PY0101ES-1-2-Cadenas

May 18, 2022

Operaciones con Cadenas

¡Bienvenido! En este cuaderno aprenderás sobre operaciones con cadenas en el Lenguaje de Programación Python. Al finalizar, conocerás acerca de las operaciones con cadenas que se pueden realizar en Python como son, indexación, operaciones y secuencias de escape.

Tabla de Contenido

```
<ul>
    <li>
        <a href="https://#strings">¿Qué es una cadena?</a>
    </li>
    <li>
        <a href="https://#index">Indexación</a>
        <ul>
            <li><a href="https://neg/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_conte
            <li><a href="https://slice/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_con
            <li><a href="https://stride/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_co
            <li><a href="https://concat/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_co
        </ul>
    </li>
    <li>
        <a href="https://#escape">Secuencias de Escape</a>
    </li>
    <li>
        <a href="https://#operations">Operaciones con Cadenas</a>
    </li>
    <li>
        <a href="https://#quiz">Exámen sobre Cadenas</a>
    </li>
</ul>
<p>
    Tiempo Estimado: <strong>15 min</strong>
</p>
```

¿Qué es una cadena?

A continuación, se muestra una cadena entre comillas dobles.

[ ]:
```
# Usa las comillas dobles para definir una cadena

"Michael Jackson"
```

También podemos usar comillas simples:

[ ]:
```
# Usa comillas simples para definir una cadena

'Michael Jackson'
```

Una cadena puede ser una combinación de espacios y dígitos:

[ ]:
```
# Dígitos y espacios en una cadena

'1 2 3 4 5 6 '
```

Una cadena también puede estar formada por caracteres especiales :

[ ]:
```
# Caracteres especiales en una cadena

'@#2_#]&*^%$'
```

Podemos imprimir en pantallas una cadena usando la sentencia print:

[ ]:
```
# Imprimir la cadea

print("hello!")
```

Podemos asignar una cadena como valor de una variable:

[ ]:
```
# Asignar una cadena a una variable

Name = "Michael Jackson"
Name
```

Indexación

Debemos pensar en una cadena como una secuencia ordenada. Cada elemento de la secuencia puede ser accedido utilizando un índice numérico, el cual representa el lugar que ocupa dicho elemento en la cadena:

Se puede acceder al primer índice de la siguiente manera:

[3]:
```
# Imprime el primer elemento de la cadena
Name="MIchael Jackson"
print(Name[0])
```

```
M
```

Podemos acceder el indice 6:

[ ]:
```
# Imprime el elemento en el índice 6 de la cadena

print(Name[6])
```

También, podemos acceder al treceavo elemento:

[ ]:
```
# Imprime el elemento en el treceavo índice de la cadena

print(Name[13])
```

Indexación Negativa

Con las cadenas tambien podemos usar la indexación negativa:

La indexación negativa nos sirve para numerar un elemento desde el final de una cadena.

El ultima elemento tiene el índice -1:

[6]:
```
# Imprime el último elemento de la cadena
Name="Adrian"
print(Name[-7])
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
/tmp/ipykernel_1781/3580024878.py in <module>
      1 # Imprime el último elemento de la cadena
      2 Name="Adrian"
----> 3 print(Name[-7])

IndexError: string index out of range
```

El primer elemento se obtiene usando como índice el -15:

[13]:
```
# Imprime el primer elemento de la cadena
len ("Adrián")
```

[13]: 6

Para conocer la cantidad de caracteres en un cadena usamos len, que es la abreviación de length, en inglés significa longitud:

[2]:
```
# Encuentra la longitud de la cadena

len("Michael Jackson")
```

[2]: 15

Slicing

Podemos obtener múltiples caracteres de una cadena usando el slicing, digamos desde primero hasta el cuarto y desde el octavo hasta el doceavo:

[3]:
```
# Asigna una seccion de la cadena dentro de la variable Name usando los indices␣
 ↪desde el 0 hasta el 3

Name[0:4]
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-3-b82e9f960225> in <module>
      1 # Take the slice on variable Name with only index 0 to index 3
      2
----> 3 Name[0:4]

NameError: name 'Name' is not defined
```

[ ]:
```
# Asigna una seccion de la cadena dentro de la variable Name usando los indices␣
 ↪desde el 8 hasta el 11

Name[8:12]
```

Stride

Para usar la técnica del stride debemos asignar un valor que a continuación se explica, si usamos el '2' por ejemplo, estaremos indicando que queremos seleccionar solamente cada segundo elemento de la cadena:

[ ]:
```
# Toma cada segundo elemento. Los elementos con índice 1, 3, 5 ...

Name[::2]
```

También Podemos combinar el slicing con el stride. Para este caso, seleccionamos los primeros cinco elementos y después aplicamos stride:

[15]:
```
# Toma cada segundo elemento desde el índice 0 al 4
Name="Adrian Nicolini"
Name[::2]
```

[15]:
```
'Ara ioii'
```

Concatenar Cadenas

Podemos concatenar o combinar cadenas usando el símbolo de suma, el resultado será una nueva cadena que surge de la combinación de ambas:

[17]:
```
# Concatenar dos cadenas

Statement = Name + " is the best"
```

```
Statement
```

[17]:
```
'Adrian Nicolini is the best'
```

Para replicar los valores en una cadena simplemente podemos multiplicarla por el numero de veces que deseemos. Para este caso, el numero de veces es tres. El resultado es una nueva cadena, la cual consiste en tres copias de la original:

[19]:
```
# Imprime la cadena 3 veces

3 * "Michael Jackson "
```

[19]:
```
'Michael Jackson Michael Jackson Michael Jackson '
```

Se puede crear una nueva cadena añadiendo otra a la original contenida en la variable. Concatenada con una nueva cadena, el resultado es otra diferente, Michael Jackson pasó a ser “Michael Jackson is the best”.

[20]:
```
# Concatenar cadenas

Name = "Michael Jackson"
Name = Name + " is the best"
Name
```

[20]:
```
'Michael Jackson is the best'
```

Secuencias de Escape

Una barra invertida representa el inicio de una secuencia de escape. Estas representan cadenas que pudieran ser difíciles de introducir. Por ejemplo, una barra invertida seguida de una “n” representa un salto de línea. En la salida de pantalla tendremos un salto de línea cada vez que una barra invertida y la letra “n” sea encontrada.

[21]:
```
# Secuencia de escape para un Salto de línea

print(" Michael Jackson \n is the best" )
```

```
 Michael Jackson
 is the best
```

De igual forma, barra invertida “t” aplicará un espacio de tabulación:

[22]:
```
# Secuencia de escape para un espacio de tabulación

print(" Michael Jackson \t is the best" )
```

```
 Michael Jackson         is the best
```

En caso de quieras que una barra invertida forme parte de una cadena, utiliza una doble barra invertida:

[23]:
```
# Incluir una barra invertida en una cadena

print(" Michael Jackson \\ is the best" )
```

```
 Michael Jackson \ is the best
```

Otra forma de hacer que se muestre la barra invertida de una cadena es colocando antes una “r”:

[ ]:
```
# La letra r le dirá a python que muestre la cadena tal cual e ignore lo que␣
 ↪haya despues de la barra invertida

print(r" Michael Jackson \ is the best" )
```

Operaciones con Cadenas

En Python existen muchos métodos para hacer operaciones con cadenas que pueden ser usados para manipular datos. Usaremos algunas operaciones básicas con cadenas.

Usemos ahora el método upper; este método convierte los caracteres de minúsculas a mayúsculas:

[24]:
```
# Convertir todos los caracteres de la cadena en mayúsculas

A = "Thriller is the sixth studio album"
print("before upper:", A)
B = A.upper()
print("After upper:", B)
```

```
before upper: Thriller is the sixth studio album
After upper: THRILLER IS THE SIXTH STUDIO ALBUM
```

El método replace reemplaza un segmento de la cadena, p.ej. una sub-cadena por una nueva. Introducimos primero la parte de la cadena que queremos cambiar. En el segundo argumento pondremos el segmento que reemplazara al anterior, el resultado será una nueva cadena con una de sus partes modificada:

[ ]:
```
# Reemplazar la vieja sub-cadena por una nueva

A = "Michael Jackson is the best"
B = A.replace('Michael', 'Janet')
B
```

La función del método find es encontrar una sub-cadena. El argumento del método es la sub-cadena que deseas encontrar, y el resultado será el numero del índice perteneciente al primer carácter de la sub-cadena. Podemos encontrar la sub-cadena jack o el.

[25]:
```
# Encontrar la sub-cadena. El resultado será unicamente el índice del primer␣
 ↪elemente de la sub-cadena dentro de la cadena principal.

Name = "Michael Jackson"
Name.find('el')
```

[25]: 5

[26]:
```
# Encotrar la sub-cadena dentro de la cadena.

Name.find('Jack')
```

[26]: 8

Si la sub-cadena no se encuentra en la cadena principal el resultado será -1. Por ejemplo, la cadena 'Jasdfasdasdf' no es una sub-cadena:

[ ]:
```
# Si no se encuentra la sub-cadena en la cadena principal

Name.find('Jasdfasdasdf')
```

Exámen sobre Cadenas

¿Cual es el valor de la variable A después de ejecutar el siguiente código?

[27]:
```
# Escribe abajo tu código y presiona Shift+Enter para ejecutarlo

A = "1"
#string
```

Haz doble click aquí para ver la solución.

¿Cual es el valor de la variable B después de ejecutar el siguiente código?

[28]:
```
# Escribe abajo tu código y presiona Shift+Enter para ejecutarlo

B = "2"
```

Haz doble click aquí para ver la solución.

¿Cual es el valor de la variable C después de ejecutar el siguiente código?

[ ]:
```
# Escribe abajo tu código y presiona Shift+Enter para ejecutarlo

C = A + B
```

Haz doble click aquí para ver la solución.

En la variable D haz uso del slicing para imprimir solo los primeros tres elementos:

[37]:
```
# Escribe abajo tu código y presiona Shift+Enter para ejecutarlo

D = "ABCDEFG"
print (D[0:3])
```

```
ABC
```

Haz doble click aquí para ver la solución.

Usa un valor de stride de 2 para imprimir cada segundo elemento de la cadena E:

[38]:
```
# Escribe abajo tu código y presiona Shift+Enter para ejecutarlo

E = 'clocrkr1e1c1t'
print(E[::2])
```

```
correct
```

Haz doble click aquí para ver la solución.

Imprime una barra invertida:

[42]:
```
# Escribe abajo tu código y presiona Shift+Enter para ejecutarlo
print ("\\")
```

```
\
```

Haz doble click aquí para ver la solución.

Convierte el mayúsculas el contenido de la variable F:

[47]:
```
# Escribe abajo tu código y presiona Shift+Enter para ejecutarlo

F = "You are wrong"
F.upper()
```

[47]:
```
'YOU ARE WRONG'
```

Haz doble click aquí para ver la solución.

En la variable G, encuentra el primer índice de la sub-cadena snow:

[55]:
```
# Escribe abajo tu código y presiona Shift+Enter para ejecutarlo

G = "Mary had a little lamb Little lamb, little lamb Mary had a little lamb \
Its fleece was white as snow And everywhere that Mary went Mary went, Mary went␣
 ↪\
Everywhere that Mary went The lamb was sure to go"
G.find ("snow")
```

[55]:
```
95
```

Haz doble click aquí para ver la solución.

En la variable G, reemplaza la sub-cadena Mary con Bob:

[58]:
```
# Escribe abajo tu código y presiona Shift+Enter para ejecutarlo
G.replace("Mary","Bob")
```

[58]:
```
'Bob had a little lamb Little lamb, little lamb Bob had a little lamb Its fleece
was white as snow And everywhere that Bob went Bob went, Bob went Everywhere
that Bob went The lamb was sure to go'
```

Haz doble click aquí para ver la solución.

¡El último ejercicio!

Felicidades, has completado tu primera lección y practica de laboratorio en Python. Sin embargo, hay algo mas que debes saber. La comunidad en Ciencia de Datos te alienta a compartir tu trabajo. La mejor forma de hacerlo es a través de GitHub. Al compartir tus cuadernos en GitHub, además de construir una reputación entre la comunidad de los científicos de datos, también te ayudará en el proceso de encontrar un trabajo. Incluso si este fuera tu primer material de trabajo, nunca es tarde para fomentar buenos hábitos. Por favor lee y analiza este artículo para aprender a compartir tu trabajo.

¡Obtén gratis IBM Watson Studio!

```
<p><a href="https://cocl.us/PY0101EN_edx_add_bbottom?utm_medium=Exinfluencer&utm_source=Exinflu
```

Acerca de los Autores:

Joseph Santarcangelo es un Científico de Datos en IBM, además posee un doctorado en Ingeniería Eléctrica. Su trabajo de investigación se centra en el uso de Aprendizaje Automático (Machine Learning), Procesamiento de Señales y Visión Artificial para determinar el impacto de los videos en el proceso cognitivo. Joseph trabaja en IBM desde la terminación de su doctorado.

Otros colaboradores: Mavis Zhou