LAB-009-Exemplar_Work with strings in Python

September 20, 2024

Modelo: Trabajar con cadenas en Python

Introducción

Los analistas de seguridad trabajan con muchos datos de cadena. Por ejemplo, algunos analistas de seguridad trabajan en la creación y actualización de identificaciones, como las identificaciones de empleados y dispositivos, que comúnmente se representan como cadenas. Como otro ejemplo, una actividad determinada de red se almacenará como datos de cadena. Para un analista de seguridad, es fundamental trabajar cómodamente con cadenas en Python.

En este laboratorio, practicarás la creación del código Python y el trabajo con cadenas. Trabajarás con un ID de empleado, un ID de dispositivo y una URL, todos representados como datos de cadena.

Consejos para completar este laboratorio

Mientras recorres este laboratorio, ten en cuenta los siguientes consejos:

— `#### YOUR CODE HERE ###` indica dónde debes escribir el código. Asegúrate de reemplazarlo con tu propio código antes de ejecutar la celda de código. — Siéntete libre de abrir las pistas para obtener información adicional a medida que trabajas en cada tarea. — Para ingresar tu respuesta a una pregunta, haz doble clic en la celda de markdown para editar. Asegúrate de reemplazar "[Haz doble clic para ingresar aquí tus respuestas.]" con tu propia respuesta. — Puedes guardar tu trabajo manualmente con un clic en Archivo y luego en Guardar en la barra de menú en la parte superior del cuaderno. — Puedes descargar tu trabajo localmente con un clic en Archivo y luego en Descargar, luego puedes especificar el formato de archivo que prefieras en la barra de menú en la parte superior del cuaderno.

0.1 Situación hipotética

Trabajas como analista de seguridad y eres responsable de escribir programas en Python para automatizar la actualización de las ID de los empleados, extraer caracteres de una ID de dispositivo y extraer componentes de una URL.

Tarea 1

En tu organización, las ID de los empleados tienen actualmente cuatro o cinco dígitos de longitud. En esta tarea, se te proporciona una ID de empleado numérica de cuatro dígitos almacenada en una variable llamada `employee_id`. Conviértela a un formato de cadena y almacena el resultado en la misma variable. Más adelante, actualizarás esta cadena de ID de empleado para que cumpla con un nuevo formato estandarizado.

Completa el siguiente código. Asegúrate de reemplazar el fragmento `### YOUR CODE HERE ###` con tu propio código antes de ejecutar la siguiente celda.

[1]:
```
# Asigna a `employee_id` un número de cuatro dígitos como valor inicial

employee_id = 4186

# Muestra el tipo de datos de `employee_id`

print(type(employee_id))

# Reasigna a `employee_id` el mismo valor pero en forma de cadena

employee_id = str(employee_id)

# Muestra el tipo de datos de `employee_id` ahora

print(type(employee_id))
```

```
<class 'int'>
<class 'str' >
```

Pista 1

Utiliza la función `str()` en Python para convertir el valor inicial de la variable `employee_id` en una cadena.

Pista 2

Pasa `employee_id` como argumento a la función `str()`.

Pregunta 1 ¿Qué observas sobre el tipo de datos de `employee_id` la primera vez que se muestra? ¿Qué observas sobre el tipo de datos de `employee_id` la segunda vez que se muestra (después de que se reasigna el valor a la variable)?

La primera vez que se muestra, el tipo de datos de `employee_id` es un número entero. La segunda vez que se muestra (después de que se reasigna el valor a la variable), el tipo de datos de `employee_id` es una cadena.

0.2 Tarea 2

Imagina que te acaban de informar de un nuevo criterio para las identificaciones de los empleados. Todas ellas deben tener cinco dígitos por motivos de normalización.

En esta tarea, escribirás una sentencia condicional que muestre un mensaje si la ID del empleado tiene menos de cinco dígitos.

Asegúrate de reemplazar el fragmento `### YOUR CODE HERE ###` con tu propio código antes de ejecutar la siguiente celda.

[2]:
```
# Asigna a `employee_id` un número de cuatro dígitos como valor inicial

employee_id = 4186

# Reasigna a `employee_id` el mismo valor pero en forma de cadena

employee_id = str(employee_id)

# Sentencia condicional que muestra un mensaje si la longitud de la ID del␣
↪empleado es inferior a 5 dígitos

if len(employee_id) < 5:
    print("Esta ID de empleado tiene menos de cinco dígitos. No cumple con los␣
↪requisitos de longitud.")
```

```
Esta ID del empleado tiene menos de cinco dígitos. No cumple con los requisitos
de longitud.
```

Pista 1

La función `len()` en Python se puede usar para obtener la longitud de `employee_id`.

Pista 2

Inicia la sentencia condicional con la palabra clave `if`.

Pista 3

Para escribir la condición en la sentencia condicional, utiliza el operador de comparación `<` para verificar si la longitud de `employee_id` es menor que `5`. Asegúrate de colocar la condición entre el `if` y los dos puntos `:`.

Tarea 3

En esta tarea, te basarás en el código anterior. Si una ID de empleado tiene solo cuatro dígitos, usarás la concatenación para crear un número de ID de empleado de cinco dígitos.

La concatenación es un proceso que te permite fusionar cadenas. El operador de suma (`+`) en Python te permite concatenar dos cadenas.

Escribe una sentencia `if` que evalúe si la longitud de `employee_id` es menor que `5`. Cuando la condición se evalúa a `True`, reasigna el valor de la variable `employee_id` concatenando `"E"` frente a la ID de empleado de cuatro dígitos para crear una con cinco caracteres. Luego, vuelve a mostrar `employee_id`. Asegúrate de reemplazar cada `### YOUR CODE HERE ###` con tu propio código antes de ejecutar la siguiente celda.

[2]:
```
# Asigna a `employee_id` un número de cuatro dígitos como valor inicial

employee_id = 4186

# Reasigna a `employee_id` el mismo valor pero en forma de cadena
```

```
employee_id = str(employee_id)

# Muestra el `employee_id` en su formato actual

print(employee_id)

# Sentencia condicional que actualiza el `employee_id` si su longitud es␣
↪inferior a 5 dígitos

if len(employee_id) < 5:
    employee_id = "E" + employee_id

# Muestra `employee_id` después de la actualización

print(employee_id)
```

```
4186
E4186
```

Pista 1

Para completar el encabezado de la sentencia condicional, utiliza la palabra clave `if` para comenzar, la función `len()` para obtener la longitud de la `employee_id` y el operador de comparación `<` para verificar si tiene una longitud inferior a `5`. Asegúrate de escribir esto antes de los dos puntos `:`.

Pista 2

Utiliza el operador de asignación `=` para actualizar el valor de la variable `employee_id`. Actualiza el valor de `employee_id` a la concatenación de `"E"` con el valor actual de la variable. La `"E"` debe aparecer a la izquierda del valor actual.

Pista 3

Utiliza la función `print()` para mostrar `employee_id` después de la actualización.

0.3 Tarea 4

Ahora pasarás a la siguiente parte de tu tarea. Imagina que los caracteres en una ID de dispositivo transmiten información técnica sobre este. Tendrás que extraer caracteres en posiciones específicas de la ID del dispositivo. Comienza por extraer el cuarto carácter.

La variable `device_id` representa una ID de dispositivo que contiene caracteres alfanuméricos, y ya está almacenada como cadena.

Asegúrate de reemplazar el fragmento `### YOUR CODE HERE ###` con tu propio código antes de ejecutar la siguiente celda.

[4]:
```
# Asigna a `device_id` una cadena que contiene caracteres alfanuméricos

device_id = "r262c36"
```

```
# Extrae el cuarto carácter de `device_id` y muéstralo

print(device_id[3])
```

2

Pista 1

Utiliza un par de corchetes, pasando el valor de indexación apropiado, para extraer el cuarto carácter de `device_id`.

Pista 2

En Python, los valores de indexación comienzan en 0.

Pista 3

Dado que los valores de indexación comienzan en 0 en Python, un valor de indexación de 3 corresponde al cuarto carácter de una secuencia.

Tarea 5 Ahora también tendrás que extraer del primer al tercer carácter de la ID del dispositivo. Toma un fragmento de la ID del dispositivo. Puedes lograrlo mediante la notación entre corchetes en Python. Luego, muestra el fragmento para examinar el resultado.

Asegúrate de reemplazar el fragmento `### YOUR CODE HERE ###` con tu propio código antes de ejecutar la siguiente celda.

[5]:
```
# Asigna a `device_id` una cadena que contiene caracteres alfanuméricos

device_id = "r262c36"

# Extrae del primer al tercer carácter de `device_id` y muestra el resultado

device_id[0:3]
```

[5]: 'r26'

Pista 1

Utiliza un par de corchetes, pasando los valores de indexación apropiados, para extraer del primer al tercer carácter de `device_id`.

Dentro de los corchetes, utiliza un `:` para separar el primer valor de indexación (el valor de indexación inicial) y el segundo valor de indexación (el valor de indexación final).

Pista 2

Ten en cuenta que el segundo valor de la indexación que se pasa a la notación entre corchetes es exclusivo.

Pista 3

Dado que el segundo valor de indexación que se pasa a la notación entre corchetes es exclusivo y la indexación en Python comienza en 0, el primer valor debe ser 0 y el segundo debe ser 3, para extraer del primer al tercer carácter de `device_id`.

0.4 Tarea 6

Ahora pasarás a la última parte de tu tarea. Esto implica extraer componentes de una URL.

Trabajarás con indexaciones de cadena para mostrar varios componentes de una URL que se almacena en la variable URL. Primero, extraerás y mostrarás el protocolo de la URL y los caracteres `://` que la siguen mediante la segmentación de cadenas. Ten en cuenta que el protocolo está en el formato seguro de `https` al determinar las indexaciones para tu segmento.

Asegúrate de reemplazar el fragmento `### YOUR CODE HERE ###` con tu propio código antes de ejecutar la siguiente celda.

[6]:
```
# Asigna a `url` una URL específica

url = "https://exampleURL1.com"

# Extrae el protocolo de `url` junto con la sintaxis que lo sigue, muestra el␣
 ↪resultado

print(url[0:8])
```

```
https://
```

Pista 1

Ten en cuenta que `https://` tiene ocho caracteres.

Pista 2

Utiliza un par de corchetes para segmentar la cadena almacenada en `url`, pasando dos valores de indexación separados por `:`. Ten en cuenta que el segundo valor de la indexación es exclusivo y que la indexación en Python comienza en 0.

Pista 3

Utiliza la función `print()` para mostrar el segmento.

0.5 Tarea 7

Más adelante en este laboratorio, extraerás la extensión del dominio. Para prepararte, utiliza el método `.index()` para identificar la indexación en que se encuentra la extensión de dominio `.com` en la URL dada.

Asegúrate de reemplazar el fragmento `### YOUR CODE HERE ###` con tu propio código antes de ejecutar la siguiente celda.

[7]:
```
# Asigna a `url` una URL específica

url = "https://exampleURL1.com"

# Muestra la indexación en que se encuentra la extensión de dominio ".com" en␣
 ↪`url`

print(url.index(".com"))
```

```
19
```

Pista 1

Aplica el método `.index()` a `url` para obtener la indexación apropiada. El método `.index()` toma una subcadena, y si esa subcadena se encuentra en la cadena original, devuelve la indexación en la que dicha subcadena comienza a aparecer en la cadena original.

Pista 2

Llama a `url.index()`, y dentro de los paréntesis, pasa la extensión del dominio objetivo como una cadena.

0.6 Tarea 8

Es una buena idea guardar datos importantes en variables al programar. Esto permite un seguimiento rápido y fácil, además de la reutilización de la información.

Almacena la salida del método `.index()` en una variable llamada `ind`, que es la abreviatura de indexación. Esta indexación representa la posición en la que la extensión de dominio `".com"` comienza en la `url`. Asegúrate de reemplazar el fragmento `### YOUR CODE HERE ###` con tu propio código antes de ejecutar la siguiente celda. Ten en cuenta que la ejecución de esta celda no producirá ningún resultado.

[8]:
```
# Asigna a `url` una URL específica

url = "https://exampleURL1.com"

# Asigna a `ind` el resultado de aplicar `.index()` a `url` para extraer la␣
 ↪indexación inicial de ".com" en `url`

ind = url.index(".com")
```

Pista 1

Para asignar la indexación extraída a la variable `ind`, utiliza el método `.index()` a la derecha del operador de asignación `=`.

Pista 2

Llama a `url.index()`, y dentro de los paréntesis, pasa la extensión de dominio objetivo como una cadena. Asegúrate de colocar esto a la derecha del operador de asignación `=`, para que el resultado

se asigne a `ind`.

0.7 Tarea 9

Puedes utilizar la segmentación de cadenas para extraer también la extensión del dominio de una URL. Para hacerlo, puedes crear un segmento. La indexación inicial debe ser la variable `ind`. Esta contiene la indexación en la que comienza la extensión de dominio. La indexación final debe ser `ind + 4` (ya que `".com"` tiene cuatro caracteres). A veces, como en esta situación, es más fácil expresar la indexación final en relación con la inicial. Examina el siguiente código, ejecútalo tal como está y observa el resultado.

[9]:
```
# Asigna a `url` una URL específica

url = "https://exampleURL1.com"

# Asigna a `ind` el resultado de aplicar `.index()` a `url` para extraer la␣
↪indexación inicial de ".com" en `url`

ind = url.index(".com")

# Extrae la extensión de dominio en `url` y muéstrala

print(url[ind:ind+4])
```

```
.com
```

Pregunta 2 ¿Qué genera este código y por qué?

Este código genera el nombre de dominio `.com` guardando primero la posición de inicio del nombre de dominio en la variable `ind` y luego extrayendo 4 caracteres consecutivos de `url` a partir de la posición guardada.

0.8 Tarea 10

Por último, extrae el nombre del sitio web de la URL dada mediante la segmentación de cadenas y la variable `ind` que definiste anteriormente. En la URL dada, el nombre del sitio web es `"exampleURL1"`. Asegúrate de reemplazar el fragmento `### YOUR CODE HERE ###` con tu propio código antes de ejecutar la siguiente celda.

[10]:
```
# Asigna a `url` una URL específica

url = "https://exampleURL1.com"

# Asigna a `ind` el resultado de aplicar `.index()` a `url` para extraer la␣
↪indexación inicial de ".com" en `url`
```

```
ind = url.index(".com")

# Extrae el nombre del sitio web en `url` y muéstralo

print(url[8:ind])
```

```
exampleURL1
```

Pista 1

A fin de extraer el nombre del sitio web en la URL dada, utiliza un par de corchetes para crear un segmento de `url`, pasando una indexación de inicio y una de finalización, separándolas con `:`.

Pista 2

La indexación inicial debe establecerse en 8, ya que esta es la posición después del protocolo y donde termina la sintaxis `://` y comienza el nombre del sitio web. La indexación final debe establecerse en la posición en la que comienza el nombre de dominio `.com`.

0.9 Conclusión

¿A qué conclusiones clave llegaste con este laboratorio? * Las cadenas son fundamentales para almacenar datos importantes relacionados con la seguridad, como ID de dispositivos y URL. * La concatenación de cadenas te permite combinar fácilmente la información de una cadena con la información almacenada en otra. * La segmentación de cadenas es una técnica poderosa que te permite extraer cualquier subsección de una cadena. * Python tiene muchas funciones y métodos que ayudan a los analistas a trabajar con valores de cadena, así como datos que desean convertir en un formato de cadena. * La función `type()` devuelve el tipo de datos de su entrada. * La función `str()` convierte el objeto de entrada en una cadena. Por ejemplo, cuando se llama a un número entero, `str()` devuelve ese valor entero convertido en una cadena. * La función `len()` devuelve el número de elementos en un objeto. Cuando se llama a una cadena, `len()` devuelve el número de caracteres en esa cadena. * El método `.index()` encuentra la primera ocurrencia de la entrada en una cadena y devuelve su ubicación. Proporciona la indexación en la que comienza la subcadena.