Guía de Ejercicios Prácticos 30 (Recuperatorio y Final)

Sitio: Universidad Virtual UTN FRC
Curso: Algoritmos y Estructuras de Datos (2020)
Libro: Guía de Ejercicios Prácticos 30 (Recuperatorio y Final)
Imprimido por: Luciana Lisette Montarce
Día: lunes, 23 de noviembre de 2020, 21:33

Descripción

Esta guía contiene enunciados de algunos ejercicios para aplicar los elementos básicos de programación en ***Python*** que se analizan en la ***Ficha 01***. 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. Servicios Agropecuarios

** Examen Final del dia 19/11/2015 para alumnos regulares*

Una empresa de servicios agropecuarios necesita un programa para gestionar los datos de los distintas productos y servicios que tiene a para ofrecer a sus clientes. Por cada producto/servicio se conoce su número de identificación (un entero), su descripción o título (una cadena), un número entero entre 0 y 14 que indica el tipo de producto/servicio, otro número entero entre 0 y 4 que indica la calidad del mismo y finalmente el precio de venta de ese producto/servicio.

Se pide:

1. Desarrollar un programa completo con menú de opciones que permita cumplir los requerimientos que se indican a continuación.
2. Cargar los datos de n productos (ingrese el valor de n por teclado) en un arreglo unidimensional. Realice las validaciones que considere necesarias. Puede desarrollar este punto haciendo que el programa genere en forma aleatoria el contenido de cada registro.
3. Mostrar los datos de todos los productos del vector, ordenados por número de identificación.
4. Generar otro arreglo que contenga los datos de todos los productos del arreglo original que sean de tipo menor a 10. El agregado en este nuevo arreglo debe hacerse de forma que vaya quedando siempre ordenado de menor a mayor por precio de venta.
5. Mostrar el nuevo arreglo, a razón de un registro por línea.
6. Usando el segundo arreglo, mostrar los datos de todos los productos cuyo precio sea menor o igual a un valor p que se carga por teclado.
7. Usando el segundo arreglo, determinar y mostrar la cantidad de productos que por cada tipo posible y por cada indicador de calidad posible. Es decir, se quiere saber cuántos existen del tipo 0 que sean de calidad 0, cuántos del tipo 0 y calidad 1, etc.
8. Grabar en un archivo todos los datos del segundo arreglo.
9. Mostrar el archivo que se acaba de crear en el punto anterior.

2. Empresa de Mudanzas

**Adaptado de uno de los enunciados del Parcial 4 (2020)*

Una empresa de mudanzas mantiene información sobre los distintos trabajos de traslado que debe realizar.

Por cada traslado se registran los datos siguientes: número de identificación del traslado (un número entero), nombre del cliente (una cadena), importe a facturar por el traslado, provincia destino del traslado (un valor entre 0 y 22 incluidos) y forma de pago (un número entero entre 0 y 4).

Se pide definir un tipo registro Traslado con los campos que se indicaron, y un programa completo con menú de opciones para hacer lo siguiente:

1. Cargar los datos de n registros de tipo Traslado en un arreglo de registros (cargue n por teclado). El arreglo debe crearse de forma que siempre quede ordenado de menor a mayor, según el número de identificación de los traslados.
2. Mostrar el arreglo creado en el punto 1, a razón de un registro por línea.
3. Buscar en el arreglo creado en el punto 1 un registro en el cual el nombre del cliente sea igual a nom (cargar nom por teclado). Si no existe, informar con un mensaje. Si existe, mostrar el registro y validar que el nombre esté compuesto sólo por letras y espacios (procesando los caracteres uno a uno); en caso de no ser válido, agregar un asterisco al final del nombre.
4. A partir del arreglo, crear un archivo de registros en el cual se copien los datos de todos los traslados cuya forma de pago sea 0 y cuyo importe a facturar se encuentre entre dos valores a y b que se cargan por teclado.
5. Mostrar el archivo creado en el punto anterior, a razón de un registro por línea en la pantalla.

2.1. interfaz

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

from procesos import *
import os.path


def mostrar_menu():
    print('=' * 100)
    print('EMPRESA DE MUDANZAS')
    print('1 - Cargar los datos')
    print('2 - Mostrar el arreglo')
    print('3 - Buscar registro')
    print('4 - Crear archivo')
    print('5 - Mostrar archivo')
    print('0 - Salir')
    print('=' * 80)
    opcion = int(input('Ingrese la opcion elegida: '))
    return opcion


def validar_mayor_que(inf, mensaje):
    n = int(input(mensaje))
    while n <= inf:
        n = int(input('Error, debe ser un valor mayor a ' + str(inf) + '. ' + mensaje))
    return n


def mostrar_registros(v):
    print(to_string_titulos())
    for reg in v:
        print(to_string(reg))


def principal():
    v = []
    fd = 'traslados.dat'
    op = -1

    while op != 0:
        op = mostrar_menu()
        if op == 1:
            n = validar_mayor_que(0, 'Ingrese la cantidad de registros que desea cargar: ')
            generar_vector(n, v)
            print('Registros cargados correctamente')
        elif op == 0:
            print('Hasta pronto!')
        elif len(v) == 0:
            print('Debe cargar primero el vector (opcion 1)')
        elif op == 2:
            mostrar_registros(v)
        elif op == 3:
            nom = input('Ingrese el nombre del cliente: ')
            pos = buscar_secuencial(nom, v)
            if pos == -1:
                print('No se encontró un cliente con ese nombre')
            else:
                if validar_nombre(v[pos].nombre) == False:
                    v[pos].nombre = v[pos].nombre + '*'
                print('Registro encontrado:\n', to_string(v[pos]))
        elif op == 4:
            a = validar_mayor_que(0, 'Ingrese el importe mínimo: ')
            b = validar_mayor_que(a, 'Ingrese el importe máximo: ')
            crear_archivo(v, a, b, fd)
            print('Archivo creado correctamente')
        elif op == 5:
            if not os.path.exists(fd):
                print('Debe generar el archivo (Opcion 4)')
```

```
            else:
                mostrar_archivo(fd)


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

2.2. procesos

```
import random
import pickle
import os.path
from registros import *


def add_in_order(v, reg):
    n = len(v)
    pos = n
    izq, der = 0, n - 1
    while izq <= der:
        c = (izq + der) // 2
        if v[c].num_id == reg.num_id:
            pos = c
            break
        if reg.num_id < v[c].num_id:
            der = c - 1
        else:
            izq = c + 1
    if izq > der:
        pos = izq
    v[pos:pos] = [reg]


def generar_vector(n, v):
    noms = ('Ana', 'Pedr0', 'Maria', 'Juan')
    aps = ('Perez', 'Gomez', 'Ruiz', 'Oliva')
    for i in range(n):
        num_id = random.randint(0, 1000)
        nombre = random.choice(noms) + ' ' + random.choice(aps)
        importe = round(random.uniform(1000, 10000), 2)
        destino = random.randint(0, 22)
        forma_pago = random.randint(0, 4)
        reg = Mudanza(num_id, nombre, importe, destino, forma_pago)
        add_in_order(v, reg)


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


def es_letra(car):
    if car >= 'a' and car <= 'z':
        return True
    if car >= 'A' and car <= 'Z':
        return True
    return False


def validar_nombre(nombre):
    validas = 0
    for car in nombre:
        if es_letra(car) or car == ' ':
            validas += 1
    return validas == len(nombre)


def crear_archivo(v, a, b, fd):
    cant = 0
    m = open(fd, 'wb')
    for reg in v:
        if reg.forma_pago == 0 and reg.importe >= a and reg.importe <= b:
            cant += 1
            pickle.dump(reg, m)
```

```
    print('Se grabaron', cant, 'registros')
    m.close()


def mostrar_archivo(nombre_archivo):
    cant = 0
    tam = os.path.getsize(nombre_archivo)
    m = open(nombre_archivo, 'rb')
    while m.tell() < tam:
        cant += 1
        if cant == 1:
            print(to_string_titulos())
        reg = pickle.load(m)
        print(to_string(reg))
    print('Se encontraron', cant, 'registros')
    m.close()
```

2.3. registros

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


class Mudanza:
    def __init__(self, num_id, nombre, importe, destino, forma_pago):
        self.num_id = num_id
        self.nombre = nombre
        self.importe = importe
        self.destino = destino
        self.forma_pago = forma_pago


def to_string(reg):
    cad = '| {:^15} | {:^20} | {:^10} | {:^10} | {:^10} |'
    return cad.format(reg.num_id, reg.nombre, reg.importe, reg.destino, reg.forma_pago)


def to_string_titulos():
    cad = '| {:^15} | {:^20} | {:^10} | {:^10} | {:^10} |'
    return cad.format('IDENTIFICACION', 'NOMBRE', 'IMPORTE', 'DESTINO', 'PAGO')
```

3. Agencia de Turismo

Una empresa que se dedica a la venta de paquetes turisticos necesita un programa para gestionar los diferentes paquetes que vende. Por cada paquete turistico se conoce un codigo, descripcion, costo por persona, ciudad de salida (0, 1, 2 ... hasta 24) y el pais de destino (0, 1, 2 ....hasta 19 posibles paises de destino).

Desarrollando un programa completo con menu de opciones, usted debe cumplir los siguientes puntos.

1. Cargar los datos de los n productos (ingrese el valor de n por teclado) en un arreglo unidimensional. Puede generar los registros en forma manual o aleatorio pero no puede mezclarlas. El arreglo debe generarse siempre en forma ordenada por codigo
2. Mostrar el arreglo generado
3. Generar un segundo arreglo con todos los paquetes turisticos, de aquellos paquetes cuya descripcion tenga una cantidad de vocales superior al de consonantes y la descripcion comience con la letra 'P'.
4. Mostrar ese nuevo arreglo.
5. A partir del arreglo del punto 1, determinar la cantidad de paquetes que se ofrecen por ciudad para cada posible destino 456 contadores). Muestre aquellas posiciones que sean distintas de cero.
6. A partir del arreglo original generado en el punto 1 buscar un paquete turistico que tenga un codigo X, pasado por paramentro. Si existe mostrar sus datos , caso contrario indique con un mensaje que no fue encontrado.
7. A partir del arreglo generado en el punto 3 buscar un paquete turistico cuyo costo por persona sea un valor mayor a X, pasado por parametro. Muestre el primero que encuentro en ese arreglo.
8. A partir del arreglo generado en el punto 1, genere un archivo binarios, a razon de un registro a la vez, de todos aquellos paquetes turisticos cuyo pais de destino No sea el valor 5 o el valor 15.
9. Mostrar el archivo generado en el punto anterior