CEPSUNI

Dirección de Responsabilidad Social y Desarrollo Sostenible

PYTHON BÁSICO - NIVEL I

TANIA ALDORADIN QUINTANILLA

Clase 4: Cadenas

Objetivos

Al finalizar con éxito este curso, podrá diseñar, crear, modificar programas usando el lenguaje de programación Python vas a:

- Comenzar con la instalación de Python.
- Conocer los diferentes IDE de Python.
- Conocer las sentencias de control.
- Conocer las sentencias de repetición.
- Conocer los arreglos lineales (Listas, Tuplas y Diccionarios).
- Conocer las operaciones con conjuntos (Unión e intercepción).

Contenido

- ¿Qué es una cadena de carácteres?
- Uso de las funciones con las cadenas de carácteres
- Índices y operadores de cadenas de carácteres
- Iteración en una cadena de carácteres
- Ejercicios de aplicación.

¿Qué es una cadena de carácteres?

Las cadenas en Python o strings son un tipo inmutable que permite almacenar secuencias de caracteres. Para crear una, es necesario incluir el texto entre comillas dobles ". Si has estado expuesto antes a otro lenguaje de programación, sabrás que necesitas declarar o escribir variables antes de que puedas almacenar cualquier cosa en ellas. Esto no es necesario cuando trabajas con cadenas de caracteres en Python. Podemos crear una cadena de caracteres simplemente encerrando contenido entre comillas después de un signo de igual (=).

```
nombre=“Rosa”
print(nombre)
print(type(nombre))
nombre=‘Rosa’
```

Comillas simples y dobles

Se puede insertar comillas simple ' o doble " dentro de una cadena.

```
miNombre = "Mi nombre es Tania Aldoradin Quintanilla"
myName = 'My name is Tania Aldoradin Quintanilla'
print(miNombre)
print(myName)
```

Caracteres

También podemos incluir un salto de línea dentro de una cadena, lo que significa que lo que esté después del salto, se imprimirá en una nueva línea.

```
mycaracter1="Este texto lleva barra invertida \\Tania"
mycaracter2="Este texto lleva apostrofe \'Tania"
mycaracter3="Este texto lleva comillas\"Tania\""
mycaracter4="Este texto lleva salto de linea\nTania"
mycaracter5="Este texto lleva tabulación \tTania"
mycaracter6="\100 este caracter se llama arroba"
mycaracter7="\110 este caracter se llama hache"
print(mycaracter1)
print(mycaracter2)
print(mycaracter3)
print(mycaracter4)
print(mycaracter5)
print(mycaracter6)
print(mycaracter7)
```

Secuencias de escape de cadenas

Secuencia	Significado
newline	No se incluye en la cadena (sirve para escribir literales de cadena que ocupen más de una línea).
`\\`	Barra invertida (*backslash*).
`\'`	Comilla simple.
`\"`	Comillas dobles.
`\a`	Campanilla (*bell*)
`\b`	Retroceso (*backspace*).
`\f`	Nueva página.
`\n`	Nueva línea.
`\r`	Retorno de carro.
`\t`	Tabulador horizontal.
`\v`	Tabulador vertical.
ooo	Carácter ASCII con código octal *ooo*.
`\x`*hh*	Carácter ASCII con código hexadecimal *hh*.

Secuencias de escape de cadenas

Si queremos evitar tener que escribir tantas barras, podemos utilizar las denominadas raw strings. Simplemente, precede la cadena por una r y no se interpretaran las secuencias de escape:

```
print(r"Este texto lleva comillas \"Tania\"")
```

```
Este texto lleva comillas \"Tania\"
```

Formateo de cadenas

Una cadena que contenga variables en su interior, como números o incluso otras cadenas. Una forma de hacerlo sería concatenando la cadena con el operador +. Se puede realizar usando que str() convierte en string lo que se pasa como parámetro.

```
m = 2.8
cadena = "El número es: " + str(m)
print(cadena)
```

```
El número es: 2.8
```

Formateo de cadenas

Un operador muy utilizado con las cadenas, especialmente para formatear la salida del programa, es %. Como operando izquierdo recibe una cadena con indicaciones de formato similares a las de printf. El operando derecho es una expresión o una tupla. El resultado es la cadena resultante de sustituir las marcas de formato en el operando izquierdo por el valor o valores del operando derecho:

```
a=10
print ("El resultado es %d" %a)
```

```
El resultado es 10
```

Caracteres de formato para el operador %

Carácter	Significado
`d`, `i`	Entero en decimal.
`o`	Entero en octal.
`x`, `X`	Entero en hexadecimal.
`e`, `E`	Número en coma flotante con exponente.
`f`, `F`	Número en coma flotante sin exponente.
`g`, `G`	Número en coma flotante con o sin exponente, según la precisión y la talla del exponente.
`s`	Transforma el objeto en cadena usando `str`.

Formateo de cadenas

```
print("%d-%d" %(3,6))
```

```
3-6
```

```
print("%d en hexadecimal es %x" %(123,123))
```

```
print("%f" %3.4)
```

```
123 en hexadecimal es 7b
3.400000
```

```
print("·····Los números son %d-%d" %(5,10))
print("·····Los números son %f-%f" %(5,10))
```

```
·····Los números son 5-10
·····Los números son 5.000000-10.000000
```

Otra forma

```
salida1= "######Los números son {} y {}".format(2, 8)
salida2 = "######Los números son {a} y {b}".format(a=2, b=8)
a = 3; b = 7
salida3 = f"######Los números son {a} y {b}"
salida4 = f"######a + b = {a+b}"

def funcion():
    return 18
salida5= f"######El resultado de la función es {funcion()}"
print(salida1)
print(salida2)
print(salida3)
print(salida4)
print(salida5)
```

```
######Los números son 2 y 8
######Los números son 2 y 8
######Los números son 3 y 7
######a + b = 10
######El resultado de la función es 18
```

Índices y operadores de cadenas de caracteres

Una operación que podemos realizar es la concatenación que consiste en unir distintas cadenas mediante el uso del signo más (+).

```
nombre = "Luke"
apellido = "Skywalker"
print(nombre + " " + apellido)
```

Índices y operadores de cadenas de caracteres

También podemos concatenar con el signo de multiplicación (*), que en este caso significa adjuntarle un determinado número de copias a la cadena.

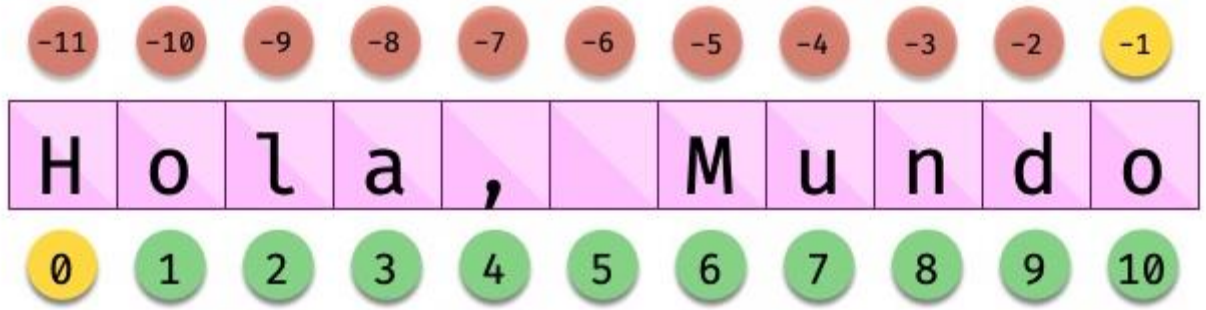
```
cadena = 'Python'
print(cadena * 4)
```

Cadenas

Los «strings» están indexados y cada carácter tiene su propia posición. Para obtener un único carácter dentro de una cadena de texto es necesario especificar su índice dentro de corchetes [...].

Caracteres :	P	y	t	h	o	n
Índice :	0	1	2	3	4	5
Índice inverso :	-6	-5	-4	-3	-2	-1

Indexación de caracteres dentro de una cadena



Iteración en una cadena de carácteres

```
miTexto = 'Python'
print("-"*30)
print(miTexto[0])
print(miTexto[1])
print(miTexto[2])
print(miTexto[3])
print(miTexto[4])
print(miTexto[5])
print("-"*30)
print(miTexto[-1])
print(miTexto[-2])
print(miTexto[-3])
print(miTexto[-4])
print(miTexto[-5])
print(miTexto[-6])
print("-"*30)
```

```
------------------------------
P
y
t
h
o
n
------------------------------
n
o
h
t
y
P
------------------------------
```

Métodos o funciones de las cadenas

len: El método len() devuelve la longitud de una cadena.

```
print(len("Esta es una cadena"))
```

capitalize(): El método capitalize() se aplica sobre una cadena y la devuelve con su primera letra en mayúscula.

lower(): El método lower() convierte todos los caracteres alfabéticos en minúscula.

upper(): El método upper() convierte todos los caracteres alfabéticos en mayúsculas.

swapcase(): El método swapcase() convierte los caracteres alfabéticos con mayúsculas en minúsculas y viceversa.

```
curso= "mi cuRso Es python"
print(curso.capitalize())
print(curso.lower())
print(curso.upper())
print(curso.swapcase())
```

```
Mi curso es python
mi curso es python
MI CURSO ES PYTHON
MI CUrSO eS PYTHON
```

Métodos o funciones de las cadenas

len: El método len() devuelve la longitud de una cadena.

```
print(len("Esta es una cadena"))
```

capitalize(): El método capitalize() se aplica sobre una cadena y la devuelve con su primera letra en mayúscula.

lower(): El método lower() convierte todos los caracteres alfabéticos en minúscula.

upper(): El método upper() convierte todos los caracteres alfabéticos en mayúsculas.

swapcase(): El método swapcase() convierte los caracteres alfabéticos con mayúsculas en minúsculas y viceversa.

```
curso= "mi cuRso Es python"
print(curso.capitalize())
print(curso.lower())
print(curso.upper())
print(curso.swapcase())
print(curso.title())
```

```
Mi curso es python
mi curso es python
MI CURSO ES PYTHON
MI CUrSO eS PYTHON
Mi Curso Es Python
```

Métodos o funciones de las cadenas

`isalnum()` devuelve cierto si la cadena es no vacía y sólo contiene letras y dígitos.

`isalpha()` devuelve cierto si la cadena es no vacía y sólo contiene letras.

`isdigit()` devuelve cierto si la cadena es no vacía y sólo contiene dígitos.

`islower()` devuelve cierto si todas las letras de la cadena son minúsculas y hay al menos una minúscula.

`isspace()` devuelve cierto si la cadena es no vacía y todos sus caracteres son espacios.

`isupper()` devuelve cierto si todas las letras de la cadena son mayúsculas y hay al menos una mayúscula.

```
cadena1 = 'Python2024'
cadena2 = 'Python'
cadena3 = '2024'
cadena4 = 'python'
cadena5 = '  '
cadena6 = 'PYTHON'
print(cadena1.isalnum())
print(cadena2.isalpha())
print(cadena3.isdigit())
print(cadena4.islower())
print(cadena5.isspace())
print(cadena6.isupper())
```

```
True
True
True
True
True
True
```

Métodos o funciones de las cadenas

strip(): nos permiten eliminar espacios en blanco. Devuelve una lista que contiene las palabras de la cadena. Si se incluye la cadena s, se utiliza como separador.

replace():reemplazar una subcadena por otra.

split(): dividir una cadena en varias subcadenas.

```
cadena7 = '\n \t  Python     2024    '
cadena8 = 'Quien mal anda mal acaba'
print(cadena7.strip())
print(cadena8.replace('mal', 'bien'))
print(cadena8.replace('mal', 'bien',1))
x = "Programa en Python"
print(x.split())
email = "tania.aldoradin@star.com"
print(email.split('@'))
```

```
Python     2024
Quien bien anda bien acaba
Quien bien anda mal acaba
['Programa', 'en', 'Python']
['tania.aldoradin', 'star.com']
```

Listas

Las listas en Python son un tipo de dato que permite almacenar datos de cualquier tipo. Son mutables y dinámicas, lo cual es la principal diferencia con los sets y las tuplas.

Las listas en Python son uno de los tipos o estructuras de datos más versátiles del lenguaje, ya que permiten almacenar un conjunto arbitrario de datos. Es decir, podemos guardar en ellas prácticamente lo que sea. Si vienes de otros lenguajes de programación, se podría decir que son similares a los arrays.

```
lista = [1, 2, 3, 4]
```

```
lista = [5, 9, 10]
for l in lista:
    print(l)
```

Gracias