Tarea 1

1. Cuál es la diferencia entre chop y chomp

R// el metodo chop elimina el ultimo carácter de una cadena de texto sin importar cual sea el carácter que se este usando al final de la cadena el metodo chop lo eliminara, mientras que el metodo chomp solo elimina el ultimo carácter solo si este es un salto de línea o un salto de carro de resto mientras no sean estos dos caracteres el metodo chomp no los eliminara a la hora de capturar el valor en una variable, en conclusión el metodo chop es más tiene utilidades dentro del lenguaje que el metodo chomp todo esto radica a los limitantes de estos, mientras que el metodo chop puede eliminar el ultimo carácter sin importar de que tipo sea el metodo chomp esta sujeto a una limitación la cual dice que a menos que sea salto de línea o de carro no se podrá eliminar este carácter por ende es un metodo que se usa solo para dos casos específicos mientas que el metodo chop es mas universal y tiene mas utilidades a la hora de la eliminación de caracteres

Reverse: es un metodo que se usa para invertir el orden de una cadena de texto, por ejemplo se usa para cambiar el sentido de una cadena haciendo que esta empiece desde el final del texto ingresado hasta el principio del texto ingresado, haciendo que este cambie su sentido

Rectángulo

Descripción generada automáticamente con confianza media

Capitalize: es un metodo que se usa para convertir la primera letra de una cadena de texto en mayúscula y hacer que las demás letras que componen la cadena se queden en minúscula, es un metodo que se usa principalmente para convertir la primera letra en mayúscula de un texto pero que no afecta al texto anterior sirviendo solo este metodo para mostrar información, en caso tal que la primera letra del texto ya se encuentre con una letra mayúscula la cadena se imprimirá de manera normal como fue definida

Forma, Rectángulo

Descripción generada automáticamente

Length: es un metodo que se usa para contar la cantidad de caracteres que tiene una cadena de texto, es un metodo que su función principal es devolver el numero de caracteres de la cadena a la cual se le indique contando no solo los espacios que cuenten con un letra o carácter si no también contando los espacios en blanco que tiene la cadena para asi entregar el resultado del tamaño de la cadena y de su longitud

Imagen que contiene Aplicación

Descripción generada automáticamente

Tarea 2

1. ¿Para qué casos crees que las matrices son más adecuadas que los hashes?

R//

* Ordenación de elementos: Los arrays mantienen el orden de los elementos según se insertan lo que puede ser útil si necesitas accede a los elementos en orden especifico. Por otro lado los hashes no garantizan ningún orden en particular, ya que los elementos se almacenan en función de sus claves. Si el orden de los elementos es importante para la operación que estes realizando es mejor usar un array
* Acceso por índice: Si necesitas acceder a los elementos de una colección de utilizando índices numéricos, las matrices son mas convenientes. Puedes accede directamente a un elemento en una matriz utilizando su posición mediante el índice. En cambio los hashes se acceden a través de claves únicas en lugar de índices, lo que significa que no puedes acceder a los elementos manera secuencial
* Operaciones basadas en índices: Si tienes que realizar operaciones como recorrer elementos en un orden especifico, eliminar o modificar elementos en función de su posición, los arrays son más apropiados. Por ejemplo si deseas eliminar el segundo elemento de una colección, puedes hacerlo fácilmente manipulando el índice correspondiente en el array. Con un hash tendrías que buscar la clave asociada al elemento y luego eliminarlo
* Datos homogéneos: Si estas trabajando con datos homogéneos, es decir, elementos del mismo tipo, las matrices son mas adecuadas. Las matrices en Ruby estan diseñadas para almacenar elementos en secuencia y acceder a ellos eficientemente, lo que es beneficioso cuando todos los elementos son del mismo tipo y tamaño. Los hashes por otro lado son mas adecuados para almacenar asociaciones clave-valor con datos heterogéneos

1. ¿Para qué casos crees que los hashes son más adecuados que las matrices?

R//

* Busqueda eficiente por la clave: Los hashes son ideales cuando necesitas acceder a los fatos utilizando una clave única en lugar de un índice numérico. Dado que los hashes estan implementados como tablas de busqueda, ofrecen una busqueda eficiente con una complejidad de tiempo constante 0(1) en promedio. Esto los hace especialmente útiles cuando necesitas buscar o recuperar rápidamente un valor en función de una clave sin tener que recorrer todos los elementos de la estructura
* Asociación Clave-Valor: Los hashes son perfectos para almacenar y manipular datos que requieren una relación clave-valor. Puedes asignar valores a claves especificas y luego acceder a esos valores mediante claves correspondientes. Esto es útil cuando necesitas almacenar informacion estructurada o cuando deseas acceder a datos en función de una identificación única
* Estructura de datos no homogéneas: Si estas trabajando con datos heterogéneos, es decir, elementos de diferentes tipos o tamaños, los hashes son más flexibles que las matrices. Los hashes no imponen restricciones de tipos de datos que puedes almacenar en ellos. Puedes tener claves y valores de diferentes tipos, lo que permite estructural y organizar datos de manera más compleja y versátil
* Flexibilidad en las claves: Los hashes permiten utilizar cualquier objeto como clave, siempre que sea inmutable. Esto significa que no estas limitado a utilizar índices numéricos o secuencias como en las matrices. Puede usar cualquier objeto de Ruby como clave, como cadenas, símbolos u objetos personalizados. Esta flexibilidad en las claves puede ser valiosa cuando necesitas organizar y acceder a datos de manera especifica

1. ¿Podemos tener hashes como elementos de arreglos? ¿Podemos tener arreglos como elementos hash? ¿Puede una matriz o un hash ser la clave de cualquier elemento hash?

R// Si en Ruby es posible tener hashes como elementos de arreglos y arreglos como elementos de hashes además tanto las matrices como los hashes pueden ser utilizadas como claves de otros hashes

Ejemplos

Hashes como elementos de arreglos: puedes tener hashes como elementos individuales dentro de un arreglo

Texto

Descripción generada automáticamente con confianza media

Arreglos como elementos de hashes: es posible tener arreglos como valores dentro de un hash

Interfaz de usuario gráfica

Descripción generada automáticamente

Arreglos como claves de hashes: es posible tener arreglos como la clave un hash pero es importante destacar que para que esto funcione la matriz debe ser inmutable, esto significa que su tamaño no debe cambiar y que una matriz mutable no puede ser utilizada como clave de un hash

Rectángulo

Descripción generada automáticamente con confianza media