Escribir una función para validar una nueva clave de acceso.

La función deberá recibir una cadena de caracteres, que contendrá la clave candidata, que fue ingresada por el usuario; y devolverá True o False, dependiendo de si cumple con las condiciones establecidas.

La clave debe:

- Tener una longitud de entre 8 y 12 caracteres inclusive

- Solo puede estar formada por letras y números

- Debe contener al menos una letra mayúscula, una letra minúscula y 1 nùmero

------------------------------------------------------------------------------

Escribir una función para validar una nueva clave de acceso.

La función deberá recibir una cadena de caracteres, que contendrá la clave candidata, ingresada por el usuario; y devolverá True o False, dependiendo de si cumple con las condiciones establecidas.

La clave debe:

- Tener una longitud de entre 6 y 10 caracteres inclusive

- Solo puede estar formada por vocales (no acentuadas) y dígitos pares

- Debe estar formada por una combinación cualquiera de:

Vocales mayúsculas y dígitos pares, o

Vocales minúsculas y dígitos pares, o

Vocales minúsculas y mayúsculas, o

Vocales minúsculas, mayúsculas y dígitos pares

------------------------------------------------------------------------------

Escribir una función para validar una nueva clave de acceso.

La función deberá recibir una cadena de caracteres, que contendrá la clave candidata, ingresada por el usuario; y devolverá True o False, dependiendo de si cumple con las condiciones establecidas.

La clave debe:

- Tener como mínimo entre 4 y 8 dígitos numéricos

- Los dígitos adyacentes no pueden ser iguales

- La clave no puede formar una secuencia ordenada creciente o decreciente

------------------------------------------------------------------------------

Escribir una función para validar una nueva clave de acceso.

La función deber recibir una cadena de caracteres, que contendrá la clave candidata, que fue ingresada por el usuario; y devolverá True o False, dependiendo de si cumple con las condiciones establecidas.

La clave debe:

- Tener una longitud total de entre 4 y 10 caracteres

- Estar formada por igual cantidad de caracteres numéricos y caracteres alfabéticos, y no puede contener ningún otro carácter.

- Los caracteres alfabéticos y numéricos deben estar intercalados, no pueden haber dos letras consecutivas, ni dos dígitos numéricos consecutivos.

------------------------------------------------------------------------------

Escribir una función para validar una nueva clave de acceso.

La función deberá recibir una cadena de caracteres, que contendrá la clave candidata, que fue ingresada por el usuario; y devolverá True o False, dependiendo de si cumple con las condiciones establecidas.

La clave debe:

- Tener una longitud total de entre 8 y 12 caracteres

- Estar formada por mayor cantidad de caracteres alfabéticos que caracteres numéricos, y no puede contener ningún otro carácter.

- No puede comenzar con un carácter numérico, ni los 3 últimos ser alfabéticos.

- La suma de los caracteres numéricos no puede ser mayor a 35.

-------------------------------------------------------------------------------

Escribir una función para validar una nueva clave de acceso.

La función deberá recibir una cadena de caracteres, que contendra la clave

candidata, que fue ingresada por el usuario; y devolvera True o False,

dependiendo de si cumple con las condiciones establecidas.

La clave debe:

- Tener una longitud total de entre 4 y 10 caracteres alfabeticos, inclusive.

- Estar formada por mayor cantidad de letras minusculas que mayusculas.

- El ultimo caracter, debe ser una letra minuscula.

- La clave puede tener como maximo 2 caracteres que sean vocales.

""" ejercicio 1

Escribir una función que, dado un texto que se pasa por parámetro,

devuelva el valor booleano falso si antes de un punto o una coma

hay un espacio, caso contrario devuelve verdadero.

No se puede utilizar el método count.

"""

def textoCorrecto(texto):

if (" ." in texto or " ," in texto):

return False

return True

""" Ejercicio 2

Solicite al usuario el ingreso de un texto.

Considerar que el usuario solo ingresa palabras separadas por blancos, sin ningún otro tipo de caracteres. De las palabras ingresadas, descartar las que tienen tres letras o menos y las que tienen 10 letras

o más, además de las repetidas.

Luego, mostrar una lista de las palabras que quedaron ordenadas,

en primer lugar por cantidad de letras (de menor a mayor) y,

a igual cantidad de letras, alfabéticamente.

"""

## ingresa palabras y devuelve una lista de esas palabras

def ingresarTexto():

texto = input("Ingrese un texto con palabras separadas por blancos: ")

return texto.split()

## verifica si la clave largo esta en el diccionario

## en caso de estar, verifica si la palabra no esta y la agrega

## si el largo no esta, agrega la clave con una lista que contiene la palabra

def verificar(dic, palabra, largo):

if (largo in dic):

if (palabra not in dic[largo]):

dic[largo].append(palabra)

else:

dic[largo] = [palabra]

## toma la lista con las palabras y arma un diccionario donde la clave

## es la cantidad de letras y tiene una lista con las palabras

def palabrasSeleccionadas(lista):

palabras = {}

for palabra in lista:

largo = len(palabra)

if (3 < largo < 10):

verificar(palabras, palabra, largo)

return palabras

## ordena cada lista de palabras del diccionario y arma una sola

def listaOrdenada(dic):

listaFinal = []

longitudOrdenada = sorted(dic)

for largo in longitudOrdenada:

listaFinal.extend(sorted(dic[largo]))

return listaFinal

## principal del ejercicio 2

def main2():

listaOriginal = ingresarTexto()

diccionarioPalabras = palabrasSeleccionadas(listaOriginal)

listaFinal = listaOrdenada(diccionarioPalabras)

print(listaFinal)

""" Ejercicio 3

Ingresar en un diccionario localidades (clave) y dos datos:

cantidad de personas en edad laboral – cantidad de empleados.

Los datos surgen de distintas planillas, por lo que una misma clave

(localidad) se puede ingresar varias veces, debiendo sumarse los valores.

Se pide:

a) Calcular el total de personas en edad laboral y empleados

para cada localidad e imprimirlo.

b) Imprimir un listado ordenado de mayor a menor por porcentaje

de desocupación. Indicando: localidad - % de desocupación

"""

## carga y devuelve el diccionario: localidad - habitantes en edad laboral

## empleados

def cargaDiccionario():

dic = {}

localidad = input("Ingrese la localidad, para salir ingrese X: ")

while (localidad != "X"):

habitantes = int(input("Ingrese los habitantes en edad laboral: "))

empleados = int(input("Ingrese los empleados: "))

agregarEnDiccionario(localidad, habitantes, empleados, dic)

localidad = input("Ingrese la localidad, para salir ingrese X: ")

return dic

## verifica si una localidad ya esta en el diccionario o no

## si esta suma los valores, si no los agrega

def agregarEnDiccionario(loc, hab, emple, dic):

if (loc in dic):

dic[loc][0] += hab

dic[loc][1] += emple

else:

dic[loc] = [hab, emple]

## imprime el diccionario

def puntoA(dic):

for localidad in dic:

print("Para la localidad ", localidad)

print("Personas en edad laboral: ", dic[localidad][0])

print("Empleados: ", dic[localidad][1])

## calcula el porcentaje de desocupacion para cada localidad

## devuelve un nuevo diccionario

def porcentajeDesocupacion(dic):

desocupacion = {}

for localidad in dic:

habitantes = dic[localidad][0]

desocupados = dic[localidad][0] - dic[localidad][1]

desocupacion[localidad] = (100 \* desocupados)/habitantes

return desocupacion

## ordena por porcentaje de desocupacion de mayor a menor

## imprime la lista

def puntoB(dic):

lista = sorted(dic.items(), key = lambda x : x[1], reverse = True)

print("Porcentaje de desocupacion por localidad")

for localidad in lista:

print("Localidad: ", localidad[0], " - desocupacion: ", localidad[1], "%")

## principal del ejercicio 3

def main3():

dic = cargaDiccionario()

puntoA(dic)

desocupacion = porcentajeDesocupacion(dic)

puntoB(desocupacion)