República Bolivariana de Venezuela

Ministerio del Poder Popular para la Educación Universitaria

UPT Trujillo "Mario Briceño Iragorry"

Núcleo "Barbarita de la Torre"

Trujillo Edo. Trujillo

MANEJO DE ARCHIVOS

EN PYTHON

Realizado por:

Diagni Linares, C.I. 31.735.114

María Cardoza, C.I. 31.368.842

Wiliander Bastidas, C.I.: 29.638.270

PNF: Informática, trayecto I, trimestre III

Unidad Curricular: Programación

Profesor: Carlos Bravo

Trujillo, Noviembre de 2024.

Introducción

En la era tecnológica actual, estamos inmersos en un entorno digital que se vuelve cada vez toma más protagonismo, donde la programación se ha convertido en una habilidad esencial y altamente valorada; el avance de dispositivos conectados, el análisis de datos y el desarrollo de aplicaciones han generado una demanda sin precedentes de soluciones innovadoras. En este contexto, existen numerosos lenguajes de programación, cada uno diseñado con características y propósitos específicos que permiten a los desarrolladores abordar una amplia variedad de desafíos.

Python es un lenguaje de programación versátil y poderoso que ofrece diversas estructuras de datos y funcionalidades para manejar información de manera eficiente. Entre estas estructuras, las pilas y las colas son fundamentales para implementar algoritmos y gestionar datos temporalmente, mientras que el manejo de archivos permite la persistencia de datos en el sistema

Manejo de archivos en Python

El manejo de archivos es un aspecto significativo en informática y la programación, ya que permite que los sistemas operativos y aplicaciones interactúen con los archivos almacenados. En el caso del manejo de archivos en Python, este ofrece varias funciones y métodos que facilitan la interacción con archivos de diferentes tipos, como archivos de texto y binarios; los de texto contienen caracteres legibles (como .txt, .csv) y se utilizan comúnmente para almacenar datos en un formato sencillo, mientras que los binarios poseen datos en un formato que no es legible por humanos (como imágenes, audio, etc.), y se utilizan para almacenar información en un formato más compacto.

En Python, cada línea de código incluye una secuencia de caracteres que juntos forman un archivo de texto, cada una de las líneas de esos archivos termina con un carácter especial denominado EOL (End of line) o caracteres de final de línea o nueva línea; con ello, sabemos dónde finaliza y donde comienza cada parte durante la lectura y la escritura de archivos en Python.

NOTA:

- **`'\n'`**: Nueva línea (LF - Line Feed). Es el carácter utilizado en sistemas Unix y Linux

Para trabajar con archivos, primero necesitas abrirlos. Usamos la función **`open()`,** esta función toma al menos dos argumentos: el nombre del archivo y el modo de apertura.

Modos de Apertura

- **'r':** Lectura (por defecto). Abre un archivo para leerlo., si el archivo no existe, se genera un error.
- **'w':** Escritura. Crea un nuevo archivo o trunca uno existente, si el archivo ya existe, se sobrescribirá.

- **'a':** Adición. Abre un archivo para agregar contenido al final sin truncar el archivo existente.
- **'b':** Modo binario. Se puede usar junto con otros modos (por ejemplo, 'rb' o 'wb') para manejar archivos binarios.
- **'x':** Creación. Crea un nuevo archivo, pero falla si el archivo ya existe.
- **'t':** Modo texto (por defecto). Se puede usar junto con otros modos (por ejemplo, 'rt', 'wt').

Para leer archivos

Una vez que el archivo está abierto en modo lectura, puedes usar varios métodos para leer su contenido:

`read()`: Lee todo el contenido del archivo.

`readline()`: Lee una línea del archivo a la vez.

`readlines()`: Lee todas las líneas y devuelve una lista donde cada elemento es una línea. Incluye los caracteres de nueva línea al final de cada línea.

`read(size)`: Este método lee el contenido del archivo y devuelve hasta size caracteres. Si no se especifica size, se leerá todo el archivo.

`read(n)`: Puedes utilizarla dentro de un bucle while para leer un archivo en bloques de tamaño específico, lo que puede ser útil para archivos grandes.

Para cerrar archivos

`close()`

Es importante cerrar los archivos después de usarlos para liberar recursos, puedes hacerlo con este método.

Pilas de Python

Las pilas son estructuras de datos fundamentales en programación y se utilizan en una variedad de aplicaciones, desde la gestión de la memoria hasta la implementación de algoritmos. En Python, las pilas se pueden implementar de varias maneras, siendo la más común el uso de listas o el módulo ´collections´.

Una pila es una estructura de datos que sigue el principio Last In**,** First Out (LIFO), lo que significa que el último elemento en entrar es el primero en salir. Las operaciones más comunes que se pueden realizar con una pila son:

- **Push**: Logra añadir un elemento a la parte superior de la pila.
- **Pop**: Elimina y devuelve el elemento de la parte superior de la pila.
- **Peek** (o Top): Devuelve el elemento en la parte superior sin eliminarlo.
- **isEmpty**: Comprueba si la pila está vacía.
- **Size**: Obtiene el número de elementos en la pila.

Colas en Python

Las colas son estructuras de datos fundamentales en programación que siguen el principio First In, First Out (FIFO). Esto significa que el primer elemento en entrar es el primero en salir, similar a una cola en un banco o en una fila de espera. Generalmente soportan las siguientes operaciones:

1. **Enqueue**: Para añadir un elemento al final de la cola.
2. **Dequeue**: Logra eliminar y devolver el elemento del frente de la cola.
3. **Peek**: Obtiene el elemento en el frente de la cola sin eliminarlo.
4. **isEmpty**: Comprueba si la cola está vacía.
5. **Size**: Obtiene el número de elementos en la cola.

En Python, podemos implementar una cola utilizando listas, aunque para un mejor rendimiento en operaciones de ´dequeue´, es recomendable utilizar el módulo ´collections.deque´, que ofrece una implementación más eficiente para colas.

Conclusión

El manejo eficiente de datos es una habilidad fundamental para cualquier desarrollador, Python, como uno de los lenguajes de programación más populares, ofrece herramientas accesibles para gestionar datos de manera efectiva. El uso de estructuras de datos como pilas y colas, junto con la capacidad de manejar archivos, permite a los programadores abordar una amplia gama de desafíos de manera eficiente. En general, la manipulación de datos en Python no solo facilita la creación de aplicaciones más dinámicas y útiles, sino que también promueve un enfoque ordenado y lógico para el desarrollo de software.

La habilidad para trabajar con diferentes tipos de datos y estructuras es esencial en un mundo donde la información se genera y consume a un ritmo vertiginoso. Además, el manejo de archivos proporciona a los desarrolladores la capacidad de almacenar y recuperar información de forma persistente, lo que es crucial para aplicaciones que requieren un seguimiento de datos a lo largo del tiempo; a raíz de esto, las pilas, con su estructura LIFO, son ideales para gestionar tareas que requieren un control de estado, como en algoritmos recursivos o funciones de deshacer. Por su parte, las colas, que operan bajo el principio FIFO, son perfectas para manejar procesos donde el orden de llegada es esencial, como en sistemas de gestión de tareas y programación de eventos.

Referencias Bibliográficas:

Esteban Canle Fernández, (2022): “Manejo de archivos en Python: ¡conviértete en experto!”.

Logan Young Ring, (2024): “Python Cola: Ejemplo FIFO, LIFO”.

Héctor Costa Guzmán, (2018): “Python, colecciones de datos, pilas”.

Héctor Costa Guzmán, (2018): “Python, colecciones de datos, colas”.

Ector Crespo, (2017) “De vuelta a lo básico, como manejar colas en Python”.

Jhontona, (2024): “EXPLORANDO LAS ESTRUCTURAS DE DATOS EN PYTHON: Pilas y Colas”.