

Comenzado el	domingo, 8 de octubre de 2023, 21:20
Estado	Finalizado
Finalizado en	domingo, 8 de octubre de 2023, 21:25
Tiempo empleado	5 minutos 56 segundos
Puntos	10/12
Calificación	8 de 10 (83%)

Pregunta **1**

Correcta

Se puntúa 1 sobre 1

¿Cuál de las siguientes afirmaciones es **FALSA** respecto de la serialización en Python?

Seleccione una:

- ☐ a. El módulo *pickle* es uno de los que brindan funciones para serializar en Python, pero no es el único (existen otros, tales como *json*).
- ☐ b. La *serialización* es un proceso por el cual el contenido de una variable se convierte automáticamente en una secuencia de bytes listos para ser almacenados en un archivo, y luego recuperarse desde el archivo y volver a crear la variable original.
- ☒ c. En Python no se pueden serializar variables de tipo simple (variables de tipo int, float, bool, etc.)
- ☐ d. El método del módulo *pickle* que permite grabar una variable directamente es *pickle.dump()* y el que permite recuperar una variable serializada es *pickle.load()*.

¡Correcto! efectivamente, esta es la afirmación falsa: en Python es perfectamente posible serializar valores de tipo simple y hemos mostrado ejemplos en el proyecto F[21] Archivos.

¡Correcto!

La respuesta correcta es:

En Python no se pueden serializar variables de tipo simple (variables de tipo int, float, bool, etc.)

Pregunta **2**

Correcta

Se puntúa 1 sobre 1

¿Cuál de las siguientes afirmaciones es **CIERTA** respecto de la operación para obtener el tamaño en bytes de un archivo en Python?

Seleccione una:

- ☒ a. La función `os.path.getsize()` toma como parámetro el nombre físico de un archivo, y retorna la longitud en bytes ✓ **¡Correcto!** de ese archivo.
- ☐ b. La función `open()` toma como parámetro el nombre físico de un archivo y el modo de apertura deseado, abre el archivo y retorna el tamaño en bytes del mismo.
- ☐ c. En Python no hay forma de obtener el tamaño en bytes de un archivo.
- ☐ d. El método `tell()` de los objetos manejadores de archivos, no toma parámetro alguno y retorna siempre el tamaño en bytes del archivo para el cual fue invocado.

¡Correcto!

La respuesta correcta es:

La función `os.path.getsize()` toma como parámetro el nombre físico de un archivo, y retorna la longitud en bytes de ese archivo.

Pregunta **3**

Correcta

Se puntúa 1 sobre 1

¿Cuál de las siguientes es claramente falsa respecto de las características y propiedades de un objeto para manejar archivos en Python (un *file object*, tal como lo crea y lo retorna la función `open()`)?

Seleccione una:

- ☒ a. Un *file object* (o un *file-like object*) representa ✓ **¡Correcto! efectivamente, esta afirmación es falsa: un file objet represente sólo archivos de texto (y nunca archivos binarios).** sólo archivos de texto (y nunca archivos binarios).
- ☐ b. Un *file object* (o un *file-like object*) en última instancia permite interpretar el contenido de un archivo como si se tratase de un arreglo de bytes en memoria externa.
- ☐ c. Un *file object* (o un *file-like object*) representa archivos en los que es posible acceder en forma directa a cualquiera de sus bytes mediante el método `seek()`.
- ☐ d. Un *file object* (o un *file-like object*) contiene métodos que permiten tanto grabar como leer datos del archivo representado, independientemente de que eso también puede hacerse con funciones de serialización incluidas en módulos separados (como *pickle* o *json*).

¡Correcto!

La respuesta correcta es:

Un *file object* (o un *file-like object*) representa sólo archivos de texto (y nunca archivos binarios).

Pregunta **4**

Incorrecta

Se puntúa 0 sobre 1

¿Cuál de las siguientes es **FALSA** respecto del *file pointer* en un *file object*?

Seleccione una:

- ☒ a. El *file pointer* de un archivo puede apuntar a algún byte que esté fuera del archivo, a **✗** **Incorrecto...** esta afirmación es cierta, pero se pidió indicar cuál era falsa...
- ☐ b. Cada vez que termina una operación de lectura o grabación en el archivo, el *file pointer* queda apuntando al byte siguiente al último que se leyó o grabó.
- ☐ c. La **única forma** en que puede cambiar el valor del *file pointer* de un archivo es invocando al método *seek()* del *file object* que lo maneja.
- ☐ d. El valor inicial del *file pointer* es asignado por la función *open()* al abrir el archivo.

La respuesta correcta es:

La **única forma** en que puede cambiar el valor del *file pointer* de un archivo es invocando al método *seek()* del *file object* que lo maneja.

Pregunta **5**

Correcta

Se puntúa 1 sobre 1

Si bien sabemos que todo archivo contiene datos representados en binario (y en ese sentido, todo archivo es un archivo binario), el hecho es que los archivos en general se clasifican en *archivos binarios* y *archivos de texto*. ¿Cuál es la diferencia entre ambos?

Seleccione una:

- ☒ a. Todos los archivos son binarios en cuanto a su contenido. La distinción se hace en cuanto a cómo se **interpreta** **✓** **¡Correcto!** ese contenido. En los archivos de texto, todos los bytes se interpretan como caracteres visualizables, salvo los saltos de línea y/o retornos de carro. Pero en los archivos binarios, cada byte puede representar cualquier tipo de dato. La interpretación depende del programa que procese a ese archivo.
- ☐ b. No hay ninguna diferencia. Todos los archivos son binarios, y esa distinción es incorrecta.
- ☐ c. No es cierto que todos los archivos son binarios. Algunos contienen efectivamente texto, otros contienen imágenes, otros contienen números, y en definitiva pueden contener datos del tipo que sea.
- ☐ d. Un archivo de texto sólo (en la plataforma *Windows*) debería contener bytes cuyos valores sean mayores a 31 (caracteres visualizables), y **sólo bytes mayores a 31**. La presencia de cualquier byte con valor menor a 32, lleva a que ese archivo sea considerado binario (por la presencia de bytes que no representan caracteres visualizables).

¡Correcto!

La respuesta correcta es:

Todos los archivos son binarios en cuanto a su contenido. La distinción se hace en cuanto a cómo se **interpreta** ese contenido. En los archivos de texto, todos los bytes se interpretan como caracteres visualizables, salvo los saltos de línea y/o retornos de carro. Pero en los archivos binarios, cada byte puede representar cualquier tipo de dato. La interpretación depende del programa que procese a ese archivo.

Pregunta 6

Correcta

Se puntúa 2 sobre 2

Analice el siguiente script Python, en el cual se están grabando registros de tipo *Libro* en un archivo por serialización:

```
__author__ = 'Catedra de AED'

import pickle

class Libro:
    def __init__(self, cod, tit, aut):
        self.isbn = cod
        self.titulo = tit
        self.autor = aut

def display(libro):
    print('ISBN:', libro.isbn, end='')
    print(' - Título:', libro.titulo, end='')
    print(' - Autor:', libro.autor)

def test():
    print('Prueba de grabación de varios registros...')
    lib1 = Libro(2134, 'Fundación', 'Isaac Asimov')
    lib2 = Libro(5587, 'Fundación e Imperio', 'Isaac Asimov')
    lib3 = Libro(3471, 'Segunda Fundación', 'Isaac Asimov')

    fd = 'libros.dat'
    m = open(fd, 'wb')
    pickle.dump(lib1, m)
    pickle.dump(lib2, m)
    m.close()
    print('Se grabaron varios registros en el archivo', fd)


    m = open(fd, 'rb')
    lib1 = pickle.load(m)
    lib2 = pickle.load(m)
    lib3 = pickle.load(m)
    m.close()

    print('Se recuperaron estos registros desde el archivo', fd, ':')
    display(lib1)
    display(lib2)
    display(lib3)

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

¿Hay algún problema con este script?

Seleccione una:

- ☐ a. Sí. El problema es que no se pueden grabar registros en un archivo si el mismo fue abierto con `open()`. Debió usar `open_register_file()`.
 - ☐ b. Sí. El problema es que el archivo fue abierto en modo wb cuando se pretendió grabar, pero en ese modo no se puede grabar.
 - ☐ c. No hay nada de malo en este segmento.
 - ☒ d. Sí. Se grabaron sólo dos registros en el archivo, pero luego se intentaron tres lecturas. La tercera fallará por  ¡Correcto!
- encontrar el final del archivo en forma inesperada.

¡Correcto!

La respuesta correcta es:

Sí. Se grabaron sólo dos registros en el archivo, pero luego se intentaron tres lecturas. La tercera fallará por encontrar el final del archivo en forma inesperada.

Pregunta **7**

Incorrecta

Se puntúa 0 sobre 1

¿Qué ocurre si se intenta leer en un archivo y el *file pointer* del mismo está apuntando en ese momento al final del archivo (o sea, al primer byte ubicado fuera del archivo)?

Seleccione una:

- ☒ a. No ocurre nada. La lectura no se realiza, pero tampoco se lanza un error. El programa simplemente ignora la orden y continúa. ✘ Incorrecto... el programa no sólo no podrá hacer la lectura, sino que también se interrumpirá...
- ☐ b. Se leen los bytes que siguen al final del archivo, bajo responsabilidad del programador, y el programa continúa normalmente.
- ☐ c. Se lanza un error de intérprete de la forma *EOFError*, y el programa se interrumpe.
- ☐ d. La pregunta no tiene sentido: el *file pointer* **no puede** estar apuntando al final del archivo.

La respuesta correcta es:

Se lanza un error de intérprete de la forma *EOFError*, y el programa se interrumpe.

Pregunta **8**

Correcta

Se puntúa 1 sobre 1

¿Qué ocurre si en un programa Python se usa el método *seek()* de un *file object* y se salta con ese método a un byte que está más allá del final del archivo?

Seleccione una:

- ☐ a. Se interrumpe el programa lanzando un error de la forma *EOFError*.
- ☐ b. El método *seek()* no puede saltar a un byte que esté fuera del archivo, por lo tanto, no hará nada.
- ☒ c. El *file pointer* del archivo quedará posicionado en ese byte, aunque esté fuera del archivo. ✔ ¡Correcto!
- ☐ d. El *file pointer* quedará posicionado en ese byte, y el tamaño del archivo se ajustará a ese byte.

¡Correcto!

La respuesta correcta es:

El *file pointer* del archivo quedará posicionado en ese byte, aunque esté fuera del archivo.

Pregunta **9**

Correcta

Se puntúa 1 sobre 1

Suponga que en un programa en Python necesita abrir un archivo de tal forma que:

- El archivo se maneje como archivo binario (y no como archivo de texto).
- Si el archivo no existía NO sea creado.
- Se permita tanto leer como grabar su contenido.
- No se destruya el contenido si el archivo ya existía.
- Se pueda leer en cualquier posición del archivo y también grabar en cualquier posición del mismo.

¿Cuál de los posibles modos de apertura disponibles para `open()` tendría que emplear al abrir ese archivo?

Seleccione una:

- ☒ a. El modo r+b. ✓ ¡Correcto!
- ☐ b. El modo r+t.
- ☐ c. El modo rb.
- ☐ d. El modo a+b.

¡Correcto!

La respuesta correcta es: El modo r+b.

Pregunta **10**

Correcta

Se puntúa 2 sobre 2

¿Qué valor tendrá el *file pointer* del archivo representado por *m* luego de terminar de realizar las operaciones que se indican? (ignore cualquier situación de error que podría producirse al abrir el archivo o al hacer las grabaciones: sólo analice las líneas dadas suponiendo un contexto correcto):

```
import pickle
import os.path

a = 25
b = 2.876
c = 'Hola mundo'

m = open('prueba.dat', 'wb')
pickle.dump(a, m) # suponer que aquí se grabaron 5 bytes...
pickle.dump(b, m) # suponer que aquí se grabaron 12 bytes...
pickle.dump(c, m) # suponer que aquí se grabaron 20 bytes..
m.close()
```

Seleccione una:

- ☐ a. 20
- ☒ b. 37 ✓ ¡Correcto!
- ☐ c. 32
- ☐ d. 5

¡Correcto!

La respuesta correcta es: 37

