Clase 5 Cadenas I

TEMAS: Cadenas de Caracteres. Operaciones. Métodos. Conversión de nros a cadenas

Una cadenas de caracteres (string) es un conjunto de 0 o mas caracteres.

Constantes de cadena

- Las constantes de cadena de caracteres pueden encerrarse indistintamente entre comillas simples o dobles.

```
cad1 = "Lunes"
cad2 = "Martes"
```

- Esto permite utilizar el otro tipo de comillas dentro de la cadena.

```
marca = "Mc.Donald's"
frase = 'Dijo "Ya voy" y partio'
```

- Una constante de cadena extensa puede ser distribuida en varias líneas, usando una barra invertida como continuación.

```
zen = " Hermoso es mejor que feo. \
      Explicito es mejor que implicito"
```

```
print(zen)
```

- Si se utilizan tres juegos de comillas la cadena retiene el formato dado por el programador y puede extenderse por multiples líneas.

```
 despacito = '''Tu , tu eres el iman y yo soy el metal
Me voy acercando y voy armando el plan
Solo con pensarlo se acelera el pulso
Despacito!!!'''
print(despacito)
```

Impression de cadenas

- Para imprimir cadenas de caracteres mediante el operador % se utiliza el especificador de conversion **%s**.
- Si se le agrega van después de la cadena. Para poner antes se debe usar anchos negativos = contraria a lo predeterminado que seria a la izq para pasar a la derecha

```
nombre = input("Ingrese su nombre: ")
edad = int(input("Ingrese su edad: "))
print("%s tiene %d años" %(nombre,edad))
```

- Tambien puede escribirse el ancho deseado entre el signo % y la letra s.

Secuencias de escape

▼ \n: Salto de Linea → más usada. **es un salto de linea.** se quiebra si continua en el renglón siguiente

```
print("Lunes\nMartes\nMier
coles")
            #Lunes
```

▼ \' : Comilla simple → agregar a los str

```
print("Lunes\'Martes\'Mier
coles")
#Lunes'Martes'Miercoles
```

▼ \" : Comilla doble → agregar a los str

```
                #Martes
                #Miercoles
```

▼ \t : Tabulador → espacios

```
print("\tLunes\tMartes\tMi
ercoles\t")
#Lunes   Martes   Miercoles
```

▼ \\ : Barra invertida → #

```
print("\\Lunes\\Martes\\Mi
ercoles\\")
#\Lunes\Martes\Miercoles\
```

```
print("\"Lunes\"Martes\"Mi
ercoles\"")
#"Lunes"Martes"Miercoles"
```

▼ \a : Campanilla → no debería provocar ninguna salida impresa. hace un sonido predeterminado

```
print("\aLunes\tMartes\tMi
ercoles\t")
#print normal y ruidito de
windows
```

Cadenas crudas

- Una **cadena cruda** (raw) se crea escribiendo la letra **r** antes de abrir las comillas.
- En una cadena cruda no se interpretan las secuencias de escape.

```
ruta = r"c:\nuevo\datos.txt"
#hay que indicar a python que no es un salto de linea ---> r
\\\\   c:\\
```

Similitudes Listas y Cadenas → soportan rebanadas excepto rebanada nula

> secuencia → tiene orden \-\ iterable → no tiene orden

- Ambas son secuencias de elementos , es decir iterables que pueden ser recorridos con un for.

- Se pueden acceder a cada elemento a traves de un subindices **(base 0)**.

```
cad = "Lunes"
print(cad[2]) #n
```

- Pueden manipularse mediante rebanadas.

```
cad = "Lunes"
subcad = cad[1:3]
print(subcad) #un
```

- Pueden ser concatenadas con el operador +.

```
nom = input("Nombre?")
saludo = "Bienvenido" + nom
print(saludo)
```

- Pueden replicarse mediante el operador *.

```
risa = "ja"
print(risa*5) #jajajajaja
separador = "-"*80
```

- Comparten muchas funciones y métodos:

```
cad = "Lunes"
len(): print(len(cad)) #5
in: if "es" in cad:
not in: if "es" not in cad
max():print(max(cad))#u  comparación alfabética
min():print(min(cad))#L  mayúsculas son las más chicas
index():print(cad.index("es"))#3
count():print(cad.count("es"))#1
```

Diferencias

- Las listas son mutables, pero las cadenas son **inmutables**.
- Significa que no pueden ser modificadas.

```
"""Para invertir una cadena s
e puede utilizar la tunica de
las rebanadas."""
```

- Las funciones y métodos para manipularlas devuelven una nueva cadena , sin alterar la original.
- Debido a que son inmutables, las cadenas carecen del método *reverse* de las listas.

```
frase = input("Ingrese una fr
ase :")
nueva = frase[::-1]
print(nueva)
if nueva != frase:
    print("no es capicua")
else:
    print("si es ")
```

EJEMPLO 1

Dibujar el borde de un cuadrado con letra X, ingresando por teclado la longitud de cada lado.

```
n = int(input("Longitud? "))

while n < 3:
    print("ERROR!")
    n = int(input("Longitud? "))
print()
print("X" * n)
for i in range(n-2):
    print("X" + "   "*(n-2) +
"X")
print("X" * n)
```

Ejemplo de ejecución:

```
Longitud del lado? (mínimo 3): 5

XXXXX
X   X
X   X
X   X
XXXXX
```

Comparación

- Las cadenas de caracteres pueden ser comparadas entre si igual que cualquier otra variable.
- Se utiliza comparación alfabética, es decir que las palabras que se clasifican como en un diccionario.

Conversion

- Un numero no puede ser concatenado a una cadena en forma directa.
- Antes es necesario convertirlo a cadena , lo que se logra con la función **str()**.

```
c = 75
mensaje = "Precio: " +str(c)
+ "dolares"
print(mensaje) #Precio: 75 do
lares
```

- Asi como existe la función str() para convertir números en cadenas, existen las funciones int() y float() que permiten efectuar la conversion en sentido contrario.
- Si la cadena no contiene un numero valido se producirá un error.

```
int(): convierte una cadena a
entero
dato = "25"
n = int(dato)
float(): convierte una cadena
a real
valor = "3.1416"
pi = float(valor)

input(): solo para ingresar t
exto
```

EJEMPLO 2

reemplazar las vocales con tilde en una cadena de caracteres por sus equivalente sin tilde

```
contilde = "áéíóúÁÉÍÓÚ" #las minusculas tienene código más ch
ico qué las mayúsculas
sintilde = "aeiouAEIOU"
frase = input("Ingrese una frase:")
nueva = ""
for caracter in frase:
    if caracter in contilde:
    posicion = contilde.index(caracter)
    nueva = nueva + sintilde[posicion]
    else:
```

```
        nueva = nueva + caracter
print(nueva)
```

Métodos para cadenas

▼ <str>.split([<sep>]): Divide <str> en una lista de cadena, buscando <sep> como separador. Si <sep> se omite se asumen los espacios.

- DEVUELVE UNA **LISTA** DE STRINGS

```
frase = "Hola Mundo"
lista = frase.split() #['Hola','Mundo']
print(lista)
fecha = "21/9/2020"
d,m,a = fecha.split('/')
print(d)
print(m)
print(a)
cad = "Hola Mundo" # 3 espacios
lista1 = cad.split()
print(lista1)
lista2 = cad.split(" ")
print(lista2)

#consola
['Hola', 'Mundo']
21
9
2020
['Hola', 'Mundo']
['Hola', 'Mundo']
```

▼ <sep>.join(<secuencia>): Devuelve un string con los elementos de <secuencia> separados por <sep>. La secuencia puede ser cadena o una lista de cadenas.

```
dias = ["lunes","martes","jueves"]
cad = '-'.join(dias)
print(cad) #lunes - martes - jueves
```

▼ <str>.replace(<vie>,<nue>[,<max>]): Devuelve una cadena nueva reemplazando todas las apariciones de <vie> por <nuevo> , hasta un máximo de <max> reemplazarlos. Este ultimo parámetro opcional. Si se omite se reemplazaran todas las apariciones. No da error si <vie> no existe.

- max es opcional si lo ponemos funciona como un tope. genera un nuevo str a partir del anterior

```
#Ejemplo del metodo replace( )
a = "Hoy es un dia frio, que frio esta!"
b = a.replace("frio" , "humedo")
print(b) #Hoy en un dia humedo, que humedo esta!
c = a.replace("frio","humedo",1)
print(c) #Hoy es un dia humedo, que frio esta!
```

EJEMPLO 3

→ Leer una frase, invertir las palabras que contiene (sin invertir las letras y mostrar la cadena invertida por pantalla.

! PRESTA ATENCIÓN A LAS CONSIGNAS

```
frase = input("Ingrese una frase :")
palabras = frase.split() #Dividimos la frase en una lista de
```

```
palabras
palabras.reverse() #Invertimos la lista con reverse()
frase = " ".join(palabras) #Construimos la cadena a partir de
la lista
print(frase)

#Tambien se puede usar un ciclo para concatenar
frase2 = ""
for palabra in palabras:
    frase2 = frase2+palabra+""
print(frase2)
```

Copia de cadenas

- Copiar una cadena siempre copia su referencia, su dirección de memoria sin importar la técnica que se utilice.
- Esto se debe a que las cadenas son inmutables, por lo que crear copias idénticas solo desperdiciaría memoria

```
a = "Hola"
b = a
c = a[:]
d = str(a)                    #121223123
e = a+""                      #121223123
for i in map(id,[a,b,c,d,e]): #121223123
    print(i)                  #121223123
```