute sine, cosine, and tangent values for given angles, expressed in degrees or radians.

3. Error Handling:

- The program uses try-except blocks to handle potential errors, such as invalid user inputs and division by zero, ensuring that the program runs smoothly without unexpected interruptions.

4. Data Structures:

- Dictionaries are used extensively for mapping user inputs to corresponding operations, such as arithmetic symbols and unit conversion factors, providing a structured and efficient way to manage data.

5. Control Flow Statements:

- Loops and conditional statements form the backbone of the menu-driven interface, enabling users to navigate between different modules and options seamlessly.

SYSTEM PLANNING

The development of the Scientific Calculator was systematically planned and executed in the following phases:

1. Requirements Analysis:

- Identify the key functionalities to be included in the calculator, such as arithmetic operations, unit conversions, and trigonometric calculations.
- Determine the user requirements for a simple and intuitive interface.

2. Design:

- Develop a modular design by creating separate Python modules for each functionality.
- Design a main program to serve as the entry point, providing a menu-driven interface for navigating between modules.

3. Implementation:

- Write the code for each module, focusing on accuracy, efficiency, and error handling.
- Implement the main program to integrate the modules and provide seamless switching between them.

4. Testing:

- Test each module independently to ensure its functionality.
- Conduct integration testing to verify that the modules work together as intended.
- Perform user testing to identify and resolve any usability issues.

5. Documentation:

- Document the program's purpose, functionality, and structure.
- Include clear instructions for users and developers, ensuring the project is easy to understand and extend.

METHODOLOGY ADOPTED AND DETAILS OF HARDWARE/SOFTWARE USED

For this project, we developed a menu-based scientific calculator program using Python. For this, we developed four programs: the main scientific calculator, the SimpleArithmeticModule, the UnitConversionModule, and the TrigFuncModule. Each module is designed to be directly used, but can also be accessed through the main scientific calculator interface.

The following Python features were used in the development of this project:

1. **Dictionaries:** Employed for storing and referencing different functions within the calculator and modules.
2. **Def Blocks:** Functions were created using def to modularize tasks and improve code readability and reusability.
3. **Error Handling:** Implemented try-except blocks to manage potential errors, such as invalid inputs or division by zero.
4. **Math Module:** Utilized Python's built-in math module to perform trigonometric calculations.

Hardware and Software

- **Hardware:** The program was developed and tested on a personal p.c. with standard specifications: Windows 10 64-bit operating system, 4GB DDR3 RAM, 512 GB SSD ROM, and an Intel Core i5-3570k processor with built-in Intel HD Graphics 4000.
- **Software:** Python 3.13 was used for programming, and the program was developed using PyCharm Community Edition as the Integrated Development Environment (IDE). Additionally, the program was developed and tested on Windows 10.

SNIPPET OF SOURCE CODE

SimpleArithmeticModule

```
#Simple Arithmetic Module
```

```
def Welcome():
    print("""
    *****
    *                SIMPLE ARITHMETIC CALCULATOR                *
    *****

    Welcome! Let's perform some arithmetic calculations.

    How to Use the Program:
    1. Enter a number when prompted.
    2. Select the operation you want to perform by entering the number
    corresponding to your choice.

    -----
    | IMPORTANT: The program does NOT follow BODMAS OR PEDMAS.
    |
    | Operations are evaluated from left to right.
    |
    | Example: 2 + 2 * 4 gives 16(incorrect) instead of 10(correct)
    |
    -----
    """)

Arithmetic_Symbols = {1: '+', 2: '-', 3: '*', 4: '/' }

total, total_output = 0, ''

def PreStart():
    global total, total_output
    while True:
        try:
            total = float(input('\nEnter the first number: '))
            total_output = str(total)
            print(f'Your input till now --> {total_output}')
            break
```

```

    except ValueError:
        print('Please enter a valid integer/float')
        continue

def addition(add):
    global total
    total += add

def subtraction(sub):
    global total
    total -= sub

def multiplication(mul):
    global total
    total *= mul

def division(div):
    if div == 0:
        print('Division by 0 is not allowed/defined')
        return
    else:
        global total
        total /= div

def StartModule():
    global total_output
    while True:
        print('\nFrom the following, input a choice:')
        print('1. Addition \t\t2. Subtraction '
              '\n3. Multiplication \t\t4. Division '
              '\n0. Exit')

        try:
            user_choice = int(input('Enter your choice (1-4 or 0 to exit): '))
        except ValueError:
            print('Please enter a valid choice!')
            continue

        if user_choice == 0:
            print(f'\nThe total is: {total_output} ==> {total:.10f}...')
            print('
            -----
            |   Exiting the Arithmetic module.   |
            -----
            ')

```

```

        break

    elif user_choice < 1 or user_choice > 4:
        print('\nPlease enter a valid choice!')

    else:
        try:
            user_input = float(input('Enter a number: '))
        except ValueError:
            print('Please enter an integer or float')
            continue
        if user_choice == 1:
            addition(user_input)
        elif user_choice == 2:
            subtraction(user_input)
        elif user_choice == 3 :
            multiplication(user_input)
        elif user_choice == 4:
            division(user_input)

        total_output += ' ' + Arithmetic_Symbols[user_choice] + ' ' +
str(user_input)
        print(f'Your current total: {total_output} ==> {total:.10f}')

if __name__ == '__main__':

    Welcome()
    PreStart()
    StartModule()

```

UnitConversionModule

#Unit Conversion Module

```
def convert_length(value, from_unit, to_unit):
    # Define conversion factors for length
    length_units = {
        "meters": 1,
        "kilometers": 1000,
        "centimeters": 0.01,
        "millimeters": 0.001,
        "miles": 1609.344,
        "yards": 0.9144,
        "feet": 0.3048,
        "inches": 0.0254
    }

    # Convert the value to meters first
    value_in_meters = value * length_units[from_unit]

    # Convert the value in meters to the target unit
    converted_value = value_in_meters / length_units[to_unit]

    return converted_value

def convert_mass(value, from_unit, to_unit):
    # Define conversion factors for mass
    mass_units = {
        'grams':1,
        'kilograms':1000,
        'milligrams':0.001,
        'micrograms':0.000001,
        'tonne':1000000,
        'ounce':28.35,
        'pound':453.6,
        'ton':9071581
    }

    # Convert the value to kilograms first
    value_in_kilograms = value * mass_units[from_unit]

    # Convert the value in kilograms to the target unit
    converted_value = value_in_kilograms / mass_units[to_unit]

    return converted_value

def convert_time(value, from_unit, to_unit):
    # Define conversion factors for time
```

```

time_units = {
    'seconds':1,
    'milliseconds':0.001,
    'microseconds':0.000001,
    'minute':60,
    'hour':3600,
    'day':86400,
    'week':604800,
    'month':2.628E6,
    'year':31536000
}

# Convert the value to seconds first
value_in_seconds = value * time_units[from_unit]

# Convert the value in seconds to the target unit
converted_value = value_in_seconds / time_units[to_unit]

return converted_value

def display_menu():
    print("\nUnit Conversion Menu:")
    print("1. Length")
    print("2. Mass")
    print("3. Time")
    print("0. Exit\n")

def StartModule():
    print("""
*****
*                               *
*           UNIT CONVERSION CALCULATOR           *
*                               *
*****

Welcome! Let's do some unit conversions!

How to Use the Program:
1. Enter a number wherever prompted.
2. Choose the physical quantity that you would like to convert.
3. Enter the unit of your value, then the unit you want to convert to
convert to.
4. At last, enter the value itself.

-----
| IMPORTANT: You can only convert between different units of length, mass,
and time. |
-----
""")

```



```

    """)

# Display menu
while True:
    display_menu()
    try:
        choice = int(input("Choose the type of conversion (1-3 or 0 to exit):
"))
    except ValueError:
        print('\nPlease enter a valid choice!')
        continue

    if choice == 0:
        print('

-----
|       Exiting the Unit Conversion module.       |
-----

')
        break

# Choose units for Length
elif choice == 1:
    print("\nLength Units:")
    length_units = ["meters", "kilometers", "centimeters", "millimeters",
"miles", "yards", "feet", "inches"]
    for index, unit in enumerate(length_units, 1):
        print(f"{index}. {unit}")

    try:
        from_choice = int(input("\nChoose the unit to convert from (1-8):
")) - 1
        to_choice = int(input("Choose the unit to convert to (1-8): ")) -
1
    except ValueError:
        print('Invalid input! Please choose from any of the options')
        continue

    if from_choice != to_choice:
        try:
            value = float(input(f"Enter the value in
{length_units[from_choice]}: "))
            converted_value = convert_length(value,
length_units[from_choice], length_units[to_choice])
            print(
                f'''\n{value} {length_units[from_choice]} ==>
{converted_value:.10f} {length_units[to_choice]}'''

```

```

        except ValueError:
            print('Please enter a valid number to convert!')
            continue
    else:
        print("You chose the same units. No conversion necessary.")

    elif choice == 2:
        print('\nMass Units:')
        mass_units =
['grams', 'kilograms', 'milligrams', 'micrograms', 'tonne', 'ounce', 'pound', 'ton']
        for index, unit in enumerate(mass_units, 1):
            print(f"{index}. {unit}")

        try:
            from_choice = int(input("\nChoose the unit to convert from (1-8):
")) - 1
            to_choice = int(input("Choose the unit to convert to (1-8): ")) -
1
        except ValueError:
            print('Invalid input! Please choose from any of the options')
            continue

        if from_choice != to_choice:
            try:
                value = float(input(f'Enter the value in
{mass_units[from_choice]}: '))
                converted_value = convert_mass(value, mass_units[from_choice],
mass_units[to_choice])
                print(f"{value} {mass_units[from_choice]} is equal to
{converted_value:.10f} {mass_units[to_choice]}")
            except ValueError:
                print('Please enter a valid number to convert!')
                continue
        else:
            print("You chose the same units. No conversion necessary.")

    elif choice == 3:
        print('\nTime Units:')
        time_units =
['seconds', 'milliseconds', 'microseconds', 'minute', 'hour', 'day', 'week', 'month', 'ye
ar']
        for index, unit in enumerate(time_units, 1):
            print(f"{index}. {unit}")

        try:
            from_choice = int(input('\nChoose the unit to convert from (1-9):
')) - 1
            to_choice = int(input('Choose the unit to convert to (1-9): ')) -

```

1

```
    except ValueError:
        print('Invalid input! Please choose from any of the options')
        continue

    if from_choice != to_choice:
        try:
            value = float(input(f'Enter the value in
{time_units[from_choice]}: '))
            converted_value = convert_time(value, time_units[from_choice],
time_units[to_choice])
            print(f'{value} {time_units[from_choice]} is equal to
{converted_value:.10f} {time_units[to_choice]}')
            except ValueError:
                print('Please enter a valid number to convert!')
                continue
        else:
            print('You chose the same units. No conversion necessary.')
            # You can add more conversions for mass and time similarly
    else:
        print("\nInvalid choice. Please choose a valid conversion type from 1-
3 or 0 to exit!")

# Start the conversion program
if __name__ == '__main__':
    StartModule()
```

TrigFuncModule

```
# Trigonometric Functions Module

import math

Trig_Functions = {1:'sin',2:'cos',3:'tan'}

def Trigonometry(angle, trigonometry_choice, user_input_angle, user_choice_angle,
trigon_choice):

    trigonometry =
    {'sin':math.sin(angle), 'cos':math.cos(angle), 'tan':math.tan(angle)}

    print(f'\nThe {trigonometry_choice} of {user_input_angle}{user_choice_angle} =
{trigonometry[trigon_choice]:.3f}')

def StartModule():
    global Trig_Functions
    print("""
*****
*                               *
*           TRIGONOMETRY CALCULATOR           *
*                               *
*****

Welcome! Let's calculate the value of trigonometric functions.

How to Use the Program:
1. Enter a number when prompted.
2. Choose the trig func of your choice.
3. Enter the unit of angle and the angle itself.

-----
| IMPORTANT: The tan of 90.0° is undefined, and not equal to
16331239353195370.000 |
-----
""")

    while True:
        print('\nFrom the following, input a choice:')
        print('1. sin\t2. cos\t3. tan')

        try:
            trig_choice = int(input('Enter your choice\t: '))
            trig_choice = Trig_Functions[trig_choice]
        except KeyError:
            print(f'\n--> Invalid choice! Please enter a number from 1-3 <--')
            continue
```

```

except ValueError:
    print(f'\n--> Invalid input! Please enter a number from 1-3 <--')
    continue

print('\nFrom the following, input a choice:')
print('1. Degree\t2. Radian')

try:
    angle_choice = int(input('Enter your choice\t: '))
except ValueError:
    print(f'\n--> Invalid input! Please enter 1 for degree and 2 for
radian <--')
    continue
if angle_choice not in (1,2):
    print(f'\n--> Invalid choice! <--')
    continue
else:
    try:
        angle_input = float(input('\nEnter the angle\t: '))
    except ValueError:
        print(f'\n--> Invalid angle input! <--')
        continue

    if angle_choice == 1:
        angle_radian, angle_choice = math.radians(angle_input), '°'
        Trigonometry(angle_radian, trig_choice, angle_input, angle_choice,
trig_choice)
    elif angle_choice == 2:
        angle_radian, angle_choice = angle_input, 'rad'
        Trigonometry(angle_radian, trig_choice, angle_input, angle_choice,
trig_choice)

    cont = input('\nDo you want to perform another calculation? (y/n) ')

    if cont.lower().strip() not in 'yes':
        print('

        -----
        |      Exiting the Trigonometric module.      |
        -----

        ')
        break

if __name__ == '__main__':
    StartModule()

```

Scientific Calculator

Menu Driven Scientific Calculator

User will be asked what would they like to do

Based on that, the corresponding custom module's functions will be called

```
count_module = 0
```

```
try:
```

```
    import SimpleArithmeticModule as SAM
```

```
except ModuleNotFoundError:
```

```
    count_module += 1
```

```
    pass
```

```
try:
```

```
    import UnitConversionModule as UCM
```

```
except ModuleNotFoundError:
```

```
    count_module += 1
```

```
    pass
```

```
try:
```

```
    import TrigFuncModule as TFM
```

```
except ModuleNotFoundError:
```

```
    count_module += 1
```

```
    pass
```

```
def display_menu():
```

```
    print('-' * 78)
```

```
    print('| Choose an option from one of the following:
```

```
    |')
```

```
    print('| 1. Simple Arithmetic\t\t\t2. Unit Conversion
```

```
    |'
```

```
        '\n| 3. Trigonometry\t\t\t\t0. Exit (Stop the scientific calculator!)
```

```
    |')
```

```
    print('-' * 78)
```

```
def StartScientificCalculator():
```

```
    print(f'{'':^180}')
```

```
    print(f'{'|'                                     |':^180}')
```

```
    print(f'{'|'          SCIENTIFIC CALCULATOR      |':^180}')
```

```
    print(f'{'|'                                     |':^180}')
```

```
    print(f'{'':^180}\n')
```

```
    while True:
```

```
        display_menu()
```

```
        try:
```

```

        user_choice = int(input('\nEnter your choice: '))
    except ValueError :
        print('Please enter a valid choice!')
        continue

    if user_choice == 0 :
        print()
        print(f'{'':^180}')
        print(f'{''|          SCIENTIFIC CALCULATOR|':^180}')
        print(f'{''|          EXITING PROGRAM|':^180}')
        print(f'{''|          |':^180}')
        print(f'{'':^180}')
        break

    elif user_choice != 0 and 1 <= user_choice <= 3:

        if user_choice == 1 :
            try:
                SAM.Welcome()
                SAM.PreStart()
                SAM.StartModule()
            except NameError:
                print('\nIt seems the SimpleArithmeticModule has not been
installed.'
                    '\nPlease do so before using this section of the
Scientific Calculator again.'
                    '\nRemember to restart the Scientific Calculator once
you have installed the required module!\n')

        elif user_choice == 2 :
            try:
                UCM.StartModule()
            except NameError:
                print('\nIt seems the UnitConversionModule has not been
installed.\n'
                    'Please do so before using this section of the
Scientific Calculator again.\n'
                    'Remember to restart the Scientific Calculator once you
have installed the required module!\n')

        elif user_choice == 3 :
            try:
                TFM.StartModule()
            except NameError:
                print('\nIt seems the TrigFuncModule has not been
installed.\n'
                    'Please do so before using this section of the

```

```

Scientific Calculator again.\n'
        'Remember to restart the Scientific Calculator once you
have installed the required module!\n')

    else:
        print('\nPlease enter a valid choice from the given options\n')

if __name__ == "__main__":
    if count_module == 3:
        print("\nIt seems none of the required modules are installed.")
        print("Please install SimpleArithmeticModule, UnitConversionModule, and
TrigFuncModule.")
        print("Exiting the Scientific Calculator...")
    else:
        if count_module > 0:
            print(
f'''
-----
| Warning: {count_module} module(s) are missing.
|
| You can still use the Scientific Calculator, but some functionalities may not
be available.
|
-----
'''
)

        StartScientificCalculator()

```


SNIPPETS OF TEST RUNS

```
Run - 11th_CS_Project
Scientific Calculator x

C:\Users\Hansika\PycharmProjects\11th_CS_Project\.venv\Scripts\python.exe "C:\Users\Hansika\PycharmProjects\11th_CS_Project\Scientific Calculator.py"

=====
|                               | SCIENTIFIC CALCULATOR |                               |
=====

-----
| Choose an option from one of the following: |
| 1. Simple Arithmetic      2. Unit Conversion |
| 3. Trigonometry          0. Exit (Stop the scientific calculator!) |
-----

Enter your choice: 1

*****
*           SIMPLE ARITHMETIC CALCULATOR           *
*****

Welcome! Let's perform some arithmetic calculations.

How to Use the Program:
1. Enter a number when prompted.
2. Select the operation you want to perform by entering the number corresponding to your choice.

-----
| IMPORTANT: The program does NOT follow BODMAS OR PEDMAS. |
| Operations are evaluated from left to right.              |
| Example: 2 + 2 * 4 gives 16(incorrect) instead of 10(correct) |
-----

Run - 11th_CS_Project
Scientific Calculator x

Enter the first number: 1
Your input till now --> 1.0

From the following, input a choice:
1. Addition      2. Subtraction
3. Multiplication 4. Division
0. Exit
Enter your choice (1-4 or 0 to exit): 1
Enter a number: 1
Your current total: 1.0 + 1.0 ==> 2.0000000000

From the following, input a choice:
1. Addition      2. Subtraction
3. Multiplication 4. Division
0. Exit
Enter your choice (1-4 or 0 to exit): 2
Enter a number: 1
Your current total: 1.0 + 1.0 - 1.0 ==> 1.0000000000

From the following, input a choice:
1. Addition      2. Subtraction
3. Multiplication 4. Division
0. Exit
Enter your choice (1-4 or 0 to exit): 3
Enter a number: 1
Your current total: 1.0 + 1.0 - 1.0 * 1.0 ==> 1.0000000000

From the following, input a choice:
1. Addition      2. Subtraction
3. Multiplication 4. Division
0. Exit
Enter your choice (1-4 or 0 to exit): 4
```

```
Run - 11th_CS_Project
Scientific Calculator x
Enter a number: 1
Your current total: 1.0 + 1.0 - 1.0 * 1.0 / 1.0 ==> 1.0000000000

From the following, input a choice:
1. Addition          2. Subtraction
3. Multiplication    4. Division
0. Exit
Enter your choice (1-4 or 0 to exit): 0

The total is: 1.0 + 1.0 - 1.0 * 1.0 / 1.0 ==> 1.0000000000...

-----
|   Exiting the Arithmetic module.   |
-----

| Choose an option from one of the following: |
| 1. Simple Arithmetic      2. Unit Conversion |
| 3. Trigonometry          0. Exit (Stop the scientific calculator!) |
|-----|

Enter your choice: 2

*****
*           UNIT CONVERSION CALCULATOR           *
*****

Welcome! Let's do some unit conversions!

How to Use the Program:
* Enter a number wherever prompted.
* Choose the physical quantity that you would like to convert.
* Enter the unit of your value, then the unit you want to convert to.
* At last, enter the value itself.
```

```
Run - 11th_CS_Project
Scientific Calculator x

How to Use the Program:
1. Enter a number wherever prompted.
2. Choose the physical quantity that you would like to convert.
3. Enter the unit of your value, then the unit you want to convert to.
4. At last, enter the value itself.

-----
| IMPORTANT: You can only convert between different units of length, mass, and time. |
-----

Unit Conversion Menu:
1. Length
2. Mass
3. Time
0. Exit

Choose the type of conversion (1-3 or 0 to exit): 1

Length Units:
1. meters
2. kilometers
3. centimeters
4. millimeters
5. miles
6. yards
7. feet
8. inches

Choose the unit to convert from (1-8): 5
Choose the unit to convert to (1-8): 2
```

```
Run - 11th_CS_Project
Scientific Calculator x
Choose the unit to convert to (1-8): 2
Enter the value in miles: 24
24.0 miles ==> 38.6242560000 kilometers
Unit Conversion Menu:
1. Length
2. Mass
3. Time
0. Exit
Choose the type of conversion (1-3 or 0 to exit): 2
Mass Units:
1. grams
2. kilograms
3. milligrams
4. micrograms
5. tonne
6. ounce
7. pound
8. US ton
Choose the unit to convert from (1-8): 7
Choose the unit to convert to (1-8): 2
Enter the value in pound: 210
210.0 pound is equal to 95.2560000000 kilograms
Unit Conversion Menu:
1. Length
2. Mass
3. Time
0. Exit
Choose the type of conversion (1-3 or 0 to exit): 3
Time Units:
1. seconds
2. milliseconds
3. microseconds
4. minute
5. hour
6. day
7. week
8. month
9. year
Choose the unit to convert from (1-9): 7
Choose the unit to convert to (1-9): 9
Enter the value in week: 104
104.0 week is equal to 1.9945205479 year
Unit Conversion Menu:
1. Length
2. Mass
3. Time
0. Exit
Choose the type of conversion (1-3 or 0 to exit): 0
```

```
Run - 11th_CS_Project
Scientific Calculator x
Choose the type of conversion (1-3 or 0 to exit): 0

-----
|   Exiting the Unit Conversion module.   |
-----

| Choose an option from one of the following: |
| 1. Simple Arithmetic      2. Unit Conversion |
| 3. Trigonometry          0. Exit (Stop the scientific calculator!) |
|-----|

Enter your choice: 3

*****
*          TRIGONOMETRY CALCULATOR          *
*****

Welcome! Let's calculate the value of trigonometric functions.

How to Use the Program:
1. Enter a number when prompted.
2. Choose the trig func of your choice.
3. Enter the unit of angle and the angle itself.

-----
| IMPORTANT: The tan of 90.0° is undefined, and not equal to 16331239353195370.000 |
|-----|

Run - 11th_CS_Project
Scientific Calculator x
| IMPORTANT: The tan of 90.0° is undefined, and not equal to 16331239353195370.000 |
|-----|

From the following, input a choice:
1. sin    2. cos    3. tan
Enter your choice : 1

From the following, input a choice:
1. Degree  2. Radian
Enter your choice : 1

Enter the angle : 30

The sin of 30.0° = 0.500

Do you want to perform another calculation? (y/n) y

From the following, input a choice:
1. sin    2. cos    3. tan
Enter your choice : 2

From the following, input a choice:
1. Degree  2. Radian
Enter your choice : 2

Enter the angle : 1.57

The cos of 1.57 rad = 0.001
```

```
Run - 11th_CS_Project
Scientific Calculator x

Do you want to perform another calculation? (y/n) n

| Exiting the Trigonometric module. |

-----
| Choose an option from one of the following: |
| 1. Simple Arithmetic      2. Unit Conversion |
| 3. Trigonometry          0. Exit (Stop the scientific calculator!) |
-----

Enter your choice: error
Please enter a valid choice!

-----
| Choose an option from one of the following: |
| 1. Simple Arithmetic      2. Unit Conversion |
| 3. Trigonometry          0. Exit (Stop the scientific calculator!) |
-----

Enter your choice: 1

*****
*           SIMPLE ARITHMETIC CALCULATOR           *
*****

Welcome! Let's perform some arithmetic calculations.

How to Use the Program:

Run - 11th_CS_Project
Scientific Calculator x

How to Use the Program:
1. Enter a number when prompted.
2. Select the operation you want to perform by entering the number corresponding to your choice.

-----
| IMPORTANT: The program does NOT follow BODMAS OR PEDMAS. |
| Operations are evaluated from left to right. |
| Example: 2 + 2 * 4 gives 16(incorrect) instead of 10(correct) |
-----

Enter the first number: error
Please enter a valid integer/float

Enter the first number: 1
Your input till now --> 1.0

From the following, input a choice:
1. Addition      2. Subtraction
3. Multiplication 4. Division
0. Exit
Enter your choice (1-4 or 0 to exit): error
Please enter a valid choice!

From the following, input a choice:
1. Addition      2. Subtraction
3. Multiplication 4. Division
0. Exit
Enter your choice (1-4 or 0 to exit): 1
Enter a number: error
Please enter an integer or float
```

```
Run - 11th_CS_Project
Scientific Calculator x
Please enter an integer or float
From the following, input a choice:
1. Addition      2. Subtraction
3. Multiplication 4. Division
0. Exit
Enter your choice (1-4 or 0 to exit): 1
Enter a number: 1
Your current total: 1.0 + 1.0 ==> 2.0000000000
From the following, input a choice:
1. Addition      2. Subtraction
3. Multiplication 4. Division
0. Exit
Enter your choice (1-4 or 0 to exit): 0
The total is: 1.0 + 1.0 ==> 2.0000000000...
| Exiting the Arithmetic module. |
| Choose an option from one of the following: |
| 1. Simple Arithmetic      2. Unit Conversion |
| 3. Trigonometry          0. Exit (Stop the scientific calculator!) |
Enter your choice: 2
Run - 11th_CS_Project
Scientific Calculator x
Enter your choice: 2
*****
*          UNIT CONVERSION CALCULATOR          *
*****
Welcome! Let's do some unit conversions!
How to Use the Program:
1. Enter a number wherever prompted.
2. Choose the physical quantity that you would like to convert.
3. Enter the unit of your value, then the unit you want to convert to.
4. At last, enter the value itself.
| IMPORTANT: You can only convert between different units of length, mass, and time. |
Unit Conversion Menu:
1. Length
2. Mass
3. Time
0. Exit
Choose the type of conversion (1-3 or 0 to exit): error
Please enter a valid choice!
Unit Conversion Menu:
1. Length
2. Mass
```

```
Run - 11th_CS_Project
Scientific Calculator x
1. Length
2. Mass
3. Time
0. Exit
Choose the type of conversion (1-3 or 0 to exit): 1
Length Units:
1. meters
2. kilometers
3. centimeters
4. millimeters
5. miles
6. yards
7. feet
8. inches
Choose the unit to convert from (1-8): error
Invalid input! Please choose from any of the options
Unit Conversion Menu:
1. Length
2. Mass
3. Time
0. Exit
Choose the type of conversion (1-3 or 0 to exit): 1
Length Units:
1. meters
2. kilometers
3. centimeters

Run - 11th_CS_Project
Scientific Calculator x
2. kilometers
3. centimeters
4. millimeters
5. miles
6. yards
7. feet
8. inches
Choose the unit to convert from (1-8): 1
Choose the unit to convert to (1-8): error
Invalid input! Please choose from any of the options
Unit Conversion Menu:
1. Length
2. Mass
3. Time
0. Exit
Choose the type of conversion (1-3 or 0 to exit): 1
Length Units:
1. meters
2. kilometers
3. centimeters
4. millimeters
5. miles
6. yards
7. feet
8. inches
Choose the unit to convert from (1-8): 1
Choose the unit to convert to (1-8): 1
```

```
Run - 11th_CS_Project
Scientific Calculator x
Choose the unit to convert from (1-8): 1
Choose the unit to convert to (1-8): 1
You chose the same units. No conversion necessary.

Unit Conversion Menu:
1. Length
2. Mass
3. Time
0. Exit

Choose the type of conversion (1-3 or 0 to exit): 1

Length Units:
1. meters
2. kilometers
3. centimeters
4. millimeters
5. miles
6. yards
7. feet
8. inches

Choose the unit to convert from (1-8): 1
Choose the unit to convert to (1-8): 2
Enter the value in meters: error
Please enter a valid number to convert!

Unit Conversion Menu:
1. Length
2. Mass
3. Time
0. Exit

Run - 11th_CS_Project
Scientific Calculator x
Choose the type of conversion (1-3 or 0 to exit): 1

Length Units:
1. meters
2. kilometers
3. centimeters
4. millimeters
5. miles
6. yards
7. feet
8. inches

Choose the unit to convert from (1-8): 1
Choose the unit to convert to (1-8): 2
Enter the value in meters: 1

1.0 meters ==> 0.0010000000 kilometers

Unit Conversion Menu:
1. Length
2. Mass
3. Time
0. Exit

Choose the type of conversion (1-3 or 0 to exit): 0

-----
|           Exiting the Unit Conversion module.           |
|-----|
```



```
Run - 11th_CS_Project
Scientific Calculator x

| Choose an option from one of the following: |
| 1. Simple Arithmetic      2. Unit Conversion |
| 3. Trigonometry          0. Exit (Stop the scientific calculator!) |
|-----|

Enter your choice: 3

*****
*          TRIGONOMETRY CALCULATOR          *
*****

Welcome! Let's calculate the value of trigonometric functions.

How to Use the Program:
1. Enter a number when prompted.
2. Choose the trig func of your choice.
3. Enter the unit of angle and the angle itself.

|-----|
| IMPORTANT: The tan of 90.0° is undefined, and not equal to 16331239353195370.000 |
|-----|

From the following, input a choice:
1. sin    2. cos    3. tan
Enter your choice : error

--> Invalid input! Please enter a number from 1-3 <--

Run - 11th_CS_Project
Scientific Calculator x

From the following, input a choice:
1. sin    2. cos    3. tan
Enter your choice : 1

From the following, input a choice:
1. Degree  2. Radian
Enter your choice : error

--> Invalid input! Please enter 1 for degree and 2 for radian <--

From the following, input a choice:
1. sin    2. cos    3. tan
Enter your choice : 1

From the following, input a choice:
1. Degree  2. Radian
Enter your choice : 1

Enter the angle : error

--> Invalid angle input! <--

From the following, input a choice:
1. sin    2. cos    3. tan
Enter your choice : 1

From the following, input a choice:
1. Degree  2. Radian
Enter your choice : 1

Enter the angle : 1
```

```
Run - 11th_CS_Project
Scientific Calculator x
1. Degree 2. Radian
Enter your choice : 1
Enter the angle : 1
The sin of 1.0° = 0.017
Do you want to perform another calculation? (y/n) n

-----
|   Exiting the Trigonometric module.   |
-----

-----
| Choose an option from one of the following: |
| 1. Simple Arithmetic      2. Unit Conversion |
| 3. Trigonometry          0. Exit (Stop the scientific calculator!) |
-----

Enter your choice: 0

=====
=====| SCIENTIFIC CALCULATOR |=====
=====| EXITING PROGRAM          |=====
=====|=====
=====|=====

Process finished with exit code 0
```

USER MANUAL

Note: Due to the difference in nature of font style and styling in Python's default IDE (IDLE) and PyCharm's IDE, it is recommended to always use the program on Pycharm. However, one may use the program using Python's default IDE as well. Please be aware that any ASCII art and styling done, may or may not look right when using Python's default IDE. Any other functionality whatsoever, will still work regardless of the IDE.

Please read through the following instructions carefully to ensure proper installation of PyCharm, and the program files:

1. Installing Python (Follow this step if you do not Python already installed in p.c., otherwise ignore this step)

- Visit this page to download the Python installer for your OS: <https://www.python.org/downloads/>
- Select Python version suitable for your OS (for eg. Python stopped support for Windows 7 with the last usable version being 3.8.15).
- Choose the correct link for your device from the options provided and proceed to download the .exe file.
- Once the download is complete, open the .exe file to launch the Python Installer Wizard. Follow the steps to complete the set-up process. If you are on Windows, you should select the box for 'Add python.exe to PATH'.
- When set-up is complete, search and open 'IDLE' from the Windows search bar. Here, confirm the version of Python you installed by reading the first line in the new window that opened.
- Once the version is confirmed, close the IDLE.

2. Installing PyCharm IDE

- Visit this page to download the installer for your OS:
<https://www.jetbrains.com/pycharm/download/?section=windows>
- Run the installer and follow the wizard steps.
Mind the following options in the installation wizard
 - i. 64-bit launcher: Adds a launching icon to the Desktop.
 - ii. Open Folder as Project: Adds an option to the folder context menu that will allow opening the selected directory as a PyCharm project.
 - iii. .py: Establishes an association with Python files to open them in PyCharm.
 - iv. Add launchers dir to the PATH: Allows running this PyCharm instance from the Console without specifying the path to it.
- Once the set-up is complete, open PyCharm by searching for it in the Windows search bar.

3. Set-up for Project Folder

- After opening PyCharm, start by creating a new project.
- Now, name your project folder as you wish, and create the project.
- In the menu to the left, right-click on your file name, and then 'Open in Associated Application' from 'Open in'.
- Now, wait in this window(i.e. Do not close this window) while we download the program files.

4. Installing the program files

- Head to this website where the program files are hosted:
https://github.com/BulkTornado/Computer_Science_Project_11thGrade
- Scroll down to read the Introduction, if you wish to.
- Otherwise, open the .zip folder.

- Now, press the following keys: Ctrl + Shift + s to download the zip folder.
- Once downloaded, extract all of the program files.
- Copy these program files, and head to the window that was opened from PyCharm, and paste it in this folder.
- Once done, close all of the windows, and open PyCharm.
- On the left menu, click on the small drop-down icon beside the 'venv' file, and double click on all of the new files that have appeared ending with the extension .py
- Now, head to any of the program file, head to the top menu and click on the little play button to start the program.
- A new small window will now open. In this specific window, on the top-right corner, click on the 'three dot' icon, 'view mode' and click on 'window'.

Congratulations, you have successfully installed all the programs, and now you can access any of them through PyCharm.

Once you are done playing around, click on 'File' top-left corner, and then 'Close Project'.

For more information on how to use PyCharm, refer to ANNEXURE - 1.

BRIEF BACKGROUND OF ORGANIZATION

...

REFERENCE

1. [school/ organization website]
2. https://www.youtube.com/playlist?list=PLZPZq0r_RZOOKUQbat8LyQii36cJf2SWT
3. https://en.wikibooks.org/wiki/Python_Programming
4. <https://wiki.python.org/moin/PythonBooks>
5. https://en.wikibooks.org/wiki/A_Beginner%27s_Python_Tutorial
6. [https://en.wikipedia.org/wiki/Python_\(programming_language\)](https://en.wikipedia.org/wiki/Python_(programming_language))
7. <https://www.geeksforgeeks.org/how-to-install-python-on-windows/>
8. <https://www.jetbrains.com/help/pycharm/installation-guide.html#standalone>

ANNEXURE - 1

If you wish to use PyCharm more for your programs, follow along.

Once you have created a new project, always click on the drop-down icon beside the 'venv' folder in the left menu.

To create a python file of your own, right click on your file name in the left menu. From here, click on 'New', and either 'File' or 'Python File'. In the new small window, name your python file. If you clicked on 'File', ensure to end the file name with the extension '.py'. For example, 'main.py'. You do not need to add the extension if you are creating the file from 'Python File'. Click on 'Enter' to confirm and create the program file.

At any point, if you wish to access the project file in your windows, just right-click on your file name, go to 'Open in' and then 'Explorer', to see all of the project files you have created with PyCharm.

If you wish to rename either your file name or your program name, always right click on them from the left menu in PyCharm, and click on 'Rename'. Renaming directly from explorer will result in PyCharm not being able to search for and execute the program.

If you wish to see a tutorial video, follow along with this one: <https://www.youtube.com/watch?v=Sg4GMVMdOPo&list=PLZPZq0rRZOOKUQbat8LyQii36cJf2SWT&index=1>

In the explorer menu, if you were to directly open the Python file, it will always open in Windows Command Prompt, which will automatically close upon completion of execution. Instead, right click on the Python file, and then click on 'Edit with IDLE' to open the file in Python's default IDE, IDLE.