Práctica de Estudio 10|: **Archivos**
Estructura de Datos y Algoritmos Grupo 01
Facultad de Ingeniería
Departamento de Computación

	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Jesus Cruz Navarro
Profesor:

Estructura de Datos y Algoritmos II
Asianatura:

01
Grupo:

10
No de Práctica(s):

Diego Santiago Gutierrez
Integrante(s):

No. de Equipo de cómputo

No. de Lista o Brigada:

Tercer Semestre
Semestre:

10/12/2020
Fecha de entrega:

Observaciones:

CALIFICACIÓN: _____

Práctica de Estudio 10|: **Archivos**
Estructura de Datos y Algoritmos Grupo 01
Facultad de Ingeniería
Departamento de Computación

REQUISITOS:
Desarrollar un programa que lea el archivo con formato CSV de casos confirmados de COVID-19 (el más actual) y procesar los datos para obtener y mostrar en la consola la siguiente información:

- Total de casos.
- Número de casos por estado.
- Estado con más casos.
- Estado con menos casos.
- Número de casos por género.
- Promedio de Edad.
- Edad máxima.
- Edad mínima.

Además, generar un archivo de nombre 'resumenCovid.eda2' con estos mismos resultados, uno por renglón (bonito). El archivo debe crearse en el directorio **'PracticaResumen'** en el directorio de trabajo, usando rutas relativas. El programa debe crear dicho directorio si no existe.

Nota: Se pueden crear directorios con el método os.makedirs('ruta_directorio'). Para usar este método, se debe importar la Librería os (import os) al inicio del programa. Si ya existe el directorio el método makedirs arroja un error, por lo que se debe considerar en un bloque try/catch

El archivo se puede descargar desde: https://serendipia.digital/2020/03/datos-abiertos-sobre-casos-de-coronavirus-covid-19-en-mexico/

TIP: Para contabilizar y “agrupar” los datos (por ejemplo, por estado) puede hacer uso de diccionarios. Al inicio el diccionario está vacío, y si existe una entrada, por ejemplo, el nombre de un estado se revisa si el nombre del estado ya existe en el diccionario:

- Si no existe, se crea la llave y se establece en 1 su valor pues es la primera vez que se contabiliza este estado.
- Si existe, el valor del diccionario para esa llave se incrementa, puesto que ya se ha contabilizado este estado y ahora hay una nueva entrada.

Práctica de Estudio 10|: **Archivos**
Estructura de Datos y Algoritmos Grupo 01
Facultad de Ingeniería
Departamento de Computación

INTRODUCCIÓN:

Existen dos formas básicas de acceder a un archivo, una es utilizarlo como un archivo de texto, que procesaremos línea por línea; la otra es tratarlo como un archivo binario, que procesaremos byte por byte. Para esto Python incorpora un tipo integrado llamado file, el cual es manipulado mediante un objeto archivo el cual fue generado a través de una función integrada en Python
n Python, para abrir un archivo usaremos la función open, que recibe el nombre del archivo a abrir. Si tiene éxito, devolverá una variable que nos permitirá manipular el archivo de diversas maneras.

Abrir archivo
La forma preferida para abrir un archivo es usando la función integrada **open**().

Leer archivo
La forma preferida para leer un archivo es usando algunas de los métodos del objeto file como **read**(), **readline**() y **readlines**().

Escribir archivo
La forma preferida para escribir un archivo es usando el método del tipo objeto file llamado **write()**.

Cerrar archivo
La forma preferida para cerrar un archivo es usando el método del tipo objeto file llamado **close()**

Archivos con modulo os:
El módulo os de Python le permite a usted realizar operaciones dependientes del Sistema Operativo como crear una carpeta, listar contenidos de una carpeta, conocer acerca de un proceso, finalizar un proceso, etc. Este módulo tiene métodos para ver variables de entornos del Sistema Operativo con las cuales Python está trabajando en mucho más.

DESARROLLO:

1) Métodos:

parseCsv:
Nos ayudará a leer las líneas del archivo Covid-19-mexico…cvs conviertiendolo en un formato "utf-8" el cual permitirá que podamos manejarlo de mejor forma. Para ello se leen las líneas de dicho archivo y las separamos obteniendo así las líneas de dicho documento. Posterior a eso, las líneas llevan el formato:
"ID_REGISTRO,Nom_Ent,SEXO,EDAD,FECHA_SINTOMAS,CLASIFICACION_FINAL" el cual nos permitirá por medio de un arreglo separar esta información por medio de un ciclo for, que posteriormente será guardado en un arreglo de userInformatión.

```python
def parseCsv():
    with open("covid-19-mexico-01122020.csv", encoding='utf-8') as file:
        lines = file.read().splitlines()
        columnNames = lines[0].split(',')
        userInformation = []
        for i in range(1, len(lines)):
            information = lines[i].split(',')
            userInformation.append((information[0], information[1], information[2], int(information[3]), information[4], information[5]))

    return columnNames, userInformation
```

Método parseCvs

stringInfo:
Permite obtener el archivo con el que se trabajará para posteriormente unir la información gracias a .join para posteriormente concatenar con el ciclo for los elementos restantes. Al final se escribe esta información en file.

```python
from typing import TextIO

def getStringInformation(file: TextIO):
    def printInfo(*toPrint):
        stringIndormation = " ".join([str(element) for element in toPrint])
        print(stringIndormation)
        file.write(stringIndormation + "\n")
    return printInfo
```

Método stringInfo

processData:
Gracias al uso de pandas es que podemos trabajar con la información de esta manera. La forma en que procesamos la información será por el método DataFrame que nos devolverá una lista de diccionarios de la información que nosotros le demos de parceCsv.
Después es necesario abrir el archivo en forma de escritura y que el archivo sea procesado por getStringInformation para poder trabajar con este de la forma más óptima.

Sabemos que el número de casos mostrados por el archivo será la cantidad de elementos que este tenga, razón por la cual utilizamos la función len sobre dataFrame .

Para conocer el numero de casos por estado, es necesario que desde dataFrame nosotros obtengamos "Nom_Ent" ya que será donde encontremos los nombres de los estados, utilizamos value_counts para saber el número de elementos que encuentra tipo "Nom_ent" y posteriormente lo imprimimos pasandolo a string. El estado con mayor número de contagios será conocido gracias a la función idmax, mientras que el menor será gracias a idmin.

Para conocer los casos por genero accedemos a dataFrame["SEXO"] que nos indicará el generó masculino y femenino en casos de Covid-19.

Por último, le edad será conocida gracias a dataFrame["EDAD"] que obtendremos el promedio con mean, el mayor con max y el menor con min ya que se tratan de enteros.

```python
# pip install pandas
import pandas
import os

from parsing import parseCsv
from stringInfo import getStringInformation


def processData():
    columns, data = parseCsv()
    dataFrame = pandas.DataFrame(data, columns=columns)

    file = open("PracticaResumen/resumenCovid.eda2", "w")
    printFile = getStringInformation(file)

    printFile("\nTotal de casos:", len(dataFrame))

    printFile("\nCasos por estado:")
    states = dataFrame["Nom_Ent"].value_counts()
    printFile(states.to_string())

    printFile("\nEstado con más casos:", states.idxmax())
    printFile("Estado con menos casos:", states.idxmin())

    printFile("\nCasos por género")
    gender = dataFrame["SEXO"].value_counts()
    printFile(gender.to_string())

    printFile("\nPromedio de edad:", dataFrame["EDAD"].mean())
    printFile("\nEdad máxima:", dataFrame["EDAD"].max())
    printFile("Edad mínima:", dataFrame["EDAD"].min())

    file.close()
```

Método processData

Resultados:

```
El directorio PracticaResumen ya existe.

Total de casos: 1048575

            Casos por estado
Ciudad de México                  223778
México                             73772
Nuevo León                         63293
Guanajuato                         59203
Sonora                             40984
Puebla                             39852
Jalisco                            39027
Coahuila de Zaragoza               38987
Veracruz de Ignacio de la Llave    37975
Tabasco                            36761
Tamaulipas                         34430
San Luis Potosí                    33017
Michoacán de Ocampo                27317
Chihuahua                          26838
Baja California                    25369
Sinaloa                            23592
Yucatán                            23098
Oaxaca                             22986
Guerrero                           22699
Querétaro                          20084
Durango                            19141
Hidalgo                            17458
Zacatecas                          15144
Quintana Roo                       13960
Baja California Sur                13721
Aguascalientes                     12820
Tlaxcala                            7991
Chiapas                             7268
Morelos                             7171
Colima                              7027
Nayarit                             6927
Campeche                            6885

Estado con más casos: Ciudad de México
```

```
Estado con menos casos: Campeche

M    535053
F    513522

Promedio de edad: 44.299566554609825

Edad máxima: 120
Edad mínima: 0
```

CONCLUCIONES:

El uso de archivos en la programación funcional involucra una gran ventaja al manejo de datos que este nos permite, ya que como se ha visto, es posible crear nuevos o leer unos ya existentes para posteriormente manipular los mismos. Como bien se ha observado en la práctica, el uso de un archivo involucra de una a adecuada implementación, ya es posible generar errores según sea la circunstancia del archivo, es por ello que se vuelve importante una vez terminado de usar dicho archivo cerrarlo para poder aprovechar toda la funcionalidad que implica un archivo,