

**Name:** Muhammad Faizan

**Roll No:** 036

**Section:** BSDS-3A

**Code:**

def calculation(expression\_from\_user = ""):

if expression\_from\_user == "":

expression\_from\_user = input("Please enter an expression: ")

expression\_from\_user = expression\_from\_user.replace(" ","")

print("expression after cleaning", expression\_from\_user)

while "(" in expression\_from\_user:

position\_of\_left\_bracket = expression\_from\_user.rfind("(")

position\_of\_right\_bracket = expression\_from\_user.find(")", position\_of\_left\_bracket)

stuff\_inside\_brackets = expression\_from\_user[position\_of\_left\_bracket+1:position\_of\_right\_bracket]

value\_inside\_brackets = calculation(stuff\_inside\_brackets)

expression\_from\_user = expression\_from\_user[:position\_of\_left\_bracket] + str(value\_inside\_brackets) + expression\_from\_user[position\_of\_right\_bracket+1:]

list\_of\_numbers\_and\_operators = []

number\_being\_built = ""

for single\_character in expression\_from\_user:

if single\_character.isdigit() or single\_character == ".":

number\_being\_built = number\_being\_built + single\_character

else:

if number\_being\_built != "":

list\_of\_numbers\_and\_operators.append(float(number\_being\_built))

list\_of\_numbers\_and\_operators.append(single\_character)

number\_being\_built = ""

if number\_being\_built != "":

list\_of\_numbers\_and\_operators.append(float(number\_being\_built))

current\_index\_for\_mul\_div = 0

while current\_index\_for\_mul\_div < len(list\_of\_numbers\_and\_operators):

if list\_of\_numbers\_and\_operators[current\_index\_for\_mul\_div] == "\*":

list\_of\_numbers\_and\_operators[current\_index\_for\_mul\_div-1] = list\_of\_numbers\_and\_operators[current\_index\_for\_mul\_div-1] \* list\_of\_numbers\_and\_operators[current\_index\_for\_mul\_div+1]

del list\_of\_numbers\_and\_operators[current\_index\_for\_mul\_div:current\_index\_for\_mul\_div+2]

elif list\_of\_numbers\_and\_operators[current\_index\_for\_mul\_div] == "/":

list\_of\_numbers\_and\_operators[current\_index\_for\_mul\_div-1] = list\_of\_numbers\_and\_operators[current\_index\_for\_mul\_div-1] / list\_of\_numbers\_and\_operators[current\_index\_for\_mul\_div+1]

del list\_of\_numbers\_and\_operators[current\_index\_for\_mul\_div:current\_index\_for\_mul\_div+2]

else:

current\_index\_for\_mul\_div = current\_index\_for\_mul\_div + 1

current\_index\_for\_plus\_minus = 0

while current\_index\_for\_plus\_minus < len(list\_of\_numbers\_and\_operators):

if list\_of\_numbers\_and\_operators[current\_index\_for\_plus\_minus] == "+":

list\_of\_numbers\_and\_operators[current\_index\_for\_plus\_minus-1] = list\_of\_numbers\_and\_operators[current\_index\_for\_plus\_minus-1] + list\_of\_numbers\_and\_operators[current\_index\_for\_plus\_minus+1]

del list\_of\_numbers\_and\_operators[current\_index\_for\_plus\_minus:current\_index\_for\_plus\_minus+2]

elif list\_of\_numbers\_and\_operators[current\_index\_for\_plus\_minus] == "-":

list\_of\_numbers\_and\_operators[current\_index\_for\_plus\_minus-1] = list\_of\_numbers\_and\_operators[current\_index\_for\_plus\_minus-1] - list\_of\_numbers\_and\_operators[current\_index\_for\_plus\_minus+1]

del list\_of\_numbers\_and\_operators[current\_index\_for\_plus\_minus:current\_index\_for\_plus\_minus+2]

else:

current\_index\_for\_plus\_minus = current\_index\_for\_plus\_minus + 1

if len(list\_of\_numbers\_and\_operators) > 0:

return list\_of\_numbers\_and\_operators[0]

else:

return 0

while True:

user\_input\_from\_terminal = input("Enter expression or end to quit: ")

if user\_input\_from\_terminal.lower() == "end":

print("closed")

break

try:

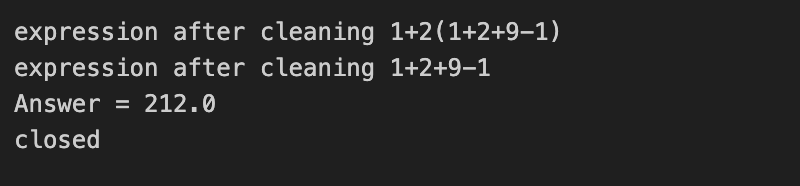
final\_answer\_after\_calculation = calculation(user\_input\_from\_terminal)

print("Answer =", final\_answer\_after\_calculation)

except Exception as error\_message\_that\_happened:

print("error", error\_message\_that\_happened)

**Output:**



**Explanation:**

The code starts with a function called calculation(expression=None). If you don't provide any expression, it asks you to type something by using input("Enter calculation: ").

**Removing Spaces**

One of the first things the function does is remove any spaces from the expression with this line:

expression = expression.replace(" ", "")

So, if you type "2 + 3", it will turn into "2+3" by stripping out the spaces.

**Handling Parentheses**

Then, it checks if there are any parentheses in the expression. If there are, it looks for the last opening parenthesis ( and finds the first closing parenthesis ) that follows it.

It grabs the part inside the parentheses and calls the calculation function again to solve that part first. Once it's solved, it replaces that part of the original expression with the result. Here's how it looks in code:

while "(" in expression:

open\_idx = expression.rfind("(")

close\_idx = expression.find(")", open\_idx)

inside = expression[open\_idx+1:close\_idx]

result = calculation(inside)

expression = expression[:open\_idx] + str(result) + expression[close\_idx+1:]

**Breaking Down the Expression**

Next, the code breaks the expression into numbers and operators. It loops through each character in the expression, checking if it's a number (or a dot for decimals). If it is, it keeps adding it to a variable num. When it reaches an operator, it adds the number to a list called tokens, then adds the operator, and resets num to start collecting the next number.

Here's that loop:

tokens = []

num = ""

for ch in expression:

if ch.isdigit() or ch == ".":

num += ch

else:

tokens.append(float(num))

tokens.append(ch)

num = ""

tokens.append(float(num))

By the end of this part, all the numbers and operators are stored in the tokens list.

**Solving Multiplication and Division First**

The next part of the code handles the actual math. It first looks for multiplication (\*) and division (/) in the tokens list and solves them. After that, it solves addition (+) and subtraction (-). Here's how it works:

for ops in [["\*", "/"], ["+", "-"]]:

i = 0

while i < len(tokens):

if tokens[i] in ops:

if tokens[i] == "\*":

tokens[i-1] \*= tokens[i+1]

elif tokens[i] == "/":

tokens[i-1] /= tokens[i+1]

elif tokens[i] == "+":

tokens[i-1] += tokens[i+1]

elif tokens[i] == "-":

tokens[i-1] -= tokens[i+1]

del tokens[i:i+2]

else:

i += 1

It first loops through multiplication and division, solving them in the order they appear. Afterward, it moves on to addition and subtraction. The del tokens[i:i+2] line removes the operator and the number that was used in the calculation.

**Returning the Result**

Finally, the function returns the result by giving back the first (and only) element in the tokens list, which should now be the final number after all the operations:

return tokens[0]

**The Main Loop**

Outside of the calculation function, there’s a loop that keeps asking you for calculations:

while True:

expr = input("Type math (or 'end' to quit): ")

if expr.lower() == "end":

break

try:

print(calculation(expr))

except:

print("Error")

This loop keeps running until you type "end". If you enter anything else, it tries to calculate the result and print it out. If something goes wrong, it catches the error and just prints "Error" instead of crashing.