Guía de Ejercicios Prácticos 17

Sitio: Universidad Virtual UTN FRC
Curso: Algoritmos y Estructuras de Datos (2020)
Libro: Guía de Ejercicios Prácticos 17
Imprimido por: Luciana Lisette Montarce
Día: lunes, 23 de noviembre de 2020, 21:26

Descripción

Esta guía contiene enunciados de algunos ejercicios para aplicar los conceptos de programación en ***Python*** que se analizan en la ***Ficha 17***. Los alumnos no deben subir nada al aula virtual: la guía se propone como fuente de ejercicios generales. Se sugiere intentar resolver cada uno de estos problemas ya sea trabajando solos o en grupos de estudio, y cuando las soluciones se publiquen, controlar lo hecho con las sugerencias propuestas por sus docentes. Utilice el foro del curso para plantear dudas y consultas, que cualquier alumno puede intentar responder.

Tabla de contenidos

1. Torneo de Futbol

Desarrollar un programa para gestionar los puntos de un torneo de futbol.

Del torneo participan n equipos (n se ingresa por teclado). Por cada equipo, cargar 3 vectores paralelos con la siguiente información:

- equipos: nombres de los mismos
- puntos: total de puntos obtenidos en el torneo (máximo 20)
- goles: total de goles anotados en el torneo

La carga debe implementarse tanto en forma manual como automática.

El programa debe luego mostrar las siguientes estadísticas

- Tabla de Posiciones: ordenar el listado por orden descendente de puntaje y mostrarlo
- Punteros: mostrar el o los equipos que tienen la mayor cantidad de puntos del torneo
- Tabla de Descenso: Mostrar los últimos 5 equipos de la tabla de posiciones
- Mejor desempeño: mostrar (en orden alfabético) los equipos que anotaron más de x goles en el torneo, siendo x un valor que se carga por teclado
- Comparativo de goles: generar una tabla donde cada fila represente un equipo y cada columna una cantidad posible de puntos. La tabla debe contener la cantidad de goles realizados por el equipo. A partir de esa tabla, informar:
  - Total de goles para los equipos que tuvieron el máximo puntaje
  - Cantidad de equipos que no obtuvieron puntos ni marcaron goles

1.1. arreglos.py

```
__author__ = 'Catedra de Algoritmos y Estructuras de Datos'

def ordenar_seleccion_directa (equipos, puntos, goles):
    for i in range(len(puntos)-1):
        for j in range(i+1, len(puntos)):
            if puntos[i] < puntos[j]:
                equipos[i],equipos[j] = equipos[j],equipos[i]
                puntos[i],puntos[j] = puntos[j],puntos[i]
                goles[i],goles[j] = goles[j],goles[i]


def buscar_mayores(equipos, puntos):
    punteros = [equipos[0]]
    for i in range(1,len(puntos)):
        if puntos[i] == puntos[0]:
            punteros.append(equipos[i])
        else:
            break
    return punteros


def buscar_ultimos(v, cantidad):
    desde = len(v) - cantidad
    hasta = len(v)
    return v[desde:hasta]


def buscar_desde_goles(equipos, goles, minimo):
    buenos = list()
    for i in range(len(equipos)):
        if goles[i] > minimo:
            buenos.append(equipos[i])
    return buenos


def ordenar_insercion_directa(v):
    n = len(v)
    for j in range(1,n):
        y = v[j]
        k = j-1
        while k >= 0 and y < v[k]:
            v[k+1] = v[k]
            k -= 1
        v[k+1] = y


def mostrar_matriz(m):
    for c in range(len(m[0])):
        if c == 0:
            print('Puntos',end='\t')
        print(c,end='\t')
    for f in range(len(m)):
        print('\nEq',f,': ',end = '\t')
        for c in range(len(m[f])):
            print(m[f][c],end='\t')


def sumar_columna(tabla,col):
    suma = 0
    for f in range(len(tabla)):
        suma += tabla[f][col]
    return suma


def contar_ceros(tabla, col):
    cant = 0
    for f in range(len(tabla)):
```

```
        if tabla[f][col] == 0:
            cant += 1
    return cant
```

1.2. principal.py

```
from general import *
from arreglos import *
import random

__author__ = 'Cátedra de Algoritmos y Estructuras de Datos'


def cargar_manual(n):
    equipos = [] * n
    puntos = [0] * n
    goles = [0] * n
    for i in range(n):
        equipos[i] = input('Ingrese nombre del equipo: ')
        puntos[i] = validar_entre(0,20,'Ingrese puntos: ')
        goles[i] = validar_mayor_igual_que(0,'Ingrese goles: ')
    return equipos, puntos, goles


def cargar_automatico(n):
    equipos = [''] * n
    puntos = [0] * n
    goles = [0] * n
    for i in range(n):
        equipos[i] = 'Equipo' + str(i)
        puntos[i] = random.randint(0,20)
        goles[i] = random.randint(0,100)
    return equipos, puntos, goles


def mostrar(equipos, puntos, goles):
    print('Equipos','Puntos','Goles',sep='\t')
    for i in range(len(equipos)):
        print(equipos[i],puntos[i],goles[i],sep='\t\t')


def generar_tabla(puntos, goles):
    tabla = [[0] * 21 for f in range(len(puntos))]
    for i in range(len(puntos)):
        tabla[i][puntos[i]] = goles[i]
    return tabla



def test():
    print('TORNEO DE FUTBOL')
    print('=' * 80)
    n = validar_mayor_que(5,'Ingrese cantidad de equipos: ')
    carga = validar_entre(1, 2, 'Elija tipo de carga: 1) Manual - 2) Automatica: ')
    if carga == 1:
        equipos, puntos, goles = cargar_manual(n)
    elif carga == 2:
        equipos, puntos, goles = cargar_automatico(n)
    print('\nTABLA DE POSICIONES')
    ordenar_seleccion_directa(equipos, puntos, goles)
    mostrar(equipos, puntos, goles)
    print('\nPUNTEROS')
    print(buscar_mayores(equipos,puntos))
    print('\nDESCENSO')
    print(buscar_ultimos(equipos,5))
    print('\nMEJOR DESEMPEÑO')
    x = int(input('Ingrese cantidad minima de goles: '))
    buenos = buscar_desde_goles(equipos,goles,x)
    ordenar_insercion_directa(buenos)
    print(buenos)
    print('\nCOMPARATIVO DE GOLES')
    tabla = generar_tabla(puntos,goles)
    mostrar_matriz(tabla)
```

```
    print()
    print('Total de goles para equipos de máximo puntaje: ',sumar_columna(tabla,20))
    print('Equipos sin puntos ni goles: ', contar_ceros(tabla,0))

if __name__ == '__main__':
    test()
```

1.3. general.py

```
__author__ = 'Catedra de Algoritmos y Estructuras de Datos'

def validar_mayor_que(desde, mensaje):
    valor = int(input(mensaje))
    while valor <= desde:
        print('¡Valor invalido!')
        valor = int(input(mensaje))
    return valor


def validar_mayor_igual_que(desde, mensaje):
    valor = int(input(mensaje))
    while valor < desde:
        print('¡Valor invalido!')
        valor = int(input(mensaje))
    return valor


def validar_entre(desde, hasta, mensaje):
    valor = int(input(mensaje))
    while valor < desde or valor > hasta:
        print('Valor Invalido!!')
        valor = int(input(mensaje))
    return valor
```

2. Mejores alumnos

En un colegio secundario se maneja la información de las notas de una materia con tres vectores de longitud n, llamados trim1, trim2 y trim3. Esos vectores almacenan las notas correspondientes a cada trimestre para cada alumno. Asimismo se mantienen los nombres de los alumnos en otro vector paralelo a los de las notas.

Se necesita un programa que permita ingresar todos esos datos y que luego de la carga genere un cuarto vector que contenga el promedio de cada alumno. Finalmente el programa debe mostrar todos los datos de los tres alumnos con mejor promedio de la materia.

2.1. arreglos.py

```
__author__ = "Cátedra de Algoritmos y Estructuras de Datos"

def calcular_promedios(trim1, trim2, trim3):
    promedios = [0] * len(trim1)

    for i in range(len(trim1)):
        promedios[i] = (trim1[i] + trim2[i] + trim3[i]) / 3

    return promedios


def ordenar_descendente(nombres, trim1, trim2, trim3, promedios):
    n = len(nombres)
    for i in range(n - 1):
        for j in range(i + 1, n):
            if promedios[i] < promedios[j]:
                nombres[i], nombres[j] = nombres[j], nombres[i]
                trim1[i], trim1[j] = trim1[j], trim1[i]
                trim2[i], trim2[j] = trim2[j], trim2[i]
                trim3[i], trim3[j] = trim3[j], trim3[i]
                promedios[i], promedios[j] = promedios[j], promedios[i]
```

2.2. principal.py

```
__author__ = "Cátedra de Algoritmos y Estructuras de Datos"
from arreglos import *

def cargar_alumnos(n):
    nombres = [None] * n
    trim1 = [None] * n
    trim2 = [None] * n
    trim3 = [None] * n

    for i in range(n):
        print("\nCarga del alumno",i+1)
        nombres[i] = input("Ingrese nombre: ")
        trim1[i] = int(input("Ingrese nota del trimestre 1: "))
        trim2[i] = int(input("Ingrese nota del trimestre 2: "))
        trim3[i] = int(input("Ingrese nota del trimestre 3: "))

    return nombres, trim1, trim2, trim3

def mostrar_mejores(cantidad, nombres, promedios):
    print("\n\nLos",cantidad,"mejores promedios son: ")
    print("{:20} {:7}".format("Nombre","Promedio"))
    for i in range(cantidad):
        print("{:20} {:5.2f}".format(nombres[i],promedios[i]))


def principal():

    n = int(input("Ingrese la cantidad de alumnos: "))

    nombres, trim1, trim2, trim3 = cargar_alumnos(n)
    promedios = calcular_promedios(trim1, trim2, trim3)
    ordenar_descendente(nombres, trim1, trim2, trim3, promedios)
    mostrar_mejores(3, nombres, promedios)


if __name__ == "__main__":
    principal()
```

3. Veterinaria

Una franquicia veterinaria quiere llevar un control de las prácticas realizadas en las distintas sucursales de la cadena. Para lo cual solicita un programa que:

- Ingrese por teclado los datos de las prácticas realizadas. De cada práctica se conoce: tipo de práctica (0-9), sucursal (0-5), importe cobrado.
- Determine el monto recaudado por cada sucursal
- Determine cual fue la práctica que mas frecuentemente se realiza
- Determinar la cantidad de veces que se realizó cada práctica en cada sucursal.

3.1. veterinaria.py

```
__author__ = "Cátedra de Algoritmos y Estructuras de Datos"

def validar(inf, mensaje, sup = None):
    n = inf - 1
    while n < inf or (sup != None and n>sup):
        n = int(input(mensaje))
        if n < inf or (sup != None and n>sup):
            print('Valor incorrecto!')
    return n

def cargar_atenciones(n, practicas, sucursales, montos):
    for i in range(n):
        p = validar(0, 'Ingrese código de práctica [0-9]: ', 9)
        s = validar(0, 'Ingrese código de sucursal [0-5]: ', 5)
        m = validar(0, 'Ingrese monto en $: ')
        practicas[i] = p
        sucursales[i] = s
        montos[i] = m

def montos_sucursal(n, sucursales, montos):
    aux = [0] * 6
    for i in range(n):
        aux[sucursales[i]] += montos[i]
    return aux

def practica_frecuente(n, practicas):
    aux = [0] * 10
    for i in range(n):
        aux[practicas[i]] += 1

    may = aux[0]
    ind = 0
    for i in range(1,10):
        if aux[i]> may:
            may = aux[i]
            ind = i
    return (ind, may)

def conteo_suc_practicas(n, practicas, sucursales):
    mat = [[0] * 10 for i in range(6)]
    for i in range(n):
        fila = sucursales[i]
        columna = practicas[i]
        mat[fila][columna] += 1
    return mat

def main():
    n = validar(0, 'Ingrese cantidad de atenciones a procesar: ')
    practicas = [0]* n
    sucursales = [0]* n
    montos = [0]* n
    # punto 1: carga de vectores
    cargar_atenciones(n, practicas, sucursales, montos)

    # punto 2: Totales cobrados por sucursal (vector de acumulación)
    acu_montos = montos_sucursal(n,sucursales,montos)
    print('Montos acumulados por sucursal:',  acu_montos)

    # punto 3: Práctica más frecuente (vector de conteo y búsqueda del mayor)
    frec, cant = practica_frecuente(n, practicas)
    print('Práctica más frecuente:', frec, ' con', cant, ' veces.')

    # punto 4: Conteo de prácticas por sucursal (Matriz de conteo)
    mat = conteo_suc_practicas(n, practicas, sucursales)
    for f in range(6):
        for c in range(10):
            if mat[f][c] != 0:
```

```
                print('Práctica:', f, ' | Sucursal:', c, ' atenciones')

if __name__ == '__main__':
    main()
```