**Python Advance Programming Assignment-02**

**1. Make a function that encrypts a given input with these steps:**

Input: "apple"

Step 1: Reverse the input: "elppa"

Step 2: Replace all vowels using the following chart:

a => 0 e => 1 i => 2 o => 2 u => 3

**"1lpp0"**

Step 3: Add "aca" to the end of the word: "1lpp0aca"

Output: "1lpp0aca"

Examples

encrypt("banana") ➞ "0n0n0baca"

encrypt("karaca") ➞ "0c0r0kaca"

encrypt("burak") ➞ "k0r3baca"

encrypt("alpaca") ➞ "0c0pl0aca"

In [6]:

**def** encrypt(string):

new\_string**=**''

reverse**=**string[::**-**1]

**for** i **in** reverse:

**if** i**==**'a':

i**=**'0'

**elif** i**==**'e':

i**=**'1'

**elif** i**==**'i':

i**=**'2'

**elif** i**==**'o':

i**=**'2'

**elif** i**==**'u':

i**=**'3'

new\_string**+=**i

new\_string**=**new\_string**+**'aca'

**return** new\_string

In [7]:

encrypt('karaca')

Out[7]:

'0c0r0kaca'

In [8]:

encrypt("karaca")

Out[8]:

'0c0r0kaca'

In [9]:

encrypt("burak")

Out[9]:

'k0r3baca'

In [10]:

encrypt("alpaca")

Out[10]:

'0c0pl0aca'

**2. Given the month and year as numbers, return whether that month contains a Friday 13th.(i.e You can check Python's datetime module)**

Examples

has\_friday\_13(3, 2020) ➞ True

has\_friday\_13(10, 2017) ➞ True

has\_friday\_13(1, 1985) ➞ False

In [45]:

**import** datetime

**def** has\_friday\_13(month,year):

d1**=**datetime**.**datetime(year,month,13)

d2**=**d1**.**strftime('%A')

**if** (d2**==**'Friday' **or** d2**==**'Fri'):

**return** **True**

**return** **False**

In [46]:

has\_friday\_13(3,2020)

Out[46]:

True

In [47]:

has\_friday\_13(10,2017)

Out[47]:

True

In [48]:

has\_friday\_13(1,1985)

Out[48]:

False

**3. Write a regular expression that will help us count how many bad cookies are produced every day. You must use RegEx negative lookbehind.**

Example

lst = ["bad cookie", "good cookie", "bad cookie", "good cookie", "good cookie"] pattern = "yourregularexpressionhere"

len(re.findall(pattern, ", ".join(lst))) ➞ 2

**Ans**

**Not yet taught in the class**

**4. Given a list of words in the singular form, return a set of those words in the plural form if they appear more than once in the list.**

Examples

pluralize(["cow", "pig", "cow", "cow"]) ➞ { "cows", "pig" }

pluralize(["table", "table", "table"]) ➞ { "tables" }

pluralize(["chair", "pencil", "arm"]) ➞ { "chair", "pencil", "arm" }

In [54]:

**def** pluralize(l):

s**=**set()

**for** i **in** l:

**if** l**.**count(i)**>**1:

s**.**add(i**+**'s')

**else**:

s**.**add(i)

**return** s

In [55]:

pluralize(["cow", "pig", "cow", "cow"])

Out[55]:

{'cows', 'pig'}

In [56]:

pluralize(["table", "table", "table"])

Out[56]:

{'tables'}

In [57]:

pluralize(["chair", "pencil", "arm"])

Out[57]:

{'arm', 'chair', 'pencil'}