













Comenzado el	domingo, 8 de octubre de 2023, 21:33
Estado	Finalizado
Finalizado en	domingo, 8 de octubre de 2023, 21:45
Tiempo empleado	12 minutos 6 segundos
Puntos	12/12
Calificación	10 de 10 (96%)

Pregunta **1**

Correcta

Se puntúa 1 sobre 1

Para cada uno de los métodos propios de un *file object* nombrados en la columna de la izquierda, seleccione en la columna de la derecha la descripción de su funcionamiento.

flush()	Graba en el archivo los buffers de grabación, sin cerrarlo. 	
writable()	Retorna True si el archivo está disponible para ser grabado. 	
closed	Es un atributo o campo que vale True si el archivo está cerrado. 	
truncate(t)	Trunca (o expande) el contenido del archivo a una cantidad igual a t bytes. 	
writelines(lines)	Graba el contenido de la lista "lines" en el archivo. 	
readable()	Retorna True si el archivo está disponible para ser leído. 	

¡Correcto!

La respuesta correcta es: flush() → Graba en el archivo los buffers de grabación, sin cerrarlo., writable() → Retorna True si el archivo está disponible para ser grabado., closed → Es un atributo o campo que vale True si el archivo está cerrado., truncate(t) → Trunca (o expande) el contenido del archivo a una cantidad igual a t bytes., writelines(lines) → Graba el contenido de la lista "lines" en el archivo., readable() → Retorna True si el archivo está disponible para ser leído.


Pregunta **2**

Correcta

Se puntúa 1 sobre 1

¿Cuántos caracteres diferentes pueden representarse usando el estándar **ASCII 7**, y cuántos pueden representarse usando **ASCII 8**?

Seleccione una:

- ☐ a. 4294967296 con **ASCII 7** y 8589934592 con **ASCII 8**.
- ☐ b. 256 con **ASCII 7** y 128 con **ASCII 8**.
- ☐ c. 32768 con **ASCII 7** y 65536 con **ASCII 8**.
- ☒ d. 128 con **ASCII 7** y 256 con **ASCII 8**.  ¡Correcto!

¡Correcto!

La respuesta correcta es:
128 con **ASCII 7** y 256 con **ASCII 8**.

Pregunta **3**

Correcta

Se puntúa 1 sobre 1

¿Cuáles de las siguientes son características propias del método `read()`, contenido en cualquier variable *file object* usada para manipular un archivo de texto en Python? (Más de una respuesta puede ser válida, por lo que marque todas las que considere correctas).

Seleccione una o más de una:

- ☐ a. Si *m* es el *file object* que representa un archivo de texto, la invocación `cad = m.read()` lee el contenido *completo* del archivo y lo retorna como una cadena de caracteres, asignando esa cadena en este caso en la variable *cad*. Los caracteres de salto de línea que pudiese contener el archivo, no se preservan: la cadena devuelta no contendrá saltos de línea de ningún tipo, en ninguna posición.
- ☒ b. Si *m* es el *file object* que representa un archivo de texto y *n* es un número entero mayor a 0, la invocación `cad = m.read(n)` lee una cantidad *n* de bytes del archivo los retorna como una cadena de caracteres, asignando esa cadena en este caso en la variable *cad*. Si el archivo no llega a tener *n* bytes, se retorna la cadena vacía (`""`). ¡Correcto!
- ☒ c. Si *m* es el *file object* que representa un archivo de texto, la invocación `cad = m.read()` lee el contenido *completo* del archivo y lo retorna como una cadena de caracteres, asignando esa cadena en este caso en la variable *cad*. ¡Correcto!
- ☒ d. Si *m* es el *file object* que representa un archivo de texto, la invocación `cad = m.read()` lee el contenido *completo* del archivo y lo retorna como una cadena de caracteres, asignando esa cadena en este caso en la variable *cad*. Los caracteres de salto de línea que pudiese contener el archivo, se preservan y se retornan con la cadena devuelta. ¡Correcto!

¡Correcto!

Las respuestas correctas son:

Si *m* es el *file object* que representa un archivo de texto, la invocación `cad = m.read()` lee el contenido *completo* del archivo y lo retorna como una cadena de caracteres, asignando esa cadena en este caso en la variable *cad*.

Si *m* es el *file object* que representa un archivo de texto y *n* es un número entero mayor a 0, la invocación `cad = m.read(n)` lee una cantidad *n* de bytes del archivo los retorna como una cadena de caracteres, asignando esa cadena en este caso en la variable *cad*. Si el archivo no llega a tener *n* bytes, se retorna la cadena vacía (`""`).

Si *m* es el *file object* que representa un archivo de texto, la invocación `cad = m.read()` lee el contenido *completo* del archivo y lo retorna como una cadena de caracteres, asignando esa cadena en este caso en la variable *cad*. Los caracteres de salto de línea que pudiese contener el archivo, se preservan y se retornan con la cadena devuelta.

Pregunta 4

Parcialmente correcta

Se puntúa 1 sobre 1

¿Cuáles de las siguientes son características propias del método `readline()`, contenido en cualquier variable *file object* usada para manipular un archivo de texto en Python? (Más de una respuesta puede ser válida, por lo que marque todas las que considere correctas).

Seleccione una o más de una:

- ☒ a. Si *m* es el *file object* que representa un archivo de texto, la invocación `cad = m.readline()` se detendrá en el primer carácter de espacio en blanco que `readline()` encuentre, o bien cuando encuentre el final del archivo. La cadena devuelta será asignada en este caso en la variable *cad*. ✗ Incorrecto... la detención no se producirá en un espacio en blanco...
- ☐ b. Si *m* es el *file object* que representa un archivo de texto, y la invocación `cad = m.readline()` no puede leer ninguna cadena (por la razón que sea...) se lanzará un error de runtime y el programa se interrumpirá.
- ☐ c. Si *m* es el *file object* que representa un archivo de texto, la invocación `cad = m.readline()` lee una sola línea de texto del archivo y la retorna como una cadena de caracteres, asignando esa cadena en este caso en la variable *cad*.
- ☒ d. Si *m* es el *file object* que representa un archivo de texto, la invocación `cad = m.readline()` se detendrá en el primer carácter de salto de línea que `readline()` encuentre, o bien cuando encuentre el final del archivo. La cadena devuelta será asignada en este caso en la variable *cad*. ✓ ¡Correcto!

Ha seleccionado correctamente 1.

Las respuestas correctas son:

Si *m* es el *file object* que representa un archivo de texto, la invocación `cad = m.readline()` lee una sola línea de texto del archivo y la retorna como una cadena de caracteres, asignando esa cadena en este caso en la variable *cad*.

Si *m* es el *file object* que representa un archivo de texto, la invocación `cad = m.readline()` se detendrá en el primer carácter de salto de línea que `readline()` encuentre, o bien cuando encuentre el final del archivo. La cadena devuelta será asignada en este caso en la variable *cad*.

Pregunta 5

Correcta

Se puntúa 1 sobre 1

¿Cuáles de las siguientes son características propias del método `readlines()` (con una `s` al final) contenido en cualquier variable `file object` usada para manipular un archivo de texto en Python? (Más de una respuesta puede ser válida, por lo que marque todas las que considere correctas).

Seleccione una o más de una:

- ☒ a. Si `m` es el `file object` que representa un archivo de texto, la invocación `v = m.readlines()` lee el contenido completo del archivo y lo retorna como una lista o arreglo de cadenas de caracteres, que contiene un elemento por cada línea de texto que el archivo tenía, asignando ese arreglo en este caso en la variable `v`. Cada una de las cadenas almacenadas en el arreglo `v` conservará el carácter de salto de línea que hubiese tenido en el archivo. ¡Correcto!
- ☒ b. Si `m` es el `file object` que representa un archivo de texto y `n` es un número entero mayor a 0, la invocación `v = m.readlines(n)` lee hasta `n bytes` del archivo y los retorna como una lista o arreglo de cadenas de caracteres, que contiene un elemento por cada línea de texto que haya logrado leer hasta completar `n bytes`, asignando ese arreglo en este caso en la variable `v`. ¡Correcto!
- ☒ c. Si `m` es el `file object` que representa un archivo de texto, la invocación `v = m.readlines()` lee el contenido completo del archivo y lo retorna como una lista o arreglo de cadenas de caracteres, que contiene un elemento por cada línea de texto que el archivo tenía, asignando ese arreglo en este caso en la variable `v`. ¡Correcto!
- ☐ d. Si `m` es el `file object` que representa un archivo de texto, la invocación `v = m.readlines()` lee el contenido completo del archivo y lo retorna como una lista o arreglo de cadenas de caracteres, que contiene un elemento por cada línea de texto que el archivo tenía, asignando ese arreglo en este caso en la variable `v`. Ninguna de las cadenas almacenadas en el arreglo `v` conservará el carácter de salto de línea que hubiese tenido en el archivo.

¡Correcto!

Las respuestas correctas son:

Si `m` es el `file object` que representa un archivo de texto, la invocación `v = m.readlines()` lee el contenido completo del archivo y lo retorna como una lista o arreglo de cadenas de caracteres, que contiene un elemento por cada línea de texto que el archivo tenía, asignando ese arreglo en este caso en la variable `v`.

Si `m` es el `file object` que representa un archivo de texto, la invocación `v = m.readlines()` lee el contenido completo del archivo y lo retorna como una lista o arreglo de cadenas de caracteres, que contiene un elemento por cada línea de texto que el archivo tenía, asignando ese arreglo en este caso en la variable `v`. Cada una de las cadenas almacenadas en el arreglo `v` conservará el carácter de salto de línea que hubiese tenido en el archivo.

Si `m` es el `file object` que representa un archivo de texto y `n` es un número entero mayor a 0, la invocación `v = m.readlines(n)` lee hasta `n bytes` del archivo y los retorna como una lista o arreglo de cadenas de caracteres, que contiene un elemento por cada línea de texto que haya logrado leer hasta completar `n bytes`, asignando ese arreglo en este caso en la variable `v`.

Pregunta **6**

Correcta

Se puntúa 1 sobre 1

¿Cuáles de las siguientes son características propias del método `write()`, contenido en cualquier variable *file object* usada para manipular un archivo de texto en Python? (Más de una respuesta puede ser válida, por lo que marque todas las que considere correctas).

Seleccione una o más de una:

- ☒ a. Si *m* es el *file object* que representa un archivo de texto y *cad* es una cadena de caracteres, la invocación `m.write(cad)` graba la cadena contenida en *cad* y retorna la cantidad de caracteres que efectivamente grabó. ¡Correcto!
- ☒ b. Si *m* es el *file object* que representa un archivo de texto y *cad* es una cadena de caracteres, la invocación `m.write(cad)` graba la cadena contenida en *cad* en forma directa y simple en un archivo de texto. ¡Correcto!
- ☒ c. Si *m* es el *file object* que representa un archivo de texto y *cad* es una cadena de caracteres, la invocación `m.write(cad)` graba la cadena contenida en *cad* pero *no agrega* un caracter de salto de línea al final, a menos que *cad* ya lo contenga previamente. ¡Correcto!
- ☐ d. Si *m* es el *file object* que representa un archivo de texto y *cad* es una cadena de caracteres, la invocación `m.write(cad)` graba la cadena contenida en *cad* sin importar en qué modo haya sido abierto el archivo (sólo importa que se haya indicado que es de texto usando una 't' en el modo de apertura, y no una 'b').

¡Correcto!

Las respuestas correctas son:

Si *m* es el *file object* que representa un archivo de texto y *cad* es una cadena de caracteres, la invocación `m.write(cad)` graba la cadena contenida en *cad* en forma directa y simple en un archivo de texto.,

Si *m* es el *file object* que representa un archivo de texto y *cad* es una cadena de caracteres, la invocación `m.write(cad)` graba la cadena contenida en *cad* y retorna la cantidad de caracteres que efectivamente grabó.,

Si *m* es el *file object* que representa un archivo de texto y *cad* es una cadena de caracteres, la invocación `m.write(cad)` graba la cadena contenida en *cad* pero *no agrega* un caracter de salto de línea al final, a menos que *cad* ya lo contenga previamente.

Pregunta **7**

Correcta

Se puntúa 1 sobre 1

¿Cuál de las siguientes **no es** una forma de codificación de caracteres para el estándar *Unicode*?

Seleccione una:

- ☐ a. UTF-16
- ☐ b. UTF-8
- ☒ c. EBCDIC ¡Correcto! La codificación EBCDIC (por **E**xtended **B**inary **C**oded **D**ecimal **I**nterchange **C**ode) es un sistema de codificación de caracteres empleado por IBM en sus computadoras mainframe, diferente de ASCII y otros estándares que se impusieron en el mercado, pero no es una forma de codificación de caracteres para Unicode.
- ☐ d. UTF-32

¡Correcto!

La respuesta correcta es:

EBCDIC

Pregunta 8

Correcta

Se puntúa 2 sobre 2

¿Cuáles de las siguientes **son ciertas** en relación al procesamiento de archivos de texto que contienen números, representados como cadenas de caracteres a razón de un número/cadena por cada línea del archivo? (Más de una respuesta puede ser válida, por lo que marque todas las que considere correctas).

Seleccione una o más de una:

- ☐ a. En realidad, no hay ningún motivo válido que justifique el uso de archivos con este criterio. Se deben almacenar directamente los números, en su formato numérico original, sin convertirlos a cadenas.
- ☒ b. Normalmente, cada línea de un archivo de este tipo contiene un "número" representado como una cadena de caracteres (sólo caracteres que representan dígitos) y cada número se separa del siguiente con un salto de línea u otro caracter clásico como un espacio en blanco, un tabulador, una coma, etc. ✓ ¡Correcto!
- ☒ c. Cuando se necesita procesar un archivo así, se leen desde el archivo las cadenas que representan números, se las convierte a números en el formato correcto (*int* o *float*), se almacenan esos números en alguna estructura de datos (como un arreglo), y finalmente se retorna esa estructura. ✓ ¡Correcto!
- ☒ d. El motivo por el que se suelen almacenar números como cadenas en un archivo de texto, es el de evitar problemas de incompatibilidad de formatos numéricos, si se prevé que distintos programadores usen lenguajes diferentes para procesar el mismo archivo. ✓ ¡Correcto!

¡Correcto!

Las respuestas correctas son:

Normalmente, cada línea de un archivo de este tipo contiene un "número" representado como una cadena de caracteres (sólo caracteres que representan dígitos) y cada número se separa del siguiente con un salto de línea u otro caracter clásico como un espacio en blanco, un tabulador, una coma, etc.,

Cuando se necesita procesar un archivo así, se leen desde el archivo las cadenas que representan números, se las convierte a números en el formato correcto (*int* o *float*), se almacenan esos números en alguna estructura de datos (como un arreglo), y finalmente se retorna esa estructura.,

El motivo por el que se suelen almacenar números como cadenas en un archivo de texto, es el de evitar problemas de incompatibilidad de formatos numéricos, si se prevé que distintos programadores usen lenguajes diferentes para procesar el mismo archivo.


Pregunta **9**

Correcta

Se puntúa 1 sobre 1

¿Cuál de los siguientes es el inconveniente principal que supone procesar el *mismo archivo de texto* en *diferentes plataformas* (o sistemas operativos)?

Seleccione una:

- ☒ a. El tratamiento dado al *salto de línea*: en algunas plataformas se representa con un caracter de control simple (`\n`) y  ¡Correcto! en otras con una pareja (`\r\n`). En general, los lenguajes de programación modernos (como Python) resuelven automáticamente este inconveniente.
- ☐ b. No hay ningún inconveniente especial. El mismo archivo de texto puede abrirse y manejarse sin problema alguno en cualquier plataforma, sin tener que tomar precauciones particulares.
- ☐ c. La forma en que se trata y reconoce el final del archivo. Cada plataforma hace esto de forma diferente en un archivo de texto, y con cada lenguaje de programación debe considerarse este problema en forma particular.
- ☐ d. La forma en que se distinguen las mayúsculas de las minúsculas dentro del archivo.

¡Correcto!

La respuesta correcta es:

El tratamiento dado al *salto de línea*: en algunas plataformas se representa con un caracter de control simple (`\n`) y en otras con una pareja (`\r\n`). En general, los lenguajes de programación modernos (como Python) resuelven automáticamente este inconveniente.


Pregunta **10**

Correcta

Se puntúa 2 sobre 2

¿Cuál de las siguientes **es cierta** respecto del uso y aplicación del método `seek()` en archivos de texto con Python?

Seleccione una:

- ☒ a. Si el archivo es de texto, en Python el método `seek()` sólo puede usarse para cambiar el valor del *file pointer*  ¡Correcto! suponiendo que el reposicionamiento se hace desde el inicio del archivo (segundo parámetro igual a `io.SEEK_SET = 0`). Se puede usar `io.SEEK_END` o `io.SEEK_CUR`, pero entonces el primer parámetro debe valer obligatoriamente 0 (cero).
- ☐ b. Si el archivo es de texto, en Python el método `seek()` puede usarse para cambiar el valor del *file pointer* suponiendo que el reposicionamiento se hace desde el inicio del archivo (segundo parámetro igual a `io.SEEK_SET = 0`) o desde la posición actual del *file pointer* (segundo parámetro igual a `io.SEEK_CUR = 1`), sin restricciones en ambos casos.
- ☐ c. Si el archivo es de texto, en Python el método `seek()` puede usarse para cambiar el valor del *file pointer* suponiendo que el reposicionamiento se hace desde el inicio del archivo (segundo parámetro igual a `io.SEEK_SET = 0`) o desde el final del archivo (segundo parámetro igual a `io.SEEK_SET = 2`), sin restricciones en ambos casos.
- ☐ d. Tanto da que el archivo sea de texto o binario, en Python el método `seek()` se aplica exactamente en la misma forma, sin restricciones.

¡Correcto!

La respuesta correcta es:

Si el archivo es de texto, en Python el método `seek()` sólo puede usarse para cambiar el valor del *file pointer* suponiendo que el reposicionamiento se hace desde el inicio del archivo (segundo parámetro igual a `io.SEEK_SET = 0`). Se puede usar `io.SEEK_END` o `io.SEEK_CUR`, pero entonces el primer parámetro debe valer obligatoriamente 0 (cero).

[◀ Materiales Adicionales para la Ficha 24](#)

Ir a...



