****

Estructuras de datos

**AUTORES**

**WILSON FABIAN CRUZ ANDRADE**

**CENTRO DE GESTIÓN DE MERCADOS, LOGÍSTICA Y TECNOLOGÍAS DE LA INFORMACIÓN**

**PROGRAMACIÓN DE SOFTWARE**

**Ficha: 2502640**

**Bogotá D.C**

**2022**

# **Estructura de datos Python**

## **Cadenas:**

Una cadena es una de las estructuras de datos más simples estructura de datos que es definida como una secuencia de caracteres de cero o más caracteres este se puede identificar con comillas dobles”” o simples ‘’ todo lo que este dentro de estas comillas se identifica como una cadena, este tipo de estructura es inmutable lo que significa es que no se puede cambiar.

Cuando se requiere usar caracteres especiales de control se puede usar una secuencia de escape

\n: Nueva línea

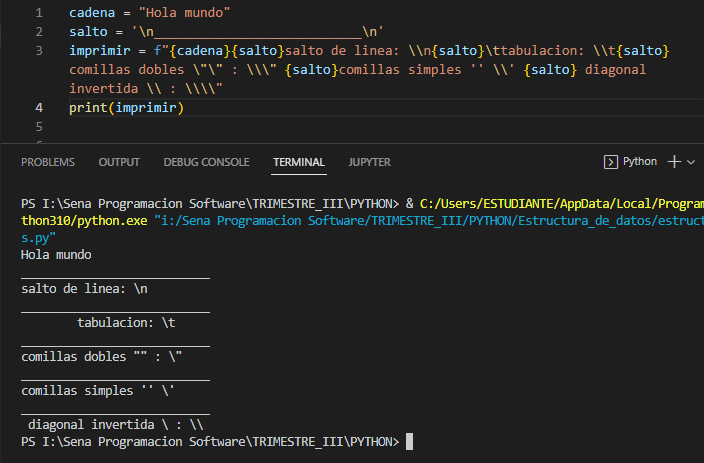
\t: Una tabulacion

\”: Una comilla doble

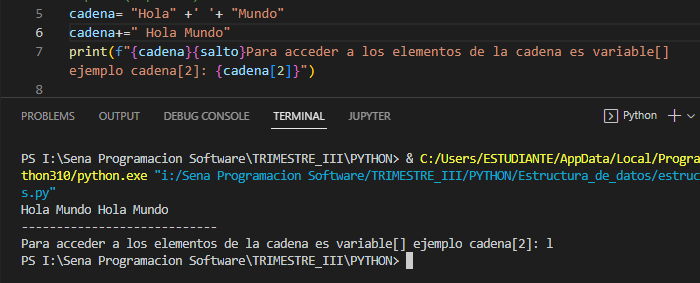
\’: Una comilla simple

\\ : El carácter de diagonal invertida (backslash)

Ejemplo:

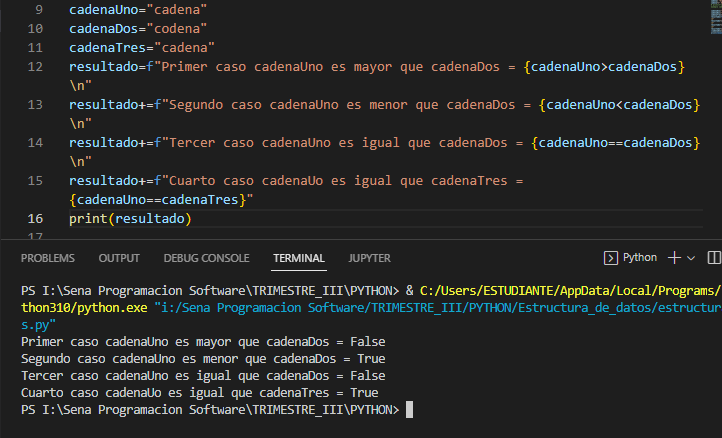


Para concatenar cadenas se puede usar de dos formas como se ve en el ejemplo:

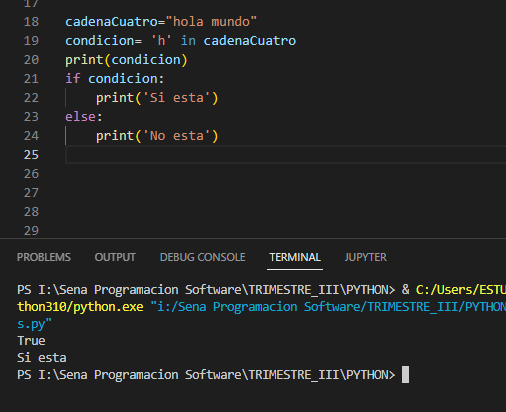


También aparte de concatenar cadenas también se puede apreciar que para acceder a un elemento de una cadena podemos usar los corchetes cuadrados e indicando una posición este nos trae el dato que está dentro de esta cadena en este ejemplo se puede ver que estamos indicándole la posición 2 que es la letra número 2.

Podemos crear consultas a una cadena por ejemplo para podes saber si un elemento se encuentra dentro de esta cadena o si dos cadenas son iguales así mismo con los operadores podemos retornar un booleano falso o verdadero, por ejemplo:



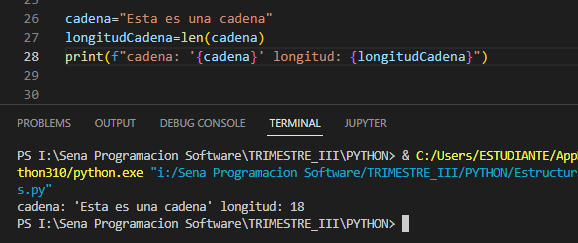
En la imagen anterior se puede evidenciar que en el primer caso el resultado es falso ya que, aunque la única letra que cambia es la ‘o’ en la “cadena” este carácter tiene un valor de memoria diferente por lo tanto la ‘a’ es un valor menor que el de la o por eso nos devuelve falso, en el segundo caso se puede apreciar que nos devuelve verdadero ya que la ‘a’ es un tipo de dato con valor menor al de ‘o’, en el tercer caso vemos que estamos preguntando si la cadenaUno es igual a cadenaDos este nos devuelve que es falso ya que no es verdadera esta consulta ya que tiene un carácter diferente por lo tanto nos devuelve un falso y por último se puede ver que la cadenaUno y la cadenaTres son iguales ya que contienen los mismos caracteres por lo tanto nos devuelve un verdadero.



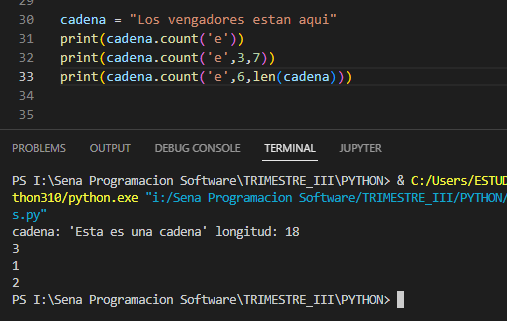
En la imagen anterior estamos guardando una condición la cual está preguntando si el carácter ‘h’ se encuentre dentro de la cadenaCuatro el resultado es que es verdadero

### **Métodos:**

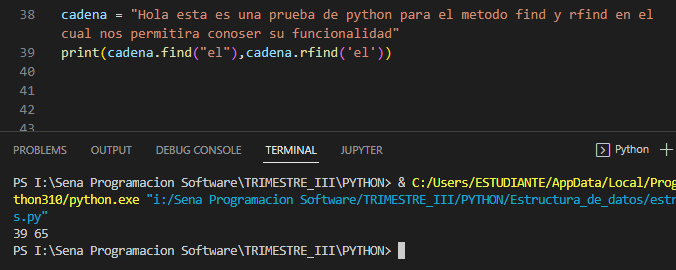
**(len)** esta función nos determina la longitud de la cadena de cuantos datos caracteres hay dentro de la cadena



**(count)** este método nos permita saber cuántas veces se repite un carácter dentro de la cadena también se le puede limitar la búsqueda en una sección específica la función es count(carácter ó subcadena , entero índice inicio, entero índice final)



**(find, rfind)** el método find retorna la ubicación de la primera ocurrencia y el rfind retorna la ubicación de la última ocurrencia encontrada en una cadena.



**(lower, upper, capitalize, title, swapcase)**

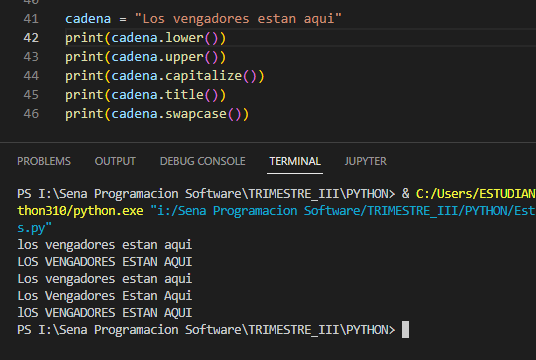
**(lower)** muestra la cadena en minúsculas

**(upper)** muestra la cadena en mayúsculas

**(capitalize)** primera letra de la cadena en mayúscula

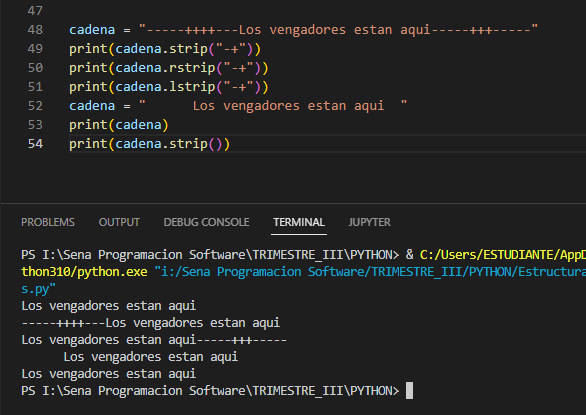
**(title)** primera letra de cada palabra a mayúscula

**(swapcase)** cambia de mayúsculas a minúsculas y de minúsculas a mayúsculas

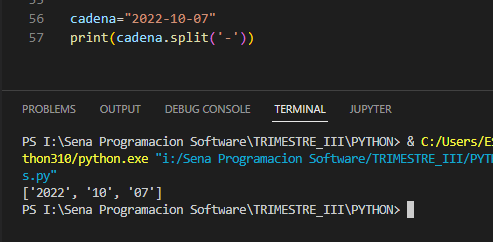


**(strip, lstrip, rstrip)**

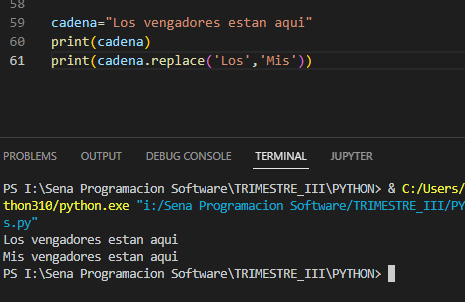
Estos métodos tienen como funcionalidad remover los caracteres que se le indique respectivamente strip: dos lados, lstrip: izquierda, rstrip: derecha de una cadena en el caso que no se proporcionen caracteres como argumento este eliminara espacios en blanco y tabulaciones.



**(split)** este método nos permite dividir la cadena indicándole un delimitador dejando las partes separadas dentro de una lista.

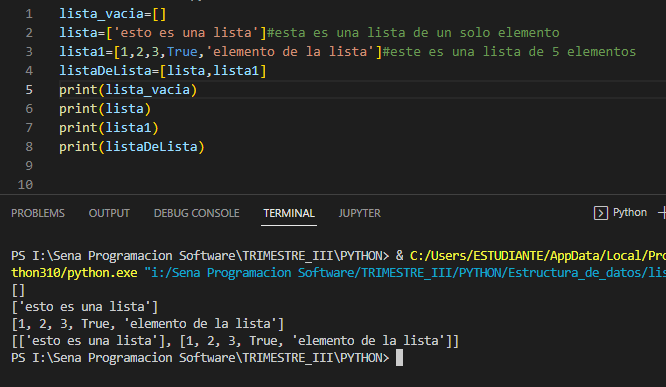


**(replace)** Este método permite reemplazar una subcadena de una cadena por otra la notación en replace(anterior, nueva).

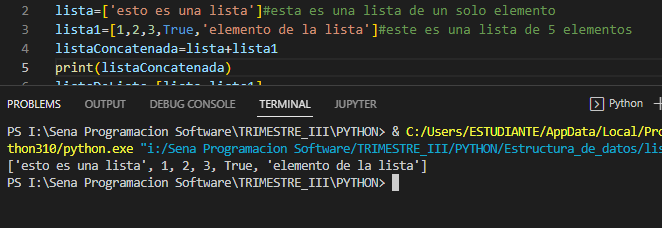


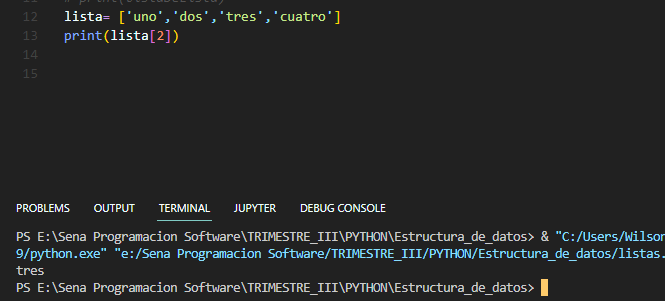
## **Listas:**

Las listas en Python son una estructura en donde se puede almacenar diferentes tipos de datos ya sean enteros flotantes cadenas boléanos hasta puede almacenar otras estructuras de datos como listas tuplas diccionarios. Las listas se delimitan con paréntesis cuadrados la secuencia de datos están separados por una coma.

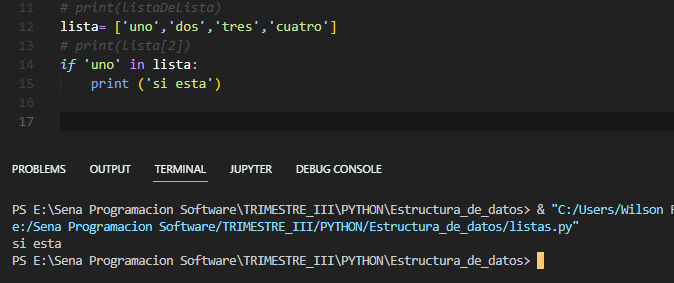


Se puede concatenar dos listas de esta forma:



Acceder a un elemento de la lista podemos igual que en las cadenas podemos usar los corchetes cuadrados y asignándole una posición este nos traerá el elemento que está en esa posición: 

Para consultar si un elemento se encuentra en una lista podemos usar una condicion if en el cual con la palabra reserbada in podevos averiguar si x elemento se encuentra en la lista respectiva.



Podemos iterar sobre una lista con un ciclo para o con un siclo mientras en donde vamos a recorrer cada elemento que tiene la lista con esto podemos entrar a ver modificar eliminar o actualizar esta lista.

Texto

Descripción generada automáticamente

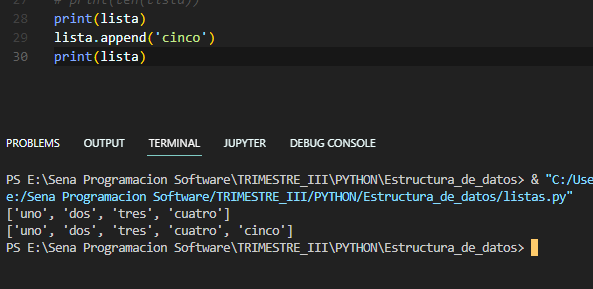
**Métodos:**

**(len)** Como pudimos ver en la imagen anterior en el ciclo mientras usamos la función len el cual nos retorna la longitud de la lista.

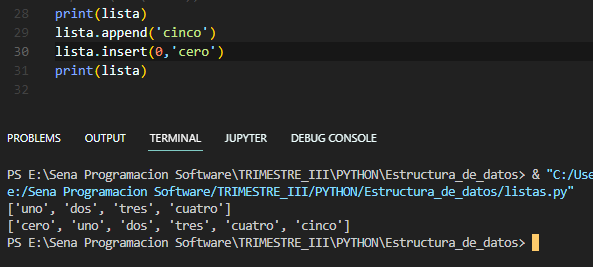
Texto

Descripción generada automáticamente

**(append)** este método nos permite agregar elementos al final de la lista su llamada seria así lista.append(elemento\_a\_agregar).



**(insert)** este método permite agregar elementos, pero a diferencia del append este nos permite agregar elementos en una ubicación especifica su llamado sería la siguiente lista.insert(numero\_posicion, elemento\_a\_agregar).

****

**(remove)**Para remover elementos de una lista usamos remove para eliminar la primera aparición del elemento indicado de la lista de izquierda a derecha su llamado sería el siguiente lista.remove(elemento\_a\_remover).

Texto

Descripción generada automáticamente

**(count)** el método count cuenta cuantas veces se repite el mismo elemento en la lista su llamado seria el siguiente lista.count(elemento)

Texto

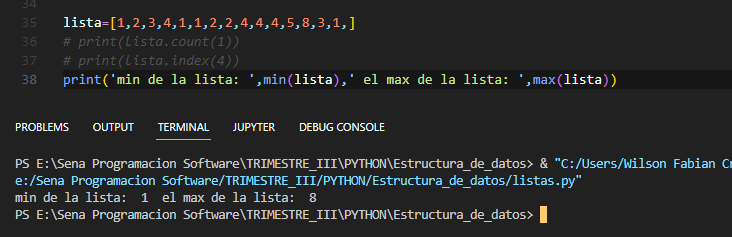
Descripción generada automáticamente

**(index)** Este método nos permite obtener la primera ocurrencia de un elemento nos devuelve la posición en la que se encuentra.

Texto

Descripción generada automáticamente

**(min-max)**El método min nos retorna el elemento mas bajo que hay en la lista y el max nos retorna el elemento con mayor valor.

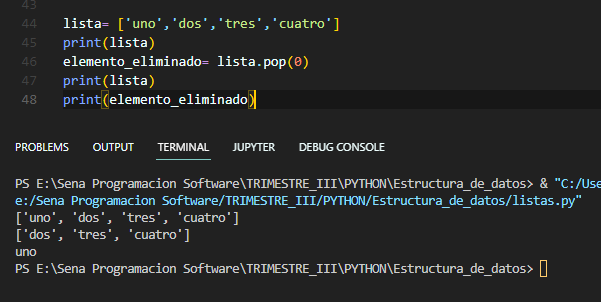


**(sort)** El método sort ordena la lista de menor a mayor valor así mismo se le puede indicar si lo retorna de mayo a menor.

Texto

Descripción generada automáticamente

(pop) Este método a comparación de remove es usado para eliminar una posición, pero así mismo este elemento eliminado se puede guardar en una variable para su posterior uso.



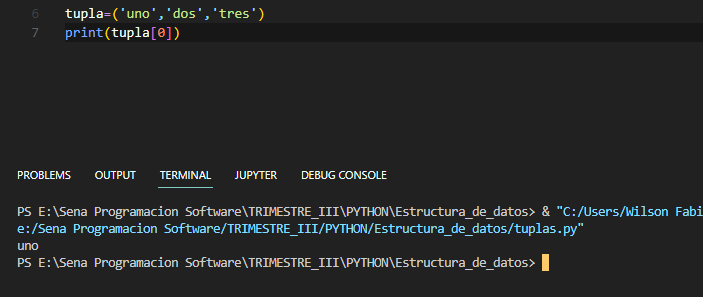
## **Tuplas**

Una tupla es una estructura de datos la cual puede almacenar datos como enteros listas diccionarios reales booleanos etc. Una tupla se puede escribir como una secuencia de datos separados por una coma y esta delimitada por paréntesis redondos para crear una tupla de un solo elemento se requiere que al final de elemento se agregue una coma para que esta sea leída como una tupla la característica principal de una tupla a diferencia de las demás estructura de datos es que es inmutable esto quiere decir que no puede se modificada después de ser creada.

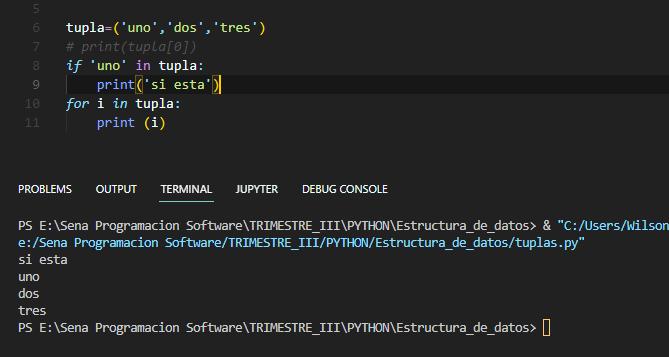
Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Al igual que las listas y las cadenas se pueden acceder a un elemento de una tupla por medio de paréntesis cuadrados



Así mismo podemos usar condiciones en una tupla t también podemos recorrer cada elemento de una tupla por medio de ciclos.

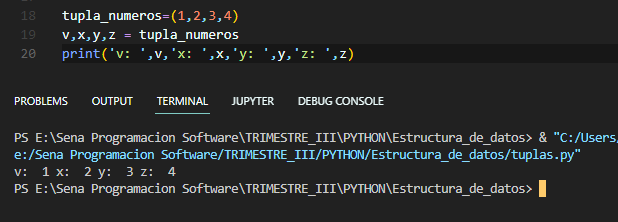


(tuple)este método se usa para poder crear tupla a partir de otros objetos como enteros cadenas etc.

Texto

Descripción generada automáticamente

Podemos desempaquetar desde una tupla a las variables.



## **Diccionarios:**

Un diccionario se define como una colección que almacena elementos por medio de una llave y valor estos elementos no se pueden repetir y así como sets estos se almacenan dentro de corchetes y la llave y el valor se determina por <llave>:<valor> y los elementos están ordenados como se creó.

Al igual que todas las colecciones se puede usar el método len(colección) el cual va a retornar el numero de elementos que contiene la colección empezando desde cero.

Para acceder a un elemento del diccionario existen diferentes formas para realizarlo una de ellas es llamar la variable donde se guarda el diccionario seguido por corchetes cuadrados dentro de ellos podemos escribir en una cadena el nombre de la llave y este nos traerá el valor de esta llave, otra forma de acceder a un elemento es por medio de la función .get(n), donde “n” es el nombre de la llave que se encuentra en el diccionario.

Diccionario[“key”]

Diccionario.get(“key)

Para poder ver las llaves que contiene un diccionario podemos usar el método.keys() el cual nos retornara un listado de todas la llaves que se encuentran en el diccionario, así mismo como hay un método para las llaves también hay uno para los valores el cual es .values() el cual retornara un listado con todos los valores que se encuentran en la lista. En el caso que se necesite tanto los valores como las llaves del diccionario existe el método.items() el cual retornara una matriz en donde en una lista se almacenara por medio de tuplas la llave y el valor para los elementos que tenía el diccionario.

X= diccionario.keys()

X= diccionario.values()

X= diccionario.items()

Texto

Descripción generada automáticamente

Para modificar el valor de una llave especifica se puede usar dos métodos el primero así como accedemos a un elemento del diccionario por medio de corchetes cuadrados identificamos en una cadena le nombre de la llave seguido por guardar el valor que se requiera ejemplo:

Diccionario[“key”]=<valor>

Otra forma de modificar los valores de una llave es por medio del método update({k:v}) en donde v es el nombre de la llave que va a ser modificada seguido por v que seria el nuevo valor que va a remplazar el anterior.

Para añadir nuevos elementos a un diccionario usamos los mismos métodos anteriores el primero seria llamar la variable donde se encuentra el diccionario y por medio de corchetes cuadrados agregamos la nueva llave que vamos a agregar y guardamos el valor que va a tener esta llave es claro aclarar que la llave que se va a agregar no puede existir anteriormente dentro del diccionario porque en tal caso lo único que se haría es modificar el valor ejemplo:

Diccionario[“nueva\_llave”]=<nuevo\_valor>

El otro método es usando update({k:v}),en donde k es la llave nueva y v es el valor nuevo que se va a añadir al diccionario.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Para remover un elemento de un diccionario hay varias formas de hacerlo, por ejemplo:

Diccionario.pop(“key”)

Este eliminara tanto la llave como el valor que s encuentran en el diccionario.

Diccionario.popitem()

Este método eliminara el ultimo ítem del diccionario.

del diccionario[“key”]

de esta forma con la palabra “del” podemos eliminar definitivamente, en este caso la llave especificada.

del diccionario

en este caso se eliminará por completo el diccionario.

diccionario.clear()

de esta forma limpiaremos toda la información dentro del diccionario y quedara vacío.