Métodos para manejo de String

Método	Descripción
upper()	Convierte toda la cadena a letras mayúsculas.
lower()	Convierte toda la cadena a letras minúsculas.
capitalize()	Convierte el primer carácter de la cadena a mayúscula y el resto a minúsculas.
title()	Convierte cada palabra en la cadena a comenzar con una letra mayúscula.
strip()	Elimina los espacios en blanco al principio y al final de la cadena.
lstrip()	Elimina los espacios en blanco al principio de la cadena.
rstrip()	Elimina los espacios en blanco al final de la cadena.
replace(old, new)	Reemplaza todas las ocurrencias de 'old' con 'new' en la cadena.
find(substring)	Encuentra la primera ocurrencia de 'substring' en la cadena y devuelve su posición.
split(delimiter)	Divide la cadena en una lista de subcadenas usando 'delimiter' como separador.
join(iterable)	Une una lista de subcadenas en una sola cadena utilizando la cadena como separador.
startswith(prefix)	Verifica si la cadena comienza con 'prefix' y devuelve True o False.
endswith(suffix)	Verifica si la cadena termina con 'suffix' y devuelve True o False.
isalpha()	Verifica si la cadena contiene solo letras y no está vacía.
isdigit()	Verifica si la cadena contiene solo dígitos y no está vacía.
len()	Devuelve la longitud (cantidad de caracteres) de la cadena.
strip(characters)	Elimina los caracteres especificados al principio y al final de la cadena.

Docente. Albeiro Muriel

```python
mensaje = "    Bienvenidos... al curso de Python   "

print("El tamaño de letras en el mensaje es de " , len(mensaje))
espacios = mensaje.strip()
print(espacios)
print("Imprimir en mayuscula ", espacios.upper())
print("Imprimir en minuscula ", espacios.lower())
print("Imprimir en mayuscula sostenica ", espacios.title())
print("Imprimir en mayuscula inicial ", espacios.capitalize())
parrafo = espacios.split('... ')
print(parrafo[0])
print(parrafo[1])
```

[ ] Listas

En Python, una lista es una estructura de datos que permite almacenar una colección ordenada y mutable de elementos. Los elementos en una lista pueden ser de diferentes tipos, como números, cadenas, objetos e incluso otras listas. Las listas son una de las estructuras de datos más versátiles y utilizadas en Python debido a su flexibilidad y amplio conjunto de métodos para manipular y operar con los elementos almacenados.

append():
Agrega un elemento al final de la lista.

remove():
Elimina un elemento específico de la lista.

len():
Devuelve la cantidad de elementos en la lista.

Método	Descripción
insert(index, element)	Inserta el elemento dado en la posición indicada. Los elementos a la derecha se desplazan para hacer espacio.
pop(index)	Elimina y devuelve el elemento en la posición indicada. Si no se proporciona un índice, se elimina y devuelve el último elemento.
index(element)	Devuelve el índice de la primera aparición del elemento en la lista.
count(element)	Devuelve la cantidad de veces que aparece el elemento en la lista.
sort()	Ordena los elementos de la lista en orden ascendente.
reverse()	Invierte el orden de los elementos en la lista.
extend(iterable)	Agrega los elementos de otro iterable (como otra lista) al final de la lista actual.
clear()	Elimina todos los elementos de la lista, dejándola vacía.

Docente. Albeiro Muriel

```python
modulos =["Lógica", "Base Datos", "Html5", "Nuevas Tecnologias"]
# imprimir una lista
print (modulos)

# cantidad de elemetos que hay en la lista
print("Nro de elementos de la lista ", len(modulos))

#Agregar un elemento en la ultima posicion de la lista
modulos.append("Web 1")
print (modulos)

#count devuelve el nro de veces que se repite un elemento
print("nro de veces de Html5 ", modulos.count("Html5"))

print("Lista ordenada " )
modulos.sort()
print (modulos)
# imprimir la posición dentro de la lista
print ("Posición del elemento en la lista Html5 ", modulos.index("Html5"))
modulos.insert(2, "Web 2")
print (modulos)
#Elimina un elemento de la posición
modulos.pop(2)
# elimina todos los elementos
modulos.clear()
```

```python
i=0
for indice in modulos:
    print (i, indice)
    i=i+1
```

Docente. Albeiro Muriel

( )
Tuplas

Una tupla en Python es una estructura de datos similar a una lista, pero con la diferencia fundamental de que es **inmutable**. Esto significa que una vez que se crea una tupla, **no se pueden modificar** sus elementos individuales **ni agregar ni eliminar** elementos. Las tuplas son útiles cuando necesitas almacenar un conjunto de valores que no deben cambiar durante la ejecución del programa, como coordenadas, valores constantes o información relacionada que debe mantenerse intacta.

Sin embargo, si intentamos modificar algún elemento de la tupla, generará un error, ya que las tuplas son inmutables. Esto contrasta con las listas, que son mutables y permiten modificar sus elementos después de su creación.

```python
estudiante =("Linda", "Nuevas Tecnologias", [4,1.7,4.9])

# Cantidad de elementos
print(len(estudiante))

nombre = estudiante[0]
modulo = estudiante[1]
notas = estudiante[2]

print(f"Estudiante {nombre}\nModulo {modulo}\nNotas {notas}")
print("Otro forma de escribir con un for")

for indices in estudiante:
    print(indices)
```

métodos que puedes utilizar con tuplas:

count(element):
Devuelve la cantidad de veces que aparece el elemento en la tupla.

index(element):
Devuelve el índice de la primera aparición del elemento en la tupla.

Docente. Albeiro Muriel