1.Write a function that takes a positive integer num and calculates how many dots exist in a pentagonal shape around the center dot on the Nth iteration.

In the image below you can see the first iteration is only a single dot. On the second, there are 6 dots. On the third, there are 16 dots, and on the fourth there are 31 dots.

image.png Examples:

pentagonal(1) ➞ 1

pentagonal(2) ➞ 6

pentagonal(3) ➞ 16

pentagonal(8) ➞ 141

def pentagonal(in\_num):

output = 1

if in\_num >=1:

for ele in range(in\_num):

output = output + (5\*ele)

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

else:

print("Enter a Positive Number as Input")

pentagonal(1)

pentagonal(2)

pentagonal(3)

pentagonal(8)

pentagonal(1) ➞ 1

pentagonal(2) ➞ 6

pentagonal(3) ➞ 16

pentagonal(8) ➞ 141

2.. 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"

def encrypt(in\_string):

vowels = {'a':'0','e':'1','i':'2','o':'2','u':'2'}

out\_string = ''

for ele in in\_string[::-1]:

if ele in vowels.keys():

out\_string += vowels[ele]

else:

out\_string += ele

out\_string += "aca"

print(f'encrypt({in\_string}) ➞ {out\_string}')

encrypt("banana")

encrypt("karaca")

encrypt("burak")

encrypt("alpaca")

encrypt(banana) ➞ 0n0n0baca

encrypt(karaca) ➞ 0c0r0kaca

encrypt(burak) ➞ k0r2baca

encrypt(alpaca) ➞ 0c0pl0aca

3.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

import datetime

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

output = False

if datetime.datetime(year,month,13).strftime('%A') == 'Friday':

output = True

print(f'has\_friday\_13{month,year} ➞ {output}')

has\_friday\_13(3, 2020)

has\_friday\_13(10, 2017)

has\_friday\_13(1, 1985)

has\_friday\_13(3, 2020) ➞ True

has\_friday\_13(10, 2017) ➞ True

has\_friday\_13(1, 1985) ➞ False

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

Examples:

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

pattern = "yourregularexpressionhere"

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

import re

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

pattern = r'(?<!good)\scookie'# Regex Negative lookbehind expression

data = re.findall(pattern,' '.join(lst))

print(f'No of Bad cookies produced per day ➞ {len(data)}')

No of Bad cookies produced per day ➞ 2

5.. 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" }

def pluralize(in\_list):

out\_set = set()

for ele in set(in\_list):

if in\_list.count(ele) > 1:

out\_set.add(ele+'s')

else:

out\_set.add(ele)

print(f'pluralize({in\_list}) ➞ {out\_set}')

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

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

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

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

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

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