In [ ]:
```
#SW2.1
#Karim Bangcola Jr.
#Sept 15
```

In [81]:
```
#evaluator.py

print("Propositional Logic Evaluator for discrete math for 2-3 variables")

variables = int(input("How many variables?"))
total_combinations = 2 ** variables

combinations_list = []

# Generate the combinations
for i in range(total_combinations):
    bin_equivalent = bin(i)[2:]
    while len(bin_equivalent) < variables:
        bin_equivalent = "0"+bin_equivalent
    combinations_list.append(tuple(int(val) for val in bin_equivalent))

#print(combinations_list)

# Main Program
expression = input("Enter the propositional logic expression: ")

if variables == 2:
    print("A B f")
    for A, B in combinations_list:
        evaluated_expression = eval(expression)
        print(A, B, evaluated_expression)
elif variables == 3:
    print("A B C f")
    for A, B, C in combinations_list:
        evaluated_expression = eval(expression)
        print(A, B, C, evaluated_expression)
```

```
Propositional Logic Evaluator for discrete math for 2-3 variables
A B f
0 0 True
0 1 True
1 0 True
1 1 1
```

In [8]:
```
#Using the open function for file handling

#filewriter.py

name = "Ken Bangcola"  # Modify by writing your name here.
file = open("newfile1.txt", 'w')
file.write(f"Hello, {name}!\n")
file.write("Isn't this amazing?\n")
file.write("that we can create and write on text files\n")
file.write("using Python.")
file.close()
```

```
#creates a new txt file named newfile1.txt with the content of the write fur

file = open("newfile2.txt", 'w')
file.write("This message was created using Python!\n")
file.close()

#creates a new txt file named newfile2.txt with different content
```

In [12]:
```
#filereader.py

file = open("newfile2.txt", 'r')
data = file.read()
print(data)
file.close()
```

```
This message was created using Python!
```

In [16]:
```
#change to read only the first 12 characters

#filereader.py

file = open("newfile2.txt", 'r')
data = file.read(12)
print(data)
file.close()
```

```
This message
```

In [18]:
```
#fileappender.py

file = open("newfile2.txt", 'a')
file.write("and also by the programmer, of course.")
file.close()

#appends to the already existing file

file = open("newfile2.txt", 'r')
data = file.read()
print(data)
file.close()
```

```
This message was created using Python!
and also by the programmer, of course.
```

In [26]:
```
#truthtablegenerator.py

def generate_truthtable(number_of_variables):
    total_combinations = 2 ** number_of_variables
    combinations_list = []

    for i in range(total_combinations):
        bin_equivalent = bin(i)[2:]

        while len(bin_equivalent) < number_of_variables:
            bin_equivalent = "0" + bin_equivalent
```

```
        combinations_list.append(tuple(int(val) for val in bin_equivalent))

    return combinations_list

print(generate_truthtable(3))

#we generate a list with 2**variable elements. each element is a possible co
#1 and 0 either 2 or 3 times in 1 element
```

```
[(0, 0, 0), (0, 0, 1), (0, 1, 0), (0, 1, 1), (1, 0, 0), (1, 0, 1), (1, 1,
0), (1, 1, 1)]
```

In [30]:
```
#truuthtablegenerator.py

#version 2

def generate_truthtable(number_of_variables=0):
    if number_of_variables == 0:
        return "You need to enter an integer"
    else:
        total_combinations = 2 ** number_of_variables
        combinations_list = []

        for i in range(total_combinations):
            bin_equivalent = bin(i)[2:]
            while len(bin_equivalent) < number_of_variables:
                bin_equivalent = "0" + bin_equivalent
            combinations_list.append(tuple(int(val) for val in bin_equivaler

        return combinations_list

print(generate_truthtable())

#we set the default paramter to 0 and added handling to exit with error if n
```

```
You need to enter an integer
```

In [83]:
```
#truthtablegenerator.py

def generate_truthtable(number_of_variables=0):
    if number_of_variables == 0:
        return "You need to enter an integer"
    else:
        total_combinations = 2 ** number_of_variables
        combinations_list = []

        for i in range(total_combinations):
            bin_equivalent = bin(i)[2:]
            while len(bin_equivalent) < number_of_variables:
                bin_equivalent = "0" + bin_equivalent
            combinations_list.append(tuple(int(val) for val in bin_equivaler

        return combinations_list

#new function created

def evaluate_propositional_logic(combinations_list):
```

```
    expression = input("Enter the propositional logic expression: ")

    if len(combinations_list) == 4: #changed from variable == 2 since we no
        print("A B f")
        for A, B in combinations_list:
            evaluated_expression = eval(expression)
            print(A, B, evaluated_expression)
    elif len(combinations_list) == 8: #changed from variable == 3 since we n
        print("A B C f")
        for A, B, C in combinations_list:
            evaluated_expression = eval(expression)
            print(A, B, C, evaluated_expression)

evaluate_propositional_logic(generate_truthtable(2))

#there is no need to use the print function when calling evaluate_propositio
#because it already calls print within the function

#it is better to define and use functions instead of writing code only seque
#as it 1) allows us to perform the same tasks repeatedly without copying the
#2) allows us to make our program go non-linearly - it may perform different
#instead of just following the set instructions from top to bottom
```

```
A B f
0 0 True
0 1 True
1 0 True
1 1 1
```

In [34]:

```
#modules1/mathmodule.py

import math

def quadratic_formula(a, b, c):
    if b**2 - (4*a*c) < 0:
        x1 = (complex(-b, math.floor(math.sqrt(abs(b**2-(4*a*c)))))) / (2*a)
        x2 = (complex(-b, -1*math.floor(math.sqrt(abs(b**2-(4*a*c)))))) / (2
        return x1, x2
    else:
        x1 = (-b + math.sqrt(b**2 - (4*a*c))) / (2*a)
        x2 = (-b - math.sqrt(b**2 - (4*a*c))) / (2*a)
        return x1, x2

print(quadratic_formula(1,2,3))
```

```
((-1+1j), (-1-1j))
```

In [36]:

```
#mathmodule2.py

import math

def angle_demo():
    angle = math.sin(math.pi/2)

    # the default input is in radius
    # angle sin(90)-1 in degree == sin(pi/2)=1 in radians
    print(angle)
```

```
    # to make it convenient, convert it to radians
    angle = math.sin(math.radians(90))
    print(angle)

    # this is also similar for cosine and other trigonometric and hyperbolic

print(angle_demo())
```

```
1.0
1.0
None
```

In [38]:
```
#dateandtime.py

import time

def pause():
    for i in range(10, 0, -1):
        print(f"The program will end in {i}...")
        time.sleep(1)

def current_time():
    t = time.strftime("%I:%M %p")
    return t

def current_date():
    d = time.strftime("%b %d %Y")
    return d

pause()
print(current_time())
print(current_date())
```

```
The program will end in 10...
The program will end in 9...
The program will end in 8...
The program will end in 7...
The program will end in 6...
The program will end in 5...
The program will end in 4...
The program will end in 3...
The program will end in 2...
The program will end in 1...
03:46 PM
Sep 15 2024
```

In [40]:
```
#main.py

import dateandtime

print("The current time is ", dateandtime.current_time())
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[40], line 3
      1 #main.py
----> 3 import dateandtime
      5 print("The current time is ", dateandtime.current_time())

ModuleNotFoundError: No module named 'dateandtime'
```

In [50]:
```
%cd C:\Users\Ken\PSMDSRC103\module3\modules1
```

```
C:\Users\Ken\PSMDSRC103\module3\modules1
```

In [52]:
```
%pwd
```

Out[52]:
```
'C:\\Users\\Ken\\PSMDSRC103\\module3\\modules1'
```

In [57]:
```
#main.py
#changed working directory to find dateandtime.py

import dateandtime

print("The current time is ", dateandtime.current_time())
```

```
The current time is  03:52 PM
```

In [61]:
```
#main.py
import dateandtime

print("The current time is ", dateandtime.current_time())
print("The current date is ", dateandtime.current_date())
```

```
The current time is  03:54 PM
The current date is  Sep 15 2024
```

In [65]:
```
#main.py
#enhance import to remove redundancy

from dateandtime import current_time, current_date

print("The current time is ", current_time())
print("The current date is ", current_date())
```

```
The current time is  03:55 PM
The current date is  Sep 15 2024
```

In [ ]:
```
#End of seatwork
```