Fundamentos de Programación

Cuaderno de Trabajo 4

Ejercicios Resueltos

1. **Escriba un programa en Python que compruebe si una lista de enteros está ordenada ascendentemente.**

SOLUCION PROPUESTA:

```
def en_orden_ascendente (lista):
    """ list -> bool
        OBJ: Devuelve True si la lista está ordenada ascendentemente
        PRE: Todos los elementos de la lista son comparables """
    if len(lista) == 1:
        ordenado=True

    if len(lista) > 1: # una lista con solo un elemento está ordenada
        i = 1
        ant = lista [0]
        while i < len(lista) and ordenado:
            if lista [i] < ant:
                ordenado = False
            else:
                ant = lista [i]
                i += 1
    return ordenado

#PROBADOR
mi_lista = [3,5,1,10]
print('Ordenada: ', en_orden_ascendente(mi_lista))
print('Ordenada: ', en_orden_ascendente([3,5,7,6))
```

Otra opción usando el esquema “discriminación a la salida”:

```
def es_ascendente lista):
    """ list -> bool
        OBJ: Devuelve True si la lista está ordenada ascendentemente
        PRE: len(lista)>0, todos los elementos comparables """
    i=1
    while i < len(lista) and lista[i]>lista [i-1]:
        i+=1
    return i==len(lista)

#PROBADOR
print('Ordenada: ', es_ascendente([3,5,7,10]), 'SI')
print('Ordenada: ', es_ascendente([3,5,7,6]), 'NO')
```

2. Dada una lista de valores enteros, indicar cuántos están por debajo de un valor dado por el usuario.

SOLUCION PROPUESTA:

```
def cuantos_por_debajo(lista, umbral):
  """ list, int -> int
      OBJ: Cuántos valores en la lista están por debajo del umbral
      PRE: la lista debe contener solo enteros   """
   i=0
   cont=0
   for valor in lista:
       if valor < umbral:
           cont+=1
   return cont

#PROBADOR
mi_lista = [3, 5, 1, 10, 41, 23, 19, 33]
umbral = 20
print(cuantos_por_debajo(mi_lista, umbral), 'elementos bajo el umbral')
```

3. Supongamos que una votación en que se dispone de una serie de n candidaturas, se desea escribir un algoritmo que realice las siguientes tareas:

a) Lectura de votos para las candidaturas (finalizar introduciendo “0”)

b) Buscar y mostrar en pantalla la candidatura ganadora

SOLUCION PROPUESTA:

```
def inicializar (lista, n):
    """ list, int -> None
        OBJ: Inicializa una lista a n elementos de valor 0"""
    for i in range(n):
         lista.append(0)
    return

def votacion_pedida(candidaturas):
    """ int -> list
        OBJ: Lista de resultados de la votación para cada una de las candidaturas"""
    lista_resultados = []
    fin_votacion = False
    inicializar(lista_resultados, len(candidaturas))
    while not fin_votacion:
        voto = input('Candidato al que vota (0 para terminar): ')
        if voto == '0':
            fin_votacion = True
        elif voto in candidaturas:
            posicion_voto = candidaturas.index(voto)
            lista_resultados[posicion_voto] += 1
        else:
            print('Voto nulo!!')
    return lista_resultados

def ganador(lista_votos):
  """lista-->int
  OBJ: Devuelve la posición del candidato con más votos """
    max = 0
    for i in lista_votos:
        if i > max:
            max = i
            candidato = lista_votos.index(i)
    return candidato

#PRUEBA
candidaturas = ['PP','PSOE','ERC','UP','CS','VOX','PNV']
resultados_votacion = votacion_pedida(candidaturas)
print(f'La candidatura ganadora es {candidaturas[ganador(resultados_votacion)]}')
```

Ejercicios propuestos

1. Escribir una función que sume dos listas de enteros de igual longitud y retorne otra lista que contenga la suma de las originales elemento a elemento.
2. Modifica la función anterior permitiendo que las listas sean desiguales. Los elementos sobrantes de la lista más larga se añadirán al final de la lista resultante.
3. Una lista de enteros original debe utilizarse para generar dos listas, una con los números pares de la original ordenados ascendentemente y otra con los impares ordenados descendentemente. La generación de las 2 listas debe hacerse a medida que se recorre la original, es decir, se toma un número de la original, se decide a qué lista (pares o impares) debe ir, y se inserta ordenado en la misma de acuerdo con el criterio de la lista (ascendente o descendente).
4. Crear una lista de enteros, inicializarlos según valores aleatorios en el rango 1..20 y computar la media de los valores, el valor más alto y el más bajo (todo ello utilizando listas).
5. Modificar una lista de números reales que representan las calificaciones de los alumnos de una clase, para sustituir los valores numéricos por sus calificaciones alfanuméricas (Suspenso, Aprobado, etc.)
6. Implementar una función que compruebe si una palabra es un palíndromo. Atención, no hagas más trabajo del necesario.
7. Implementar una función que pone en mayúsculas la primera letra de cada una de las palabras de una frase, sin usar el método `title()`.
8. Crear una función que compruebe si dos cadenas de caracteres son iguales, sin comparar las cadenas completas y sin usar el operador in.
9. Implemente una función que indique si una palabra contiene las cinco vocales: por ejemplo "murciélago". Modifique posteriormente la función para que detecte sólo aquellas palabras que contienen una única vez cada vocal.
10. Escriba un programa que "codifique" una frase modificando todas las vocales según el siguiente código: a por 4, e por 3, i por 1, o por 0 y u por el símbolo #. Por ejemplo, la frase: "*Un perro del hortelano*", deberá devolverse: "*#n p3rr0 d3l h0rt3l4n0*".
11. Un texto contiene comandos en forma de frases separadas por puntos. En cada frase, la primera palabra contiene el código de la operación y la última el resultado. Ejemplo:

    SUMAR 45 50 95. AND A B TRUE. MULT 10 20 200. Etc.

    Cree una lista de parejas [código-resultado] utilizando como entrada un texto con el formato indicado.
12. Escribir una función que permita mostrar los caracteres de una cadena del final al principio, pero nunca mostrando la letra "a". Ejemplo: si la entrada es "barco amarillo", la función devolverá: "ollirm ocrb".
13. Realizar un programa que lea palabras hasta que se introduzca "*fin*", mostrando un recuento de las longitudes de las palabras, es decir, el número total de palabras de longitud 1 que se hayan introducido, el total de longitud 2, etc. La máxima longitud de las palabras deberá ser de 15 caracteres. Una posible salida de este programa sería:

```
Palabras longitud 1: ninguna
Palabras longitud 2: 10
 …
Palabras longitud 15: 1
```