1. Given a sentence as txt, return True if any two adjacent words have this property: One word ends with a vowel, while the word immediately after begins with a vowel (a e i o u).

**Examples**

vowel\_links("a very large appliance") ➞ True

vowel\_links("go to edabit") ➞ True

vowel\_links("an open fire") ➞ False

vowel\_links("a sudden applause") ➞ False

**Ans:**

def vowel\_links(string):

lst=str(string).split()

match = False

for i in range(len(lst)-1):

if lst[i].endswith(('a', 'e', 'i', 'o', 'u')) and lst[i+1].startswith(('a', 'e', 'i', 'o', 'u')):

match=True

break

print(match)

2. You are given three inputs: a string, one letter, and a second letter.

Write a function that returns True if every instance of the first letter occurs before every instance of the second letter.

**Examples**

first\_before\_second("a rabbit jumps joyfully", "a", "j") ➞ True

# Every instance of "a" occurs before every instance of "j".

first\_before\_second("knaves knew about waterfalls", "k", "w") ➞ True

first\_before\_second("happy birthday", "a", "y") ➞ False

# The "a" in "birthday" occurs after the "y" in "happy".

first\_before\_second("precarious kangaroos", "k", "a") ➞ False

**Ans:**

def first\_before\_second(string, a, b):

lst1=[]

lst2=[]

result=True

for i in string:

if i==a:

lst1.append(string.find(i))

string=string.replace(i, '#', 1)

if i==b:

lst2.append(string.find(i))

string=string.replace(i, '#', 1)

for i in lst1:

for j in range(len(lst2)):

if i<lst2[j]:

continue

result=False

break

print(result)

3. Create a function that returns the characters from a list or string r on odd or even positions, depending on the specifier s. The specifier will be "odd" for items on odd positions (1, 3, 5, ...) and "even" for items on even positions (2, 4, 6, ...).

**Examples**

char\_at\_pos([2, 4, 6, 8, 10], "even") ➞ [4, 8]

# 4 & 8 occupy the 2nd & 4th positions

char\_at\_pos("EDABIT", "odd") ➞ "EAI"

# "E", "A" and "I" occupy the 1st, 3rd and 5th positions

char\_at\_pos(["A", "R", "B", "I", "T", "R", "A", "R", "I", "L", "Y"], "odd") ➞ ["A", "B", "T", "A", "I", "Y"]

**Ans:**

def char\_at\_pos(lst, s):

if s=="even":

start=2

else:

start=1

n\_lst=[]

for i in range(start, len(lst)+1, 2):

n\_lst.append(lst[i-1])

print(n\_lst)

4. Write a function that returns the greatest common divisor of all list elements. If the greatest common divisor is 1, return 1.

**Examples**

GCD([10, 20, 40]) ➞ 10

GCD([1, 2, 3, 100]) ➞ 1

GCD([1024, 192, 2048, 512]) ➞ 64

**Ans:**

def GCD(lst):

m=min(lst)

result=1

for i in range(1, m+1):

div=True

for j in lst:

if j%i==0:

continue

else:

div=False

break

if div:

result=i

print(result)

5. A number/string is a palindrome if the digits/characters are the same when read both forward and backward. Examples include "racecar" and 12321. Given a positive number n, check if n or the binary representation of n is palindromic. Return the following:

- "Decimal only." if only n is a palindrome.

- "Binary only." if only the binary representation of n is a palindrome.

- "Decimal and binary." if both are palindromes.

- "Neither!" if neither are palindromes.

**Examples**

palindrome\_type(1306031) ➞ "Decimal only."

# decimal = 1306031

# binary = "100111110110110101111"

palindrome\_type(427787) ➞ "Binary only."

# decimal = 427787

# binary = "1101000011100001011"

palindrome\_type(313) ➞ "Decimal and binary."

# decimal = 313

# binary = 100111001

palindrome\_type(934) ➞ "Neither!"

# decimal = 934

# binary = "1110100110"

**Ans:**

def palindrome\_type(num):

num=str(num)

binary=bin(int(num))[2:]

dec\_palin=num==num[::-1]

bin\_palin=binary==binary[::-1]

if dec\_palin and bin\_palin:

print("Decimal and binary.")

if not dec\_palin and not bin\_palin:

print("Neither!")

if not dec\_palin and bin\_palin:

print("Binary only.")

if dec\_palin and not bin\_palin:

print("Decimal only.")