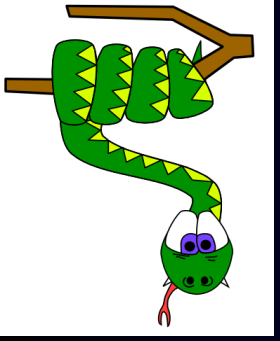

Programación en Python

Sesión 3

Cadenas en Python

Víctor Melchor

Cadenas en Python

1.1 Introducción a Cadenas

1.2 Operaciones con Cadenas

Cadenas en Python

1.1 Introducción a Cadenas

1.2 Operaciones con Cadenas

1.3 Métodos de Cadenas

Introducción a Cadenas

Las cadenas son una colección de caracteres encerradas dentro de las comillas.

Los siguientes son ejemplos de cadenas...

```
"Ana estuvo aquí"
"Ana estuvo aquí el viernes 2 de Agosto de 2019"
"9396633"
"A"
"7"
```

Las cadenas están compuestas por caracteres

Las cadenas son una colección de caracteres encerradas dentro de las comillas.

Cada carácter tiene un código numérico y pertenece a un conjunto de caracteres conocido como el conjunto de caracteres Unicode.

Strings y Caracteres Numéricos

En los primeros días de las computadoras, cada fabricante usaba su propia codificación de números para los caracteres.

- El sistema ASCII (Código estándar americano para el intercambio de información) utiliza 127 códigos de bits. Python es compatible con Unicode (más de 100,000 caracteres)

El conjunto de caracteres Unicode

El conjunto de caracteres Unicode incluye los caracteres A - Z, los números 0 - 9, los signos de puntuación, el carácter de espacio, así como muchos otros caracteres ...

¿Qué es Unicode?

La tabla de caracteres de Unicode

El conjunto de caracteres Unicode incluye el conjunto de caracteres del código ASCII.

El conjunto de caracteres del código ASCII

ASCII = American Standard Code for Information Interchange

Las cadenas están contenidas entre comillas

Las cadenas son una colección de caracteres encerradas dentro de las comillas.

Las cadenas pueden estar contenidas dentro de comillas simples, dobles o triples ...

Un programa de ejemplo que usa cadenas

Las cadenas pueden estar contenidas dentro de comillas simples, dobles o triples ...

```
'Ana estuvo aquí'

"9396633"

'''Ana estuvo aquí
   el viernes
   2 de Agosto de 2019'''
```

Impresión de comillas dobles dentro de una cadena

Si desea imprimir una comilla doble (") dentro de una cadena, delimitar la cadena en comillas simples (') ...

```
print('Aqui vemos una comilla doble ", y "mas" ')
```

Imprimiendo una cadena que contiene un apóstrofe

Si desea imprimir un apóstrofo (o una comilla simple) dentro de una cadena, delimite la cadena entre comillas dobles ...

```
print ("Este es el Ana's Bar")
```

Imprimiendo una cadena que contiene un apóstrofe

La función **ord** devuelve el código numérico (ordinal) de un solo carácter.

La función **chr** convierte un código numérico al carácter correspondiente.

```
>>> ord("A")
65
>>> ord("a")
97
>>> chr(97)
'a'
>>> chr(65)
'A'
```

Cadenas en Python

1.1 Introducción a Cadenas

1.2 Operaciones de Cadenas

1.3 Métodos de Cadenas

Los números pueden ser tratados como caracteres

Dentro de una cadena, (es decir, contenida entre comillas), un número se trata como un carácter.

No se puede realizar aritmética regular en números almacenados como caracteres ...

Repetir (multiplicar) cadenas utilizando *

No se puede realizar aritmética regular en números almacenados como caracteres, pero ...

puede "multiplicar" (repetir) cadenas:

`"3" * 4` resulta `"3333"`

Las cadenas se pueden unir usando el símbolo +

Unir cadenas mediante el símbolo + es un proceso conocido como concatenación.

```
"Ana " + "estuvo " + ("aqui " * 3)
```

resulta en `"Ana estuvo aqui aqui aqui "`

Indexación de Cadenas utilizando el operador []

Se puede acceder a cualquier elemento (carácter) de una cadena mediante la indexación.

```
s1 = "Victor Melchor"
print (s1[0], s1[7])
```

se muestra `V M`

Extraer subcadenas usando el operador []

Cualquier subcadena de una cadena puede obtenerse usando el operador [].

```
s1 = "Victor Melchor"
print (s1[0:1],s1[7:9])
print (s1[8:11])
```

imprime

```
V Me
elc
```

Encontrar la longitud de una cadena usando `len`

La longitud de cualquier cadena se puede determinar utilizando el método len.

```
s1 = "Victor"
s2 = "Melchor"
s3 = ""
print (len(s1), end=" ")
print (len(s2), end=" ")
print (len(s3))
```

imprime `6 7 0`

Impresión de cadenas y números

Podemos imprimir valores de cadena y valores numéricos desde la misma declaración de impresión separándolos con comas:

```
d = 10
c = 75
print ('Total es: ', d, 'soles y', c, ' centimos ')
```

```
>>>
 Total es:  10 soles y 75  centimos
```

Los números se pueden imprimir desde una sola cadena usando un método especial conocido como **formato de cadena.**

Formato de cadena y %

En el formato de cadena, usamos el símbolo%.

El operador % también se puede usar para un propósito diferente como operador de módulo (encontrar el resto después de una división entera).

Se dice que el símbolo % está sobrecargado.

Formato de cadena y %

Un operador sobrecargado se comporta de manera diferente dependiendo del contexto.

En el siguiente ejemplo, vemos que el operador% se usa para especificar cómo se debe imprimir una cadena (es decir, el formato de cadena).

Imprimir enteros dentro de una cadena

```
x = 20
y = 75
print ('La suma de %d y %d es %d' % (x, y, x + y))
```

```
>>>
La suma de 20 y 75 es 95
```

Los tres códigos de formato% d representan donde deben imprimirse los enteros decimales que se muestran entre paréntesis después del símbolo%.

Imprimir reales dentro de una cadena

```
x = 20.512
y = 15.269
print ('La suma de %f y %f es %f' % (x, y, x + y))

>>>
La suma de 20.512000 y 15.269000 es 35.781000
```

Los tres % f representan donde deben imprimirse los valores flotantes mostrados entre paréntesis después del símbolo%.

Especificando el número de decimales

```
x = 20.512
y = 15.269
print ('La suma de %0.2f y %0.2f es %0.2f' % (x, y, x + y))
```

```
>>>
La suma de 20.51 y 15.27 es 35.78
```

Los tres% 0.2f representan donde deben imprimirse los valores flotantes (con 2 decimales) mostrados entre paréntesis después del símbolo%.

Códigos de formato de cadena

% d y% f son solo dos de un conjunto de códigos de formato disponibles para el formato de cadena

Código de formato de cadena `%s`

`%s` es el código de formato que representa una cadena.

```
print ('Python es un %s lenguaje.' % 'gran')
```

```
>>>
Python es un gran lenguaje.
```

La secuencia %s representa donde se debe imprimir el valor de la cadena (que se muestra después del símbolo %).

Secuencias de escape

Una secuencia de escape es una barra invertida (\) seguida de uno o más caracteres, que se inserta en una cadena para realizar una tarea especial. Por ejemplo \ n genera una nueva línea y \ t genera una tabulación dentro de la cadena. . .

Una cadena contiene secuencias de escape

El siguiente ejemplo muestra una cadena con dos secuencias de escape:

```
s = 'uno\ndos\ttres'
print (s)
```

La salida:

```
>>>
uno
dos     tres
```

Una secuencia de escape cuenta como un caracter

El siguiente ejemplo muestra que cada secuencia de escape cuenta como un carácter:

```
s = 'uno\ndos\ttres'
print (s)
print (len(s))
```

The output:

```
>>>
uno
dos     tres
12
```

Iteración de cadena y membresía

El siguiente ejemplo muestra que puede iterar sobre cadenas en bucles usando declaraciones **for** y probar la membresía con el operador **in**:

```
s = 'Ana estuvo aqui'
for c in s:
    print(c, end=" ")
print ('n' in s, end=" ")
print (' ' in s, end=" ")
print ('x' in s)
```

La salida:

```
>>>
A n a   e s t u v o   a q u i  True True False
```

Caracteres Unicode dentro de una cadena

El siguiente ejemplo muestra cómo insertar caracteres especiales en una cadena. Ejecuta el programa de ejemplo y asegúrate de leer los comentarios..

```
# Para ver los gráficos de caracteres, vaya a:
# http://www.unicode.org/charts/
# http://www.unicode.org/charts/PDF/U2580.pdf (Block Elements)

# \u2588 es un bloque completo que se puede usar para construir un
   cuadrado negro
str1 = u'Hola\u2588alli'   # bloque negro sólido dentro del texto
print (str1)
```

Cadenas en Python

1.1 Introducción a Cadenas
1.2 Operaciones con Cadenas
1.3 Métodos de Cadenas

Métodos de Cadenas

- Hay una serie de métodos de cadena:

- s.capitalize () - Copia toda la cadena s con solo el primer carácter en mayúscula.
- s.center (ancho) - Centra s en un campo de ancho dado.
- s.count(sub) – Cuenta el número de ocurrencias de sub en s.
- s.endswith(sufijo) – Retorna True si la cadena termina con el sufijo especificado; de lo contrario, devuelve False.
- s.find(sub) – Retorna la primera posición donde ocurre sub en s. Retorna -1 si no lo encuentra.

Encontrar una cadena dentro de una cadena

El método de cadena `find` se utiliza para localizar la posición de una cadena dentro de otra.

```
s1 = 'aeropuerto'
x = s1.find('pue')
print (x)
```

La salida:

```
>>>
4
```

Métodos de Cadenas

- Hay una serie de métodos de cadena:

- s.isalpha () - Devuelve True si la cadena s contiene solo letras.
- s.isdigit() - Devuelve True si la cadena s contiene solo dígitos.
- s.islower() – Devuelve True si la cadena s contiene solo letras minúsculas.
- s.isspace() – Devuelve True si solo hay caracteres de espacio en blanco en la cadena s y la cadena no está vacía, de lo contrario retorna False.
- s.isupper() – Devuelve True si la cadena s contiene solo letras mayúsculas.

Métodos de Cadenas

- Hay una serie de métodos de cadena:

- s.join(list) – Concatena la lista de cadenas en una cadena grande usando s como separador.
- s.ljust(width) – Como center, pero s está justificado a la izquierda.
- s.lower() – Retorna una copia de s con todos los caracteres convertidos a minúsculas.
- s.lstrip() – Copia de s con espacios en blanco iniciales eliminados

Métodos de Cadenas

- Hay una serie de métodos de cadena:

- s.replace(oldsub, newsub) – Retorna una copia de s donde se reemplaza las ocurrencias de oldsub en s con newsub.
- s.rfind(sub) – Como find, pero devuelve la posición más a la derecha.
- s.rjust(width) – Como center, pero s es justificado a la derecha.
- s.rstrip() – Copia de s con los espacios en blanco al final eliminados.

Reemplazar texto dentro de una cadena

El método de cadena `replace` se utiliza para reemplazar texto dentro de una cadena con texto nuevo.

```
s1 = 'caldo de gallina con presa'
s2 = s1.replace('con','sin')
print (s2)
```

La salida:

```
>>>
caldo de gallina sin presa
```

Métodos de Cadenas

- Hay una serie de métodos de cadena:

- s.split(w) – Divide s en una lista de subcadenas, delimitados por w.
- s.startswith(prefijo) – Retorna True si la cadena empieza con el prefijo especificado; de lo contrario, devuelve False.
- s.strip(w) – Retorna una copia de s con todos los caracteres iniciales y finales que aparecen en w eliminados.

Métodos de Cadenas

- Hay una serie de métodos de cadena:

- s.swapcase () - Devuelva una copia de la cadena con caracteres en mayúsculas convertidos a minúsculas y viceversa.
- s.title () - Copia de s; el primer carácter de cada palabra en mayúscula.
- s.upper() – Retorna una copia de s con todos los caracteres convertidos a mayúsculas

Strings

- La mayoría de los programas de Python del mundo real contienen cadenas.

Strings

- Las cadenas te permiten recopilar caracteres para que puedas tratarlos como un grupo.

Strings

- Las cadenas tienen ordenamiento posicional de izquierda a derecha, con capacidad de índice.

Strings

- A diferencia de las listas, las cadenas son inmutables, lo que significa que no se pueden cambiar. Pero los nuevos objetos de cadena se pueden crear a partir de objetos de cadena existentes.

Strings

- Los Strings son homogéneas , es decir, consisten solamente de caracteres.

Strings

- La mayoría de los programas de Python del mundo real contienen cadenas.
- Las cadenas te permiten recopilar caracteres para que puedas tratarlos como un grupo.
- Las cadenas tienen ordenamiento posicional de izquierda a derecha, con capacidad de índice.
- A diferencia de las listas, las cadenas son inmutables, lo que significa que no se pueden cambiar. Pero los nuevos objetos de cadena se pueden crear a partir de objetos de cadena existentes.
- Los Strings son homogéneas , es decir, consisten solamente de caracteres.