```python
# TIPOS DE STRINGS:
# index(): string para conocer posicion de letras.
# format[]: string para dar formato a los strings.
# upper(): pasar a mayusculas
# lower(): pasar a minusculas
# split(): separar en partes (lista)
# join(): unir items usando separador
# find(): encontrar un substring
# replace(): reemplazar a minuscar un substring


# Metodo index: solo se puede aplicar a los strings

# string(str) "hola","adios",'prueba' -> cualquier caracter entre comillas "" o '' es un string .

# 1. Conocer la posicion de un caracter:
    mi_texto = "hola"
    print(mi_texto.index("o"))

# 2. Conocer que caracter hay en un indice
# Indice positivo. Empieza en 0.
    mi texto = "hola"
    print(mi_texto[3])

# Indice negativo. Empieza en 0 pero sigue desde atras.s
    mi_texto = "hola"
    print(mi_texto[-3])

# Ejercicio 1: Encuentra y muestra en pantalla que caracter ocupa la quinta posicion dentro de la siguiente palabra: "ordenador"
palabra = "ordenador"
print(palabra[5])

# Ejercicio 2: Encuentra y muestra en pantalla el indice de la primera aparicion de la palabra "practica" en la siguiente frase:
#
# "En teoria, la teoria y la practica son los mismos. En la practica, no lo son."
frase = "En teoria, la teoria y la practica son los mismos. En la practica, no lo son."
print(frase.index("practica"))

# Encuentra y muestra en pantalla el indice de la ultima aparicion de la palabra "practica" en la siguiente frase:
frase = "En teoria, la teoria y la practica son los mismos. En la practica, no lo son."
frase = "En teoria, la teoria y la practica son los mismos. En la practica, no lo son."
print(frase.rfind("practica"))


# METODO FORMAT

# Extraer substrings
texto = "ABCDEFGH"
fragmento = texto[2:5] # Coge desde el segundo caracter hasta el quinto, sin incluirlo
print(fragmento)

# SALTAR LETRAS
texto = "ABCDEFGH"
fragmento = texto[2:5:2] # Coge desde el segundo caracter hasta el quinto, sin incluirlo pero saltando de dos en dos
print(fragmento)

# Ejercicio 1: Extrae la primera palabra de la siguiente frase utilizando slicing, y muestrala en pantalla:
frase = "Controlar la complejidad es la esencia de la programacion"

fragmento = frase[:9]
print(fragmento)

# Ejercicio 2: Toma cada tercer caracter empezando desde el noveno hasta el final de la frase, e imprime el resultado.
frase = "Nunca confies en un ordenador que no puedas lanzar por una ventana"
fragmento = frase[8::3]
print(fragmento)

# Ejercicio 3: Invierte la posicion de todos los caracteres de la siguiente frase y muestra el resultado en pantalla.
frase = "Es genial trabajar con ordenadores. No discuten, lo recuerdan todo y no se beben tu cerveza"
fragmento = frase[::-1]
print(fragmento)


# UPPER
texto = "prueba svlad"
resultado = texto.upper()
print(resultado)

# LOWER
texto = "prueba svlad"
resultado = texto.lower()
print(resultado)

# SPLIT
texto = "prueba svlad"
resultado = texto.split( )
print(resultado)

a = "prueba"
b = "svlad"
c = " ".join([a,b])
print(c)

# FIND
texto = "prueba svlad"
resultado = texto.find("s")
print(resultado)

# REPLACE
texto = "prueba svlad"
resultado = texto.replace("svlad","dani")
```

```python
print(resultado)


# EJERCICIO 1: Imprime el siguiente texto en mayusculas, empleando el metodo especifico de strings:O 1:
frase = "Especialmente en las comunicaciones electronicas, la escritura enteramente en mayusculas equivale a gritar."
print(frase.upper())

# EJERCICIO 2:Une la siguiente lista en un string, separando cada elemento con un espacio.
# Utiliza el metodo apropiado de listas/strings, y muestra en pantalla el resultado.
lista_palabras = ["La","legibilidad","cuenta."]
unir=" ".join(lista_palabras)
print(unir)

# EJERCICIO 3: Reemplaza en la siguiente frase:  "Si la implementacion es dificil de explicar, puede que sea una mala idea."
#los siguientes pares de palabras: "dificil" --> "facil" "mala" --> "buena" y muestra en pantalla la frase con ambas palabras modificadas.

texto = "Si la implementacion es dificil de explicar, puede que sea una mala idea."
resultado = texto.replace("dificil", "facil").replace("mala", "buena")
print(resultado)

# PROPIEDADES DE STRINGS
# - Son inmutables: su contenido no se puede modificar.
# - Pueden concatenarse mediante el signo +. Tambien se pueden multiplicar *
# - Multilineales: permite realizar saltos de linea.
# - Lenght(): calcular largura string.

# SUMA STRINGS
n1 = "Ste"
n2 = "fan"
print(n1+n2)

# MULTIPLICACION STRINGS
n1 = "Ste"
n2 = "fan"
print(n1*10) # Repite la palabra 10 veces

# MULTILINEALES: triple comillas
n1 = """Que
tal
estas
"""
print("estas" in n1) # te dice si el caracter esta dentro del string

# LENGHT: te dice el numero de caracteres
n1 = "SteLSFKJÑLSJCÑAL"
print(len(n1))


# EJERCICIO 1: Concatena 15 veces el texto "Repeticion" y muestra el resultado en pantalla. Por suerte, conoces
# que los strings son multiplicables y puedes realizar esta actividad de forma simple y elegante.
a = "Repeticion"
print(a*15)

# EJERCICIO 2: Verifica si la palabra "agua" no se encuentra en el siguiente haiku. Debes imprimir el booleano.
a = """Tierra mojada
mis recuerdos de viaje,
entre las lluvias"""
print("agua" not in a)

# EJERCICIO 3: Muestra en pantalla el largo (en numeros de caracteres) de la palabra electroencefalografista.
a = "electroencefalografista"
print(len(a))

# LISTAS: secuencia ordenada de objetos., da igual el tipo. va entre corchetes [] y separados por coma , . El orden de los elementos puede variar.
mi_lista = ['a', 'b', 'c']
print(type(mi_lista))

mi_lista2 = [142, 'hola', 2342.4565] # pueden almacenar objetos de diferentes tipos
mi_lista3 = mi_lista + mi_lista2 # las listas se pueden concatenar
mi_lista3[0] = 'stefan' # en la posicion 0 del indice , modificame el campo que haya por el texto
print(mi_lista3)

print(len(mi_lista3)) # tamaño de la lista

# METODOS LISTAS

mi_lista3.append('g') # .append añade valores a la lista
print(mi_lista3)

mi_lista3.pop() # .pop elimina elementos de la lista. Sin parametros, elimina el ultimo registro.
print(mi_lista3)

mi_lista3.pop(3) # .pop elimina elementos de la lista. Debemos indicar el indice de la lista del elemento a eliminar.
print(mi_lista3)

eliminado = mi_lista3.pop(2) # puedes guardar el elemtento eliminado
print(eliminado)

desordenado = ['s', 'g', 'o', 'a', 'z']
desordenado.sort() # .sort ordena alfabeticamente o numericamente los valores
print(desordenado)
# MUY IMPORTANTE: .sort reordena la lista que existe, NO crea algo a partir de lo que ya hay.
# Si metemos el resultado en una variable, nos dara error. Para ver el resultado, volver a llamar
# a la variable ordenada.

desordenado.reverse() # ordena pero al reves, de la Z -> A y 9 -> 0.
print(desordenado)


# EJERCICIO 1: Crea una lista con 5 elementos, dentro de la variable mi_lista.
# Puedes incluir strings, booleanos, numeros, etc.
```

```python
mi_lista = ['a', 'b', 'c', 12345, 392.42]

# EJERCICIO 2: Agrega el elemento "motocicleta" a la siguiente lista de medios de transporte:
medios_transporte = ["avion", "auto", "barco", "bicicleta"]
medios_transporte.append('motocicleta')

# EJERCICIO 3: Utiliza el metodo pop() para quitar el tercer elemento de la siguiente lista llamada frutas, y almacenalo en una variable llamada eliminado.
# Utiliza metodos de listas sin alterar la linea de codigo ya suministrada.
frutas = ["manzana", "banana", "mango", "cereza", "sandia"]
eliminado = frutas.pop(2)



# DICCIONARIOS: {'clave':'valor','arte':'cine'} -> pares de palabras agrupados, la primera se denomina clave y la segunda valor. Van entre {}
diccionario = {'c1':'valor1', 'c2':'valor2'}
print(diccionario)

resultado = diccionario['c1'] # esta es la forma de realizar busquedas de claves en un diccionario
print(resultado)

cliente = {'nombre':'stefan', 'apellido':'vlad', 'edad':22, 'peso':85.5}
consulta = cliente['apellido']
print(consulta)

dic = {'c1':55, 'c2':[10, 20, 30],'c3':{'s1':100, 's2':200}} # diccionario creado pero con INT, lista y otro diccionario dentro
print(dic['c2'][2]) # busca dentro de la clave 'c2' del diccionario, el segundo parametro
print(dic['c3']['s2']) # busca dentro de la clave 'c3' del diccionario la clave 's2' de ese diccionario

dic = {'c1':['a','b','c'], 'c2':['d','e','f']}
print(dic['c2'][1].upper()) # Muestra resultado en mayuscula

cliente = {'nombre':'stefan', 'apellido':'vlad', 'edad':22, 'peso':85.5}
cliente['nacionalidad'] = 'rumania' # para añadir valores nuevos al diccionario
print(cliente)

cliente = {'nombre':'stefan', 'apellido':'vlad', 'edad':22, 'peso':85.5}
cliente['nombre'] = 'klk' # para modificar valores existentes del diccionario
print(cliente)

# METODOS
print(cliente.keys()) # imprime solamente las claves del diccionario
print(cliente.values()) # imprime solamente los valores del diccionario
print(cliente.items()) # muestra los tupples


# Crea un diccionario llamado mi_dic que almacene la siguiente informacion de una persona:
#     nombre: Karen
#     apellido: Jurgens
#     edad: 35
#     ocupacion: Periodista
# Los nombres de las claves y valores deben ser iguales a la consigna.

#nombre: Karen
#apellido: Jurgens
#edad: 35
#ocupacion: Periodista
mi_dic = {'nombre':'Karen','apellido':'Jurgens','edad':35,'ocupacion':'Periodista'}

# Crea una funcion print que devuelva del segundo item de la lista llamada points2, dentro del siguiente diccionario.
mi_dict = {"valores_1":{"v1":3,"v2":6},"puntos":{"points1":9,"points2":[10,300,15]}}
print(mi_dict["puntos"]["points2"][1])

# Actualiza la informacion de nuestro diccionario llamado mi_dic  (reasignando nuevos valores a las claves segun corresponda),
# y agrega una nueva clave llamada "pais" (sin tilde). Los nuevos datos son:
#     nombre: Karen
#     apellido: Jurgens
#     edad: 36
#     ocupacion: Editora
#     pais: Colombia



# TUPLES: ("sal", 1.5, -3) -> lista de objetos, da igual el tipo. va entre parentesis () pero a diferencia de la lista, el orden de los elementos NO varia
# Ocupan menos espacio de memoria y son inmutables.

t = (1,2,3,4)
print(t)

print(t[0]) # se pueden filtrar por indices

t = (1,2,("a","b","c")) # tuple dentro de un tuple
print(t[2][1]) # busca dentro del tuple la posicion 1

t = (1,2,3)
x,y,z = t # se pueden asignar variables al contenido del tupple
print(x,y,z)

# METODO COUNT: cuenta cuantas veces aparece el numero 1 dentro del tuple
t = (1,2,3,1)
y = t.count(1) # cuenta cuantas veces aparece el numero 1 dentro del tuple
print(y)


# METODO INDEX: te indica en que indice se encuentra un valor
t = (1,2,3,1)
y = t.index(2)
print(y)


# Utiliza un metodo de tuplas para contar la cantidad de veces que aparece el valor 2 en la siguiente tupla, y muestra el resultado (integer) en pantalla:
mi_tupla = (1, 2, 3, 2, 3, 1, 3, 2, 3, 3, 3, 1, 3, 2, 2, 1, 3, 2)
```

```python
print(mi_tupla.count(2))


# Convierte a lista la siguiente tupla, y almacenala en una variable llamada mi_lista.
mi_tupla = (1, 2, 3, 2, 3, 1, 3, 2)
mi_lista = list(mi_tupla)


# Extrae los elementos de la siguiente tupla en cuatro variables: a, b, c, d
mi_tupla = (1, 2, 3, 4)
a,b,c,d = mi_tupla


# SETS: ista de objetos UNICOS, no se pueden repetir. Van entre {}
set1 = set((1,2,3,4,5)) # se puede definir con set() pero dentro, los datos tienen que ir entre parentesis otra vez set((datos))
print(mi_set)

set2 = {"a","b","c","d"} # tambien se puede definir entre llaves {} como el diccionario
print(set2)

# METODO UNION: Unir sets.
s1 = {1,2,3}
s2 = {3,4,5}
s3 = s1.union(s2)
print(s3)

# METODO ADD: añadir valores al set.
s1 = {1,2,3}
s1.add(4)
print(s1)

# METODO REMOVE: eliminar valores del set
s1 = {1,3,2}
s1.remove(2) # ELIMINA EL NUMERO 2, NO EL VALOR 2
print(s1)

# METODO DISCARD: igual que el remove, pero si no existe, no da error.
s1 = {1,3,2}
s1.discard(8) # ELIMINA EL NUMERO 2, NO EL VALOR 2
print(s1)

# METODO POP: ELIMINA UN NUMERO ALEATORIO DEL SET.
s1 = {1,3,2}
s1.pop()
print(s1)

# METODO CLEAR: VACIA EL SET ENTERO
s1 = {1,3,2}
s1.clear()
print(s1)


# EJERCICIOS:
# Une los siguientes sets en uno solo, llamado mi_set_3:
mi_set_1 = {1, 2, "tres", "cuatro"}
mi_set_2 = {"tres", 4, 5}
mi_set_3 = mi_set_1.union(mi_set_2)

# Elimina un elemento al azar del siguiente set, utilizando metodos de sets.
sorteo = {"Camila", "Margarita", "Axel", "Jorge", "Miguel", "Monica"}
sorteo.pop()

# Agrega el nombre Damian al siguiente set, utilizando metodos de sets:
sorteo = {"Camila", "Margarita", "Axel", "Jorge", "Miguel", "Monica"}
sorteo.add("Damian")



# BOOLEANOS: True, false -> solo puede tener dos valores. Se usa para comprobar si las condiciones se cumplen
var1 = True
var2 = False

numero = 5 > 2+4 # esto es una operacion booleana debido al signo de comparacion
print(type(numero))
print(numero)

lista = [1,2,3,4]
control = 5 in lista
print(type(control))
print(control)

# OPERADORES LOGICOS:
# Mayor >
# Menor <
# Mayor o igual >=
# Menor o igual <=
# Igual ==
# Diferente !=


# EJERCICIOS:
# Realiza una comparacion que arroje como resultado un booleano y almacena el resultado (True/False) en una variable llamada prueba
prueba  = 2 > 5

# Verifica si 17834/34 es mayor que 87*56 y muestra el resultado (booleano) en pantalla utilizando print()
a = 17834/34 > 87*56
print(a)

# Verifica si la raiz cuadrada de 25 es igual a 5 y muestra el resultado (booleano) en pantalla utilizando print()
a = 25**0.5 == 5
print(a)
```