1. def vowel\_links(in\_string):

in\_list = in\_string.split(" ")

vowel\_list = ['a','e','i','o','u']

end,start,output = False, False,False

for ele in range(len(in\_list)):

temp = True if in\_list[ele][-1] in vowel\_list else False

start = True if in\_list[ele][0] in vowel\_list else False

if start == end == True:

output=True

break

end=temp

print(f'vowel\_links({in\_string}) ➞ {output}')

vowel\_links("a very large appliance")

vowel\_links("go to edabit")

vowel\_links("an open fire")

vowel\_links("a sudden applause")

Output:

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

1. # Approach 1 Using index() and() rindex() functions

def first\_before\_second\_index(in\_string,in\_first,in\_second):

last\_occur\_first = in\_string.rindex(in\_first)

first\_occur\_second = in\_string.index(in\_second)

output = True if last\_occur\_first < first\_occur\_second else False

print(f'first\_before\_second\_index({in\_string}, {in\_first}, {in\_second}) ➞ {output}')

# Approach 2 Using find() and() rfind() functions

def first\_before\_second\_find(in\_string,in\_first,in\_second):

last\_occur\_first = in\_string.rfind(in\_first)

first\_occur\_second = in\_string.find(in\_second)

output = True if last\_occur\_first < first\_occur\_second else False

print(f'first\_before\_second\_find({in\_string}, {in\_first}, {in\_second}) ➞ {output}')

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

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

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

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

print()

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

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

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

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

Output:

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

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

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

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

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

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

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

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

1. def char\_at\_pos(in\_list,mode):

out\_list = []

for ele in range(len(in\_list)):

if mode == 'even' and (ele+1)%2 == 0:

out\_list.append(in\_list[ele])

elif mode == 'odd' and (ele+1)%2 != 0:

out\_list.append(in\_list[ele])

print(f'char\_at\_pos{in\_list,mode} ➞ {out\_list}')

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

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

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

Output:

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

char\_at\_pos('EDABIT', 'odd') ➞ ['E', 'A', 'I']

char\_at\_pos(['A', 'R', 'B', 'I', 'T', 'R', 'A', 'R', 'I', 'L', 'Y'], 'odd') ➞ ['A', 'B', 'T', 'A', 'I', 'Y']

1. def GCD(in\_list):

small = min(in\_list)

gcd = -1

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

output = []

for ele in in\_list:

output.append(ele%i)

if len(set(output)) == 1 and list(set(output))[0] == 0:

gcd = i

print(f'GCD({in\_list}) ➞ {gcd}')

GCD([10, 20, 40])

GCD([1, 2, 3, 100])

GCD([1024, 192, 2048, 512])

Output:

GCD([10, 20, 40]) ➞ 10

GCD([1, 2, 3, 100]) ➞ 1

GCD([1024, 192, 2048, 512]) ➞ 64

1. def palindrome\_type(in\_num):

output = None

if str(in\_num) == str(in\_num)[::-1] and str(bin(in\_num)[2:]) == str(bin(in\_num)[2:])[::-1]:

output = 'Decimal and binary.'

elif str(in\_num) == str(in\_num)[::-1] and str(bin(in\_num)[2:]) != str(bin(in\_num)[2:])[::-1]:

output = 'Decimal only.'

elif str(bin(in\_num)[2:]) != str(bin(in\_num)[2:])[::-1] and str(in\_num) == str(in\_num)[::-1]:

output = 'Binary only.'

else:

output = 'Neither!'

print(f'palindrome({in\_num}) ➞ {output}')

palindrome\_type(1306031)

palindrome\_type(427787)

palindrome\_type(313)

palindrome\_type(934)

Output:

palindrome(1306031) ➞ Decimal only.

palindrome(427787) ➞ Neither!

palindrome(313) ➞ Decimal and binary.

palindrome(934) ➞ Neither!