Challenge 1:

store\_number = [] # to store all products of two three-digit numbers

for a in range(100,999): # set the range of three-digit numbers

for b in range(100,999):

number = a \* b

string = str(number) # convert number to string to compare the index

if string[:] == string[::-1]: # check if the string has same forwards and backwards sequence

store\_number.append(number) # only store the numbers if condition above is met

print(max(store\_number)) # print the max number from the values store in store\_number

Challenge 2:

def check\_prime(num):

if num in range(0,2000): # set a range from 0 to 2000 for numerator

for i in range(2,num): # set a range from 2 to max of numerator for denominator

if num % i == 0: # if the remainder of the division is 0, the numerator is a prime number

return False

else:

return True

sum = 0

for number in range(0,2000):

if check\_prime(number) == True: # find out the prime numbers below 2000

sum += number # sum up all the above prime numbers

print(sum)

Challenge 3:

def mul\_three(num3):

if num3 % 3 == 0: # set criteria to get multiples of 3

return True

else:

return False

def mul\_five(num5):

if num5 % 5 == 0: # set criteria to get multiples of 5

return True

else:

return False

sum = 0

for number in range(0,1000):

if mul\_three(number) == True or mul\_five(number) == True: # get multiples of 3 and 5 from range between 0 to 1000

sum += number # sum up the above numbers

print(sum)

Challenge 4:

def compress(string):

alphanumeric = '' # alphanumeric starts empty

count = 1 # character count in alphanumeric starts from 1

alphanumeric += string[0] # add the first character of string argument into variable 'alphanumeric'

for i in range(0,len(string)-1): # for loop to check over the characters in the string

if(string[i] == string[i+1]): # if the current string equal to next word

count+=1 # character count will increase by 1 from base of 1

else: # if the current character does not equal to the next character

alphanumeric += str(count) # if next character does not repeat, the count will be 1

alphanumeric += string[i+1] # alphanumeric will add on with next character in the string

count = 1 # reset counter to 1 for characters after index 0

if(count > 0):

alphanumeric += str(count) # all the alphanumeric and the count will be captured here

return alphanumeric # store the resulted alphanumeric in the variable 'alphanumeric'

string = "aabcccccaaa"

print(compress(string))

#print(len(compress(string)))

#print(len(string))

if len(compress(string)) < len(string):

print(string)

else:

print(compress(string))

Bonus Challenge:

def print\_num(num):

group = []

for i in range(1,101): # list out numbers from 1 to 100

group.append(i)

return (group)

print(print\_num(i))

def mul\_three(num3):

if num3 % 3 == 0: # set criteria to get multiple of 3

return True

else:

return False

def mul\_five(num5):

if num5 % 5 == 0: # set criteria to get multiple of 5

return True

else:

return False

def mul\_threefive(num35):

if num35 % 3 == 0 and num35 % 5 == 0: # set criteria to get multiples of 3 and 5

return True

else:

return False

def print\_alphanumeric(num1):

group = [] # set a empty list to store numbers and words from the subsequent codings

for j in range(1,101): # set the range of targeted numbers

if mul\_threefive(j) == True: # check if the number is a multiple of 3 and 5

group.append("FizzBuzz") # replace the number with "FizzBuzz" and store in 'group'

elif mul\_three(j) == True: # check if the number is a multiple of 3

group.append("Fizz") # replace the number with "Fizz" and store in 'group'

elif mul\_five(j) == True: # check if the number is a multiple of 5

group.append("Buzz") # replace the number with "Buzz" and store in 'group'

else:

group.append(j) # all other numbers store in 'group' as it is

return (group)

print(print\_alphanumeric(j))