ALGORITMOS: Guía de ejercicios N° 1

Una compañía dedicada a la venta de seguros para automotores ha incorporado a su oferta un nuevo producto: seguros de vida. Ya que se cuenta con una cartera de clientes inicial se pretende identificar los clientes potenciales para este nuevo producto.
En principio el procesamiento se realizará de forma manual, sobre pequeñas muestras datos de diferentes promotores de seguros, por lo que la cantidad a procesar no es conocida y se propone ingresar datos mientras el valor de la edad sea válido (mayor a cero).
Se debe determinar el porcentaje de clientes potenciales (cantidad de clientes con un criterio particular dividido la cantidad total de clientes considerados multiplicado por 100). El primer criterio que se utilizará para considerar un cliente como potencial es su edad, debido estar en el rango entre 35 y 45 años (ambos valores inclusive).

a) Partiendo de un script Python con la estructura indicada a continuación realice la implementación requerida.

```
def main():
    edadStr = input("Ingrese la primer edad: ")
    edadInt = int(edadStr)
    while (edadInt > 0):
        #
        # Agregue aquí el procesamiento a realizar
        #
        edadStr = input("Siguiente: ")
        edadInt = int(edadStr)
    #
    # Realice las operaciones requeridas
    # y presente los resultados correspondientes
    #

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

b) ¿Qué sucede cuando el primer valor ingresado es cero? Si se produce algún error o mensaje incorrecto ¿Cómo podría remediar este inconveniente?

c) A fin de separar de un modo más claro el criterio de selección de los clientes potenciales respecto del algoritmo de procesamiento, implemente una función que, considerando la edad de un cliente, retorne verdadero o falso a fin de indicar si se trata o no de un posible cliente. Modifique la aplicación del punto anterior a fin de utilizar esta función.

d) Se propone incorporar otro indicador al procesamiento realizado, el promedio de edades considerado y cuantas edades del conjunto se encuentran por encima del promedio. Para esto realice las modificaciones requeridas a fin de que se almacenen las edades según se ingresan en una lista.
Puede determinar el promedio utilizando las funciones **sum** y **len** e indicando como argumento la lista de edades.
Luego implemente una función que, tomando como argumento la lista de edades y su promedio, retorne la cantidad por encima del promedio.

e) A fin de evitar la carga manual de información y aprovechando que se cuenta con archivos de texto donde se tiene, en cada renglón, el número de DNI y la edad del cliente separados por el carácter “;” se desea utilizar la información disponible para el procesamiento anterior. Para esto, realice las siguientes acciones:

- Cree un archivo de texto utilizando el Block de Notas o el editor de Texto de Spyder que contenga los siguientes valores:

```
12345678; 63
23435213; 52
26678123; 45
32564876; 37
```

  Nombre al archivo como “datos.txt”. Es importante que el archivo contenga solo texto plano (no puede ser un documento de Word).

- Cree un nuevo archivo script de Python utilizando Spyder y almacénelo en la misma carpeta que el archivo anterior con el siguiente fragmento de código fuente:

```
def main():
    fp = open('datos.txt', 'r', encoding='utf8')
    for linea in fp:
        lineaLimpia = linea.strip()
        datosCliente = lineaLimpia.split(";")
        if(len(datosCliente) == 2):
          dni = int(datosCliente[0].strip())
          edad = int(datosCliente[1].strip())
          print(f"DNI: {dni} Edad: {edad}")
    fp.close()

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

  Al ejecutar el archivo script, encontrará que se accederá a cada línea del archivo de texto llamado “datos.txt” en cada iteración del ciclo **for**. Cada renglón es accedido como una cadena de caracteres. A fin de eliminar separadores (espacios en blanco, tabulaciones, etc.) se utiliza la función **strip**. Tanto sobre el renglón leído como sobre cada cadena obtenida luego de dividir el renglón según la posición del carácter “;”.
  Solo se procesan las líneas que tienen dos elementos campos (el DNI y la edad), omitiendo de este modo líneas mal formadas o renglones en blanco. Las variables **edad** y **dni** almacenan los datos como valores enteros.
  Al finalizar las iteraciones, el archivo se cierra a través de una invocación al método **close()**. Los archivos deben abrirse a fin de poder acceder a su contenido y, una vez finalizadas las tareas de interés, deben cerrarse posibilitando su uso por otras aplicaciones.
  A partir de estas ideas, modifique la aplicación original a fin de que el procesamiento se realice utilizando los datos contenidos en el archivo de texto en vez de ser ingresados por el usuario.

f) Ahora se desea contactar a los clientes potenciales, para esto, es necesario identificar cada cliente por su nombre y apellido. Esto es posible debido a que se cuenta con archivos de texto

conteniendo el número de documento y el nombre completo de cada cliente con un formato similar al del punto anterior.
A fin de lograr la implementación realice las siguientes operaciones:

- Utilizando el block de notas o el editor de texto de Spyder cree un archivo de texto llamado nombres.txt con el siguiente contenido:

```
12345678; Juan Carlos Nicoletti
23435213; Ana María Trigati
26678123; Alicia Mercado
32564876; Gustavo Lamperti
```

- Cree un nuevo archivo script, e incorpore el siguiente fragmento de código fuente:

```
def main():
    dicNombres = {}
    fp = open('nombres.txt', 'r', encoding='utf8')
    for linea in fp:
        lineaLimpia = linea.strip()
        datosCliente = lineaLimpia.split(";")
        if(len(datosCliente) == 2):
          dni = int(datosCliente[0].strip())
          nombre = datosCliente[1].strip()
          dicNombres.update({dni: nombre})
    fp.close()

    for vDni in dicNombres.keys():
        nom = dicNombres[vDni]
        print(f"DNI: {vDni} - Nombre: {nom}")

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

Al ejecutar este archivo notará que se presenta la lista de todos los nombres pero accediendo a los mismos a través de un **diccionario**. De modo similar al caso anterior, aquí se toma cada línea del archivo de texto, separando sus partes (DNI como valor numérico y nombre como cadena de caracteres). Con esta información se construye un diccionario, donde la clave es el número de DNI.
Luego de construir el diccionario, se itera sobre los valores de las claves a fin de recuperarlas en ciclo **for**. Por cada valor de clave se accede al nombre del cliente.
Utilizando estas ideas, modifique la aplicación anterior de modo que, al comienzo, se cree y cargue el diccionario con los datos de los clientes. Luego, se procesen los datos del archivo que contiene las edades y, por cada cliente potencial, se muestre su nombre utilizando el diccionario creado.