**Emmanuel Alvarez**

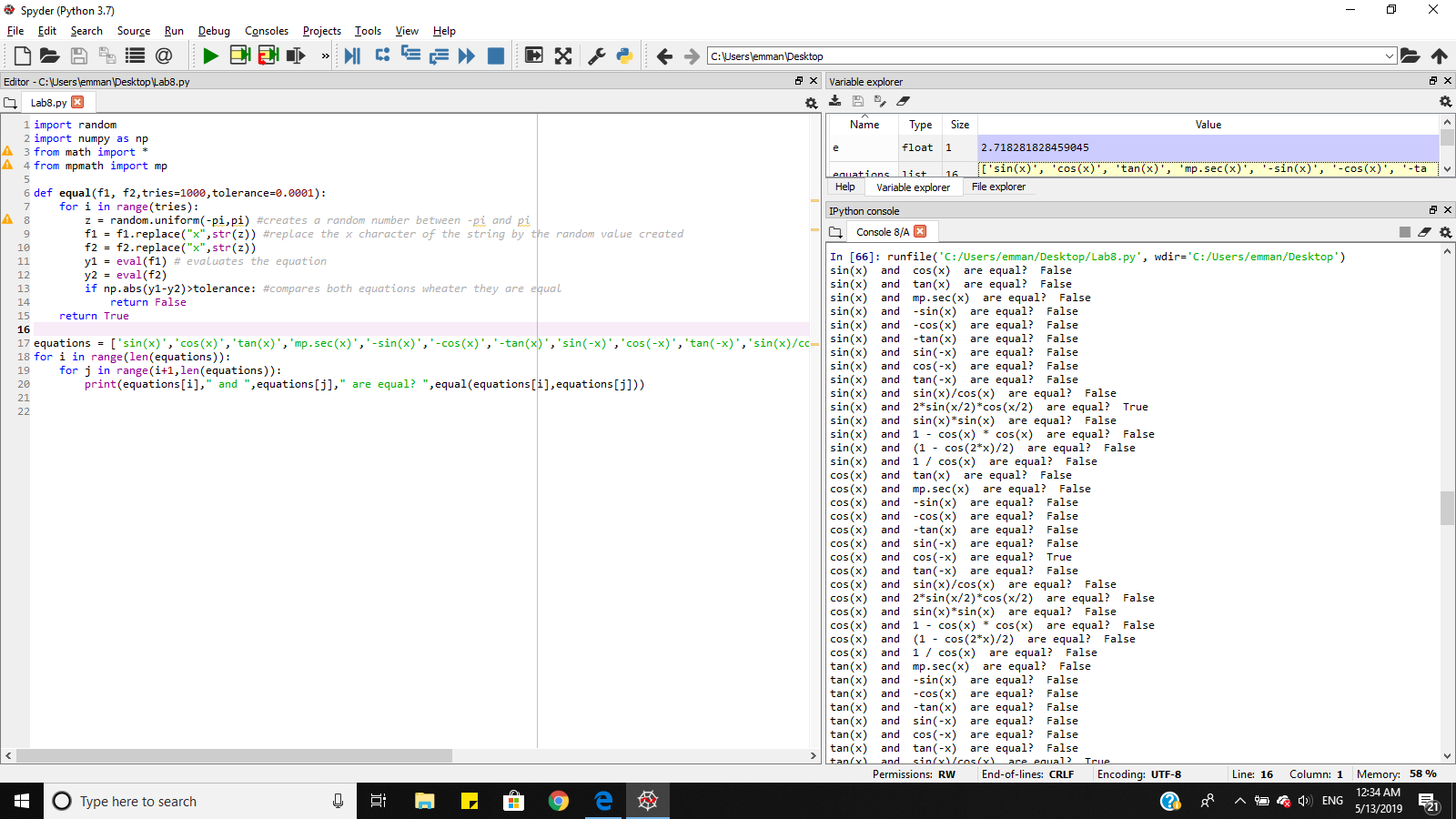
**80567137**

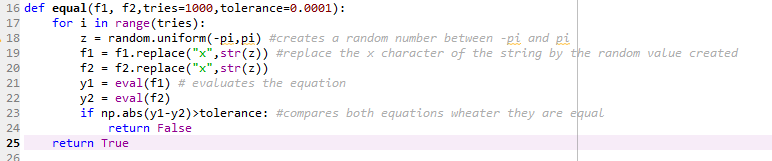
**Introduction**

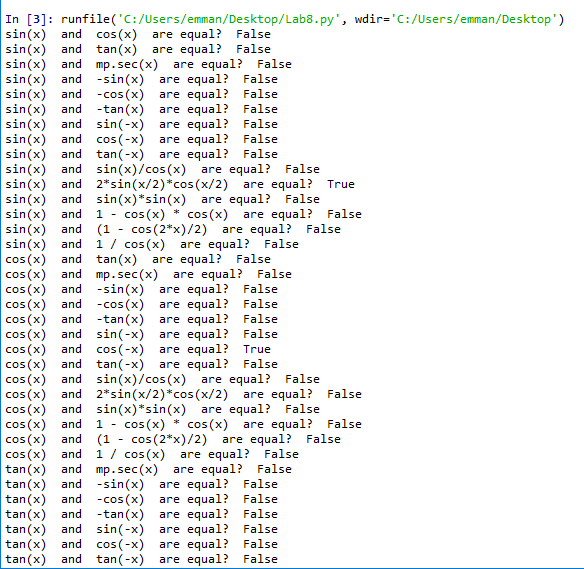
The purpose of this lab is to apply randomization algorithm and backtracking algorithm to solve two different problems: The first problem is creating a method that compares two different equations and return if they are equal (randomization algorithm) and the second problem is creating a method that receives a set of integers and return two different subsets where the union is the set and the intersection among them is an empty set and also, they add up the same integer.

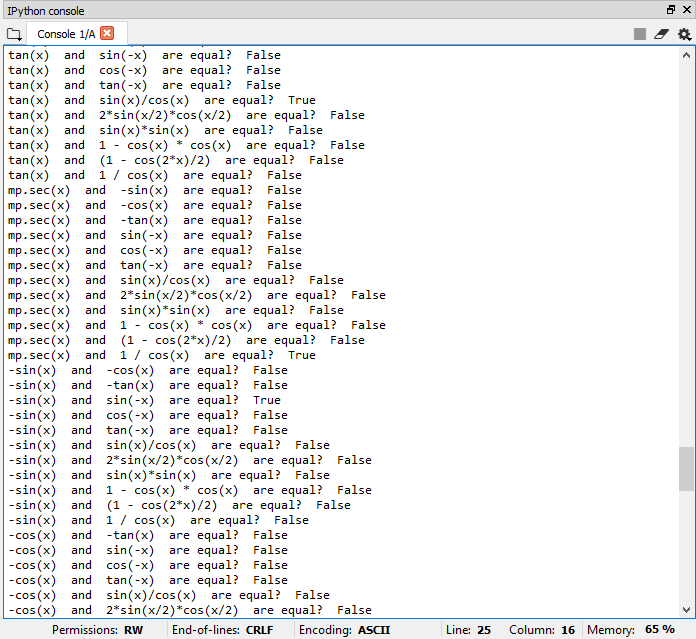
**Proposed solution and design implementation**

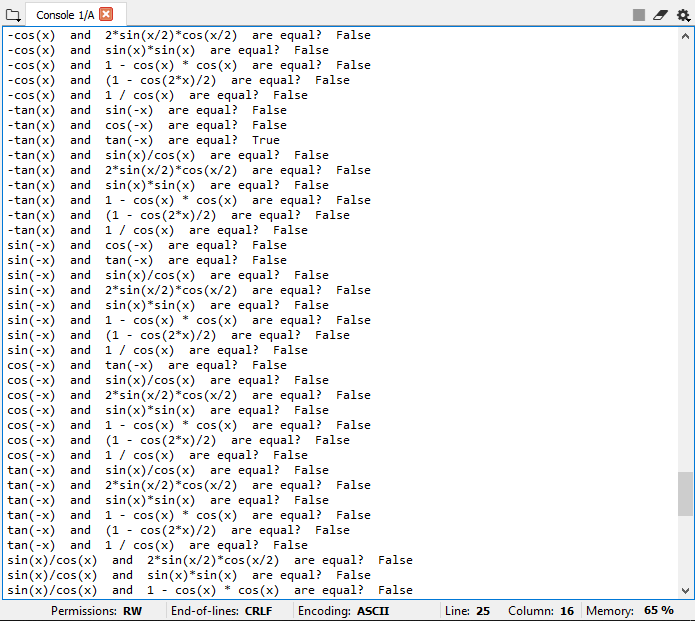
For the first method that compares two different equations, I first created a String array with every equation and called the method “equal” comparing every equation with each other using two for loops. The equal method receives four parameters: first equation, second equation, number of tries and tolerance. In this method, first of all, I used a for loop that will be executed n tries. Then, I created a random variable that goes from -pi to pi. Then, I changed the variable “x” of every equation to the random variable number in order to use the eval method. Then, I created two variables that receives the “eval” value of each equation and I subtracted them. Then, if the result is less then tolerance value n tries, it returns True. Otherwise, it will return False. The outputs are shown below.

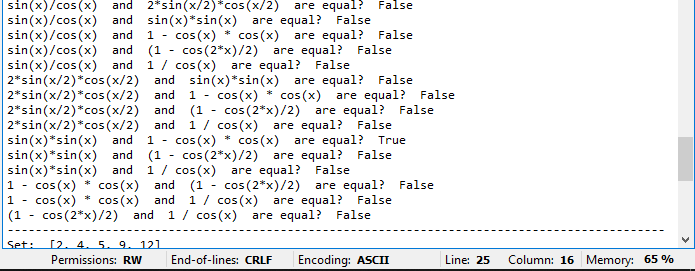




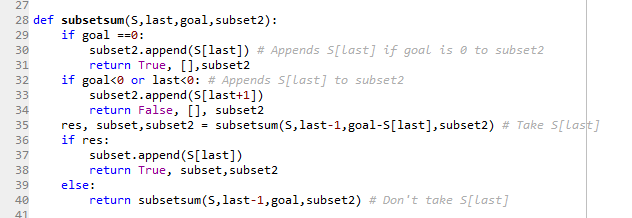


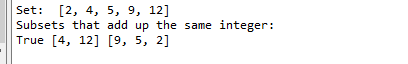


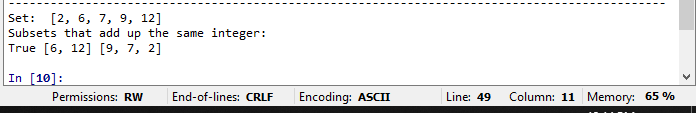


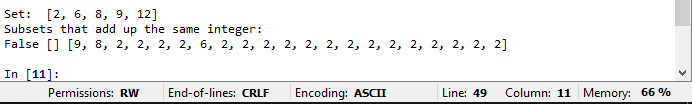


For the second method, it receives a set of integers and return if there are subsets that add up the same integer and the union among them is the original set and the intersection among them is an empty set. To accomplish it, I called the method subsetsum, and it receives four parameters: “S”, it is the entire set, “last”, it is the last index of the array “S”, “goal”, it is the number that we want to reach adding up the numbers of the subsets and finally, subset2, it will be the subset of “S” that will save the integers that were left after the other subset reached the goal. Having said that, I called the method with the main set, the last index of this set, the goal that is half of the result of adding up all the integers in the original set, and an empty subset. At the beginning of the method, it verifies if the goal is equal to 0, that will mean that we find a path that will add up the desired goal, and it will append the current integer to the subset2. Then, if goal is less than 0 or last is less than 0, it will return false after appending the current integer to the subset2. Then, if neither of them where True, it will call the same method subtracting 1 to last, and subtracting the current goal and the current integer of the index last. Then, if one of those calls is True it will append the current integer to the subset1. The purpose of this method is that whether one set reach the goal desired, the other set must reach the same goal, if the first subset did not reach the goal desired, the second subset will not reach the goal. The outputs are shown below.









**Big O notation**

Randomization: The equal method uses only one for loop with n tries. So, the Big O is O(n)

Backtracking: The subsetsum method uses two recursive calls 2T(n -1). Thus, the Big O is O(nlogn)

**Appendix**

"""

Emmanuel Alvarez 80567137

Instructor: Olac Fuentes

Last day modified: 05-13-2019

The purpose of this code is to compare two different equations and return is they

are equal or not using randomization algorithm. Also, given a Set, return if it has two

different subsets that add up the same integer.

"""

import random

import numpy as np

from math import \*

from mpmath import mp

def equal(f1, f2,tries=1000,tolerance=0.0001):

for i in range(tries):

z = random.uniform(-pi,pi) #creates a random number between -pi and pi

f1 = f1.replace("x",str(z)) #replace the x character of the string by the random value created

f2 = f2.replace("x",str(z))

y1 = eval(f1) # evaluates the equation

y2 = eval(f2)

if np.abs(y1-y2)>tolerance: #compares both equations wheater they are equal

return False

return True

def subsetsum(S,last,goal,subset2):

if goal ==0:

subset2.append(S[last]) # Appends S[last] if goal is 0 to subset2

return True, [],subset2

if goal<0 or last<0: # Appends S[last] to subset2

subset2.append(S[last+1])

return False, [], subset2

res, subset,subset2 = subsetsum(S,last-1,goal-S[last],subset2) # Take S[last]

if res:

subset.append(S[last])

return True, subset,subset2

else:

return subsetsum(S,last-1,goal,subset2) # Don't take S[last]

equations = ['sin(x)','cos(x)','tan(x)','mp.sec(x)','-sin(x)','-cos(x)','-tan(x)','sin(-x)','cos(-x)','tan(-x)','sin(x)/cos(x)','2\*sin(x/2)\*cos(x/2)','sin(x)\*sin(x)','1 - cos(x) \* cos(x)','(1 - cos(2\*x)/2)','1 / cos(x)']

for i in range(len(equations)):

for j in range(i+1,len(equations)):

print(equations[i]," and ",equations[j]," are equal? ",equal(equations[i],equations[j]))

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

S = [2,6,8,9,12]

total = 0

for t in S:

total+=t

print("Set: ", S)

print("Subsets that add up the same integer: ")

subset2 = []

a,b,c = subsetsum(S,len(S)-1,total/2,subset2)

print(a,b,c)

“I certify that this project is entirely my own work. I wrote, debugged, and tested the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provided inappropriate assistance to any student in the class.”