Manejo de Cadenas de Caracteres

1 GENERALIDADES

— Las cadenas son cualitativamente diferentes de los otros cuatro tipos, porque están compuestas de piezas más pequeñas: los `caracteres`.

— Las cadenas de caracteres (STRING) se implementan como una secuencia de caracteres cuya longitud es variable.

— No existe la declaración formal de una variable de este tipo, solamente basta con darle el valor de inicio una cadena vacía ""

2 OPERACIONES CON CADENAS DE CARACTERES.

2.1 Asignación.

La asignación se realiza empleando el signo igual, como en cualquier variable.

```
Pais = “Venezuela”
```

Las cadenas de caracteres son inmutables (sus elementos no se pueden modificar). Si se requieren modificaciones, se debe construir una cadena nueva.

```
Texto    = "Hola mundo"
Texto[3] = "A"                    # ¡ERROR, esto no se puede hacer!
```

2.2 Porciones de texto.

La selección de una porción de una cadena de caracteres es similar a la selección un carácter:

P	e	d	r	o	,		P	a	b	l	o		y		M	a	r	i	o
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

```
Cadena = "Pedro, Pablo y Mario"
print(Cadena[0:5]    )            # Despliega: Pedro
print(Cadena[7:12])               # Despliega: Pablo
print(Cadena[15:20])              # Despliega: Mario
print(Cadena[-4:-1])              # Despliega: ari

Pais = "Panamá"
print(Pais[:3])                   # Despliega: Pan
print(Pais[3:])                   # Despliega: amá

Dato = "Hola Mundo"
print(Dato[0])                    # Corresponde a la letra H.
```

2.3 Concatenación.

Para concatenar strings, se puede ocupar el símbolo +.

```
print("HOLA" + "QUE" + "TAL")                    # Despliega: HOLAQUETAL
print("HOLA" + " " + "QUE" + " " + "TAL")        # Despliega: HOLA QUE TAL
```

También es posible multiplicar una cadena:

```
Fruta = "PERA”
Fruta = Fruta * 3
print(Fruta)                                     # Despliega: PERAPERAPERA
```

2.4 Comparación.

Se pueden ocupar los operadores de relación =, <, >, <=, >= y <>, produciéndose un resultado booleano.

Ejemplo:

```
"a" < "b"                          # es TRUE
"A" < "B"                          # es TRUE
"a" < "B"                          # es FALSE, pues "a" está ubicado en posiciones más
                                     altas en la tabla ASCII.
"HOLA" == "HOLA "                  # es FALSE, por el carácter blanco en la segunda cadena.
```

La operación se hace sobre los valores ASCII de los caracteres.

2.5 El operador in

El operador **in** prueba si una cadena es una subcadena de otra:

```
"z"  in "manzana"                  # True
"i"  in "manzana"                  # False
"nz" in "manzana'"                 # True
"az" in "manzana"                  # False
"za" in "manZana"                  # False
```

2.6 Recorrido de cada carácter de una cadena.

Para acceder a cada carácter dentro de una cadena mediante un recorrido en ella, existen las siguientes posibilidades:

Forma 1:

```
Fruta  = "PERA"
Indice = 0
while Indice < len(Fruta):
    Letra = Fruta[Indice]
    print(Letra)
    Indice = Indice + 1
```

Forma 2:

```
Fruta = "PERA"
for Indice in range(0,len(Fruta)):
    print(Fruta[Indice])
```

Forma 3:

```
Fruta = "PERA"
for Letra in Fruta:
    print(Letra)
```

3 MÉTODOS DE LAS CADENAS.

Python maneja las cadenas de caracteres como objetos que poseen propiedades (valores) y métodos (procesos). Su forma de usar es la siguiente:

Cadena.metodo

capitalize()

Devuelve una copia de la cadena con el primer carácter en mayúscula.

```
Pais = "CHILE CAMPEÓN"
print(Pais.capitalize())              # Despliega: Chile campeón
```

swapcase()

Devuelve una copia de la cadena con las mayúsculas pasadas a minúsculas y viceversa.

```
Cadena = "Hola Mundo"
print(cadena.swapcase())              # Despliega: hOLA mUNDO
```

title()

Devuelve una versión con formato título, es decir, con todas las palabras en minúsculas excepto la primera letra, que va en mayúsculas.

```
Cadena = "HOLA MUNDO"
print(cadena.title())                 # Despliega: Hola Mundo
```

split([sep [,maxsplit]])

Devuelve una lista de las palabras de la cadena, usando sep como delimitador de palabras. Si se indica maxsplit, se devolverán como mucho maxsplit valores (el último elemento contendrá el resto de la cadena). Si no se especifica sep o es None, cualquier espacio en blanco sirve de separador.

```
Cadena = "HOLA MUNDO"
print(Cadena.split(" "))              # Despliega: ["HOLA", "MUNDO"]
```

replace(old, new[, maxsplit])

Devuelve una copia de la cadena en la que se han sustituido todas las apariciones de old por new. Si se proporciona el argumento opcional maxsplit, sólo se sustituyen las primeras maxsplit apariciones.

```
Cadena = "HOLA MUNDO"
print(Cadena.replace("O","xx"))       # Despliega: HxxLA MUNDxx
```

count(sub[, start[, end]])

Devuelve cuántas veces aparece la subcadena (sub) en la cadena. Los argumentos opcionales start y
end se interpretan según la notación de corte.

```
Cadena = "HOLA MUNDO"
print(Cadena.count("O"))                # Despliega: 2

Cadena = "HOLA MUNDO"
print(Cadena.count("O",0,1))            # Despliega: 0

Cadena = "HOLA MUNDO"
print(Cadena.count("O",0,2))            # Despliega: 1

Cadena = "HOLAAAMUNDO"
print(Cadena.count("A",3,5))            # Despliega: 2

Cadena = "HOLAAAMUNDO"
print(Cadena.count("A",3,6))            # Despliega: 3

Cadena = "HOLA MUNDO"
print(Cadena.count("O",5))              # Despliega: 1 → A partir del carácter 5 se
                                        encuentra una sola vez, hacia la derecha, la letra O

Cadena = "HOLA MUNDO"
print(Cadena.count("O",6,3))            # Despliega: 0 → A partir del carácter 6 y hasta 3
                                        más a la derecha no se encuentra la letra O (Busca la
                                        letra O dentro del substring MUN)
```

find(sub[, start[, end]])

Devuelve el menor índice de la cadena para el que sub se encuentre, de tal modo que sub quede contenido en el rango [start, end). Los argumentos opcionales start y end se interpretan según la notación de corte. Devuelve -1 si no se halla sub.

```
Cadena = "HOLA MUNDO"
print(Cadena.find("O"))                 # Despliega: 1 (Recuerde que la letra H ocupa el índice
                                        0 → Cadena[0])
Cadena = "HOLA MUNDO"
print(Cadena.find("X"))                 # Despliega: -1
```

index(sub[, start[, end]])

Como find(), pero lanza ValueError si no se encuentra la subcadena.

```
Cadena = "HOLA MUNDO"
print(Cadena.index("X"))                # Error…
```

isalnum()

Devuelve verdadero si todos los caracteres de la cadena son alfanuméricos (letras y dígitos) y hay al menos un carácter. En caso contrario, devuelve falso.

```
Cadena = "3"
```

```
print(Cadena.isalnum())                # Despliega: True

Cadena = "3A"
print(Cadena.isalnum())                # Despliega: True

Cadena = "3A@"
print(Cadena.isalnum())                # Despliega: False
```

isalpha()

Devuelve verdadero si todos los caracteres de la cadena son alfabéticos y hay al menos un carácter. En caso contrario, devuelve falso.

```
Cadena = "3"
print(Cadena.isalpha())                # Despliega: False

Cadena = "3A"
print(Cadena.isalpha())                # Despliega: False

Cadena = "A"
print(Cadena.isalpha())                # Despliega: True

Cadena = "A3"
print(Cadena.isalpha())                # Despliega: False

Cadena = "3A@"
print(Cadena.isalpha())                # Despliega: False
```

isdigit()

Devuelve verdadero si todos los caracteres de la cadena son dígitos y hay al menos un carácter. En caso contrario, devuelve falso.

```
Cadena = "3"
print(Cadena.isdigit())                # Despliega: True

Cadena = "3A"
print(Cadena.isdigit())                # Despliega: False

Cadena = "A"
print(Cadena.isdigit())                # Despliega: False

Cadena = "A3"
print(Cadena.isdigit())                # Despliega: False

Cadena = "3A@"
print(Cadena.isdigit())                # Despliega: False
```

islower()

Devuelve verdadero si todos los caracteres alfabéticos de la cadena están en minúscula y hay al menos un carácter susceptible de estar en minúsculas. En caso contrario, devuelve falso.

isspace()

Devuelve verdadero si todos los caracteres de la cadena son espacio en blanco (lo que incluye tabuladores, espacios y retornos de carro) y hay al menos un carácter. En caso contrario, devuelve falso.

isupper()

Devuelve verdadero si todos los caracteres alfabéticos de la cadena están en mayúscula y hay al menos un carácter susceptible de estar en mayúsculas. En caso contrario, devuelve falso.

join(seq)

Devuelve una cadena formada por la concatenación de todos los elementos de la secuencia seq. Los elementos se separan por la cadena que proporciona el método. Se lanza TypeError si alguno de los elementos no es una cadena.

```
Cadena = "-"
Secuencia = ("a", "b", "c")             # Despliega a-b-c
print(Cadena.join(Secuencia))
```

__len__()

Devuelve el número de caracteres de una cadena.

```
Pais = “Venezuela”
print(Pais.__len__())                   # Despliega el valor 9
```

lower()

Devuelve una copia de la cadena convertida en minúsculas.

```
Cadena = "hola MUNDO"
print(Cadena.lower())                   # Despliega: hola mundo
```

upper()

Devuelve una copia de la cadena en mayúsculas.

```
Cadena = "hola MUNDO"
print(Cadena.upper())                   # Despliega: HOLA MUNDO
```

lstrip()

Devuelve una copia de la cadena con el espacio inicial eliminado.

```
Cadena = "    HOLA MUNDO"
print(Cadena.lstrip())                  # Despliega: HOLA MUNDO (Quita los espacios de la
                                        izquierda)
```

rstrip()

Devuelve una copia de la cadena con el espacio al final suprimido.

```
Cadena = "HOLA MUNDO     "
print(Cadena.rstrip())                # Despliega: HOLA MUNDO (Quita los espacios de la
                                      derecha)
```

strip()

Devuelve una copia de la cadena con el espacio del inicio y final suprimido.

```
Cadena = "    HOLA MUNDO     "
print(Cadena.strip())                 # Despliega: HOLA MUNDO (Quita los espacios de la
                                      izquierda y derecha)
```

rjust(width)

Devuelve la cadena justificada a la derecha en una cadena de longitud width. Se rellena la cadena con espacios. Se devuelve la cadena original si width es menor que len(s).

```
Cadena = "HOLA MUNDO"
print(Cadena.rjust(15))               # Despliega: .....HOLA MUNDO[1]
```

ljust(width)

Devuelve la cadena justificada a la izquierda en una cadena de longitud width. Se rellena la cadena con espacios. Se devuelve la cadena original si width es menor que len(s).

```
Cadena = "HOLA MUNDO"
print(Cadena.ljust(15))               # Despliega: HOLA MUNDO.....[2]
```

center(width)

Devuelve la cadena centrada en una cadena de longitud width. Se rellena con espacios.

```
Cadena = "HOLA MUNDO"
print(Cadena.center(20))              # Despliega: .....HOLA MUNDO.....
```

4 Otros.

Función len()

Devuelve la cantidad de caracteres que tiene una cadena de caracteres. Cuando el argumento corresponde a un vector, entrega la cantidad de elementos que el vector tiene.

---

[1] Cada punto no forma parte del resultado, solo señala un espacio en blanco.
[2] Cada punto no forma parte del resultado, solo señala un espacio en blanco.

```
Texto = "HOLA MUNDO"
print(len(Texto))                    # Despliega: 10
```

Función str()

Devuelve valor del argumento convertido a cadena de caracteres.

```
Numero = 123
print(str(Numero))                   # Despliega: "123" (sin las comillas)
```

Función int()

Devuelve el valor del argumento convertido en número entero.

```
Texto = "123"
print(int(Texto))                    # Despliega: 123
```