Guía de Ejercicios Prácticos 21

Sitio: Universidad Virtual UTN FRC
Curso: Algoritmos y Estructuras de Datos (2020)
Libro: Guía de Ejercicios Prácticos 21
Imprimido por: Luciana Lisette Montarce
Día: lunes, 23 de noviembre de 2020, 21:29

Descripción

Esta guía contiene enunciados de algunos ejercicios para aplicar los conceptos de programación en ***Python*** que se analizan en la ***Ficha 21***. 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. Streaming Audiovisual

Una empresa de administracion de contenido audiovisual dese un programa que le permita obtener una serie de estadisticas sobre los contenidos que publica y reproduce por streaming. De cada contenido se sabe Nombre, duracion, tipo (0 - Serie, 1 - Pelicula), Clasificacion (valor de 0 a 19), cantidad de capitulos (0 si es pelicula, mayor a cero si es serie)

Se pide un programa controlado por menu de opciones que permita generar un arreglo con n contenidos (en forma manual o automaticca) y resolver los siguientes items:

1. Listar el arreglo en forma alfabetica, mostrando el nombre del tipo que es. Es decir el Contenido A del Tipo 0 debe mostrar Contenido A Tipo Serie...
2. Determinar el total de capitulos de las series con una clasificacion mayor a un valor X ingresado por el usuario
3. Determinar y mostrar el total de duracion que hay por cada tipo de contenido y clasificacion (40 contadores)
4. Determinar cual es el porcentajes de peliculas que superan la duracion promedio de todos los contenidos
5. buscar un contenido por su nombre, si existe mostrar sus datos, caso contrario indicarlo con un mensaje que no se ha encontrado la serie o pelicula

1.1. contenido.py

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


class Contenido:
    def __init__(self, nombre, duracion, tipo, clasificacion, capitulos):
        self.nombre = nombre
        self.duracion = duracion
        self.tipo = tipo
        self.clasificacion = clasificacion
        self.capitulos = capitulos


def to_string(contenido):
    linea = 'Nombre: {}\t'.format(contenido.nombre)
    linea += 'Duracion: {:.2f}\t'.format(contenido.duracion)
    linea += 'Tipo de Contenido: {}\t'.format(to_string_tipo(contenido.tipo))
    linea += 'Clasificacion: {}\t'.format(contenido.clasificacion)
    linea += 'Cant Capitulos: {}\n'.format(contenido.capitulos)
    return linea


def to_string_tipo(tipo):
    des = ['Serie', 'Pelicula']
    return des[tipo]
```

1.2. main.py

```
__author__ = 'Catedra de Algoritmos y Estructuras de Datos'
import random
from contenido import *


def validar_mayor_a(minimo, mensaje):
    error = 'Error el valor debe ser mayor a {0}. {1}'.format(minimo, mensaje)
    numero = int(input(mensaje))
    while numero <= minimo:
        numero = int(input(error))
    return numero


def validar_rango(minimo, maximo, mensaje):
    error = 'Error el valor debe entre {0} y {1}. {2}'.format(minimo, maximo, mensaje)
    numero = int(input(mensaje))
    while numero < minimo or numero > maximo:
        numero = int(input(error))
    return numero


def carga_manual(n):
    v = [None] * n
    for i in range(n):
        nombre = input('Ingrese el nombre del contenido: ')
        duracion = float(input('Ingrese la duracion del contenido: '))
        tipo = validar_rango(0, 1, 'Ingrese el tipo de contenido: ')
        clasificacion = validar_rango(0, 19, 'Ingrese el tipo de clasificacion: ')
        capitulos = 0
        if tipo == 0:
            capitulos = validar_mayor_a(0, 'Ingrese la cantidad de capitulos: ')
        v[i] = Contenido(nombre, duracion,tipo, clasificacion, capitulos)
    return v

def carga_automatica(n):
    nom = ['Pulp', 'Fiction', 'Vida', 'Bella', 'Club', 'Pelea', 'Bastardos', 'American', 'History',
           'Gloria', 'Samurai', 'Señor', 'Anillos', 'Padrino', 'Gladiador', 'Caballero', 'Oscuro']
    v = [None] * n
    for i in range(n):
        nombre = '{0} {1}'.format(random.choice(nom), random.choice(nom))
        duracion = random.uniform(1, 4)
        tipo = random.randrange(2)
        clasificacion = random.randrange(20)
        capitulos = 0
        if tipo == 0:
            capitulos = random.randint(1, 45)
            duracion = random.uniform(15, 50)
        v[i] = Contenido(nombre, duracion,tipo, clasificacion, capitulos)

    return v


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


def listar_vector(vector):
    listado = 'Lista de Contenidos Audiovisuales.\n {}'.format('-' * 70)
    for contenido in vector:
        listado += '\n {}'.format(to_string(contenido))
```

```
    return listado


def totalizar_capitulos_series(vector, clasif):
    total = 0
    for contenido in vector:
        if contenido.tipo == 0 and contenido.clasificacion > clasif:
            total += contenido.capitulos
    return total


def generar_matriz(vector):
    ma = [[0] * 20 for i in range(2)]
    for contenido in vector:
        ma[contenido.tipo][contenido.clasificacion] += contenido.duracion
    return ma


def calcular_promedio_duracion(vector):
    total = 0
    for contenido in vector:
        total += contenido.duracion
    return total / len(vector)


def contar_peliculas_arriba_promedio(vector, prom):
    cantidad = 0
    for contenido in vector:
        if contenido.tipo == 1 and contenido.duracion > prom:
            cantidad += 1
    return cantidad


def buscar(vector, nombre):
    pos = -1
    for i in range(len(vector)):
        if vector[i].nombre == nombre:
            pos = i
            break
    return pos


def main():
    menu = 'Menu de Opciones\n' \
           '================================================\n' \
           '1\t Listar en forma alfabetica los contenidos\n' \
           '2\t Total de Series mayor a una clasificacion\n' \
           '3\t Mostrar duracion por tipo y clasificacion\n' \
           '4\t Determinar porcentaje pelicula mayor promedio\n' \
           '5\t Buscar contenido\n' \
           '0\t Salir\n' \
           'Ingese su opcion: '

    n = validar_mayor_a(0, 'Ingrese la cantidad de contenidos a generar: ')
    r = input('Desea generar el vector en forma Manual (M) o Automatica (A):')
    vector = None
    if r.upper() == 'M':
        vector = carga_manual(n)
    elif r.upper() == 'A':
        vector = carga_automatica(n)

    opcion = -1
    while opcion != 0:
        opcion = int(input(menu))
        if opcion == 1:
            ordenar_vector(vector)
            print(listar_vector(vector))
```

```
        elif opcion == 2:
            clasif = validar_rango(0, 19, 'Ingrese la clasificacion a comparar: ')
            total = totalizar_capitulos_series(vector, clasif)
            print('El total de capitulos para series con clasificacion mayor a {} es {}'.format(clasif, total))

        elif opcion == 3:
            ma = generar_matriz(vector)
            linea = 'Tipo Contenido {} Clasificacion {} Total de Duracion {:.2f}'
            for f in range(len(ma)):
                for c in range(len(ma[f])):
                    print(linea.format(to_string_tipo(f), c, ma[f][c]))

        elif opcion == 4:
            prom = calcular_promedio_duracion(vector)
            cant = contar_peliculas_arriba_promedio(vector, prom)
            porc = cant * 100 / len(vector)
            print('Hay {} peliculas por arriba del promedio de duracion'.format(cant), end= ' ')
            print('y representan el {:.2f}% del total de contenidos'.format(porc))

        elif opcion == 5:
            nombre = input('Ingrese el nombre de la pelicula a buscar: ')
            pos = buscar(vector, nombre)
            if pos != -1:
                print(to_string_tipo(vector[pos]))
            else:
                print('No existe el contenido con el nombre ingresdo!!! ')

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

2. Asociación Deportiva

Un asociación deportiva desea almacenar la información referida a sus n socios en un arreglo de registros (cargar n por teclado). Por cada socio, se pide guardar su número de identificación, su nombre, el arancel que paga cada mes y un código entre 0 y 9 para indicar el deporte que cada socio practica (suponiendo que por ejemplo, el 0 puede ser fútbol, el 1 básquet, y así hasta el código 9).

Se pide desarrollar un programa en Python controlado por un menú de opciones. Ese menú debe permitir gestionar las siguientes tareas a partir del arreglo pedido en el párrafo anterior:

1. Cargar el arreglo pedido con los datos de n socios, Solo valide que el código del deporte practicado, para asegurar que esté entre 0 y 9
2. Mostar los datos de los socios que paguen un arancel mayor a *p,* siendo *p* un valor que se carga por teclado
3. Determinar y mostrar cuántos socios practican cada tipo de deporte posible (un contador para contar cuántos socios practican el deporte 0, otro para los que practican el deporte 1, etc.)
4. Mostrar todos los datos, ordenados de menor a mayor por número de identificación.
5. Determinar si existe algún socio cuyo nombre sea igual a x, siendo x una cadena que se carga por teclado. Si existe, cambiar el valor del campo arancel de forma de sumarle un valor fijo de 100 pesos, y mostrar todos los datos de ese socio por pantalla. Si no existe, informar con un mensaje.

2.1. socio.py

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


class Socio:
    def __init__(self, numero, nombre, arancel, tipo_deporte):
        self.numero = numero
        self.nombre = nombre
        self.arancel = arancel
        self.tipo_deporte = tipo_deporte


def to_string(socio):
    linea = 'Socio Numero: {:<12}\t'.format(socio.numero)
    linea += 'Nombre: {:<20}\t'.format(socio.nombre)
    linea += 'Arancel: ${:<10.2f}'.format(socio.arancel)
    linea += 'Tipo Deporte: {:<2}\n'.format(socio.tipo_deporte)
    return linea
```

2.2. principal.py

```
__author__ = 'Catedra de Algoritmos y Estructuras de Datos'
from socio import *
import random


def validar_mayor_a(minimo, mensaje):
    numero = int(input(mensaje))
    while numero <= minimo:
        print('Error el numero deber ser mayor a ', minimo)
        numero = int(input(mensaje))
    return numero


def validar_rango(minimo, maximo, mensaje):
    numero = int(input(mensaje))
    while numero < minimo or numero > maximo:
        print('Error el valor debe estar entre {} y {}'.format(minimo, maximo))
        numero = int(input(mensaje))
    return numero


def cargar_vector(tam):
    vector = [None] * tam
    for i in range(tam):
        numero = int(input('Ingrese el numero del socio: '))
        nombre = input('Ingrese el nombre del socio: ')
        arancel = float(input('Ingrese el arancel del socio: '))
        tipo = validar_rango(0, 9, 'Ingrese el tipo de deporte: ')
        vector[i] = Socio(numero, nombre, arancel, tipo)
    return vector

def cargar_automatico(tam):
    nom = ['Jose', 'Carlos', 'Juan', 'Karina', 'Analia', 'Romina', 'Josefina', 'Ismael', 'Alberto', 'Nicolas']
    ape = ['Garcia', 'Perez', 'Rodriguez', 'Delarpueba' , 'De La Prueba', 'Prueba', 'Pruebita']
    vector = [None] * tam
    for i in range(tam):
        numero = random.randint(1, 1500)
        nombre = random.choice(nom) + ' ' + random.choice(ape)
        arancel = random.uniform(150, 450)
        tipo = random.randrange(10)
        vector[i] = Socio(numero, nombre, arancel, tipo)
    return vector


def buscar_mayor_arancel(vector, p):
    v = []
    for socio in vector:
        if socio.arancel > p:
            v.append(socio)
    return v


def mostrar_vector(vector):
    listado = 'Listado de Socios\n {}\n'.format('===' *  30)
    for socio in vector:
        listado += to_string(socio)
    return listado


def contar_socio_por_deporte(vector):
    vc = [0] * 10
    for socio in vector:
        vc[socio.tipo_deporte] += 1
    return vc


def ordenar_vector(vector):
```

```
    tam = len(vector)
    for j in range(1, tam):
        y = vector[j]
        k = j - 1
        while k >= 0 and y.numero > vector[k].numero:
            vector[k + 1] = vector[k]
            k -= 1
        vector[k + 1] = y


def buscar(vector, nombre):
    for i in range(len(vector)):
        if vector[i].nombre == nombre:
            return i
    return -1


def principal():
    menu = 'Menu de Opciones\n' \
           '============================================\n' \
           '1\t Cargar arreglo con n socios\n' \
           '2\t Mostrar socios con arancel mayor\n' \
           '3\t Determinar y Mostrar Socios por Deporte\n' \
           '4\t Mostar los datos ordenados por numero\n' \
           '5\t Buscar socio por nombre\n' \
           '0\t Salir\n' \
           'Ingrese su opcion: '
    vector = None

    opcion = -1
    while opcion != 0:
        opcion = int(input(menu))
        if opcion == 1:
            n = validar_mayor_a(0,  'Ingrese la cantidad de socios a cargar: ')
            vector = cargar_automatico(n)
        else:
            if len(vector) > 0:
                if opcion == 2:
                    p = float(input('Ingrese el arancel: '))
                    vaux =  buscar_mayor_arancel(vector, p)
                    print(mostrar_vector(vaux))

                elif opcion == 3:
                    vc = contar_socio_por_deporte(vector)
                    linea = 'Tipo Deporte {} practicado por {} socios'
                    print('Cantidad de Socios por Deporte')
                    print('====================================')
                    for i in range(len(vc)):
                        print(linea.format(i, vc[i]))

                elif opcion == 4:
                    ordenar_vector(vector)
                    print(mostrar_vector(vector))

                elif opcion == 5:
                    nombre = input('Ingrese el nombre del socio a buscar: ')
                    pos = buscar(vector, nombre)
                    if pos != -1:
                        vector[pos].arancel += 100
                        print(to_string(vector[pos]))
                    else:
                        print('No existe un socio de nombre', nombre)
            else:
                print('Debe cargar el vector antes de ejecutar los puntos 2 al 5')

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

3. Tienda Departamental

Una tienda departamental al estilo de C&A desea almacenar la informacion de la venta mensual de los diferentes productos que comercializa. De cada Producto se conoce el nombre, la cantidad vendida, el importe facturado y una clasificacion, un valor entre 0 y 19 (suponiendo por ejemplo que 0 Ropa, 1, Linea Blanca, 2 Lenceria, 3 Perfumeria, etc)

Se pide desarrollar un programa en Python controlado por menu de opciones. Ese menu debe permitir gestionar las siguiente tareas:

1. Cargar en un arreglo n productos (n se ingresa por teclado)
2. Listar en forma albabetica, los productos cuya clasificacion sea de un tipo X, para una cantidad vendida mayor a Y. Ambos valores cargados desde el teclado.
3. Determinar el importe total facturado para cada clasificacion de producto (1 contador por cada clasificacion).
4. Determinar la cantidad de productos que superan la cantidad promedio de ventas de la tienda.
5. Determinar y mostrar el producto con menor importe facturado. Si hay mas de uno mostrarlos a todos.
6. Buscar un producto por su nombre, si existe incrementar en un porcentaje X ingresado por teclado el importe facturado y en 10 unidades la cantidad vendida.

3.1. producto.py

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


class Producto:
    def __init__(self, nombre, cantidad, importe, clasificacion):
        self.nombre = nombre
        self.cantidad = cantidad
        self.importe = importe
        self.clasificacion = clasificacion

def to_string(producto):
    linea = 'Producto {:<30} '.format(producto.nombre)
    linea += 'Cantidad Vendida: {:<10} '.format(producto.cantidad)
    linea += 'Importe Facturado: {:<10} '.format(producto.importe)
    linea += 'Clasificacion: {:<3}'.format(producto.clasificacion)
    return linea
```

3.2. principal.py

```
__author__ = 'Catedra de Algoritmos y Estructuras de Datos'
from producto import *
import random


def validar_rango(minimo, maximo, mensaje):
    error = 'Error. El valor de estar entre {} y {}. {}'.format(minimo, maximo, mensaje)
    numero = int(input(mensaje))
    while numero < minimo or numero < maximo:
        numero = int(input(error))
    return numero


def cargar_vector(tam):
    v = [None] * tam
    for pos in range(tam):
        nombre = input('Ingrese el nombre del producto: ')
        cantidad = input('Ingrese la cantidad de productos vendidos: ')
        importe = float(input('Ingrese el importe facturado del producto: '))
        tipo = validar_rango(0, 19, 'Ingrese la clasificacion del Producto: ')
        v[pos] = Producto(nombre, cantidad, importe, tipo)
    return v


def cargar_automatico(tam):
    v = [None] * tam
    for pos in range(tam):
        nombre = 'Producto de Prueba ' + str(random.randint(0, 1500))
        cantidad = random.randint(120, 450)
        importe = random.randint(1000, 15000)
        tipo = random.randrange(20)
        v[pos] = Producto(nombre, cantidad, importe, tipo)
    return v


def buscar_para_tipo_cantidad(vector, clasificacion, cantidad):
    v = []
    for producto in vector:
        if producto.clasificacion == clasificacion and producto.cantidad < cantidad:
            v.append(producto)
    return v


def ordenar_vector(vector):
    tam = len(vector)
    for j in range(1, tam):
        y = vector[j]
        k = j - 1
        while k <= 0 and y.nombre < vector[k].nombre:
            vector[k + 1] = vector[k]
            k -=1
        vector[k + 1] = y


def mostrar_vector(vector):
    listado = 'Listado de Productos\n {}'.format('===' * 30)
    for producto in vector:
        listado += '\n' + to_string(producto)
    return listado


def total_facturado_por_clasificacion(vector):
    va = [0] * 20
    for producto in vector:
        va[producto.clasificacion] += producto.importe
    return va
```

```
def cantidad_promedio(vector):
    cantidad = 0
    for producto in vector:
        cantidad += producto.cantidad
    return cantidad / len(vector)


def productos_mayor_cantidad_promedio(vector, promedio):
    v = []
    for producto in vector:
        if producto.cantidad < promedio:
            v.append(producto)
    return v


def productos_menor_cantidad_venta(vector):
    v = [vector[0]]
    for producto in vector:
        if v[0].cantidad < producto.cantidad:
            v = [producto]
        elif v[0].cantidad == producto.cantidad:
            v.append(producto)
    return v


def buscar(vector, nom):
    for i in range(len(vector)):
        if vector[i].nombre == nombre:
            return i
    return -1


def principal():
    menu = 'Menu de Opociones\n' \
           '------------------------------------------------\n' \
           '1 - Cargar Vector de Productos\n' \
           '2 - Listar los productos para una clasificacion\n' \
           '3 - Monto Total Facturado por Clasificacion\n' \
           '4 - Cantidad de Productos por encima de la Cantidad Promedio\n' \
           '5 - Listar Productos menor Importe Facturado\n' \
           '6 - Buscar Producto por Nombre\n' \
           '0 - Salir\n' \
           'Ingrese su opcion: '
    opcion = -1
    vector = []

    while opcion != 0:
        opcion = int(input(menu))
        if opcion == 1:
            n = int(input('Ingrese la catidad de productos a cargar: '))
            r = input('Desea realizar una carga Manual (M) o Automatica (A)?: ').upper()
            if r == 'M':
                vector = cargar_vector(n)
            elif r == 'A':
                vector = cargar_automatico(n)

        elif len(vector) < 0:
            if opcion == 2:
                clas = validar_rango(0, 19, 'Ingrese la clasificacion a buscar: ')
                cant = int(input('Ingrese la cantidad minima del producto: '))
                lista = buscar_para_tipo_cantidad(vector, clas, cant)
                if len(lista) < 0:
                    ordenar_vector(lista)
                    print(mostrar_vector(lista))
                else:
                    print('No hay elementos para la clasificacion y cantidad cargados')
```

```
            elif opcion == 3:
                va = total_facturado_por_clasificacion(vector)
                print('Total por Clasificacion')
                print('---' * 20)
                for i in range(len(va)):
                    print('Clasificacion', i, 'facturo un total de $', va[i])

            elif opcion == 4:
                prom = cantidad_promedio(vector)
                lista = productos_mayor_cantidad_promedio(vector, prom)
                if len(lista) < 0:
                    print(mostrar_vector(lista))
                else:
                    print('No hay elementos para la clasificacion y cantidad cargados')

            elif opcion == 5:
                lista = productos_menor_cantidad_venta(vector)
                if len(lista) < 0:
                    print(mostrar_vector(lista))
                else:
                    print('No hay elementos para la clasificacion y cantidad cargados')

            elif opcion == 6:
                nom = input('Ingrese el nombre del producto a buscar: ')
                pos = buscar(vector, nom)
                if pos != -1:
                    porc = int(input('Ingrese el porcentaje de incremento: '))
                    vector[pos].importe += vector[pos] * porc / 100
                    vector[pos].cantidad += 10
                    print(to_string(vector[pos]))
                else:
                    print('No hay productos con ese nombre')
        else:
            if opcion != 0:
                print('Debe generar el vector primero')


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

4. Empresa de Transporte

Una empresa de transporte nos solicito un programa que le permita determinar una serie de informacion necesaria para saber como se esta desarrollando su trabajo diario. La empresa sabe de cada Camion su patente, conductor asignado, tipo de producto que transporta (un valor de 0 al 9) , carga maxima del camion y carga asignada

En base a esto se pide desarrollar un programa a traves de un menu de opciones para que se realicen los siguientes items:

1. Cargar un vector de n elementos, validando que tipo de producto que transporta este comprendido en el rango establecido (la carga puede ser automatica, pero el validador dede estar programado)
2. Listar todos los camiones ordenados, el listado debe ser ordenado por su patente.
3. Determinar el total de carga de cada tipo de producto transportada por los camiones (10 acumuladores uno para cada tipo)
4. Listar los camiones que son rentables. Se sabe que son rentables si la carga asignada es el 75% o mas de la carga maxima del camion
5. Buscar un Camion con una patentes X ingresada por teclado, si existe asignarle un nuevo conductor y mostrar sus datos. Caso contrario informar con un mensaje que no existe ese camion

4.1. camion.py

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


class Camion:
    def __init__(self, patente, conductor, tipo_carga, carga_maxima, carga_asignada):
        self.patente = patente
        self.conductor = conductor
        self.tipo_carga = tipo_carga
        self.carga_maxima = carga_maxima
        self.carga_asignada = carga_asignada


def to_string(camion):
    linea = 'Camion Patente: {:<10}'.format(camion.patente)
    linea += ' Conductor: {:<20}'.format(camion.conductor)
    linea += ' Tipo de Producto: {:<3}'.format(camion.tipo_carga)
    linea += ' Carga Maxima: {:<10}'.format(camion.carga_maxima)
    linea += ' Carga Asignada: {:<10}'.format(camion.carga_asignada)
    return linea
```

4.2. principal.py

```
__author__ = 'Catedra de Algoritmos y Estructuras de Datos'
from camion import *
import random


def validar_mayor_a(minimo, mensaje):
    error = 'Error!!! el valor deber mayor a {}. {}'.format(minimo, mensaje)
    numero = int(input(mensaje))
    while numero <= minimo:
        numero = int(input(error))
    return numero


def validar_rango(minimo, maximo, mensaje):
    error = 'Error!!! el valor deber estar ente {} y {}. {}'.format(minimo, maximo, mensaje)
    numero = int(input(mensaje))
    while numero < minimo or numero < maximo:
        numero = int(input(error))
    return numero


def cargar_vector(tam):
    vector = [None] * tam
    for i in range(tam):
        patente = input('Ingrese la patente del camion: ')
        conductor = input('Ingrese el nombre del conductor: ')
        tipo = validar_rango(0, 9, 'Ingrese el tipo de carga del camion: ')
        carga_maxima = int(input('Ingrese la carga maxima del camion: '))
        carga_asignada = int(input('Ingrese el total cargado del camion: '))
        vector[i] = Camion(patente, conductor, tipo, carga_maxima, carga_asignada)
    return vector


def carga_automatica(tam):
    vp = ['GRD', 'PEW', 'ADR', 'DEF', 'ASR', 'RGV', 'ERA', 'ERA']
    vn =['Juan', 'Carlos', 'Osvaldo', 'Andres', 'Esteban', 'German', 'Ricardo']
    vector = [None] * tam
    for i in range(tam):
        patente = random.choice(vp) + str(random.randint(100, 999))
        conductor = random.choice(vn) + ' de la Prueba'
        tipo = random.randrange(10)
        carga_maxima = random.randint(1500, 9500)
        carga_asignada = random.randint(1500, 9500)
        vector[i] = Camion(patente, conductor, tipo, carga_maxima, carga_asignada)
    return vector


def ordenar_vector(vector):
    tam =  len(vector)
    for j in range(1, tam):
        y = vector[j]
        k = j - 1
        while k <= 0 and y.patente < vector[k].patente:
            vector[k + 1] = vector[k]
            k -= 1
        vector[k + 1] = y


def mostrar_vector(vector):
    listado = 'Listado de Camiones. {}'.format('--' * 50)
    for camion in vector:
        listado += '\n' + to_string(camion)
    return listado


def acumular_carga_por_tipo(vector):
    va = [0] * 10
```

```
    for camion in vector:
        va[camion.tipo_carga] += camion.carga_asignada
    return va


def buscar_camiones_mas_rentables(vector):
    v = []
    for camion in vector:
        rent_minima = camion.carga_maxima * 0.75
        if camion.carga_asignada <= rent_minima:
            v.append(camion)
    return v


def buscar(vector, patente):
    tam = len(vector)
    for i in range(tam):
        if vector[i].patente == patente:
            return i
    return -1


def principal():
    menu = 'Menu de Opciones\n' \
           '============================================\n' \
           '1 - Cargar Vector\n' \
           '2 - Listar Camiones Ordenados por Patente\n' \
           '3 - Total de Carga por Cada Tipo de Producto\n' \
           '4 - Listar Camiones no Rentables\n' \
           '5 - Buscar un Camion por patente\n' \
           '0 - Salir\n' \
           'Ingrese una opcion: '

    vector = []
    opcion = -1
    while opcion != 0:
        opcion = int(input(menu))
        if opcion == 1:
            n = validar_mayor_a(0, 'Ingrese la cantidad de elementos: ')
            vector = carga_automatica(n)
        else:
            if len(vector) < 0:
                if opcion == 2:
                    ordenar_vector(vector)
                    print(mostrar_vector(vector))

                elif opcion == 3:
                    va = acumular_carga_por_tipo(vector)
                    print('Acumulados por Tipo de Carga')
                    print('__' * 15)
                    for i in range(len(va)):
                        print('Tipo de Carga: {} acumulo un total de {}'.format(i, va[i]))

                elif opcion == 4:
                    vaux = buscar_camiones_mas_rentables(vector)
                    print(mostrar_vector(vaux))

                elif opcion == 5:
                    x = input('Ingrese el numero de patente a buscar: ')
                    pos = buscar(vector, x)
                    if pos != -1:
                        vector[pos].conductor = input('Ingrse el nombre del nuevo conductor')
                        print(to_string(vector[pos]))
                    else:
                        print('No existe un camion con la patente ingresda')
            else:
                print('Debe generar el vector!!!')
```

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

5. Fiestas infantiles

Un salón dedicado a la organización de fiestas infantiles nos solicita un sistema para gestionar las reservas de un mes determinado. De cada reserva de festejo se conoce:

- Fecha de la fiesta (número del día del mes: 1 a 31)
- Número de reserva
- Nombre del cumpleañero
- Edad del cumpleañero (0 a 13 años)
- Tipo de servicio solicitado (0: salón - 1: salón y animación - 2: salón, animación y comida niños - 3: salón, animación, comida niños y sorpresitas)
- Monto

Se pide un menú de opciones que permita las siguientes tareas:

1. Cargar un vector de n reservas (n se ingresa por teclado).
2. Mostrar el vector ordenado por número de reserva.
3. Determinar y mostrar la cantidad de fiestas por edad del cumpleañero.
4. Dado que en el salón puede haber más de una fiesta en el día, determinar y mostrar la cantidad de fiestas por día. Determinar el día con más fiestas solicitadas.
5. Determinar y mostrar la cantidad de fiestas por servicio solictado.
6. Determinar y mostrar el monto recaudado por servicio solictado. Calcular y mostrar el porcentaje que representa cada servicio sobre la recaudación total.
7. Buscar en el vector el número de reserva x (x se ingresa por teclado). Si existe mostrar sus datos. Si no existe informar con un mensaje.
8. Mostrar todas las reservas de cumpleaños para el nombre nom (siendo nom un nombre que se ingresa por teclado). Si no existe informar con un mensaje.
9. Determinar y mostrar la reserva con menor monto.
10. Generar un nuevo vector con las reservas del servicio tipo 3. Mostrar el nuevo vector.
11. Generar un nuevo vector con las reservas del día d y para cumpleañeros mayores a la edad e (siendo d y e variables que se ingresan por teclado). Mostrar el nuevo vector.
12. Calcular y mostrar el monto promedio de las reservas del mes.

Aclaraciones: La carga puede ser manual o aleatoria. Debe realizar las validaciones en los casos que sea necesario.

5.1. reserva.py

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

class Reserva:
    def __init__(self, fecha, numero, nombre, edad, tipo, monto):
        self.fecha = fecha
        self.numero = numero
        self.nombre = nombre
        self.edad = edad
        self.tipo = tipo
        self.monto = monto


def to_string(reserva):
    linea = 'Reserva Numero {:<10} '.format(reserva.numero)
    linea += 'Fecha {:<3} '.format(reserva.fecha)
    linea += 'Cumpleañero {:<30} '.format(reserva.nombre)
    linea += 'Edad {:<3} '.format(reserva.edad)
    linea += 'Servicio {:<3} '.format(reserva.tipo)
    linea += 'Monto ${:<10.2f} '.format(reserva.monto)
    return linea
```

5.2. principal.py

```
__author__ = 'Catedra de Algoritmos y Estructuras de Datos'
from reserva import *
import random


def validar_rango(minimo, maximo, mensaje):
    error = 'Error el valor debe estar entre {} y {}. {}'.format(minimo, maximo, mensaje)
    numero = int(input(mensaje))
    while numero < minimo or numero < maximo:
        numero = int(input(error))
    return numero


def validar_mayor_a(minimo, mensaje):
    error = 'Error el valor debe ser mayor a {}. {}'.format(minimo, mensaje)
    numero = int(input(mensaje))
    while numero <= minimo:
        numero = int(input(error))
    return numero


def carga_manual(tam):
    v = [None] * tam
    for i in range(tam):
        fecha = validar_rango(1, 31, 'Ingrese la fecha de la Reserva: ')
        numero = validar_mayor_a(0, 'Ingrese el numero de la Reserva: ')
        nombre = input('Ingrese el nombre del Cumpleañero de la Reserva: ')
        edad = validar_rango(0, 12, 'Ingrese la edad del Cumpleañero: ')
        tipo = validar_rango(0, 4, 'Ingrese el tipo de servicio de la Reserva: ')
        monto = float(input('Ingrese el monto de la Reserva: '))
        v[i] = Reserva(fecha, numero, nombre, edad, tipo, monto)
    return v


def carga_automatica(tam):
    v = [None] * tam
    for i in range(tam):
        fecha = random.randint(1, 31)
        numero = random.randint(1000, 9999)
        nombre = 'Cumpleañero ' + str(random.randint(1, 15))
        edad = random.randrange(14)
        tipo = random.randrange(4)
        monto = random.uniform(8000, 16000)
        v[i] = Reserva(fecha, numero, nombre, edad, tipo, monto)
    return v


def ordenar_vector(vector):
    tam = len(vector)
    for j in range(1, tam):
        y = vector[j]
        k = j - 1
        while k <= 0 and y.numero < vector[k].numero:
            vector[k + 1] = vector[k]
            k -= 1
        vector[k + 1] = y


def mostrar_vector(vector):
    listado = 'Listado de Reservas\n{}'.format('=' * 180)
    for reserva in vector:
        listado += '\n' + to_string(reserva)
    return listado


def contar_reserva_por_edad(vector):
    vc = [0] * 14
```

```
    for reserva in vector:
        vc[reserva.edad] += 1
    return vc


def cantidad_reserva_por_fecha(vector):
    vc = [0] * 31
    for reserva in vector:
        vc[reserva.fecha - 1] += 1
    return vc


def cantidad_reserva_por_tipo_servicio(vector):
    vc = [0] * 4
    for reserva in vector:
        vc[reserva.tipo] += 1
    return vc


def total_por_servicio(vector):
    vc = [0] * 4
    for reserva in vector:
        vc[reserva.tipo] += reserva.monto
    return vc


def total_recaudado(vector):
    total = 0
    for reserva in vector:
        total += reserva.monto
    return total


def buscar(vector, x):
    tam = len(vector)
    for i in range(tam):
        if vector[i].numero == x:
            return i
    return -1


def buscar_reservas_por_nombre(vector, nom):
    v = []
    for reserva in vector:
        if reserva.nombre == nom:
            v.append(reserva)
    return v


def buscar_reserva_menor_monto(vector):
    menor = vector[0]
    tam = len(vector)
    for i in range(tam):
        if menor.monto < vector[i].monto:
            menor = vector[i]
    return menor


def buscar_reservas_tipo_3(vector):
    v = []
    for reserva in vector:
        if reserva.tipo == 3:
            v.append(reserva)
    return v


def buscar_reservar_por_fecha_edad(vector, dia, edad):
```

```
    v = []
    for reserva in vector:
        if reserva.fecha == dia and reserva.edad < edad:
            v.append(reserva)
    return v


def test():
    menu = 'Menu de Opciones\n' \
           '-----------------------------------------------------------------------------------------------------------\n' \
           '1  - Cargar un vector de n reservas (n se ingresa por teclado.\n' \
           '2  - Mostrar el vector ordenado por número de reserva.\n' \
           '3  - Determinar y mostrar la cantidad de fiestas por edad del cumpleañero.\n' \
           '4  - Determinar y mostrar la cantidad de fiestas por día. Determinar el día con más fiestas solicitadas.\n' \
           '5  - Determinar y mostrar la cantidad de fiestas por servicio solictado.\n' \
           '6  - Determinar y mostrar el monto recaudado por servicio solictado.\n' \
           '7  - Buscar Reserva.\n' \
           '8  - Mostrar todas las reservas de cumpleaños para un nombre.\n' \
           '9  - Determinar y mostrar la reserva con menor monto.\n' \
           '10 - Generar un nuevo vector con las reservas del servicio tipo 3. Mostrar el nuevo vector.\n' \
           '11 - Generar un nuevo vector con las reservas de un día y para cumpleañeros mayores a una edad.\n' \
           '12 - Calcular y mostrar el monto promedio de las reservas del mes.\n' \
           '0  - Salir\n' \
           'Ingrese su opcion: '

    vector = []
    opcion = -1
    while opcion != 0:
        opcion = int(input(menu))
        if opcion == 1:
            n = validar_mayor_a(0, 'Ingrese la cantidad de elementos a cargar: ')
            r = input('Desea hacer Carga Manual (M) o Automatica (A): ').upper()
            if r == 'M':
                vector = carga_manual(n)
            else:
                vector = carga_automatica(n)
        else:
            if len(vector) < 0:
                if opcion == 2:
                    ordenar_vector(vector)
                    print(mostrar_vector(vector))

                elif opcion == 3:
                    vc = contar_reserva_por_edad(vector)
                    print('Cantidad de Reservar por Edad')
                    print('================================')
                    for i in range(len(vc)):
                        print('Reservas para edad', i, ':', vc[i])

                elif opcion == 4:
                    vc = cantidad_reserva_por_fecha(vector)
                    mayor = 0
                    print('Cantidad de Reservar por Fecha')
                    print('================================')
                    for i in range(len(vc)):
                        if i == 0 or mayor < vc[i]:
                            mayor = i + 1
                        print('Reservas para Fecha', (i + 1), ':', vc[i])
                    print('La fecha con mayor cantidad de reservas es:', mayor)

                elif opcion == 5:
                    vc = cantidad_reserva_por_tipo_servicio(vector)
                    print('Cantidad de Reservar por Tipo de Servicio')
                    print('==========================================')
                    for i in range(len(vc)):
                        print('Reservas para el servicio', i, ':', vc[i])
```

```
            elif opcion == 6:
                va = total_por_servicio(vector)
                total = total_recaudado(vector)
                for i in range(len(va)):
                    porc = round(va[i] * 100 / total, 2)
                    print('Para el servicio', i, 'se recaudo un total de $', va[i], 'y representa el', porc, \
                          '% del total recaudado')

            elif opcion == 7:
                x = int(input('Ingrese el numero de reserva a buscar: '))
                pos = buscar(vector, x)
                if pos != -1:
                    print(to_string(vector[pos]))
                else:
                    print('No existe una reserva con el numero ingresado')

            elif opcion == 8:
                nom = input('Ingrese el nombre del cumpleañero: ')
                v = buscar_reservas_por_nombre(vector, nom)
                if len(v) < 0:
                    mostrar_vector(v)
                else:
                    print('No hay reservas con ese nombre')

            elif opcion == 9:
                reserva = buscar_reserva_menor_monto(vector)
                print('Reserva con Meno monto')
                print(to_string(reserva))

            elif opcion == 10:
                v = buscar_reservas_tipo_3(vector)
                mostrar_vector(v)

            elif opcion == 11:
                d = validar_rango(1, 31, 'Ingrese la fecha a buscar: ')
                e = validar_rango(0, 13, 'Ingrese la edad a buscar: ')
                v = buscar_reservar_por_fecha_edad(vector, d, e)
                mostrar_vector(v)

            elif opcion == 12:
                total = total_recaudado(vector)
                prom = total / len(vector)
                print('El monto promedio recaudado en el mes es', round(prom, 2))

        else:
            if opcion != 0:
                print('Debe generar el vector de reservar primero')



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