Universidad del Desarrollo
Facultad de Ingeniería

Taller de Programación

Examen

28 de noviembre de 2023

Nombre:.....

Instrucciones:

- Lea atentamente el enunciado de cada uno de los problemas.
- El problema 1 y el 2 son **obligatorios**. Debe elegir un problema adicional entre el problema 3 y 4.
- Para cada problema cree un archivo.py distinto. El nombre del archivo debe ser el número del problema (uno.py, dos.py, tres.py, cuatro.py).
- Comprima los problemas en un solo archivo ZIP y súbalo a la sección Evaluación en http://canvas.udd.cl. Solo tiene dos oportunidades para subir sus respuestas.
- No se puede utilizar librerías adicionales ni estilos de sintaxis o métodos avanzados de programación que no se hayan revisado en clases o que no domine.
- En caso de que su profesor(a) tenga dudas sobre la originalidad del código, podrá interrogarlo oralmente para verificar que no hubo faltas a la ética.
- De detectarse copia entre pares o cualquier otra situación que de cuenta de falta a la ética, su certamen será evaluado con la NOTA MÍNIMA y será reportado(a) al comité de ética de la Facultad, lo que puede derivar en una causal de eliminación de la universidad.

Ejercicio 1 Obligatorio: Registro de Ventas de una Automotora (2.5 pts)

Crear un programa que maneje los datos de ventas de una automotora y genere un resumen específico de una de las marcas. El archivo de registro de ventas contiene una fila por auto vendido y tiene la siguiente estructura:

Columnas: Fecha, Marca, Modelo, Precio.

Crear un programa que lea el archivo CSV `ventas_automotora.csv` y escriba un archivo de resumen de ventas **sólo para la marca Ford** llamado `resumen_ventas_Ford.csv`.

Este archivo debe contener la marca, modelo y la cantidad total de autos vendida para la marca "Ford".

Ejemplo de archivo de salida:

```
Marca;Modelo;CantidadTotal
Ford;Focus;4
Ford;Fiesta;12
```

```
#Solución Correcta 2.5 pts
#Leer el archivo y contar 1.5 pts
with open ('ventas_automotora.csv', 'r') as f: #abrir archivo 0.1 pts
    dicc_count=dict() #crear estructura de edatos para almacena 0.1 pts
    next(f)
    for line in f:
        aux=line.split(';') #separar y seleccionar columnas 0.5 pts
        marca=aux[1]
        modelo=aux[2]
        if marca=='Ford': #contar correctamente 0.8 pts
            if modelo in dicc_count:
                dicc_count[modelo]+=1

            else:
                dicc_count[modelo]=1

print(dicc_count)


#Escribir correctamente el archivo 1.0 pt
with open ('resumen_ventas_Ford.csv','w') as g:# crear archivo 0.1 pts
    g.write('Marca;'+'Modelo;'+ 'CantidadTotal' + '\n') #escribir el encabezadc
    for modelo, numero in dicc_count.items(): #iterar sobre el diccionarop 0.3
        g.write('Ford;'+modelo+ ';' + str(numero) + '\n') #escribir correctamnt
```

Ejercicio 2 Obligatorio: Diccionarios y Funciones (2.5 pts)

Crea una **función** `contar_letras` que tome una cadena de texto como entrada y

devuelva un diccionario donde las claves sean las letras presentes en el texto y los valores sean la cantidad de veces que cada letra aparece.

Nota: - La función `letra.isalpha()` devuelve `True` si un caracter es una letra. - La función `letra.lower()` en el caso de que una letra sea mayuscula la transforma una letra minuscula.

```
#Solución Correcta 2.5 pts
def contar_letras(texto): #iniciar creacion de funcion y definir variables 0.3
    dicc_cuentas=dict()

    #pasar a mayuscula  y/o selecionar sólo digitos 0.2 pts
    texto=texto.lower()
    for letra in texto: #recorrrer el texto 0.3 pts
        if letra.isalpha()==True:
            if letra in dicc_cuentas:#llena el diccionario correctamente 1.0 pt
                dicc_cuentas[letra]+=1
            else:
                dicc_cuentas[letra]=1

    return dicc_cuentas  #retornar diccionario 0.2 pts

texto_prueba='Me encanta programar en Python'

print(contar_letras(texto_prueba)) #usar correctamente la funcion e impirimir r
```

Ejercicio 3: Tuplas (1.0 pt)

Crea un programa que reciba como entrada N nombres y apellidos de personas separados por espacio o coma `,` y genere una lista de tuplas con esos datos. Luego, desempaqueta cada tupla y muestra por pantalla los valores obtenidos. No necesita validar la entrada.

Ejemplo de entrada: `Juan Ríos`

```
#Solución Correcta 1.0 pt
n=4
lista_final=[]
for i in range(n):
    nombre=input('Ingrese nombre y apellido separados por coma:') #solicitar ir
    nombre_lista=nombre.split(',') #separar correctamente 0.4 pts
    lista_final.append(tuple(nombre_lista)) #agregar a lista 0.2 pts

for nombre_tupla in lista_final: #desempaquetar tuplas 0.2 pts
    print(nombre_tupla)
```

Ejercicio 4: Condicionales y Ciclos (1.0 pt)

Diseña una **función** `suma_cuadrados_impares` que reciba un número `n` y calcule la suma de los cuadrados de todos los números impares desde 1 hasta `n`. Recuerde que un número impar puede ser representado con la expresión $2\dot{k} + 1$.

```
# Alternativa 1 Maximo 0.8 pts
# Solución Parcialmente Correcta
def suma_cuadrados_impares(n):
    suma=0
    for k in range(n):
        expresion=(2*k+1)**2
        suma+=expresion
    return suma

print('Suma', suma_cuadrados_impares(10))
```

```
# Alternativa 2 1.0 pt
# Solución Correcta
def suma_cuadrados_impares2(n): #definicion correcta 0.1 pts
    suma=0
    for k in range(n): #uso de ciclo correcto 0.1 pts
        termino=2*k+1
        if termino <=n:
            suma+=termino**2 #suma correcta 0.5 pts
    return suma

print('Suma', suma_cuadrados_impares2(10)) #uso correcto de la funcion 0.3 pts
```

```
# Alternativa 3 1.0 pt
# Solución Correcta
def suma_cuadrados_impares3(n): #definicion correcta 0.1 pts
    suma=0
    for k in range(n): #uso de ciclo correcto 0.1 pts
        if k%2==1:
            suma+=k**2 #suma correcta 0.5 pts
    return suma

print('Suma', suma_cuadrados_impares3(10)) #uso correcto de la funcion 0.3 pts
```