Python calculator logic documentation:

Files:

python cf.py

python test\_run.py

June 30, 2021

* Seems like virtual environment can run built in packages of python
* Eval() can’t run trigonometry
* Eval() can run floor division, exponent and modulus
* Eval() is currently having a float error due to the inclusion of decimals in the equation.

Functions: (pretend that you are pressing the calculator, so one() makes input 1, division() inputs / etc..)

calculate()

clear\_all()

addition()

subtraction()

division()

multiplication()

exponent()

negative\_sign()

parenthesis\_open()

parenthesis\_close()

one()

two()

three()

four()

five()

six()

seven()

eight()

nine()

zero()

July 5, 2021

* Eval() can now run trigonometry

def examine\_trigo() -> None:

    """This function is used to replace all trigo function in the equation to their answer"""

    find\_asin()

    find\_acos()

    find\_atan()

    find\_sin()

    find\_cos()

    find\_tan()

def find\_sin() -> None:

    """This converts the sin in the equation to numerical value"""

    global equation, syntax\_trigonometry

    while "sin" in equation:

        i = equation.index("sin")

        num = find\_number\_trigo(i)

        if num:

            ans = round(math.sin(math.radians(float(num))),10)

            equation = equation[:i] + "the" + equation[i:]

            replace\_word = f"thesin({num})"

            equation = equation.replace(replace\_word, str(ans))

            replace\_word = f"thesin{num}"

            equation = equation.replace(replace\_word, str(ans))

        else:

            replace\_word = "sin"

            equation = equation.replace(replace\_word, "")

            syntax\_trigonometry += 1

def find\_cos() -> None:

    "This converts the cos in the equation to numerical value"

    global equation, syntax\_trigonometry

    while "cos" in equation:

        i = equation.index("cos")

        num = find\_number\_trigo(i)

        if num:

            ans = round(math.cos(math.radians(float(num))), 10)

            equation = equation[:i] + "the" + equation[i:]

            replace\_word = f"thecos({num})"

            equation = equation.replace(replace\_word, str(ans))

            replace\_word = f"thecos{num}"

            equation = equation.replace(replace\_word, str(ans))

        else:

            replace\_word = "cos"

            equation = equation.replace(replace\_word, "")

            syntax\_trigonometry += 1

def find\_tan() -> None:

    "This converts the tan in the quation to numerical value"

    global equation, syntax\_trigonometry

    while "tan" in equation:

        i = equation.index("tan")

        num = find\_number\_trigo(i)

        if num:

            ans = round(math.tan(math.radians(float(num))), 10)

            equation = equation[:i] + "the" + equation[i:]

            replace\_word = f"thetan({num})"

            equation = equation.replace(replace\_word, str(ans))

            replace\_word = f"thetan{num}"

            equation = equation.replace(replace\_word, str(ans))

        else:

            replace\_word = "tan"

            equation = equation.replace(replace\_word, "")

            syntax\_trigonometry += 1

def find\_asin() -> None:

    """This converts the arcsin in the equation to numerical value"""

    global equation, syntax\_trigonometry, syntax\_inversetrigonometryn

    while "arcsin" in equation:

        i = equation.index("arcsin")

        num = find\_number\_trigo(i)

        if num:

            try:

                ans = math.asin(float(num))

                equation = equation[:i] + "the" + equation[i:]

                replace\_word = f"thearcsin({num})"

                equation = equation.replace(replace\_word, str(math.degrees(ans)))

                replace\_word = f"thearcsin{num}"

                equation = equation.replace(replace\_word, str(math.degrees(ans)))

            except:

                syntax\_trigonometry += 1

        else:

            replace\_word = "arcsin"

            equation = equation.replace(replace\_word, "")

            syntax\_trigonometry += 1

def find\_acos() -> None:

    """This converts the arccos in the equation to numerical value"""

    global equation, syntax\_trigonometry, syntax\_inversetrigonometryn

    while "arccos" in equation:

        i = equation.index("arccos")

        num = find\_number\_trigo(i)

        if num:

            try:

                ans = math.acos(float(num))

                equation = equation[:i] + "the" + equation[i:]

                replace\_word = f"thearccos({num})"

                equation = equation.replace(replace\_word, str(math.degrees(ans)))

                replace\_word = f"thearccos{num}"

                equation = equation.replace(replace\_word, str(math.degrees(ans)))

            except:

                syntax\_trigonometry += 1

        else:

            replace\_word = "arccos"

            equation = equation.replace(replace\_word, "")

            syntax\_trigonometry += 1

def find\_atan() -> None:

    """This converts the arctan in the equation to numerical value"""

    global equation, syntax\_trigonometry, syntax\_inversetrigonometryn

    while "arctan" in equation:

        i = equation.index("arctan")

        num = find\_number\_trigo(i)

        if num:

            try:

                ans = math.atan(float(num))

                equation = equation[:i] + "the" + equation[i:]

                replace\_word = f"thearctan({num})"

                equation = equation.replace(replace\_word, str(math.degrees(ans)))

                replace\_word = f"thearctan{num}"

                equation = equation.replace(replace\_word, str(math.degrees(ans)))

            except:

                syntax\_trigonometry += 1

        else:

            replace\_word = "arctan"

            equation = equation.replace(replace\_word, "")

            syntax\_trigonometry += 1

def find\_number\_trigo(i: int) -> str:

    """to extract the number from a trigo expression;i is the index of the expression"""

    global equation

    loop = True

    while loop:

        n = ""

        try:

            data = float(equation[i])

        except:

            data = "str"

        if data != "str":

            while True:

                try:

                    data = float(equation[i])

                except:

                    if i > len(equation) - 1:

                        break

                    if equation[i] == ".":

                        n += equation[i]

                        i += 1

                    elif equation[i] == "﹣":

                        n += equation[i]

                        i += 1

                    else:

                        loop = False

                        break

                n += equation[i]

                i += 1

        i += 1

        if i > len(equation) - 1:

            break

    return n

* Eval() can also run logarithmic functions like e and ln.

def examine\_logarithm() -> None:

    convert\_e()

    find\_ln()

    find\_log()

def convert\_e() -> None:

    "This converts the tan in the quation to numerical value"

    global equation

    while "e" in equation:

        ans = math.e

        equation = equation.replace("e", str(ans))

def find\_ln() -> None:

    "This converts the tan in the quation to numerical value"

    global equation,syntax\_inversetrigonometry

    while "ln" in equation:

        i = equation.index("ln")

        num = find\_number\_ln(i)

        if num:

            try:

                ans = round(math.log(float(num)), 10)

            except:

                syntax\_inversetrigonometry += 1

            equation = equation[:i] + "the" + equation[i:]

            replace\_word = f"theln({num})"

            equation = equation.replace(replace\_word, str(ans))

            replace\_word = f"theln{num}"

            equation = equation.replace(replace\_word, str(ans))

        else:

            syntax\_inversetrigonometry += 1

            replace\_word = "ln"

            equation = equation.replace(replace\_word, "")

def find\_log() -> None:

    "This converts the tan in the quation to numerical value"

    global equation,syntax\_inversetrigonometry

    while "log" in equation:

        i = equation.index("log")

        num = find\_number\_ln(i)

        if num:

            try:

                ans = round(math.log10(float(num)), 10)

            except:

                syntax\_inversetrigonometry += 1

            equation = equation[:i] + "the" + equation[i:]

            replace\_word = f"thelog({num})"

            equation = equation.replace(replace\_word, str(ans))

            replace\_word = f"thelog{num}"

            equation = equation.replace(replace\_word, str(ans))

        else:

            syntax\_inversetrigonometry += 1

            replace\_word = "log"

            equation = equation.replace(replace\_word, "")

def find\_number\_ln(i: int) -> str:

    """to extract the number from an ln expression

    i is the index of the expression"""

    global equation

    loop = True

    while loop:

        n = ""

        try:

            data = float(equation[i])

        except:

            data = "str"

        if data != "str":

            while True:

                try:

                    data = float(equation[i])

                except:

                    if i > len(equation) - 1:

                        break

                    if equation[i] == ".":

                        n += equation[i]

                        i += 1

                    elif equation[i] == "﹣":

                        n += equation[i]

                        i += 1

                    else:

                        loop = False

                        break

                n += equation[i]

                i += 1

        i += 1

        if i > len(equation) - 1:

            break

    return n

def examine\_sqrt() -> None:

    """This function converts sqrt into numerical values"""

    global equation

    while "√" in equation:

        i = equation.index("√")

        num = find\_number\_ln(i)

        if num:

            try:

                ans = round(math.sqrt(float(num)), 12)

            except:

                syntax\_inversetrigonometry += 1

            equation = equation[:i] + "the" + equation[i:]

            replace\_word = f"the√({num})"

            equation = equation.replace(replace\_word, str(ans))

            replace\_word = f"the√{num}"

            equation = equation.replace(replace\_word, str(ans))

        else:

            syntax\_inversetrigonometry += 1

            replace\_word = "√"

            equation = equation.replace(replace\_word, "")

#--------------------------For the exponent and negative sign symbols-------------

def exponent\_negativesign\_convert():

    global equation

    if "^" or "﹣" in equation:

        equation = equation.replace("^", "\*\*")

        equation = equation.replace("﹣", "-")

#---------------------------For the ANS button---------------------

def find\_ans() -> None:

    global equation

    while "ans" in equation:

        i = equation.index("ans")

        equation = equation[:i] + "the" + equation[i:]

        equation = equation.replace("theans", answer\_button)

#---------------------------------Functions for inputing numbers --------------------

def one():

    global equation

    """Adding number 5 to the equation"""

    equation += "1"

def two():

    global equation

    """Adding number 5 to the equation"""

    equation += "2"

def three():

    global equation

    """Adding number 5 to the equation"""

    equation += "3"

def four():

    global equation

    """Adding number 5 to the equation"""

    equation += "4"

def five():

    global equation

    """Adding number 5 to the equation"""

    equation += "5"

def six():

    global equation

    """Adding number 5 to the equation"""

    equation += "6"

def seven():

    global equation

    """Adding number 5 to the equation"""

    equation += "7"

def eight():

    global equation

    """Adding number 5 to the equation"""

    equation += "8"

def nine():

    global equation

    """Adding number 5 to the equation"""

    equation += "9"

def zero():

    global equation

    """Adding number 5 to the equation"""

    equation += "0"