PYTHON PRACTICAL

**1: Write a program for computing GCD of 2 numbers with optimal data structures and less time-consuming.**

"""

This program for calculating GCD of 2 numbers.

"""

word\_to\_digit\_dict = {

'zero': '0',

'one': '1',

'two': '2',

'three': '3',

'four': '4',

'five': '5',

'six': '6',

'seven': '7',

'eight': '8',

'nine': '9'

}

digit\_to\_word\_dict = {

'0': 'zero',

'1': 'one',

'2': 'two',

'3': 'three',

'4': 'four',

'5': 'five',

'6': 'six',

'7': 'seven',

'8': 'eight',

'9': 'nine'

}

def word\_to\_num(words):

"""

This function takes numeric words string as an argumetent and it's converted into digit value.

Args:

words (string): numeric words string

Returns:

string: converted number.

"""

start\_pos = 0

end\_pos = 0

converted\_num = ''

while end\_pos < len(words):

if word\_to\_digit\_dict.get(words[start\_pos:end\_pos+1]):

converted\_num += word\_to\_digit\_dict.get(words[start\_pos:end\_pos+1])

start\_pos = end\_pos + 1

end\_pos += 1

return converted\_num

def num\_to\_word(num):

"""

This function takes numeric string value as an argumetent and it's converted into numeric words string.

Args:

nums (string): digit string

Returns:

string: converted numeric string.

"""

answer = ''

i = 0

while i < len(num):

answer += digit\_to\_word\_dict.get(num[i])

i += 1

return answer

def find\_gcd(num1, num2):

"""

This function takes two integer value as an argumetent and calculating GCD of 2 numbers and retruning answer.

Args:

num1 (int): digit one for finding gcd of two number.

num2 (int): digit two for finding gcd of two number.

Returns:

int: gcd of two numbers.

"""

i = 1

if num2 > num1:

min\_number = num1

else:

min\_number = num1

while i < min\_number+1:

if num1 % i == 0 and num2 % i == 0:

gcd = i

i += 1

return gcd

try:

print("Enter numeric words zero to nine only.")

num1 = input('Enter first value ')

num2 = input('Enter second value ')

converted\_num1 = int(word\_to\_num(num1))

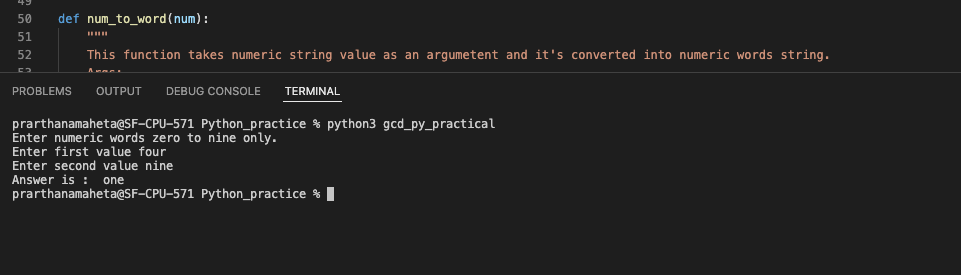
converted\_num2 = int(word\_to\_num(num2))

print('Answer is : ', num\_to\_word(

str(find\_gcd(converted\_num1, converted\_num2))))

except ValueError as e:

print("Please Enter valid numeric words")



2. **Given n pairs of parentheses, write a function to generate all combinations of well-formed parentheses.**

"""This program is used to find all the possible brackets"""

lst = []

def generatebrac(n, openb, closeb, string, res):

"""This function will generate brackets"""

if openb == 0 and closeb == 0:

# appending string of brackets if no close or open brackets remain

res.append(string)

return

if openb > 0:

# append ( if there are still open brackets left

string += '('

# Recursive call

generatebrac(n, openb-1, closeb, string, res)

# Backtrack

string = string[:len(string)-1]

if closeb > openb:

# append ) if number of close is greater than open brackets remain

string += ')'

# recursive call

generatebrac(n, openb, closeb-1, string, res)

# backtrack

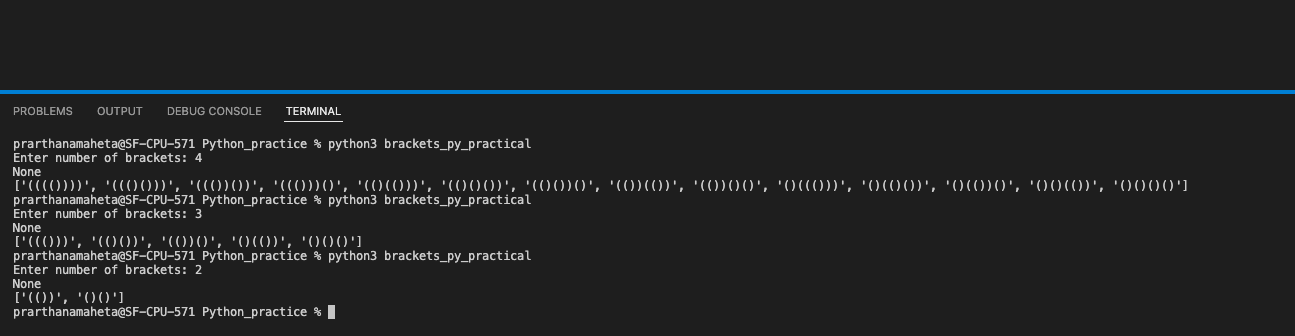
string = string[:len(string) - 1]

if \_\_name\_\_ == '\_\_main\_\_':

a = int(input('Enter number of brackets: '))

print(generatebrac(a, a, a, "", lst))

print(lst)



**3. Given an array of strings strs, group the anagrams together. You can return the answer in any order.**

**An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.**

"""

This program for find the anagrams of words from the given word list.

"""

def anagram(word\_list):

"""

This function takes the word list as an argumetent then finds the anagram from the list and returns it in the same group of anaragram.

Args:

word\_list (list): list of the words.

Returns:

list: It's returning the group of anagram list.

"""

anagram\_dict = {}

for i in word\_list:

sorted\_string = ''.join(sorted(i))

if sorted\_string in anagram\_dict:

anagram\_dict[sorted\_string].append(i)

else:

anagram\_dict[sorted\_string] = [i]

return anagram\_dict

word\_list = input().split(',')

print(list(anagram(word\_list).values()))

