# This is the starting number for the numerator.

# The denominator will always start at 1 more than this.

lowerBound=10

# This is the ending number for the numerator.

# The numerator will run from lowerBound to

# one less than this number.

upperBound=1000

# This import makes the built-in combinations function

# and the program run timer available.

import itertools

import timeit

# This grabs the start time for the program.

start = timeit.default\_timer()

# This function converts an integer to a list.

def numberToList(num):

return([int(i) for i in str(num)])

# This function converts a list to an integer.

def listToNumber(digitList):

return(int(''.join(map(str,digitList))))

# This function returns an array of the positions of

# digit matches for the numerator and denominator.

# It also returnes the number of unique matches.

def getMatchList(top, bottom):

matchList = []

for topIndex in range(0, len(numeratorList)):

for bottomIndex in range(0, len(denominatorList)):

if numeratorList[topIndex] == denominatorList[bottomIndex]:

matchList = matchList + [[topIndex, bottomIndex]]

return matchList

# This function gets ALL possible n-tuples of size m.

# It us used to find ALL possible ways to cancel m

# digits from the fraction, but it includes duplicates

# that must be filtered out later.

def getAllTuples (theList, m):

AllTuples = list(itertools.combinations(theList, m))

return AllTuples

# This function gets the final list of matching

# digit position pairs with no duplicates. It filters

# the list from the function above.

def getTheFinalList(combos):

finalList = []

for i in range (0, len(combos)):

fine = True

for j in range (0, len(combos[0]) - 1):

if (combos [i][j][0] == combos [i][j+1][0]) or ((combos [i][j][1] == combos [i][j+1][1])):

fine = False

if fine == True:

finalList = finalList + [combos[i]]

return finalList

# This function does the final fraction check for

# SINGLE-DIGIT canceling and prints the results.

def goCheckFractions1 (h):

for a in range (0, h):

tupNumList1 = []

tupDenList1 = []

for b in range (0, len(numeratorList)):

if b != theMatchingDigits[a][0]:

tupNumList1 = tupNumList1 + [numeratorList[b]]

if b != theMatchingDigits[a][1]:

tupDenList1 = tupDenList1 + [denominatorList[b]]

tempNum1 = listToNumber(tupNumList1)

tempDen1 = listToNumber(tupDenList1)

if tempDen1 != 0:

if tempNum1/tempDen1 == numerator/denominator:

print(numerator, '/', denominator, '=', tempNum1, '/', tempDen1)

# This function creates a new fraction by including digits

# that are not cancelled. Then it checks the original fraction

# against the new fraction and prints the results when there

# is equality.

def goCheckFractions (v, w):

for m in range (0, v):

tupNumList = []

tupDenList = []

for n in range (0, w):

tupNumList = tupNumList + [uniqueTuples[m][n][0]]

tupDenList = tupDenList + [uniqueTuples[m][n][1]]

trialNum = []

trialDen = []

finalAnswer = 'No'

for y in range (0, len(numeratorList)):

if y not in tupNumList:

trialNum = trialNum + [numeratorList[y]]

for z in range (0, len(denominatorList)):

if z not in tupDenList:

trialDen = trialDen + [denominatorList[z]]

if (len(trialNum) != 0) and (len(trialDen) != 0):

tempNum = listToNumber(trialNum)

tempDen = listToNumber(trialDen)

if tempDen != 0:

if tempNum/tempDen == numerator/denominator:

finalAnswer = 'Yes'

print(numerator, '/', denominator, '=', tempNum, '/', tempDen)

# This is the main body of the program.

checkingOK = True

for numerator in range(lowerBound,upperBound):

for denominator in range(numerator+1,upperBound+1):

numeratorList = numberToList(numerator)

denominatorList = numberToList(denominator)

theMatchingDigits = getMatchList(numeratorList, denominatorList)

totalMatches = len(theMatchingDigits)

# This next if statement throws out fractions whose

# numerator and denominator are both multiples of 10.

if (numerator % 10 == 0) and (denominator % 10 == 0):

checkingOK = False

if (totalMatches != 0) and checkingOK:

goCheckFractions1(totalMatches)

if (totalMatches > 1) and checkingOK:

for d in range(2, len(numeratorList)):

if d <= totalMatches:

completeListOfTuples = getAllTuples(theMatchingDigits, d)

uniqueTuples = getTheFinalList(completeListOfTuples)

if uniqueTuples != []:

goCheckFractions(len(uniqueTuples), len(uniqueTuples[0]))

checkingOK = True

# This grabs the stop time for the program.

stop = timeit.default\_timer()

# These lines let me know the program is done running.

print(' ')

print("These are all of the fractions I could find.")

print (stop - start)