ESTRUCTURA DE DATOS Y PROGRAMACIÓN ORIENTADA A OBJETOS

Tema Nº12:Manejo de archivos

Indicador de logro Nº12:Ejecuta archivos de texto open, close, read, readline, readlines, write a través de un lenguaje de programación.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº12:**

Manejo de archivos

**Subtema 12.1:**

Manejo de archivos

Si te fijas en el escritorio de tu ordenador, ¿qué ves? Cientos de archivos que se extienden por todas partes, ¿verdad? Casi todo el mundo cae en este penoso hábito, nos resulta imposible, por un extraño sendero del destino, el poner en orden los ficheros de nuestro equipo. La clave de este capítulo es saber cómo tratar todos esos archivos, vamos a hablar de Python, y de cómo trabajar con ficheros utilizando Python.

Con el fin de abrir un archivo utilizando Python, usaremos la función **open()**.

**open()** devuelve un objeto file, y comúnmente se utiliza con dos argumentos: ***open(nombre del archivo, modo)***. El nombre del archivo hace referencia al nombre (y la ruta) del archivo que desees abrir. El modo es un valor string que indica cómo se va a abrir el archivo. Aquí puedes ver los distintos modos en los que se puede abrir un archivo:

* **r** : Abrir fichero para lectura. El puntero se posiciona al principio del fichero
* **r+** : Abrir fichero para lectura y escritura. El puntero se posiciona al principio del fichero
* **w** : Trunca a cero la longitud o crea un fichero de texto para escritura. El puntero se posiciona al principio del fichero
* **w+** : Abrir fichero para lectura y escritura. Si el fichero no existe, se crea, de lo contrario se trunca. El puntero se posiciona al principio del fichero
* **a** : Abrir fichero para lectura. Se creará el fichero si no existe. El puntero se posiciona al final del fichero.
* **a+** : Abrir fichero para lectura y escritura. Se creará el fichero si no existe. El puntero se posiciona al final del fichero.
* **ACCEDIENDO A ARCHIVOS DESDE EL CÓDIGO PYTHON:**

Una actividad muy común en un programa es el almacenamiento y recuperación de datos almacenado en un dispositivo secundario (disco duro). Existen muchos modos de almacenar datos como son los archivos de texto, archivos binarios, bases de datos etc. Veremos en este concepto como almacenar y recuperar datos de un archivo de texto.

* **ARCHIVO DE TEXTO:**

Un archivo de texto contiene un conjunto de caracteres estructurados en distintas líneas. Es un formato de archivo ampliamente utilizado como pueden ser:

* El código fuente de un script en Python se almacena en un archivo de texto (igual que cualquier otro lenguaje de programación).
* Archivos HTML, CSS, XML se almacenan en archivos de texto.
* Archivos JSON etc.
* **CREACIÓN DE UN ARCHIVO DE TEXTO Y ALMACENAMIENTO DE DATOS:**

Como es una actividad tan común en todo programa el lenguaje Python incluye por defecto todas las funcionalidades para trabajar con archivos de texto.

Veamos con un ejemplo como crear y almacenar caracteres en un archivo de texto.

**EJEMPLO:**

Crear un archivo de texto llamado **datos.txt**, almacenar tres líneas de texto. Abrir luego el archivo creado con un editor de texto.

archi1 = open("datos.txt", "w")

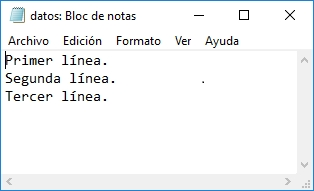
archi1.write("Primer línea.\n")

archi1.write("Segunda línea.\n")

archi1.write("Tercer línea.\n")

archi1.close()

Luego de ejecutar este programa (no muestra nada por pantalla) debemos abrir con un editor de texto el archivo de texto que se acaba de crear llamado **datos.txt**:



**EXPLICACIÓN:**

1. Para crear un archivo de texto debemos llamar a la función open y pasar dos parámetros, el primero indica el nombre del archivo a crear y el segundo un string con el caracter **"w"** (la **"w"** write indica crear el archivo de texto):

archi1 = open("datos.txt", "w")

Si el archivo **datos.txt** ya existe luego se crea uno nuevo y se borra el actual.

1. El archivo se crea en la misma carpeta donde se está ejecutando el script de Python, si necesitamos que se cree en otra carpeta podemos indicar el path del mismo:

archi1 = open("c:/administracion/datos.txt", "w")

Si indicamos un path inexistente se genera un error.

1. Para grabar caracteres en el archivo de texto utilizamos el método **write** y le pasamos un string a grabar:

archi1.write("Primer línea.\n")

Mediante la sintaxis **\n** indicamos que debe almacenarse un salto de línea en el archivo de texto.

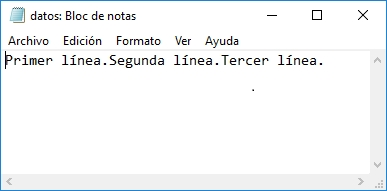
1. Si no incluimos los respectivos **\n**:

archi1.write("Primer línea.")

archi1.write("Segunda línea.")

archi1.write("Tercer línea.")

El archivo de texto tiene una sola línea:



1. Luego de trabajar con el archivo de texto debemos liberarlo para que otro programa pueda acceder a su contenido:

archi1.close()

* **LECTURA DE UN ARCHIVO DE TEXTO:**

**EJEMPLO:**

Leer el contenido del archivo de texto **datos.txt**.

archi1 = open("datos.txt", "r")

contenido = archi1.read()

print(contenido)

archi1.close()

**EXPLICACIÓN:**

1. Luego de ejecutar este programa:
2. Llamamos a la función **open** y le pasamos el nombre del archivo a leer y un string con el caracter **"r"** (**read**):

archi1 = open("datos.txt", "r")

1. El método **read** recupera todas las líneas del archivo de texto (recupera el archivo completo):

contenido = archi1.read()

1. Luego de trabajar con el archivo de texto debemos liberarlo para que otro programa pueda acceder a su contenido:

archi1.close()

1. El método **read** sin parámetros retorna todos los caracteres almacenados en el archivo, opcionalmente podemos pasar un entero que represente la cantidad de caracteres a leer:

archi1 = open("datos.txt", "r")

contenido = archi1.read(6)

print(contenido) # imprime: Primer

archi1.close()

* **LECTURA DE UN ARCHIVO DE TEXTO LÍNEA A LÍNEA:**

En algunas situaciones podemos necesitar leer el contenido de un archivo de texto línea a línea. Disponemos de un método llamado **readline** que lee una línea completa del archivo, inclusive retorna el carácter **\n** de fin de línea.

**EJEMPLO:**

Leer el contenido del archivo de texto **datos.txt** línea a línea.

archi1 = open("datos.txt", "r")

línea = archi1.readline()

while línea != '':

print(linea, end = '')

línea = archi1.readline()

archi1.close()

**EXPLICACIÓN:**

1. Antes de la estructura repetitiva **while** leemos la primera línea del archivo de texto:

linea = archi1.readline()

1. Luego mientras la variable **linea** sea distinta a un string vacío imprimimos el contenido de la variable **linea** evitando que genere un salto de línea la función print mediante la asignación del parámetro end con el valor de ' ':

while línea != '':

print(linea, end = '')

línea = archi1.readline()

También dentro de la estructura **while** leemos las siguientes líneas.

1. Podemos recorrer el archivo leyendo línea a línea utilizando la estructura repetitiva **for**:

archi1 = open("datos.txt", "r")

for linea in archi1:

print(linea, end = '')

archi1.close()

Es una forma más compacta de recorrer cada una de las líneas del archivo de texto.

* **ALMACENAR UN ARCHIVO DE TEXTO EN UNA LISTA:**

Mediante el método **readlines** podemos recuperar cada una de las líneas del archivo de texto y almacenarlas en una lista.

**EJEMPLO:**

Leer el contenido del archivo de texto **datos.txt** y almacenar sus líneas en una lista. Imprimir la cantidad de líneas que tiene el archivo y su contenido.

archi1 = open("datos.txt", "r")

líneas = archi1.readlines()

print('El archivo tiene', len(lineas), 'líneas')

print('El contenido del archivo')

for linea in lineas:

print(linea, end = '')

archi1.close()

**EXPLICACIÓN:**

1. Mediante la llamada al método **readlines** obtenemos una lista con todas las líneas del archivo **datos.txt**:

lineas = archi1.readlines()

1. Imprimimos seguidamente la cantidad de elementos de la lista que coincide con la cantidad de líneas que tiene el archivo de texto:

print('El archivo tiene', len(lineas), 'líneas')

1. Para imprimir el contenido del archivo de texto procedemos a imprimir la lista:

print('El contenido del archivo')

for linea in lineas:

print(linea, end = '')

1. Una variante del método **readlines** es pasar como un parámetro entero que represente que línea queremos recuperar:

linea = archi1.readlines(2)

print(linea, end = '') # imprime ['Primer línea.\n']

* **ABRIR UN ARCHIVO DE TEXTO PARA AÑADIR LÍNEAS:**

Hemos visto que cuando llamamos a la función **open** el segundo parámetro puede ser **"w"**, **"r"** y si queremos que se abra para añadir sin borrar las líneas actuales del archivo debemos hacerlo con el parámetro **"a"** (**append**).

**EJEMPLO:**

Abrir el archivo de texto **datos.txt** y luego agregar 2 líneas. Imprimir luego el archivo completo.

archi1 = open("datos.txt", "a")

archi1.write("nueva línea 1\n")

archi1.write("nueva línea 2\n")

archi1.close()

archi1 = open("datos.txt", "r")

contenido = archi1.read()

print(contenido)

archi1.close()

**EXPLICACIÓN:**

1. Abrimos el archivo **datos.txt** en modo agregar:

archi1 = open("datos.txt", "a")

1. Almacenamos dos líneas en el archivo:

archi1.write("nueva línea 1\n")

archi1.write("nueva línea 2\n")

1. Cerramos el archivo y lo volvemos a abrir para leer su contenido:

archi1.close()

archi1 = open("datos.txt", "r")

contenido = archi1.read()

print(contenido)

archi1.close()

* **ABRIR UN ARCHIVO PARA LEER Y AGREGAR DATOS:**

Hay una cuarta forma de abrir un archivo indicando en el segundo parámetro de la función open el string **"r+"**, con dicha opción podemos leer y escribir.

**EJEMPLO:**

Abrir un archivo de texto con el parámetro **"r+"**, imprimir su contenido actual y agregar luego dos líneas al final.

archi1 = open("datos.txt", "r+")

contenido = archi1.read()

print(contenido)

archi1.write("Otra línea 1\n")

archi1.write("Otra línea 2\n")

archi1.close()

* **CODIFICACIÓN DE CARACTERES UTF-8:**

Actualmente se utiliza mucho la codificación de caracteres **UTF-8** la que nos permite representar una infinidad de caracteres de distintos idiomas y símbolos.

En Python debemos indicar cuando abrimos el archivo de texto mediante el parámetro **encoding** la codificación de caracteres utilizada.

**EJEMPLO:**

Crear un archivo de texto llamado **datos.txt** con una codificación **UTF-8**, almacenar tres líneas de texto. Abrir luego el archivo creado con el editor VS Code.

archi1 = open("datos.txt", "w", encoding = "utf-8")

archi1.write("Primer línea.\n")

archi1.write("Segunda línea.\n")

archi1.write("Tercer línea.\n")

archi1.close()

Ahora abrimos el archivo de texto creado con un editor que permite trabajar con archivos con codificación **UTF-8** y verán el resultado.

En esta parte del proyecto se implementarán los métodos para que funcione los archivos de texto:

* Clientes.txt
* DetalleVenta.txt
* Factura.txt
* Producto.txt

**arregloClientes.py**

from Controlador.clientes import \*

class ArregloClientes:

    def \_\_init\_\_(self):

        self.dataClientes = []

        self.cargar()

    def cargar(self):

        archivo = open("Modelo/Clientes.txt", "r", encoding = "utf-8")

        for linea in archivo.readlines():

            columna = str(linea).split(",")

            dni = columna[0]

            nombres = columna[1]

            apellido\_paterno = columna[2]

            apellido\_materno = columna[3]

            direccion = columna[4]

            telefono = columna[5].strip()

            objCli = Cliente(dni, nombres, apellido\_paterno, apellido\_materno, direccion, telefono)

            self.adicionaCliente(objCli)

        archivo.close()

    def grabar(self):

        archivo = open("Modelo/Clientes.txt", "w+", encoding = "utf-8")

        for i in range(self.tamañoArregloCliente()):

            archivo.write(str(self.devolverCliente(i).getDniCliente()) + ","

            + str(self.devolverCliente(i).getNombresCliente()) + ","

            + str(self.devolverCliente(i).getApellidoPaternoCliente()) + ","

            + str(self.devolverCliente(i).getApellidoMaternoCliente()) + ","

            + str(self.devolverCliente(i).getDireccionCliente()) + ","

            + str(self.devolverCliente(i).getTelefonoCliente()) + "\n")

        archivo.close()

    def adicionaCliente(self, objCli):

        self.dataClientes.append(objCli)

    def devolverCliente(self, pos):

        return self.dataClientes[pos]

    def tamañoArregloCliente(self):

        return len(self.dataClientes)

    def buscarCliente(self, dni):

        for i in range(self.tamañoArregloCliente()):

            if dni == self.dataClientes[i].getDniCliente():

                return i

        return -1

    def eliminarCliente(self, indice):

        del(self.dataClientes[indice])

**arregloDetalleVenta.py**

from Controlador.detalleVenta import \*

from Controlador.arregloClientes import \*

from Controlador.detalleVenta import \*

from Controlador.arregloProductos import \*

class ArregloDetalleVenta:

    def \_\_init\_\_(self):

        self.dataDetalleVenta = []

        self.cargar()

    def cargar(self):

        archivo = open("Modelo/DetalleVenta.txt", "r", encoding = "utf-8")

        for linea in archivo.readlines():

            columna = str(linea).split(",")

            numero\_documento = columna[0]

            numero\_item = columna[1]

            codigo\_producto = columna[2]

            precio\_venta = columna[3]

            cantidad = columna[4].strip()

            objDetVent = DetalleVenta(numero\_documento, numero\_item, codigo\_producto, precio\_venta, cantidad)

            self.adicionaDetalleVenta(objDetVent)

        archivo.close()

    def grabar(self):

        archivo = open("Modelo/DetalleVenta.txt", "w+", encoding = "utf-8")

        for i in range(self.tamañoDetalleVenta()):

            archivo.write(str(self.devolverDetalleVenta(i).getNDocumentoVenta()) + ","

            + str(self.devolverDetalleVenta(i).getNItem()) + ","

            + str(self.devolverDetalleVenta(i).getCodigoProducto()) + ","

            + str(self.devolverDetalleVenta(i).getPrecioVenta()) + ","

            + str(self.devolverDetalleVenta(i).getCantidad()) + "\n")

        archivo.close()

    def adicionaDetalleVenta(self, objDetVent):

        self.dataDetalleVenta.append(objDetVent)

    def devolverDetalleVenta(self, pos):

        return self.dataDetalleVenta[pos]

    def tamañoDetalleVenta(self):

        return len(self.dataDetalleVenta)

    def buscarDetalleVenta(self, nDocumentoVenta):

        for i in range(self.tamañoDetalleVenta()):

            if nDocumentoVenta == self.dataDetalleVenta[i].getNDocumentoVenta:

                return i

        return -1

    def eliminarDetalleVenta(self, indice):

        del(self.dataDetalleVenta[indice])

    def imprimirDetalleVenta(self, nDocumentoVenta, aPro):

        for objDetVent in self.dataDetalleVenta:

            if objDetVent.getNDocumentoVenta() == nDocumentoVenta:

                pos = aPro.buscarProducto(objDetVent.getCodigoProducto())

                objPro = aPro.devolverProducto(pos)

                objDetVent.imprimirLineaDetalleVenta(objPro)

**arregloFactura.py**

from Controlador.factura import \*

class ArregloFactura:

    def \_\_init\_\_(self):

        self.dataFactura = []

        self.cargar()

    def cargar(self):

        archivo = open("Modelo/Factura.txt", "r", encoding = "utf-8")

        for linea in archivo.readlines():

            columna = str(linea).split(",")

            numero\_documento = columna[0]

            fecha = columna[1]

            igv = columna[2]

            total\_precio\_venta = columna[3]

            dni = columna[4].strip()

            objFact = Factura(numero\_documento, fecha, igv, total\_precio\_venta, dni)

            self.adicionaFactura(objFact)

        archivo.close()

    def grabar(self):

        archivo = open("Modelo/Factura.txt", "w+", encoding = "utf-8")

        for i in range(self.tamañoFactura()):

            archivo.write(str(self.devolverFactura(i).getNDocumentoVenta()) + ","

            + str(self.devolverFactura(i).getFecha()) + ","

            + str(self.devolverFactura(i).getIgv()) + ","

            + str(self.devolverFactura(i).getTotalPrecioVenta()) + ","

            + str(self.devolverFactura(i).getRuc()) + ","

            + str(self.devolverFactura(i).getEstado()) + "\n")

        archivo.close()

    def adicionaFactura(self, objFact):

        self.dataFactura.append(objFact)

    def devolverFactura(self, pos):

        return self.dataFactura[pos]

    def tamañoFactura(self):

        return len(self.dataFactura)

    def buscarFactura(self, nDocumentoVenta):

        for i in range(self.tamañoFactura()):

            if nDocumentoVenta == self.dataFactura[i].getNDocumentoVenta():

                return i

        return -1

    def eliminarFactura(self, indice):

        del(self.dataFactura[indice])

    def nroSerie(self):

        self.series = []

        for i in range(self.tamañoFactura()):

            self.series.append(self.dataFactura[i].getNDocumentoVenta())

        return len(self.series)

**arregloProductos.py**

from Controlador.productos import \*

class ArregloProductos:

    def \_\_init\_\_(self):

        self.dataProductos = []

        self.cargar()

    def cargar(self):

        archivo = open("Modelo/Productos.txt", "r", encoding = "utf-8")

        for linea in archivo.readlines():

            columna = str(linea).split(",")

            codigo = columna[0]

            nombre = columna[1]

            descripcion = columna[2]

            stock\_minimo = columna[3]

            stock\_actual = columna[4]

            precio\_costo = columna[5]

            precio\_venta = columna[6]

            proveedor = columna[7]

            almacen = columna[8].strip()

            objPro = Producto(codigo, nombre, descripcion, stock\_minimo, stock\_actual,

precio\_costo, precio\_venta, proveedor, almacen)

            self.adicionaProducto(objPro)

        archivo.close()

    def grabar(self):

        archivo = open("Modelo/Productos.txt", "w+", encoding = "utf-8")

        for i in range(self.tamañoArregloProducto()):

            archivo.write(str(self.devolverProducto(i).getCodigo()) + ","

            + str(self.devolverProducto(i).getNombre()) + ","

            + str(self.devolverProducto(i).getDescripcion()) + ","

            + str(self.devolverProducto(i).getStockMinimo()) + ","

            + str(self.devolverProducto(i).getStockActual()) + ","

            + str(self.devolverProducto(i).getPrecioCosto()) + ","

            + str(self.devolverProducto(i).getPrecioVenta()) + ","

            + str(self.devolverProducto(i).getProveedor()) + ","

            + str(self.devolverProducto(i).getAlmacen()) + "\n")

        archivo.close()

    def adicionaProducto(self, objPro):

        self.dataProductos.append(objPro)

    def devolverProducto(self, pos):

        return self.dataProductos[pos]

    def tamañoArregloProducto(self):

        return len(self.dataProductos)

    def buscarProducto(self, codigo):

        for i in range(self.tamañoArregloProducto()):

            if codigo == self.dataProductos[i].getCodigo():

                return i

        return -1

    def eliminarProducto(self, indice):

        del(self.dataProductos[indice])

    def actualizarStock(self, cantidad, codigoProducto):

        for i in range(self.tamañoArregloProducto()):

            #objPro = self.dataProductos[i]

            if self.dataProductos[i].getCodigo() == codigoProducto:

                self.dataProductos[i].setStockActual(cantidad)

                #objPro.setStockActual(cantidad)