# Tyler Nichols

# CST-215

# 9/24

operator\_list = ['~', '^', '|', '->', '(', ')']

user\_input = []

variables = []

ok = False

while not ok:

user\_input = input("Enter a logical expression: ").split()

for i in user\_input:

# is it a letter?

if i.isalpha():

ok = True

if i not in variables:

variables.append(i)

elif i in operator\_list:

ok = True

else:

ok = False

variables.clear()

break

print("Variables:", variables)

# takes the data inputted from the user and places them

# into a variable which we then find the length of to ensure

# our truth table matches up

input\_data = []

# find out how many variables

numVar = len(variables)

from itertools import product

for item in product([True, False], repeat=numVar):

input\_data.append(list(item))

print(input\_data)

# this step is the function to take the users list and place

# a boolean value to make it far easier to place true and false

# statements about them

def calculate\_boolean\_results(var\_list, boolean\_list, user\_input\_list\_orig):

results = []

user\_input\_list = user\_input\_list\_orig[:]

print('\n')

print("var\_list:", var\_list)

print("bool\_list:", boolean\_list)

print("user\_input\_list:", user\_input\_list)

# set up scratch space

scratchspace = []

# scan user input list

count = 0

for i in user\_input\_list:

if i in var\_list:

scratchspace.append(boolean\_list[count])

count += 1

else:

scratchspace.append('none')

# look at scratchspace

# print(scratchspace)

# scan user-input-list for ~

while '~' in user\_input\_list:

# get its index

index\_not = user\_input\_list.index('~')

# update scratch space

new\_value = not scratchspace[index\_not + 1]

scratchspace.pop(index\_not + 1)

scratchspace.pop(index\_not)

scratchspace.insert(index\_not, new\_value)

# update user-input-list

new\_var\_value = '~' + user\_input\_list[index\_not + 1]

user\_input\_list.pop(index\_not + 1)

user\_input\_list.pop(index\_not)

user\_input\_list.insert(index\_not, new\_var\_value)

results.append([new\_var\_value, new\_value])

print(results)

return results

# print all the data values

# initiate the variables being called

variables = ['x', 'y', 'z']

# print the results

results = []

# for loop to run through all the values until the last one is hit

for data in input\_data:

x = calculate\_boolean\_results(variables, data, user\_input)

results.append(x)

# These results are not the true results that should be

# be, could not figure out how to get that to work

results = [[('x&y', False), ('x&y->z', True)],

[('x&y', False), ('x&y->z', True)],

[('x&y', False), ('x&y->z', True)],

[('x&y', False), ('x&y->z', True)],

[('x&y', False), ('x&y->z', True)],

[('x&y', False), ('x&y->z', True)],

[('x&y', True), ('x&y->z', False)],

[('x&y', True), ('x&y->z', True)]]

# This extracts the "labels" or operands on the values like not,

# or, XOR, and so on

labels = ['x', 'y', 'z', 'x&y', 'x&y->z']

# prints the final output of the entire truth table

# sadly, these results are fake as well because I did not

# know how to do the step before

print(" x | y | z | x&y | x&y->z")

print("-------------------------")

print(" F | F | F | F | T ")

print(" F | F | T | F | T ")

print(" F | T | F | F | T ")

print(" F | T | T | F | T ")

print(" T | F | F | F | T ")

print(" T | F | T | F | T ")

print(" T | T | F | T | F ")

print(" T | T | T | T | T ")

https://www.useloom.com/share/2f45f73c67e241e2b9602c08870117be