Guía de Ejercicios Prácticos - Ficha 16

Sitio: Universidad Virtual UTN FRC
Curso: Algoritmos y Estructuras de Datos (2020)
Libro: Guía de Ejercicios Prácticos - Ficha 16
Imprimido por: Luciana Lisette Montarce
Día: lunes, 23 de noviembre de 2020, 21:25

Descripción

Esta guía contiene enunciados de algunos ejercicios para aplicar los conceptos de programación en ***Python*** que se analizan en la ***Ficha 16***. 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. Empresa de colectivos

Una empresa de colectivos desea generar estadística sobre sus pasajeros. Sus líneas están enumeradas de 0 a n, y las paradas de 0 a m.

En primer lugar, se debe ingresar por teclado la cantidad de líneas de colectivo y el número de paradas a procesar.

Luego cargar una tabla de doble entrada, donde cada fila represente a una de las líneas y cada columna una parada, indicando la cantidad de pasajeros que subieron al colectivo.

Con esa información establecer:

1. Cuál fue el total de pasajeros transportado por cada una de las líneas
2. Cuál fue la cantidad promedio de pasajeros, para una parada que se ingresa por teclado
3. Cuál fue la menor cantidad de pasajeros, para una línea que se ingresa por teclado.
4. Cuál fue la recaudación de la empresa en el período analizado, sabiendo que el precio del boleto es de $8,5.

1.1. Solución - ModuloEjercicio1

```
__author__ = 'Algoritmos y Estructuras de Datos'
# Modulo ModuloEjercicio1

import random

def cargar_matriz_pasajeros(lineas, paradas):
    pasajeros = [[0] * paradas for i in range(lineas)]

    for i in range(lineas):
        print("Linea", i)
        for j in range(paradas):
            cantidad = random.randint(0,20)
            #cantidad = int(input("Cantidad de pasajeros en la parada " + str(j) + ": "))
            pasajeros[i][j] = cantidad

    return pasajeros


def pasajeros_por_linea(matriz, lineas, paradas):
    cont = [0] * lineas
    for i in range(lineas):
        for j in range(paradas):
            cont[i] += matriz[i][j]
    return cont


def promedio_por_parada(parada, matriz, lineas):
    suma = 0
    for i in range(lineas):
        suma += matriz[i][parada]
    return suma / lineas


def total_pasajeros(matriz):
    suma = 0
    for i in range(len(matriz)):
        for j in range(len(matriz[i])):
            suma += matriz[i][j]

    return suma

def parada_menor(linea_menor, matriz):
     par_menor =0
     men = matriz[linea_menor][0]
     for j in range(1,len(matriz[linea_menor])): # j toma cada valor parada de la linea_menor
         if matriz[linea_menor][j] < men :
              par_menor = j
              men = matriz[linea_menor][j]

     return par_menor
```

1.2. Solución - Principal

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

from ModuloEjercicio1 import *

def main():
    lineas = int(input("Ingrese cantidad de líneas: "))
    paradas = int(input("Ingrese la cantidad de paradas: "))

    # Carga de datos
    matriz = cargar_matriz_pasajeros(lineas, paradas)
    print(matriz)

    # Punto 1
    total_por_linea = pasajeros_por_linea(matriz, lineas, paradas)
    for i in range(len(total_por_linea)):
        print("La cantidad de pasajeros de la linea", i, "fue de: ", total_por_linea[i])

    # Punto 2
    parada = int(input("Ingrese parada a consultar: "))
    promedio = promedio_por_parada(parada, matriz, lineas)
    print("El promedio de la parada es de: ", promedio)

    # Punto 3

    # menor cantidad de pasajeros en una linea dada
    linea_menor = int(input("Ingrese linea a controlar: "))
    parada_menor_pasajeros = parada_menor(linea_menor, matriz)
    print("La parada de la linea ", linea_menor, "con menor cantidad de pasajeros es : ", parada_menor_pasajeros, "con
:",matriz[linea_menor][ parada_menor_pasajeros], "Pasajeros")

    # Punto 4
    total = total_pasajeros(matriz)
    print("La recaudación fue de: $", (total * 8.5))


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

2. Estadística con Dados (continuación)

Hace algunas semanas se desarrolló una aplicación estadística, lanzando simultáneamente 2 dados y anotando en 2 vectores paralelos los resultados obtenidos, para n lanzamientos (n se ingresa por teclado).

El mismo cliente nos pide ahora que agreguemos la generación de una tabla de doble entrada, donde cada fila represente uno de los dados y cada columna uno de sus valores posibles. La tabla debe contar cuantas veces apareció cada valor en los lanzamientos efectuados.

El programa deberá:

- Mostrar la tabla generada
- Indicar qué valores aparecieron la misma cantidad de veces en ambos dados
- Informar cuál fue el número con más apariciones para cada dado

2.1. Solución

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

import random

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


def cargar(n):
    dado1 = list()
    dado2 = list()
    for i in range(n):
        dado1.append(random.randint(1,6))
        dado2.append(random.randint(1,6))
    return dado1, dado2


def generar_tabla(dado1, dado2):
    tabla = [[0]*7 for f in range(2)]
    for i in range(len(dado1)):
        tabla[0][dado1[i]] += 1
        tabla[1][dado2[i]] += 1
    return tabla


def mostrar_tabla(tabla):
    print('Valores',end=' ')
    for i in range(1,7):
        print(i,end= '\t')
    for fila in range(len(tabla)):
        print('\nDado',fila,':',end=' ')
        for col in range(1,len(tabla[fila])):
            print(tabla[fila][col],end='\t')


def buscar_mayor(tabla, fila):
    mayor = 0
    for col in range(1,len(tabla[fila])):
        if tabla[fila][col] > tabla[fila][mayor]:
            mayor =  col
    return mayor


def buscar_iguales(tabla):
    iguales = list()
    for col in range(1,len(tabla[0])):
        if tabla[0][col] == tabla [1][col]:
            iguales.append(col)
    return iguales


def main():
    print('ESTADÍSTICAS CON DADOS')
    n = validar_mayor_que(0,'Ingrese cantidad de lanzamientos: ')
    dado1, dado2 = cargar(n)
    print('Dado 1:',dado1)
    print('Dado 2:',dado2)
    print('-' * 80)
    print('Repeticiones')
    tabla = generar_tabla(dado1,dado2)
    mostrar_tabla(tabla)
    print('-' * 80)
    iguales = buscar_iguales(tabla)
```

```
    print('Valores que aparecieron la misma cantidad de veces:',iguales)
    mayor1 = buscar_mayor(tabla, 0)
    mayor2 = buscar_mayor(tabla, 1)
    print('\nEl valor más repetido en el dado 1 fue:',mayor1)
    print('El valor más repetido en el dado 2 fue:',mayor2)


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

3. Major League Baseball

La Major League Baseball (MLB) nos solicito un programa que permita obtener estadisticas de los jugadores que se encuentran jugando en sus equipos. De cada jugador se conoce el nombre, la posicion en la que juegan (un valor de 1 al 9) y la cantidad de hits que han pegado, es decir la cantidad de veces que llegaron a primera base. En base a esto se pide generar un arreglo de n jugadores y a partir de alli, con un menu de opciones resolver los siguientes puntos

Una vez cargado realizar los siguientes puntos:

a) Obtener las estadisticas de hits de posiciones por mes, para ellos armar un matriz de n x m, donde las filas son las posiciones y las columnas la cantidad de hits (un valor hasta 20), y contar la cantidad de jugadores que batearon esta cantidad de veces

b) Mostrar la cantidad total de hits para un posicion x pasada por parametro (usar la matriz)

c) Acumular por posicion (vector de acumulacion) el porcentaje de bateo de los bateadores, sobre 100 tunos al bate posibles en el mes

d) Obtener el nombre del jugador con el mejor porcentaje de bateo, sobre 100 turnos al bat posibles en el mes

3.1. Solucion

```
from registro import *
import random

__author__ = 'Algoritmos y Estructuras de Datos'


def menu():
    menu = 'Menu de Opciones \n' + \
           '=' * 60 + '\n' + \
           '1  ---------- Cargar Jugadores\n' + \
           '2  ---------- Generar hits por posiciones (matriz)\n' + \
           '3  ---------- Mostrar Hits por posiciones\n' + \
           '4  ---------- Acumular porcentajes de efectividad por posicion\n' + \
           '5  ---------- Nombre del jugador con mejor porcentaje de bateo\n' + \
           '0  ---------- Salir\n' + \
           'Ingrese su opcion: '
    return int(input(menu))


def validar_mayor_que(minimo, mensaje='Ingrese un numero'):
    numero = minimo - 1
    while numero <= minimo:
        numero = int(input(mensaje))
        if numero <= minimo:
            print('Error !!!! El valor ingresado de ser mayor a ', minimo)
    return numero


def validar_rango(desde, hasta, mensaje='Ingrese un numero: '):
    numero = desde - 1
    while numero <= desde or numero >= hasta:
        numero = int(input(mensaje))
        if numero <= desde or numero >= hasta:
            print('Error cargue de nuevo, valores de ', desde, 'a', hasta)
    return numero


# Press the green button in the gutter to run the script.
def validar_si_existe_jugador(vector, nombre):
    for elemento in vector:
        if elemento.nombre == nombre:
            return 1
    return 0


def cargar_nombre(vector):
    nombres = ['pra', 'con', 'lis', 'cam', 'man', 'are', 'atro', 'pac', 'ino']
    existe = 1
    nombre = ''
    while existe == 1:
        nombre = random.choice(nombres) + random.choice(nombres) + random.choice(nombres) + ' ' + \
                 random.choice(nombres) + random.choice(nombres) + random.choice(nombres)
        existe = validar_si_existe_jugador(vector, nombre)
        if existe == 1:
            print('Error!!! Ese jugador ya se encuentra cargado, se va a generar uno nuevo')
    return nombre


def cargar_vector(vector, n):
    for i in range(n):
        nombre = cargar_nombre(vector)
        posicion = random.randint(1, 9)
        hits = random.randint(0, 20)
        jugador = Pelotero(nombre, posicion, hits)
        vector.append(jugador)
```

```
def display(vector):
    print('Listado de Jugadores de Beisbol')
    print('-' * 60)
    for elemento in vector:
        print(to_string(elemento))


def cargar_estadistica_bateo(vector):
    matriz = [[0] * 20 for i in range(9)]
    for elemento in vector:
        fila = elemento.posicion - 1
        columna = elemento.hits - 1
        matriz[fila][columna] += 1
    return matriz


def cantidad_total(mat, pos):
    total = 0
    fila = pos - 1
    for j in range(len(mat[fila])):
        total += mat[fila][j]
    return total


def acumular_por_posicion(vector):
    va = [0] * 9
    for elemento in vector:
        pos = elemento.posicion - 1
        va[pos] += (elemento.hits / 100) * 100
    return va


def buscar_mas_bateador(vector):
    mayor = vector[0]
    for i in range(1, len(vector)):
        if vector[i].hits > mayor.hits:
            mayor = vector[i]
    return mayor.nombre


def principal():
    opcion = -1
    jugadores = []
    matriz_generada = False

    while opcion != 0:
        opcion = menu()
        if opcion == 1:
            n = validar_mayor_que(0, 'Ingrese la cantidad de jugadores: ')
            cargar_vector(jugadores, n)
            display(jugadores)
        else:
            if len(jugadores) > 0:
                if opcion == 2:
                    matriz_generada = True
                    mat = cargar_estadistica_bateo(jugadores)
                    for i in range(len(mat)):
                        for j in range(len(mat[i])):
                            print('m[' + str(i) + '][' + str(j) + '] = ' + str(mat[i][j]))

                elif opcion == 3:
                    if matriz_generada:
                        pos = validar_rango(1, 9, 'Ingrese la posicion que quiere ver: ')
                        total = cantidad_total(mat, pos)
                        print('La cantidad total de hits para la posicion', pos, 'fue de', total)
                    else:
                        print('Primero tiene que contar por posicion y cantidad de hits (opcion 2)')
```

```
                elif opcion == 4:
                    va = acumular_por_posicion(jugadores)
                    for i in range(len(va)):
                        print('El porcentaje de bateo para la posicion', i + 1, 'fue de', va[i], '%')

                elif opcion == 5:
                    nom = buscar_mas_bateador(jugadores)
                    print('El nombre del jugador que mas palos pego es', nom)
            else:
                print('Debe cargar los jugadores para poder ejecutar las opciones')


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

3.2. Registro

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


class Pelotero:
    def __init__(self, nombre, posicion, hits):
        self.nombre = nombre
        self.posicion = posicion
        self.hits = hits


def to_string(jugador):
    return 'Jugador ' + jugador.nombre + \
           ' juega en la posicion ' + str(jugador.posicion) + \
           ' y ha llegado a primera base ' + str(jugador.hits) + ' veces'
```

4. Analisis de Proyectos

Una empresa de tecnología solicito un programa que permita obtener estadísticas sobre el desempeño de sus proyectos.

Para lograrlo informo que actualmente tiene n proyectos en cursos, y para cada proyecto hay m roles involucrados, haciendo la salvedad que todos los roles son ocupados en los proyectos.

Usted deberá generar una tabla o matriz solicitando la cantidad de proyectos que tiene en ese momento la empresa, junto con la cantidad de roles que la empresa involucra en los proyectos, cada componente de la matriz sera la cantidad de horas que insumió para un proyecto un rol.

A partir de alli, determinar:

a) Para un rol ingresado por parámetro, indicar el total de horas que tuvo en los proyectos de la empresa

b) Saber la cantidad de horas que un proyecto x llevo

c) Las horas promedios para un rango de proyectos solicitados al usuario

d) Sabiendo que se cobra un promedio de $175 la hora, obtener el total que se le cobrar a cada cliente por proyecto (n contadores)

4.1. Solución

```
from random import randint

__author__ = 'Catedra de Algoritmos y Estructuras de Datos'


def validar_mayor(limite, mensaje):
    valor = int(input(mensaje))
    while valor <= limite:
        print('Error!!! El valor ingresado debe ser >', limite)
        valor = int(input(mensaje))
    return valor


def validar_rango(inferior, superior, mensaje):
    numero = int(input(mensaje))
    while numero < inferior or numero > superior:
        print('Error! El numero ingresado debe estar comprandido en el rango [', inferior, ':', superior, ']')
        numero = int(input(mensaje))
    return numero


def menu():
    print('Menu de Opciones')
    print('-' * 90)
    print('1 - Total de horas para un rol')
    print('2 - Total de horas para un proyecto')
    print('3 - Promedio de Horas para Rango de Proyectos')
    print('4 - Total a Cobrar por proyecto')
    print('5 - Salir')
    return int(input('Ingrese su opcion: '))


def generar_matriz(filas, columnas):
    m = [[0] * columnas for _ in range(filas)]
    for f in range(filas):
        for c in range(columnas):
            m[f][c] = validar_mayor(0, 'Ingrese la cantidad de horas para el proyecto ' + str(f) + ' del rol' + str(c))
    return m


def generar_matriz_aleatoria(filas, columnas):
    m = [[0] * columnas for _ in range(filas)]
    for f in range(filas):
        for c in range(columnas):
            m[f][c] = randint(1, 60)
    return m


def totalizar_rol(matriz, rol):
    total = 0
    for f in range(len(matriz)):
        total += matriz[f][rol]
    return total


def totalizar_proyecto(matriz, proyecto):
    total = 0
    for c in range(len(matriz[proyecto])):
        total += matriz[proyecto][c]
    return total


def totalizar_horas(matriz, desde, hasta):
    total = 0
    for f in range(desde, hasta + 1):
        for c in range(len(matriz[desde])):
            total += matriz[f][c]
```

```
    return total


def costo_total_por_proyecto(matriz):
    va = [0] * len(matriz)
    for proy in range(len(matriz)):
        va[proy] = totalizar_proyecto(matriz, proy) * 175
    return va


def test():
    print('Empresa de Tecnologia - Analisis de Proyectos')
    print('_' * 90)

    n = validar_mayor(0, 'Ingrese la cantidad de proyectos de la empresa: ')
    m = validar_mayor(0, 'Ingrese la cantidad de roles que tiene la empresa: ')

    tipo_carga = input('Desea generar la matriz en forma (M)anual o (A)utomatica? ')
    if tipo_carga == 'M':
        matriz = generar_matriz(n, m)
    else:
        matriz = generar_matriz_aleatoria(n, m)

    opcion = 0
    while opcion != 5:
        opcion = menu()
        if opcion == 1:
            rol = validar_rango(0, m, 'Ingerse el rol que desea totalizar: ')
            total = totalizar_rol(matriz, rol)
            print('El total de horas cargadas por el rol', rol, 'fueron:', total)

        elif opcion == 2:
            proyecto = validar_rango(0, n, 'Ingrese el proyecto que desea totalizar: ')
            total = totalizar_proyecto(matriz, proyecto)
            print('El total de horas cargadas por el proyecto', proyecto, 'fueron:', total)

        elif opcion == 3:
            proy_desde = validar_rango(0, n, 'Ingrese el proyecto que desea totalizar: ')
            proy_hasta = validar_rango(proy_desde, n, 'Ingrese el proyecto que desea totalizar: ')
            total_horas = totalizar_horas(matriz, proy_desde, proy_hasta)
            print('Las horas promedio para los proyecto comprendidos entre', proy_desde, 'y', proy_hasta, end=' ')
            print('fue de ', round(total_horas / (proy_hasta - proy_desde), 2))

        elif opcion == 4:
            va = costo_total_por_proyecto(matriz)
            for i in range(len(va)):
                print('El total a cobrar para el proyecto ', i, ' sera de $', va[i], sep='')


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

5. Cobro de Servicios

Desarrollar un programa que permita cargar, para una empresa de cobro de servicios, el detalle de n cobranzas realizadas (n se carga por teclado) conteniendo la siguiente información: número de caja (0-9 inclusive), turno (0=mañana, 1= tarde, 2 = noche) y monto cobrado.

A medida que se cargan los datos, se pide generar una tabla resumen, dando a elegir al usuario una carga manual o automática, donde cada fila represente una caja y cada columna un turno, totalizando los montos cobrados.

Mostrar el contenido de la tabla, y luego informar:

- Recaudación promedio, para una caja que se ingresa por teclado
- Total recaudado por cada turno, cuál es el mayor y qué porcentaje representa sobre la recaudación total.
- Qué caja y turno tuvo la menor recaudación acumulada

5.1. principal.py

```
from general import *
from vector import *
import random

__author__ = 'Catedra de Algoritmos y Estructuras de Datos'


def cargar_manual(n):
    tabla = [[0] * 3 for f in range(10)]
    for i in range(n):
        caja = validar_entre(0, 9, 'Ingrese el numero de caja: ')
        turno = validar_entre(0, 2, 'Ingrese el turno: ')
        monto = validar_mayor_que(0, 'Ingrese el monto a cobrar del servicio: ')
        tabla[caja][turno] += monto
    return tabla


def cargar_automatico(n):
    tabla = [[0] * 3 for f in range(10)]
    for i in range(n):
        caja = random.randrange(10)
        turno = random.randrange(3)
        monto = random.randint(100, 1000)
        tabla[caja][turno] += monto
    return tabla


def mostrar(tabla):
    print('Turno', 'Mañana', 'Tarde', 'Noche', sep='\t', end='')
    for f in range(len(tabla)):
        print('\nCaja ', f, ': ', sep='', end='')
        for c in range(len(tabla[f])):
            print('\t$', tabla[f][c], sep='', end=' ')


def calcular_promedio(tabla, caja):
    total = 0
    turnos = len(tabla[caja])
    for col in range(turnos):
        total += tabla[caja][col]
    return round(total / turnos, 2)


def totalizar_turnos(tabla):
    va = [0] * len(tabla[0])
    for col in range(len(tabla[0])):
        for fila in range(len(tabla)):
            va[col] += tabla[fila][col]
    return va


def menor_recaudacion(tabla):
    menor = tabla[0][0]
    caja, turno = 0, 0
    for fila in range(len(tabla)):
        for col in range(len(tabla[fila])):
            if tabla[fila][col] < menor:
                menor = tabla[fila][col]
                caja, turno = fila, col
    return caja, turno


def test():
    n = validar_mayor_que(0, 'Ingrese la cantidad de servicios: ')

    tipo_carga = input('Ingrese si la carga sera (M)anual o (A)utomatica: ')
    if tipo_carga == 'M':
```

```
        tabla = cargar_manual(n)
    else:
        tabla = cargar_automatico(n)

    mostrar(tabla)

    opcion = -1
    while opcion != 0:
        print('Menu de Opciones')
        print('1 - Recaudacion Promedio de una Caja')
        print('2 - Total Recaudado por turno')
        print('3 - Caja/Turno con menor recaudacion')
        print('0 - Salir')
        opcion = int(input('Ingrese su opcion: '))
        if opcion == 1:
            caja = validar_entre(0, 9, 'Ingrese la caja a calcular el promedio recaudado: ')
            promedio = calcular_promedio(tabla, caja)
            print('El promedio recaudado para la caja', caja, 'es', promedio)

        elif opcion == 2:
            vec_total_turnos = totalizar_turnos(tabla)
            mayor = buscar_mayor(vec_total_turnos)
            total = sumar_vector(vec_total_turnos)
            porcentaje = calcular_porcentaje(mayor, total)
            print('Total por turnos:', vec_total_turnos)
            print('El mayor monto fue', mayor, 'y representa el', porcentaje, '% del total')

        elif opcion == 3:
            caja, turno = menor_recaudacion(tabla)
            print('La menor recaudacion fue de la caja', caja, 'en el turno', turno)


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

5.2. vector.py

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


def buscar_mayor(v):
    mayor = v[0]
    for i in range(1, len(v)):
        if v[i] > mayor:
            mayor = v[i]
    return mayor


def sumar_vector(v):
    total = 0
    for valor in v:
        total += valor
    return total
```

5.3. 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_entre(desde, hasta, mensaje):
    valor = int(input(mensaje))
    while valor < desde or valor > hasta:
        print('Valor Invalido!!')
        valor = int(input(mensaje))
    return valor


def calcular_porcentaje(cantidad, total):
    porc = 0
    if total != 0:
        porc = cantidad * 100 / total
    return round(porc, 2)
```

6. Recorridos en matriz rectangular

Hacer un programa que permita cargar una matriz de m x n números enteros, con m y n cargados previamente. Luego de la carga calcular y mostrar:

- Suma de todos los números
- Suma de cada fila
- Suma de cada columna
- Suma del contorno
- Suma de la mitad de arriba
- Suma del cuarto inferior derecho

6.1. Solución

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

def cargar_matriz(f, c):
    mat = [[0]*c for i in range(f)]
    for i in range(f):
        for j in range(c):
            print("Ingrese fila {0}, columna {1}".format(i,j), end=" ")
            mat[i][j] = int(input())
    return mat

def suma_todo(mat):
    suma = 0
    for i in range(len(mat)):
        for j in range(len(mat[i])):
            suma += mat[i][j]
    return suma

def suma_por_filas(mat):
    f = len(mat)
    suma = [0] * f
    for i in range(f):
        for j in range(len(mat[i])):
            suma[i] += mat[i][j]
    return suma


def suma_por_columnas(mat):
    c = len(mat[0])
    suma = [0] * c
    for i in range(len(mat)):
        for j in range(c):
            suma[j] += mat[i][j]
    return suma


def suma_contorno(mat):
    f = len(mat)
    c = len(mat[0])
    suma = 0
    for i in range(f):
        suma += mat[i][0] + mat[i][c-1]
    for j in range(1,c-1):
        suma += mat[0][j] + mat[f-1][j]
    return suma

def suma_mitad_arriba(mat):
    f = len(mat)
    suma = 0
    for i in range(f//2):
        for j in range(len(mat[i])):
            suma += mat[i][j]
    return suma

def suma_esquina_inferior_derecha(mat):
    f = len(mat)
    c = len(mat[0])
    suma = 0
    for i in range(f//2, f):
        for j in range(c//2, c):
            suma += mat[i][j]
    return suma

def test():
    f = int(input("Ingrese cantidad de filas"))
    c = int(input("Ingrese cantidad de columnas"))

    matriz = cargar_matriz(f,c)
```

```
    print("Suma de todo",suma_todo(matriz))
    print("Suma por filas",suma_por_filas(matriz))
    print("Suma por columnas",suma_por_columnas(matriz))
    print("Suma del contorno",suma_contorno(matriz))
    print("Suma mitad de arriba",suma_mitad_arriba(matriz))
    print("Suma de la esquina",suma_esquina_inferior_derecha(matriz))


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