Fundamentos de Programación. Primer Curso de ASIR.

UT 02.03 – Funciones y métodos para cadenas.

UT 02.03 – Funciones y métodos para cadenas.
1.- Representación de Cadenas.
1.1.- Forma general.

- Una cadena es una secuencia inmutable de caracteres Unicode, delimitada por **comillas simples o dobles**.

  Las cadenas **se deben cerrar con las mismas comillas con las que se abrieron**, de lo contrario estaremos cometiendo un error de sintaxis
- La función **print()** muestra por pantalla el contenido de la cadena, pero no las comillas delimitadoras de las cadenas.
- Se pueden escribir comillas simples en cadenas delimitadas con comillas dobles y viceversa.

```
>>> print('Esto es una cadena delimitada por comillas simples.')
Esto es una cadena delimitada por comillas simples.
>>> print("Esto es una cadena delimitada por comillas dobles.")
Esto es una cadena delimitada por comillas dobles.
>>> print("La comilla simple ' se pueden incluir en una cadena.")
La comilla simple ' se pueden incluir en una cadena.
>>> print('Las comillas dobles " delimitan cadenas.')
Las comillas dobles " delimitan cadenas.
```

UT 02.03 – Funciones y métodos para cadenas.
1.- Representación de Cadenas.
1.2.- Caracteres especiales.

- Los caracteres especiales empiezan por una contrabarra **“\”**.
- Algunos caracteres especiales son los siguientes:
  - **Comilla doble. \".**
  - **Comilla simple. \'.**
  - **Salto de línea. \n.**
  - **Tabulador. \t.**

```
>>> print("Hay una comilla doble \" en esta cadena.")
Hay una comilla doble " en esta cadena.
>>> print('Hay una comilla simple \' en esta cadena.')
Hay una comilla simple \' en esta cadena.
>>> print("Esta cadena\nocupa dos lineas.")
Esta cadena
ocupa dos lineas.
>>> print("Uno\tDos\tTres")
Uno     Dos     Tres
```

UT 02.03 – Funciones y métodos para cadenas.
1.- Representación de Cadenas.
1.3.- Cadenas de gran tamaño.

- Las comillas triples permiten que las **cadenas ocupen más de una línea**.
- Pero las comillas triples se utilizan para documentar código. Son las llamadas **docstrings**.
- Para obtener la docstring de una sección de código se usa la sintaxis.

  print(seccion.__doc__)

  Antes y después de la palabra doc van dos guiones bajos.

```
>>> print("""Esta cadena
tiene una longitud
  de tres líneas.""")
Esta cadena
tiene una longitud
  de tres lineas.
```

```
>>> def ejemplo():
...    """Aqui se hace una descripcion de lo que hace la funcion."""
>>> print(ejemplo.__doc__)
Aqui se hace una descripcion de lo que hace la funcion.
```

UT 02.03 – Funciones y métodos para cadenas.
1.- Representación de Cadenas.
1.4.- Formateado de la salida. Cadenas f-string (I).

- En **Python 3.6** se añadió (**PEP 498**) una nueva notación para cadenas llamada cadenas **f-string**.
- Estas simplifican la inserción de variables y expresiones en las cadenas.
- Una f-string contiene variables y expresiones entre llaves **{}** que **se sustituyen directamente por su valor**. Si la variable va entre dobles llaves **{{}}** se escribe el nombre de la variable en si misma.
- Las f-string se reconocen porque **comienzan por una letra f antes de las comillas de apertura**.
- Veamos algunos ejemplos:

```
>>> nombre = "Roberto"
>>> edad = 19
>>> print(f"Me llamo {nombre} y tengo {edad} años.")
Me llamo Roberto y tengo 19 años.
```

```
>>> nombre = "Roberto"
>>> print(f"La variable {{nombre}} contiene el valor {nombre}.")
La variable nombre contiene el valor 19.
```

UT 02.03 – Funciones y métodos para cadenas.
1.- Representación de Cadenas.
1.4.- Formateado de la salida. Cadenas f-string (II).

```
>>> valor1=2500
>>> valor2=500
>>> cadena=f"""
... {valor1:5}
... {valor2:5}
... -----
... {suma:5}
... """
>>> print(cadena)
 2500
  500
-----
 3000
```

- Con la sintaxis **{variable:n.mf}** se muestra un número ocupando **n caracteres** (tanto números como el punto decimal) de los cuales **m son decimales**. **Se acaba con la letra f (float)**.

```
>>> pi = 3.141592
>>> print(f"La variable {pi:5.3f} ocupa cinco caracteres y tiene tres decimales.")
La variable 3.142 ocupa cinco caracteres y tiene tres decimales.
```

UT 02.03 – Funciones y métodos para cadenas.
1.- Representación de Cadenas.
1.4.- Formateado de la salida. Cadenas f-string (III).

- Con la sintaxis **{variable:.n%}** se realiza el cálculo del porcentaje (como si se tratara de una hoja de cálculo. El valor de n es el número de decimales presentes.

```
>>> porcentaje = 0.1263852
>>> print(f"El porcentaje es {porcentaje:.2%}.")
El porcentaje es 12.64%.
```

UT 02.03 – Funciones y métodos para cadenas.
1.- Representación de Cadenas.
1.4.- Formateado de la salida. Cadenas f-string (IV).

- Podemos formatear cadenas con la siguiente sintaxis:
  - **{cadena :<n}**. Alinea la cadena a la izquierda a n caracteres (**por defecto**).
  - **{cadena :^n}**. Alinea la cadena al centro a n caracteres.
  - **{cadena :>n}**. Alinea la cadena a la izquierda a n caracteres.
- Tambien podemos rellenar con un carácter específico mediante la sintaxis:

  {cadena :[carácter][alineacion]:[numCaracteres]}

```
>>> cadena = "Esta es una prueba"
>>> print(f"{cadena}")
Esta es una prueba
>>> print(f"{cadena :>20}")
  Esta es una prueba
>>> print(f"{cadena :>30}")
            Esta es una prueba
>>> print(f"{cadena :^30}")
      Esta es una prueba
>>> print(f"{cadena :*^30}")
******Esta es una prueba******
```

UT 02.03 – Funciones y métodos para cadenas.
2.- Concatenación de cadenas. Longitud de cadena.

- Se pueden concatenar cadenas mediante la suma de dichas cadenas.

```
>>> cadena1="Hola"
>>> cadena2="Mundo"
>>> cadena = cadena1 + cadena2
>>> print(cadena)
HolaMundo
```

- Podemos clonar una cadena un número n de veces, **se multiplica la cadena por n**.

```
>>> cadena = cadena1 * 3
>>> print(cadena)
HolaHolaHola
```

- La longitud de una cadena se obtiene mediante la funcion **len(cadena)**.

```
>>> cadena = "Esta es una prueba."
>>> len(cadena)
19
```

UT 02.03 – Funciones y métodos para cadenas.
3.- Indexación de cadenas.

- Cada uno de los caracteres de una cadena **tiene asignado un índice**.
- Este indice nos permite seleccionar su carácter asociado haciendo **referencia a él entre corchetes ([]) en el nombre de la variable que almacena la cadena**.
- Si consideremos el orden **de izquierda a derecha**, el indice empieza en 0 para el primer carácter, etc.
- También se puede considerar el orden **de derecha a izquierda**, en cuyo caso al último carácter le corresponde el índice -1, al penúltimo -2 y así sucesivamente.

  Este método es útil si por ejemplo queremos acceder a caracteres en las ultimas posiciones de una cadena con muchos caracteres de la cual no conocemos su longitud.

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

```
>>> cadena = "Esta es una prueba"
>>> print(cadena[5])
e
>>> print(cadena[-3])
e
>>> print(cadena[-6])
p
```

UT 02.03 – Funciones y métodos para cadenas.
4.- Subcadenas de una cadena (I).

- Podemos realizar a una cadena es seleccionar solamente una parte de ella.
- Para ello se usa la notación **[inicio:fin:salto]** con el nombre de la variable de la cadena, donde:
  - **Inicio**. Es el índice del primer carácter de la porción de la cadena que queremos seleccionar.
  - **Fin**. Último carácter no incluido de la porción de la cadena que queremos seleccionar.
  - **Salto**. Indica cada cuantos caracteres seleccionamos entre las posiciones de inicio y fin.

  Los índices de inicio y final pueden ser **negativos**. En este caso se empieza a contar desde el final.

```
>>> cadena = "Esta es una prueba"
>>> print(cadena[5])
e
>>> print(cadena[-6])
p
>>> print(cadena[3:7])
a es
>>> print(cadena[3:12:2])
ae n
>>> print(cadena[7:])
 una prueba
```

- Podemos realizar a una cadena es seleccionar solamente una parte de ella.
- Si se omite el final se sobreentiende que se quiere seleccionar desde la posición de inicio hasta el final de la cadena. Por el contrario, si se omite el inicio se seleccionará desde el inicio de la cadena hasta la posición indicada

UT 02.03 – Funciones y métodos para cadenas.
4.- Subcadenas de una cadena (II).

- Si se omite el final se sobreentiende que se quiere seleccionar desde la posición de inicio hasta el final de la cadena. Por el contrario, si se omite el inicio se seleccionará desde el inicio de la cadena hasta la posición indicada

```
>>> print(cadena[5:])
es una prueba
>>> print(cadena[:9])
Esta es u
>>> print(cadena[::3])
Easnpe
```

UT 02.03 – Funciones y métodos para cadenas.
5.- Iteración sobre una cadena.

- Una cadena es un elemento iterable. Por ello, puede ser recorrida por un bucle.
- Veamos un ejemplo.

```
cadena = “Prueba”
for letra in cadena:
    print(f”En la cadena ‘{cadena}’ hay una ‘{letra}’.”)


En la cadena ‘Prueba’ hay una ‘P’.
En la cadena ‘Prueba’ hay una ‘r’.
En la cadena ‘Prueba’ hay una ‘u’.
En la cadena ‘Prueba’ hay una ‘e’.
En la cadena ‘Prueba’ hay una ‘b’.
En la cadena ‘Prueba’ hay una ‘a’.
```

UT 02.03 – Funciones y métodos para cadenas.
6.- Métodos de cadenas.
6.1.- Métodos de análisis.

cadena = “Esta es una cadena”			
Método	**Función**	**Ejemplo**	**Resultado**
cadena.count()	Cuenta el n.º de veces que aprece la cadena pasada como parámetro.	cadena.count(“a”)	3
cadena.find()	Encuenta la primera ocurrencia de la cadena que se introduce como parámetro.	cadena.find("prueba") cadena.find("cadena")	-1 12
cadena.startswith()	Comprueba si una cadena empieza por una determinada secuancia de caracteres.	cadena.startswith(“Hola”) cadena.startswith(“Es”)	False True
cadena.endswith()	Comprueba si una cadena empieza por una determinada secuancia de caracteres.	cadena.endswith(“cadena”) cadena.endswith(“fin”)	True False
cadena.isdecimal()	Comprueba si la cadena está formada por dígitos decimales.	"abc123".isdigit()	False
cadena.isalpha()	Comprueba si la cadena está formada por caracteres alfabéticos.	"abc123".isalpha()	False

Para mas información sobre métodos de cadenas consultar la documentación de Python en la URL:

https://docs.python.org/es/3/library/stdtypes.html#string-methods

UT 02.03 – Funciones y métodos para cadenas.
6.- Métodos de cadenas.
6.2.- Métodos de transformación.

Método	Función	Ejemplo	Resultado
cadena.upper()	Obtiene la cadena en mayúsculas.	“prueba”.upper()	“PRUEBA”
cadena.lower()	Obtiene la cadena en minúsculas.	“PrueBA”.upper()	“prueba”
cadena.capitalize()	Obtiene la cadena capitalizada; es decir, con la primera de las letras en mayúscula.	“prueba”.capitalize()	“Prueba”
cadena.center()	Centra la cadena un número de caracteres dado.	“Hola”.center(10)	“ Hola “
cadena.rjust()	Justifica a la derecha.	“Hola”.rjust(10)	“ Hola“
cadena.ljust()	Justifica a la izquierda.	“Hola”.ljust(10)	“Hola “
cadena.strip() **Cadena.lstrip()** **cadena.rstrip()**	Elimina los espacios en blanco a ambos lados, a la izquierda y a la derecha.	“ Hola “.strip() “ Hola “.lstrip() “ Hola “.rstrip()	“Hola” “Hola “ “ Hola”
cadena.replace()	Sustituye en una cadena una secuencia de caracteres por otra.	“Hola Pedro”.replace(“Pedro”, “Juan”)	“Hola Juan”

Para mas información sobre métodos de cadenas consultar la documentación de Python en la URL:

https://docs.python.org/es/3/library/stdtypes.html#string-methods

UT 02.03 – Funciones y métodos para cadenas.
6.- Métodos de cadenas.
6.3.- Métodos de separación y unión de cadenas.

Método	Función	Ejemplo	Resultado
cadena.split()	Devuelve una lista con las subcadenas obtenidas al particionar la cadena usando el carácter como separador. Por defecto el separador es el espacio o el retorno de carro (\n).	“Es una prueba”.split()	[“Es”,”una”,”prueba”]
lista.join()	Crea una cadena con las cadenas contenidas en la lista usando como separador el carácter indicado.	[“Es”,”una”,”prueba”].join(“*”)	“Es*una*prueba”

Para mas información sobre métodos de cadenas consultar la documentación de Python en la URL:

https://docs.python.org/es/3/library/stdtypes.html#string-methods