**Módulo de 12:**

**Manejo de información en archivos externos**

El manejo de archivos es una tarea común y útil para leer y escribir información en files.

Primero veremos el manejo del archivo local, enla misma carpeta donde tenemos nuestri archivo Python.  
Luego veremos los sistemas de carpetas y discos que depende en mucho del sistema operativo del equipo.

|  |
| --- |
| **Es muy importante aclarar que el flujo del programa se altera con try except y finally** |

Antes se usa simplemente open para abrir el archivo externo y luego close para cerrarlo. Hoy en día se recomienda el uso de with que permite una apertura y cierre casi inmediato.

En principio veremos solamente dos argumentos para open el archivo o **file** y el **modo**

El nombre del archivo debe tener una extensión que corresponda al tipo de escritura que contiene.

Texto plano txt

Excel xls, xlsx o xlsm

Comma Separated Values csv

JavaScript Object Notation. json

Pickle pkl

eXtensible Markup Language xml

etc

La primera diferencia es si se graba como texto plano (el que podemos leer con un block de notas, vim, etc) y los que se escriben en binario.

Los textos planos usan la t de text o la omiten, los binarios deben usarse con la b de binary

Se pueden escribir (write) , leer (read), agregar al final (append) y x para crearlo sin contenido

·

|  |  |  |  |
| --- | --- | --- | --- |
| Modo  texto | Modo  binario | Admite  **+** | Descripción de la acción |
| r / rt  (por defecto) | rb | + Si  lectura y escritura | Read  Abre un archivo de solo lectura (texto o binario).(Debe existir)  El puntero de archivo se coloca al principio del archivo. |
| w / wt | wb | + Si  escritura y lectura | Write  Abre un archivo (nuevo si no existe) solo para escritura (texto o binario).  Sobrescribe el archivo si existe. |
| a / at | ab | + Si  agregar y leer | Append  Abre un archivo para agregar información en el (texto o binario).  Si el archivo no existe, crea un nuevo archivo para escribir. |
| x |  |  | Crea un archivo vació.  Genera un error si ya existe |

.

Hay muchos mas argumentos que no son parte del curso.

buffering=-1, encoding=None, errors=None, newline=None, closefd=True, opener=None

|  |
| --- |
| <https://es.wikipedia.org/wiki/Windows-1251> https://es.wikipedia.org/wiki/Windows-1252  https://es.wikipedia.org/wiki/UTF-8  <https://es.wikipedia.org/wiki/UTF-16> <https://es.wikipedia.org/wiki/Unicode> |

Para decodificar ASCII, CP1251 / 52 o cualquier otra codificación a UTF-8 o Unicode en Python, puedes usar el método decode de las cadenas de texto o el Módulo codecs

|  |
| --- |
| **.encode y .decode** debe darse al menos como idea, principalmente porque windows tiene cp1251 y UTF8 - Unicode en Python y veremos que hay mucho material que requiere este concepto  cp1251\_text = b'\xd1\xf2\xe0\xee\xf2\xe8\xf0\xe0'  utf8\_text = cp1251\_text.decode('cp1252').encode('utf-8')  print(utf8\_text.decode('utf-8'))  # Decoding from ASCII to UTF-8  ascii\_text = b'Hello, World!'  utf8\_text = ascii\_text.decode('ascii').encode('utf-8')  print(utf8\_text.decode('utf-8'))  Aqui decodificamos el texto codificado desde la codificación especificada (CP1251 o ASCII) usando el método decode. Luego, lo codificamos en UTF-8 usando el método encode. Finalmente, decodificamos el texto UTF-8 para obtener la cadena Unicode. |

En Python, los objetos de cadena de texto (str) utilizan el formato de codificación Unicode por defecto.

Unicode es un estándar que asigna un número único (llamado punto de código) a cada carácter utilizado en la mayoría de los sistemas de escritura del mundo.

Cuando se trata de codificación de bytes, Python utiliza el formato UTF-8 como codificación predeterminada.

UTF-8 es una forma de codificar los puntos de código Unicode en secuencias de bytes variables, lo que permite representar de manera eficiente caracteres de múltiples idiomas.

|  |
| --- |
| En resumen, Python utiliza objetos de cadena de texto Unicode (str) internamente y utiliza UTF-8 como codificación predeterminada al trabajar con bytes. |

·

|  |
| --- |
| salida="""En Python, los objetos de cadena de texto (str) utilizan el formato de codificación Unicode por defecto.  Cuando se trata de codificación de bytes, Python utiliza el formato UTF-8 como codificación predeterminada. """  objeto\_io = open ("borrar.txt",mode="w")  objeto\_io.write(salida)  objeto\_io.close() |

·

|  |
| --- |
| contenido\_objeto\_python='En Python, los objetos de cadena de texto (str) utilizan el formato de codificación Unicode por defecto.\nCuando se trata de codificación de bytes, Python utiliza el formato UTF-8 como codificación predeterminada. ' |

**Escritura:**

Formato antiguo

|  |
| --- |
| salida="""En Python, los objetos de cadena de texto (str) utilizan el formato de codificación Unicode por defecto.  Cuando se trata de codificación de bytes, Python utiliza el formato UTF-8 como codificación predeterminada. """  objeto\_io = open ("borrar.txt",mode="w")  objeto\_io.write(salida)  objeto\_io.close() |

·

Formato moderno

|  |
| --- |
| with open ("borrar.txt",mode="w") as objeto\_io:  objeto\_io.write(salida) |

·

|  |
| --- |
| Ir a consola al directorio donde están nuestros archivos y veremos que encontraremos el archivo **borrar.txt** con fecho y hora del momento que corriste |

·

**Lectura:**

·

Para leer el contenido de un archivo, se utilizan los métodos read(), readline() o readlines()

·

Formato antiguo

|  |
| --- |
| objeto\_InOut = open("borrar.txt", "r")  contenido\_objeto\_python = objeto\_InOut.read()  print (f"{contenido\_objeto\_python=}")  objeto\_InOut.close() |

·

Formato moderno

|  |
| --- |
| with open("borrar.txt", "r") as objeto\_InOut:  contenido\_objeto\_python = objeto\_InOut.read()  print (f"{contenido\_objeto\_python=}") |

·

ambos me daran el mismo resultado

|  |
| --- |
| contenido\_objeto\_python='En Python, los objetos de cadena de texto (str) utilizan el formato de codificación Unicode por defecto.\nCuando se trata de codificación de bytes, Python utiliza el formato UTF-8 como codificación predeterminada. ' |

·

|  |
| --- |
| En realidad hay diferencias muy importantes.  No requiere cerrar (close) ya que al salir del bloque with esto se hace automáticamente.  Por otro lado el objeto de intercambio no existe fuera del with y ocupa mucha menos memoria |

·

**Lee una línea del archivo**

|  |
| --- |
| with open("borrar.txt", "r") as objeto\_InOut:  linea = objeto\_InOut.readline() # Lee una línea del archivo  print (f"{linea=}") |

·

|  |
| --- |
| linea='En Python, los objetos de cadena de texto (str) utilizan el formato de codificación Unicode por defecto.\n' |

Se puede trabajar en un bucle for leyendo line a linea

**Lee la totalidad de líneas del archivo y las guarda cada linea como objeto de una lista**

·

|  |
| --- |
| with open("borrar.txt", "r") as objeto\_InOut:  lineas = objeto\_InOut.readlines() # Lee una línea del archivo  print (f"{lineas=}") |

·

|  |
| --- |
| lineas=['En Python, los objetos de cadena de texto (str) utilizan el formato de codificación Unicode por defecto.\n', 'Cuando se trata de codificación de bytes, Python utiliza el formato UTF-8 como codificación predeterminada. '] |

·

La serialización de archivos es el proceso de convertir una estructura de datos o un objeto en una secuencia de bytes para que pueda ser almacenado en un archivo o transmitido a través de una red. La serialización permite guardar el estado de un objeto de manera persistente, lo que significa que se puede recuperar posteriormente y restaurar en su forma original.

La serialización es útil cuando necesitas almacenar objetos complejos en disco, transferir datos entre diferentes plataformas o comunicarte con sistemas distribuidos. Al serializar un objeto, se guarda su estado actual, incluyendo los valores de sus atributos y la estructura de los datos.

La serialización puede realizarse en diferentes formatos, como JSON (JavaScript Object Notation), XML (eXtensible Markup Language), protocol buffers, entre otros. Estos formatos definen reglas y convenciones para convertir los datos en una representación legible por máquina.

En Python, el Módulo por excelencia es **pickle** proporciona funcionalidad para serializar y deserializar objetos en formato binario. También existen otras bibliotecas populares de serialización como json y xml.etree.ElementTree que permiten la serialización en formatos JSON y XML, respectivamente.

**Serialización:**

Consiste en convertir un objeto de Python (normalmente una lista o diccionario) en un string.

**Deserialización:**

Consiste en convertir un string en un objeto de Python (normalmente una lista o diccionario).

|  |
| --- |
| import **pickle**  obj\_diccionario={"uno":{"ingles":"One","francés":"Un"},  "dos":{"ingles":"Two","francés":"Deux"},  "tres":{"ingles":"Three","francés":"Trois"},  "cuatro":{"ingles":"Four","francés":"Quatre"},  "cinco":{"ingles":"Five","francés":"Cinq"}  }  with open ("borrar.**pkl**" , mode="w**b**") as objeto\_InOut:  **pickle.dump**(obj\_diccionario,objeto\_InOut)  #------------------------------------------------------------------------------------  with open ("borrar.**pkl**" , mode="r**b**") as objeto\_InOut:  nuevo=**pickle.load**(objeto\_InOut)  print (f"{nuevo=}") |

·

|  |
| --- |
| nuevo={'uno': {'ingles': 'One', 'francés': 'Un'}, 'dos': {'ingles': 'Two', 'francés': 'Deux'}, 'tres': {'ingles': 'Three', 'francés': 'Trois'}, 'cuatro': {'ingles': 'Four', 'francés': 'Quatre'}, 'cinco': {'ingles': 'Five', 'francés': 'Cinq'}} |

·

|  |  |
| --- | --- |
| DUMP: Graba un objeto Python a un archivo JSON  DUMPS: Convierte un objeto Python a uno JSON (objetos json se envian por puertos a otras app) | |
| LOAD: Lee un archivo JSON a un objeto Python  LOADS: Convierte un objeto JSON a uno Python (objetos json se envian por puertos a otras app) | |
| Objetos Python | Objetos JSON |
| list, tuple | array |
| str | string |
| int, float, int & float-derived Enums | number |
| True | true |
| False | false |
| dict | object |
| None | null |

·

|  |
| --- |
| **indent** define el numero de indents  **sort\_keys** ordena el resultado  **separators**=(". ", " = ") Cambia el separador default  Syntax:  json.dumps(  obj,  \*,  skipkeys=False,  ensure\_ascii=True,  check\_circular=True,  allow\_nan=True,  cls=None,  indent=None,  separators=None,  default=None,  sort\_keys=False,  \*\*kw ) |
| Parameters:  obj: serializar obj como una secuencia con formato JSON  skipkeys:  Si skipkeys es verdadero (predeterminado: Falso), las teclas de dictado que no sean de un tipo básico (str, int, float, bool, None) se omitirán en lugar de generar un TypeError.  ensure\_ascii:  Si asegurar\_ascii es verdadero (el valor predeterminado), se garantiza que la salida tendrá todos los caracteres que no sean ASCII entrantes escapados. si asegurar\_ascii es falso, estos caracteres se mostrarán tal cual.  check\_circular :  Si check\_circular es falso (predeterminado: verdadero), se omitirá la verificación de referencia circular para los tipos de contenedor y una referencia circular dará como resultado un OverflowError (o algo peor).  allow\_nan :  Si allow\_nan es falso (predeterminado: verdadero), entonces será un ValueError para serializar valores flotantes fuera de rango (nan, inf, -inf) en estricto cumplimiento de la especificación JSON. Si allow\_nan es verdadero, se utilizarán sus equivalentes de JavaScript (NaN, Infinity, -Infinity).  Indent:  Si la sangría – indentacion - es un entero no negativo o una cadena, los elementos de la matriz JSON y los miembros del objeto se imprimirán con ese nivel de sangría. Un nivel de sangría de 0, negativo o "" solo insertará líneas nuevas. Ninguno (el valor predeterminado) selecciona la representación más compacta. El uso de una sangría de entero positivo sangra tantos espacios por nivel. Si la sangría es una cadena (como "\\t"), esa cadena se usa para sangrar cada nivel.  Separators :  Si se especifica, los separadores deben ser una tupla (item\_separator, key\_separator). El valor predeterminado es ("",":") si la sangría es Ninguno y ("",":") de lo contrario. Para obtener la representación JSON más compacta, debe especificar que se eliminen los espacios en blanco.  Default:  Si se especifica, el valor predeterminado debe ser una función que se llame para objetos que de otro modo no se pueden serializar. Debería devolver una versión codificable JSON del objeto o generar un TypeError. Si no se especifica, se genera TypeError.  sort\_keys :  Si sort\_keys es verdadero (predeterminado: falso), la salida de los diccionarios se ordenará por clave. |

·

|  |
| --- |
| import **json**  obj\_diccionario={"uno":{"ingles":"One","francés":"Un"},  "dos":{"ingles":"Two","francés":"Deux"},  "tres":{"ingles":"Three","francés":"Trois"},  "cuatro":{"ingles":"Four","francés":"Quatre"},  "cinco":{"ingles":"Five","francés":"Cinq"}  }  with open ("borrar.**json**" , mode="w") as objeto\_InOut:  **json**.**dump**(obj\_diccionario,objeto\_InOut)  #------------------------------------------------------------------------------------  with open ("borrar.**json**" , mode="r") as objeto\_InOut:  nuevo=**json.load**(objeto\_InOut)  print (f"{nuevo=}") |

·

|  |
| --- |
| nuevo={'uno': {'ingles': 'One', 'francés': 'Un'}, 'dos': {'ingles': 'Two', 'francés': 'Deux'}, 'tres': {'ingles': 'Three', 'francés': 'Trois'}, 'cuatro': {'ingles': 'Four', 'francés': 'Quatre'}, 'cinco': {'ingles': 'Five', 'francés': 'Cinq'}} |

·

Los archivos json no son binarios y se pueden abrir con muchas aplicaciones, entre ellas el navegador web

|  |
| --- |
| import json  obj\_diccionario={"uno":{"ingles":"One","francés":"Un"},  "dos":{"ingles":"Two","francés":"Deux"},  "tres":{"ingles":"Three","francés":"Trois"},  "cuatro":{"ingles":"Four","francés":"Quatre"},  "cinco":{"ingles":"Five","francés":"Cinq"}  }  with open(f"borrar.json", mode="w",encoding='utf-8') as objeto\_InOut:  json.dump(obj\_diccionario,objeto\_InOut ,ensure\_ascii=False, indent=4)  #------------------------------------------------------------------------------------  with open ("borrar.json" , mode="r" ,encoding='utf-8') as objeto\_InOut:  nuevo=json.load(objeto\_InOut)  print (f"{nuevo=}") |

·

|  |
| --- |
| nuevo={'uno': {'ingles': 'One', 'francés': 'Un'}, 'dos': {'ingles': 'Two', 'francés': 'Deux'}, 'tres': {'ingles': 'Three', 'francés': 'Trois'}, 'cuatro': {'ingles': 'Four', 'francés': 'Quatre'}, 'cinco': {'ingles': 'Five', 'francés': 'Cinq'}}  #en un lector de texto, block de notas o navegador web  {  "uno": {  "ingles": "One",  "francés": "Un"  },  "dos": {  "ingles": "Two",  "francés": "Deux"  },  "tres": {  "ingles": "Three",  "francés": "Trois"  },  "cuatro": {  "ingles": "Four",  "francés": "Quatre"  },  "cinco": {  "ingles": "Five",  "francés": "Cinq"  }  } |

·

Los archivos json no son binarios y se pueden abrir con muchas aplicaciones, entre ellas el navegador web

|  |
| --- |
| Tener en cuenta que al utilizar la serialización, es recomendable tomar precauciones de seguridad, especialmente cuando se deserializan datos provenientes de fuentes no confiables. La deserialización de datos maliciosos podría conducir a vulnerabilidades de seguridad, como la ejecución de código no autorizado. |

**Manejo de archivos y directorios:**

El Módulo os en proporciona funcionalidades para interactuar con archivos y directorios mediante cualquier sistema operativo (Windows, Linux o MacOS).

|  |
| --- |
| import os   * + os.path.isfile() -------------->booleana   + os.path.isdir() -------------→booleana   + os.mkdir ("carpeta\_nueva")   + os.rename ("carpeta\_antes","carpeta\_despues")   + os.rmdir ("carpeta\_adios")   + path\_actual = os.getcwd()   + listado=os.listdir(“path\_actual”)   + os.chdir (f"{directorio}\\{archivo}")   (f"{directorio}/{archivo}")   * + os.remove (archivo) |

if os.path.isfile(**"borrar.json"**):

devuelve True si el archivo **"borrar.json"** esta en el mismo directorio en donde esta el archivo python que estamos ejecutando

if os.path.isdir (**"imagenes"**):

devuelve True si el directorio o carpeta **"imagenes"** esta dentro del directorio en donde esta el archivo python que estamos ejecutando

|  |  |
| --- | --- |
| El curso no contempla ver toda la estructura de directorios y alias en nombre de carpetas windows.  Tenga en cuenta que el directorio usuarios no existe en Windows sino que se escriben en ingles | |
| **Directorio reales** | **Alias Windows** |
| C:\Program Files | C:\Archivos de programa |
| C:\Program Files (x86) | C:\Archivos de programa (x86) |
| C:\Users | C:\Usuarios |

os.mkdir ("carpeta\_nueva")

make directory (construir carpeta)

os.rmdir ("carpeta\_adios")

remove directory (remover eliminar carpeta)

En windows

path\_actual='J:\\carpeta\_principal\\sub\_carpeta\\sub\_sub\_carpeta'

el MacOS o Linux

path\_actual='home/sub\_carpeta/sub\_sub\_carpeta'

|  |
| --- |
| La doble barra invertida \\ es debido a los caracteres de escape que se vieron anteriormente, la primera permite que la segunda sea leída.  Tener en cuenta que a partir de las nuevas versiones de Windows empieza a ser compatible la barra invertida \ en reemplazo de la estándar /  El Linux y MacOS no hay disco. |

os.listdir(path\_actual)

·

|  |
| --- |
| Windows  ATTRIB [+R | -R] [+A | -A ] [+S | -S] [+H | -H] [[drive:] [path] filename] [/S [/D]]  + Activa un atributo.  - Desactiva un atributo.  R Atributo de sólo lectura.  A Atributo de archivo.  S Atributo de sistema.  H Atributo de archivo oculto.  /S Procesa todos los archivos en todos los directorios de una ruta específica.  /D Procesa los directorios también. |

·

|  |
| --- |
| Linux  Mencionar los atributos de archivos (ocultos o de sistema) para acceso o visualización 777,755, 655, 644, etc   * 0: Sin permisos * 1: Ejecución * 2: Escritura * 3: Lectura y escritura * 4: Lectura * 5: Lectura y ejecución * 6: Lectura y escritura * 7: Lectura, escritura y ejecución   777  **1er valor: Propietario**  Es la persona que ha creado el archivo o la carpeta  **2do valor: Grupo**  Dentro de este parámetro definiremos el grupo de usuarios que tendrán acceso al archivo o carpeta  **3er valor: Otros**  Dentro de este parámetro están incluidos los usuarios particulares |

·

|  |
| --- |
| MacOS  Read  Read & write   * **Leer y escribir:** Permite a los usuarios abrir la carpeta o archivo y modificarlo. * **Solo lectura:** Permite a los usuarios abrir la carpeta o archivo, pero no modificar su contenido. * **Solo escribir** (Buzón de Entrega): Convierte una carpeta en un buzón de entrega. Los usuarios pueden copiar la carpeta o archivo al buzón de entrega, pero no pueden abrirlo. Solo el propietario del buzón de entrega puede abrirlo. * **Sin acceso:** Bloquea totalmente el acceso a la carpeta o archivo.   **1er valor: Propietario**  Es la persona que ha creado el archivo o la carpeta  **2do valor: staff**  Dentro de este parámetro definiremos el staff de usuarios que tendrán acceso al archivo o carpeta  **3er valor: everyone**  Dentro de este parámetro están incluidos los usuarios particulares |

·

os.rename ("nombre\_antes","nombre\_despues")

rename file or directory (renombrar archivo o carpeta)

path\_actual = os.getcwd()

genera un string con el camino desde el disco a la carpeta actual

|  |
| --- |
| path\_actual = os.getcwd()  print(f"{path\_actual=}") |

listado = os.listdir(path\_actual)

genera una lista de strings los nombres de archivos y carpetas del directorio indicado

|  |
| --- |
| listado = os.listdir(path\_actual)  print(f"{listado=}") |

os.chdir (f"{directorio}\\{sub\_directorio}")

(f"{directorio}/{sub\_directorio}")

La función os.chdir() permite cambiar el directorio en el que se están realizando las operaciones de archivo y directorio.

|  |
| --- |
| import os  nuevo\_directorio = f"{directorio}/{sub\_directorio}\_nuevo"  os.chdir(nuevo\_directorio) |

os.remove (archivo)

Eliminar un archivo específico del sistema de archivos.

archivo\_a\_eliminar = "archivo.txt"

os.remove(archivo\_a\_eliminar)

|  |
| --- |
| import os  eliminar = "borrar.txt"  if os.path.isfile (eliminar):  os.remove(eliminar)  print("El archivo eliminado.")  else:  print("El archivo no existe.") |

·

|  |
| --- |
| Se pueda hacer con un try, except |