VITYARTHI PROJECT

ENTRY-LEVEL SCIENTIFIC CALCULATOR
TYPE- PYTHON PROJECT

Dr Lokesh Malviya

BY- KUNAL SHARMA

25BCE11303

ABOUT THE PROJECT

An **entry-level scientific calculator** is an affordable, non-programmable electronic calculator that includes the **fundamental scientific functions** required for high school and introductory college-level math and science courses (like Algebra, Geometry, and Physics).

KEY FUNCTIONS OF ENTRY LEVEL SCIENTIFIC CALCULATOR

•**Trigonometry:** Sine (sin), Cosine (cos), Tangent (tan), and their inverses.
•**Exponents/Logarithms:** Powers (x^y), square/cube roots (sqrt{x}), common (log), and natural (ln) logarithms.
•**Scientific Notation:** A dedicated key (often **EXP** or **EE**) to easily enter and display very large or very small numbers.
•**Constants:** Quick access to (pi) and e (Euler's number).
•**Factorials:** The x! function, crucial for probability and statistics.

CHARACTERSTICS OF ENTRY LEVEL SCIENTIFIC CALCULATOR

•**Multi-line Display (Common):** Most modern entry-level models feature a two-line or multi-line display. The **top line** shows the full expression you are typing, and the **bottom line** shows the result.
 This makes checking your work much easier.
•**Order of Operations:** They strictly adhere to the correct mathematical order (e.g., PEMDAS/BODMAS).
•**Memory:** Multiple memory registers to store and recall numbers during multi-steP
 problems.

PYTHON CODE

```
import math

import numpy as np

class operations():

  def inverse_sin():

    return math.asin(input1)

 def inverse_cos():

          return math.acos(input1)

 def inverse_tan():

    if math.cos(input1)==0:

       return "invalid input"

  return math.atan(input1)

  def inverse_cot():

    # math.acot is not a standard function in math module, using 1/tan instead.

    if math.tan(input1)==0:

      return "invalid input"

    return 1/math.atan(input1)

def inverse_sec():

    # math.asec is not a standard function in math module, using 1/cos instead

    if math.cos(input1)==0:

.       return "invalid input“

   return 1/math.acos(input1)
```

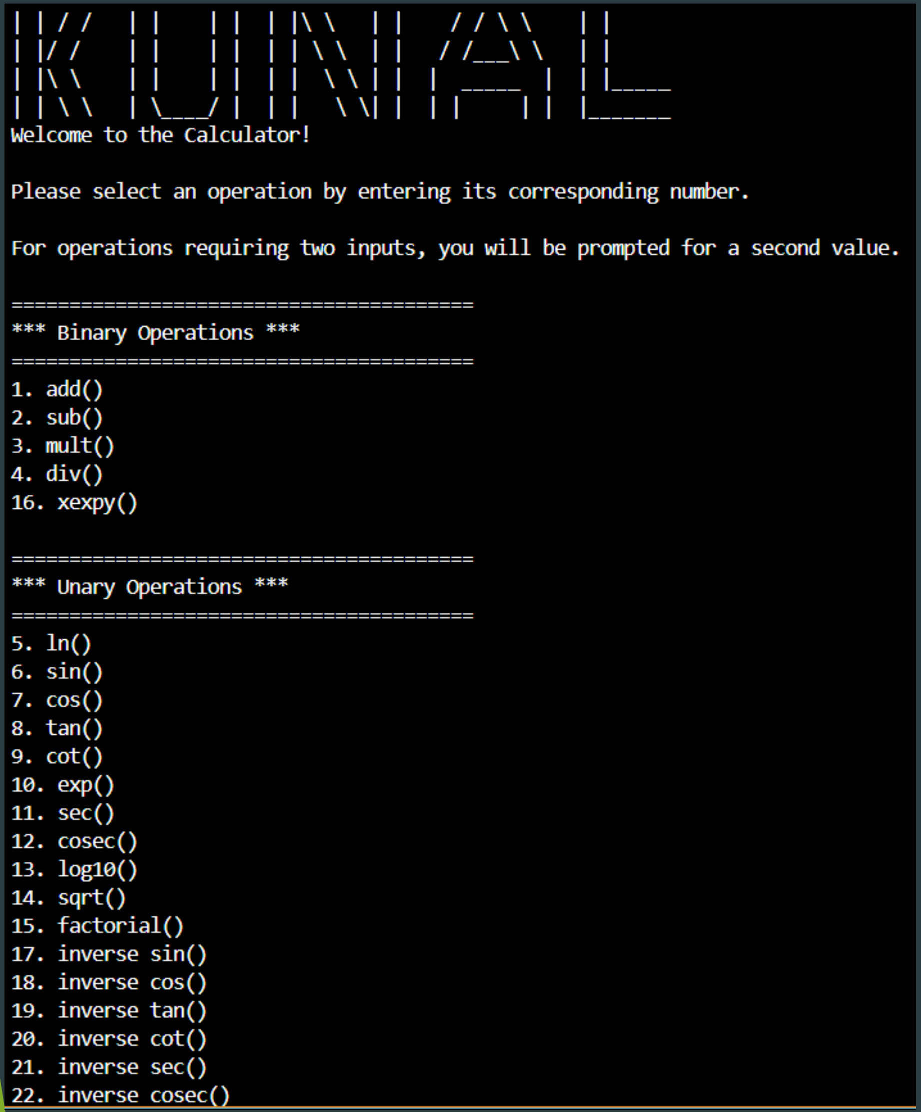
```
    if math.cos(input1)==0:
      return "invalid input"
    return 1/math.acos(input1)
  def inverse_cosec():
    # math.acosec is not a standard function in math module, using 1/sin
instead.
    if math.sin(input1)==0:
      return "invalid input"
    return 1/math.asin(1/input1)
  def xexpy():
    return input1**input2
 def factorial():
    num = int(input1) # Factorial is typically for integers
    if not input1.is_integer() or num < 0:
      return "invalid input as factorial is not defined for non-integers or
negatives"
    if num == 0:
      return 1
    res = 1
    for i in range(1, num + 1):
      res *= i
    return res
def sqrt():
    if input1<0:
      return "Invalid input. Cannot calculate square root of a negative number."
    return math.sqrt(input1)
def add():
    return input1 + input2


  def sub():
    return input1 - input2


  def mult():
    return input1 * input2


  def div():
    if input2 == 0:
      return "Division by zero is not allowed."
    return input1 / input2
```

```
def sin_op():
  return math.sin(input1)
def cos_op():

    return math.cos(input1)

 def tan_op():

    # Tangent is undefined

     if math.cos(input1) == 0 :

        return "Tangent undefined for input.“

     return math.tan(input1)

  def sec_op():

    if math.cos(input1) == 0 :

        return "Secant undefined for input."

    return 1 / math.cos(input1)

 def cosec_op():

    if math.sin(input1) == 0 :

        return "Cosecant undefined for input."

    return 1 / math.sin(input1)

  def cot_op():

    if math.tan(input1) == 0 : # cot is 1/tan, tan is 0
when sin is 0

        return "Cotangent undefined for input."

    return 1 / math.tan(input1)

  def ln_op():

    if input1 <= 0 :

        return "Logarithm undefined for non-positive
input(s)."

    return np.log(input1)
```

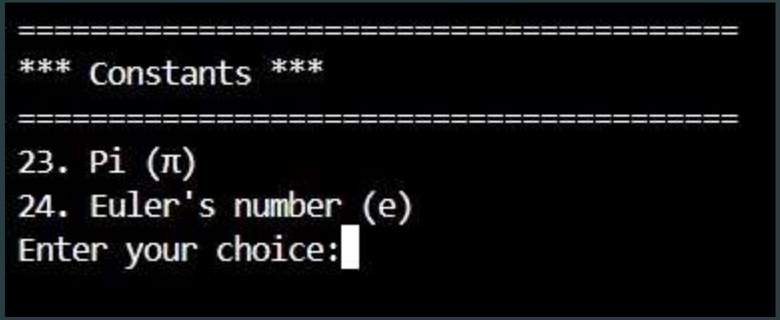
Continue…………

```
 def log10_op():
   if input1 <= 0 :
       return "Logarithm base 10 undefined for non-positive
input(s)."
   return np.log10(input1)
 def exp_op():
   return np.exp(input1)
# Display menu
print("""
| | / /   | |    | |  | |\ \   | |   / / \ \      | |
| |/ /    | |    | |  | | \ \  | |  / /___\ \     | |
| |\ \   | |    | |  | | \ \ | |  |  _____  |   | |_____
| | \ \  | \__/  |   | |   \ \| |  | |        | |  |________    """)
print("Welcome to the Calculator!\n")
print("Please select an operation by entering its
corresponding number.\n")
print("For operations requiring two inputs, you will be
prompted for a second value.\n")
print("=" * 40)
print("*** Binary Operations ***")
print("=" * 40)
print("1. add()")
print("2. sub()")
print("3. mult()")
print("4. div()")
print("16. xexpy()")
print("\n" + "=" * 40)
print("*** Unary Operations ***")
print("=" * 40)
print("5. ln()")
print("6. sin()")
print("7. cos()")
print("8. tan()")
print("9. cot()")
print("10. exp()")
print("11. sec()")
print("12. cosec()")
print("13. log10()")
print("14. sqrt()")
print("15. factorial()")
print("17. inverse sin()")
print("18. inverse cos()")
print("19. inverse tan()")
print("20. inverse cot()")
print("21. inverse sec()")
print("22. inverse cosec()")

print("\n" + "=" * 40)
print("*** Constants ***")
print("=" * 40)
print("23. Pi (\u03c0)")
print("24. Euler's number (e)")
# --- Calculator execution flow ---
binary_operations = [1, 2, 3, 4, 16]
unary_operations = [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17,
18, 19, 20, 21, 22]
constant_operations = [23, 24]
# Get user choice without error checking
choice = int(input("Enter your choice:"))
# Input handling based on operation type
if choice in unary_operations or choice in
binary_operations:
    input1 = float(input("Enter first input:"))
    if choice in binary_operations:
        input2 = float(input("Enter second input:"))
elif choice in constant_operations:
    pass # No input needed for constants
else:
    print("Invalid choice. Please select a number from 1 to
24.")
# Execute chosen operation
if choice == 1:
  print(operations.add())
elif choice == 2:
  print(operations.sub())
elif choice == 3:
  print(operations.mult())
elif choice == 4:
  print(operations.div())
elif choice == 5:
  print(operations.ln_op())
elif choice == 6:
  print(operations.sin_op())
elif choice == 7:
  print(operations.cos_op())
elif choice == 8:
  print(operations.tan_op())
elif choice == 9:
  print(operations.cot_op())
elif choice == 10:
  print(operations.exp_op())
elif choice == 11:
  print(operations.sec_op())
elif choice == 12:
  print(operations.cosec_op())
elif choice == 13:
  print(operations.log10_op())
elif choice==14:
  print(operations.sqrt())
elif choice==15:
  print(operations.factorial())
elif choice==16:
  print(operations.xexpy())
elif choice==17:
  print(operations.inverse_sin())
elif choice==18:
  print(operations.inverse_cos())
elif choice==19:
  print(operations.inverse_tan())
elif choice==20:
  print(operations.inverse_cot())
elif choice==21:
  print(operations.inverse_sec())
elif choice==22:
  print(operations.inverse_cosec(
))
elif choice==23:
  print(math.pi)
elif choice==24:
  print(math.e)
else:
  print("Invalid choice. Please
select a number from 1 to 24.")
```

Output on running code

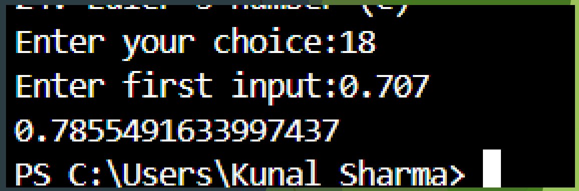


```
| | / /  | |   | | | |\ \ | |   / / \ \   | |
| |/ /   | |   | | | | \ \| |  / /___\ \  | |
| |\ \   | |   | | | |  \ \ | | _____ |  | |____
| | \ \  | \___/ | | |   \ | | |      | |  |______
Welcome to the Calculator!

Please select an operation by entering its corresponding number.

For operations requiring two inputs, you will be prompted for a second value.

========================================
*** Binary Operations ***
========================================
1. add()
2. sub()
3. mult()
4. div()
16. xexpy()

========================================
*** Unary Operations ***
========================================
5. ln()
6. sin()
7. cos()
8. tan()
9. cot()
10. exp()
11. sec()
12. cosec()
13. log10()
14. sqrt()
15. factorial()
17. inverse sin()
18. inverse cos()
19. inverse tan()
20. inverse cot()
21. inverse sec()
22. inverse cosec()
```



```
========================================
*** Constants ***
========================================
23. Pi (π)
24. Euler's number (e)
Enter your choice:
```

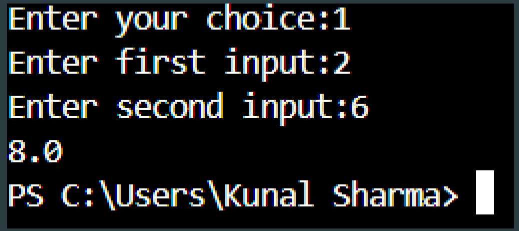
UNARY OPERATIONS



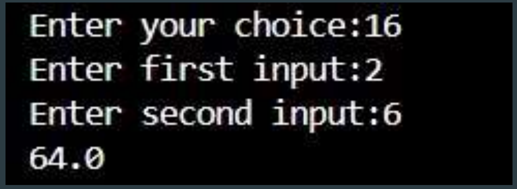
```
Enter your choice:18
Enter first input:0.707
0.7855491633997437
PS C:\Users\Kunal Sharma>
```

CHOICE AND INPUTS GIVES OUTPUT

BINARY OPERATIONS

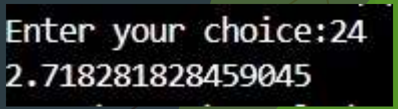


```
Enter your choice:1
Enter first input:2
Enter second input:6
8.0
PS C:\Users\Kunal Sharma>
```

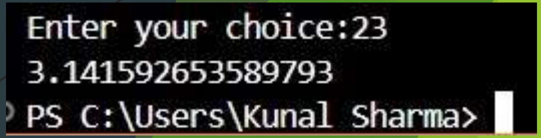


```
Enter your choice:16
Enter first input:2
Enter second input:6
64.0
```

CONSTANTS



```
Enter your choice:24
2.718281828459045
```



```
Enter your choice:23
3.141592653589793
PS C:\Users\Kunal Sharma>
```

Explaination

This code implements a command-line calculator program using the `math` and `numpy` libraries.

It defines a class `operations` which contains methods for various mathematical calculations, including binary operations (like addition, subtraction, multiplication, division, and exponentiation),
unary operations (like logarithms, trigonometric functions, square root, and factorial),
and inverse trigonometric functions. The program first displays a menu of these operations to the user.
Based on the user's choice, it prompts for the necessary inputs (one or two numbers) and
then performs the selected calculation using the corresponding method from the `operations` class.
Finally, it prints the result of the calculation.

CONCEPTS COVERED

This Python code covers several key concepts:

1.**Object-Oriented Programming (OOP)**: It uses a `class` named `operations` to encapsulate related functionalities (mathematical operations) as methods.

2.**Functions/Methods**: Each mathematical operation (e.g., `add`, `sin_op`, `factorial`) is implemented as a method within the `operations` class.

3.**Conditional Logic**: `if-elif-else` statements are used extensively for decision-making, such as handling invalid inputs (e.g., square root of negative numbers,
division by zero, undefined tangents),
selecting operations based on user choice, and defining base cases for recursive functions like `factorial`.

4.**Input/Output Operations**: The program interacts with the user to take inputs using `input()` and displays results using `print()`.

5.**Mathematical Operations**: It demonstrates a wide range of mathematical operations including:

- **Basic Arithmetic**: Addition, subtraction, multiplication, division.
- **Exponents**: `xexpy()` for `x^y`.
- **Trigonometry**: Sine, cosine, tangent, secant, cosecant, cotangent.
- **Inverse Trigonometry**: Inverse sine, cosine, tangent, cotangent, secant, cosecant.
- **Logarithms**: Natural logarithm (`ln`) and base-10 logarithm (`log10`).
- **Exponential Function**: `exp_op()` for e^x.
- **Square Root**: `sqrt()`.
- **Factorial**: `factorial()`.

5.**Error Handling (Basic)**: Although the explicit `try-except` blocks were removed, the code still includes conditional checks to prevent mathematical errors
like division by zero or taking the square root of a negative number.

6.**Libraries**: It extensively uses the `math` and `numpy` libraries for mathematical functions and operations.

7.**Global Variables**: `input1` and `input2` are used as global variables to pass data between the main execution flow and the class methods.

CONCLUSION

THANK YOU