Ejercicios de Archivos

Ejercicio 1

Escribir una función que pida un número entero entre 1 y 10 y guarde en un archivo con el nombre tabla-n.txt la tabla de multiplicar de ese número, donde n es el número introducido.

```
def tabla_multiplicar(n):
  t= []
  for m in range(10):
    m+=1
    t.append(f'{n} x {m} = {n*m} \n')
  return t

def ingresar():
  v = False
  while not v:
    n = int(input("Ingrese un numero entre 1 y 10 : "))
    condition = 10 > n  and n > 1
    if not condition:
      print("Error, debe ingresar un numero entra 1 y 10 !!")
    else:
      v = True
      return n

def guardar_txt(n, array):
  archivo = open(f'tabla-{n}.txt', 'w')
  for a in array:
    archivo.write(a)
  archivo.close()

  l = open(f'tabla-{n}.txt','r')
  print(l.read())

n = ingresar()
tbl = tabla_multiplicar(n)
txt = guardar_txt(n, tbl)
```

```
Ingrese un numero entre 1 y 10 : 12
Error, debe ingresar un numero entra 1 y 10 !!
Ingrese un numero entre 1 y 10 : 5
5 x 1 = 5
```

```
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

Ejercicio 2

Escribir una función que pida un número entero entre 1 y 10, lea el archivo tabla-n.txt con la tabla de multiplicar de ese número, donde n es el número introducido, y la muestre por pantalla. Si el fichero no existe debe mostrar un mensaje por pantalla informando de ello.

```
def ingresar():
  v = False
  while not v:
    try:
      n = int(input("Ingrese un numero entre 1 y 10 : "))
      condition = 10 > n  and n > 1
      if not condition:
        print("Error, debe ingresar un numero entra 1 y 10 !!")
      else:
        v = True
        return n
    except ValueError:
      print('Ingrese un nombre valido')

def leer_txt(n, array):
  try:
    name = f'tabla-{n}.txt'
    archivo = open(name, 'r')
  except FileNotFoundError:
    print('No existe el archivo ' + name)
  else:
    print(archivo.read())

n = ingresar()
tbl = tabla_multiplicar(n)
txt = leer_txt(n, tbl)
```

```
Ingrese un numero entre 1 y 10 : 213213
Error, debe ingresar un numero entra 1 y 10 !!
Ingrese un numero entre 1 y 10 : 2
No existe el archivo tabla-2.txt
```

Ejercicio 3

Escribir una función que pida dos números n y m entre 1 y 10, lea el archivo tabla-n.txt con la tabla de multiplicar de ese número, y muestre por pantalla la línea m del archivo. Si el archivo no existe debe mostrar un mensaje por pantalla informando de ello.

```
def tabla_multiplicar(n):
  t= []
  for m in range(10):
    m+=1
    t.append(f'{n} x {m} = {n*m} \n')
  return t

def ingresar():
  v = False
  while not v:
    try:
      n = int(input("Ingrese un numero entre 1 y 10 : "))
      condition = 10 > n  and n > 1
      if not condition:
        print("Error, debe ingresar un numero entra 1 y 10 !!")
      else:
        v = True
        return n
    except ValueError:
      print('Ingrese un nombre valido')

def guardar_txt(n, array):
  archivo = open(f'tabla-{n}.txt', 'w')
  for a in array:
    archivo.write(a)
  archivo.close()

def leer_txt(n, m):
  archivo = f'tabla-{n}.txt'
  try:
    a = open(archivo, 'r')
  except FileNotFoundError:
      print('El archivo no existe en la tabla del numero ', n)
  else:
      linea = a.readlines()
      print(linea[m - 1])

n = ingresar()
tbl = tabla_multiplicar(n)
txt = guardar_txt(n, tbl)
print("########## Ingrese el N y luego el M ###########")
```

```
n1 = ingresar()
m1 = ingresar()

leer_txt(n1,m1)
```

Ejercicio 4

Escribir un programa que acceda a un archivo de Internet mediante su url y muestre por pantalla el número de palabras que contiene.

```
def archivo(url):
    from urllib import request
    from urllib.error import URLError
    try:
        file = request.urlopen(url)
    except URLError:
        return(f'La url {url} no fue posible encontrarla')
    else:
        content = file.read()
        return len(content.split())

print(archivo('https://www.google.rl'))
print(archivo('https://www.gutenberg.org/files/2/2.txt'))
```

```
 La url https://www.google.rl no fue posible encontrarla
 1771
```

Ejercicio 5

Escribir un programa que abra el archivo con información sobre el PBI per cápita de los países de la Unión Europea, pregunte por las iniciales de un país y muestre el PBI per cápita de ese país de todos los años disponibles.

```
from urllib import request
from urllib.error import URLError

def validar_codigo(c):
  codigos = ["AT", "BE", "BG", "CH", "CY", "CZ", "DE", "DK", "EA19", "EE", "EL", "ES", "EU27'
  pais = c.upper()
  if c.upper() in codigos:
    url ='https://ec.europa.eu/eurostat/estat-navtree-portlet-prod/BulkDownloadListing?file=d
    print('Producto Interior Bruto per cápita de', pais)
    print('Año', '\t', 'PIB')
```

```
    for año, pib in pbi(url,pais).items():
        print(año, '\t', pib)
  else:
    print("Error el codigo no existe!")

def pbi(url, pais):

    try:
        f = request.urlopen(url)
    except URLError:
        return("Url ingresada no existe ({})".format(url))
    else:


          data = f.read().decode('utf-8').split('\n')
          data = [i.split('\t') for i in data]
          data = [list(map(str.strip, i)) for i in data]
          for i in data:
              i[0] = i[0].split(',')[-1]
          data[0][0] = 'years'
          data = {i[0]:i[1:] for i in data}
          result = {data['years'][i]:data[pais][i] for i in range(len(data['years']))}
          return result


pais = input('Ingresa el código de un país: ')
validar_codigo(pais)
```

```
 Ingresa el código de un país: asdasd
 Error el codigo no existe!
```

Ejercicio 6

Escribir un programa para gestionar un listado telefónico con los nombres y los teléfonos de los clientes de una empresa. El programa debe tener funciones para crear el archivo con el listado si no existe, para consultar el teléfono de un cliente, añadir el teléfono de un nuevo cliente y eliminar el teléfono de un cliente. El listado debe estar guardado en el archivo de texto listado.txt donde el nombre del cliente y su teléfono deben aparecer separados por comas y cada cliente en una línea distinta.

```
file_name = "listado.txt"

def ingresar_opcion():
  print('''
  ############Opciones##############
```

```
    1: Consultar si existe el archivo
    2: Añadir teléfono de un cliente
    3: Consultar teléfono de un cliente
    4: Eliminar teléfono de un cliente
    ##################################
    ''')
    v = False
    while not v:
      option = int(input("Ingresar una opción: "))
      condition = 5 > option  and option > 0
      if not condition:
        print("Error, debe ingresar una de las opciones de la tabla!!")
      else:
        v = True
    return direccionar(option)

  def direccionar(op):
    if op == 1:
      validar_archivo()
    elif op == 2:
      agregar_telefono()
    elif op == 3:
      consultar_telefono()
    elif op == 4:
      eliminar_telefono()

  def agregar_telefono():
    nom = input('Ingrese nombre del cliente : ')
    tel = input('Ingrese telefono del cliente : ')
    try:
      txt = open(file_name, 'a')
    except FileNotFoundError:
      print('No se encuentra el archivo.')
    else:
      txt.write(nom + ',' + tel + '\n')
      txt.close()
      return print('El teléfono fue ingresado correctamente.')


  def consultar_telefono():
      nom = input('Ingrese nombre del cliente : ')
      try:
          txt = open(file_name, 'r')
      except FileNotFoundError:
          print('No se encuentra el archivo.')
      else:
          t = txt.readlines()
          txt.close()
          t = dict([tuple(i.split(',')) for i in t])
          if nom in t:
              print('El telefono de ' + nom + ' es ' + t[nom])
```

```
            else:
                print('No existe el cliente ' + nom)

    def eliminar_telefono():
        nom = input('Ingrese el nombre del cliente que desea eliminar : ')
        try:
            txt = open(file_name, 'r')
        except FileNotFoundError:
            print('No se encuentra el archivo.')
        else:
            t = txt.readlines()
            txt.close()
            t = dict([tuple(i.split(',')) for i in t])
            if nom in t:
                del t[nom]
                new_t = open(file_name, 'w')
                for key, value in t.items():
                    new_t.write(key + ',' + value)
                new_t.close()
                print('El cliente ' + nom  + ' fue eliminado exitosamente')
            else:
                print('El cliente ' + nom  + ' no se encuentra')

    def validar_archivo():
      try:
        open(file_name)
        print("El archivo existe")
      except FileNotFoundError:
        print("El archivo no existe")
        print("----Creando archivo-----")
        file = open(file_name, "w")
        print(f'######Se creo el archivo con el nombre {file_name} #######')

    ingresar_opcion()


           ############Opciones##############
           1: Consultar si existe el archivo
           2: Añadir teléfono de un cliente
           3: Consultar teléfono de un cliente
           4: Eliminar teléfono de un cliente
           ##################################
```

Ejercicio 7

El archivo cotizacion.csv contiene las cotizaciones de las empresas del IBEX35 con las siguientes columnas: Nombre (nombre de la empresa), Final (precio de la acción al cierre de bolsa), Máximo

(precio máximo de la acción durante la jornada), Mínimo (precio mínimo de la acción durante la jornada), Volumen (Volumen al cierre de bolsa), Efectivo (capitalización al cierre en miles de euros).

1. Construir una función reciba el archivo de cotizaciones y devuelva un diccionario con los datos del archivo por columnas.
2. Construir una función que reciba el diccionario devuelto por la función anterior y cree un archivo en formato csv con el mínimo, el máximo y la media de dada columna.

```
def limpiar(cifra):

    cifra = cifra.replace('.', '')
    cifra = cifra.replace(',','.')
    return float(cifra)

def preprocesado(ruta):

    try:
        f = open(ruta, 'r')
        #print ("El fichero si existe.")
    except FileNotFoundError:
        print('El fichero no existe.')
        return

    lineas = f.readlines()
    f.close()
    claves = lineas[0]
    claves = claves[:-1].split(';')
    cotizaciones = {}
    for i in claves:
        cotizaciones[i] = []
    for linea in lineas[1:]:
        linea = linea[:-1].split(';')
        cotizaciones[claves[0]].append(linea[0])
        for i in range(1, len(cotizaciones)):
            cotizaciones[claves[i]].append(limpiar(linea[i]))
    return cotizaciones


def resumen_cotizacion(cotizaciones, ruta):
    try:
        del (cotizaciones['Nombre'])
        f = open(ruta, 'w')
        f.write('Nombre')
        for clave in cotizaciones.keys():
            f.write(';' + clave)
        f.write('\nMínimo')
        for valores in cotizaciones.values():
```

```
                f.write(';' + str(min(valores)))
            f.write('\nMáximo')
            for valores in cotizaciones.values():
                f.write(';' + str(max(valores)))
            f.write('\nMedia')
            for valores in cotizaciones.values():
                f.write(';' + str(sum(valores) / len(valores)))
            f.close()
            return
        except FileNotFoundError as f:
          print("El fichero no existe." +  str(f))
        except TypeError as t:
          print("Error de tipo." + str(t))

    cotizaciones = preprocesado('cotizacion.csv')
    resumen_cotizacion(cotizaciones, 'resumen-cotizacion.csv')

    try:
      archivo_creado = open('resumen-cotizacion.csv')
      linea_c=archivo_creado.readline()
      while linea_c != '':
          linea_c=archivo_creado.readline()
          print(linea_c)
    except FileNotFoundError as f:
          print("El fichero no existe." + str(f))
```

```
Mínimo;1.0165;4.0675;1.0165;1221.0;2343.09

Máximo;19705.0;19875.0;19675.0;36129692.0;145765.44

Media;2796.7687571428573;3170.1133571428572;3136.510471428572;4252278.514285714;31767.7
```

Ejercicio 8

El archivo [calificaciones.csv]() contiene las calificaciones de un curso. Durante el curso se realizaron dos exámenes parciales de teoría y un examen de prácticas. Los alumnos que tuvieron menos de 4 en alguno de estos exámenes pudieron repetirlo al final del curso (convocatoria ordinaria). Escribir un programa que contenga las siguientes funciones:

1. Una función que reciba el archivo de calificaciones y devuelva una lista de diccionarios, donde cada diccionario contiene la información de los exámenes y la asistencia de un alumno. La lista tiene que estar ordenada por apellidos.

2. Una función que reciba una lista de diccionarios como la que devuelve la función anterior y añada a cada diccionario un nuevo par con la nota final del curso. El peso de cada parcial de teoría en la nota final es de un 30% mientras que el peso del examen de prácticas es de un 40%.
3. Una función que reciba una lista de diccionarios como la que devuelve la función anterior y devuelva dos listas, una con los alumnos aprobados y otra con los alumnos suspensos. Para aprobar el curso, la asistencia tiene que ser mayor o igual que el 75%, la nota de los exámenes parciales y de prácticas mayor o igual que 4 y la nota final mayor o igual que 5

```
def nota(cifra):

    cifra = cifra.replace(',','.')
    return float(cifra)

def calificaciones(ruta):

    try:
        f = open(ruta, 'r')
    except FileNotFoundError:
        #print('El fichero no existe.')
        calificaciones="error"
        return("error")
    else:
      lineas = f.readlines()

      f.close()

      claves = lineas[0][:-1].split(";")

      calificaciones = []

      for i in lineas[1:]:

          valores = i[:-1].split(";")

          alumno = {}

          for j in range(len(valores)):
              alumno[claves[j]] = valores[j]

          calificaciones.append(alumno)
      return calificaciones

def añadir_nota_final(calificaciones):
     if calificaciones=="error":
       return
     else:
      def nota_final(alumno):
```

```
            if alumno['Ordinario1']:
                parcial1 = nota(alumno['Ordinario1'])
            elif alumno['Parcial1']:
                parcial1 = nota(alumno['Parcial1'])
            else:
                parcial1 = 0
            if alumno['Ordinario2']:
                parcial2 = nota(alumno['Ordinario2'])
            elif alumno['Parcial2']:
                parcial2 = nota(alumno['Parcial2'])
            else:
                parcial2 = 0
            if alumno['OrdinarioPracticas']:
                practicas = nota(alumno['OrdinarioPracticas'])
            elif alumno['Practicas']:
                practicas = nota(alumno['Practicas'])
            else:
                practicas = 0
            alumno['Final1'] = parcial1
            alumno['Final2'] = parcial2
            alumno['FinalPracticas'] = practicas
            alumno['NotaFinal'] = parcial1 * 0.3 + parcial2 * 0.3 + practicas * 0.4
            return alumno


      return list(map(nota_final, calificaciones))

def aprobados_suspensos(calificaciones):

    aprobados = []
    suspensos = []

    for alumno in calificaciones:

        if all([int(alumno['Asistencia'][:-1]) >= 75, alumno['Final1'] >= 4, alumno['Final2']
            aprobados.append(alumno['Apellidos'] + ', ' + alumno['Nombre'])
        else:
            suspensos.append(alumno['Apellidos'] + ', ' + alumno['Nombre'])
    return aprobados, suspensos


print(añadir_nota_final(calificaciones('calificaciones.csv')))
if calificaciones('calificaciones.csv') == "error":
  print("Archivo no encontrado")
else:
  aprobados, suspensos = aprobados_suspensos(añadir_nota_final(calificaciones('calificaciones
  print('Lista de aprobados:', aprobados)
  print('Lista de suspensos:', suspensos)
```

```
[{'Apellidos': 'Anido Bonet', 'Nombre': 'David', 'Asistencia': '90%', 'Parcial1': '5,5'
Archivo no encontrado
```

✓ 0 s completado a las 8:33