**Python Advance Programming Assignment-08**

**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

**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**

In [114]:

**def** first\_before\_second(string,first,second):

string**=**''**.**join(string)

second\_index**=**[i **for** i **in** range(len(string)) **if** (string[i:i **+** len(second)] **==** second)]

first\_index**=**[i **for** i **in** range(len(string)) **if** (string[i:i **+** len(first)] **==** first)]

zip\_list**=**list(zip(first\_index,second\_index))

val**=**[]

**for** i **in** zip\_list:

**if** i[0]**<**i[1]:

val**.**append(**True**)

**else**:

val**.**append(**False**)

**if** all(val)**==True**:

**return** **True**

**return** **False**

In [109]:

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

Out[109]:

False

**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"]

In [126]:

**def** char\_at\_pos(iterable,specifier):

l**=**[]

**if** specifier**.**lower()**==**'even':

**for** i **in** iterable:

**if** (iterable**.**index(i)**+**1)**%2**==0:

l**.**append(i)

**elif** specifier**.**lower()**==**'odd':

**for** i **in** iterable:

**if** (iterable**.**index(i)**+**1)**%2**!=0:

l**.**append(i)

**return** l

In [127]:

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

Out[127]:

[4, 8]

In [129]:

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

Out[129]:

['E', 'A', 'I']

In [130]:

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

Out[130]:

['A', 'B', 'T', 'A', 'Y']

**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

**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"

In [25]:

**def** palindrome\_type(num):

binary**=** str(bin(num)**.**replace("0b", ""))

decimal**=**str(num)

flag\_binary**=False**

flag\_decimal**=False**

rev\_binary**=**binary[::**-**1]

rev\_decimal**=**decimal[::**-**1]

**if** rev\_binary**==**binary:

flag\_binary**=True**

**if** rev\_decimal**==**decimal:

flag\_decimal**=True**

print('Decimal = ',decimal)

print('Binary = ',binary)

**if** flag\_binary**==True** **and** flag\_decimal**==False**:

**return** 'Binary only'

**if** flag\_binary**==False** **and** flag\_decimal**==True**:

**return** 'Decimal only'

**if** flag\_binary**==True** **and** flag\_decimal**==True**:

**return** 'Decimal and Binary'

**if** flag\_binary**==False** **and** flag\_decimal**==False**:

**return** 'Neither'

In [26]:

palindrome\_type(1306031)

Decimal = 1306031

Binary = 100111110110110101111

Out[26]:

'Decimal only'

In [27]:

palindrome\_type(427787)

Decimal = 427787

Binary = 1101000011100001011

Out[27]:

'Binary only'

In [28]:

palindrome\_type(313)

Decimal = 313

Binary = 100111001

Out[28]:

'Decimal and Binary'

In [29]:

palindrome\_type(934)

Decimal = 934

Binary = 1110100110

Out[29]:

'Neither'