Secuencias

Programación de Inteligencia Artificial

Junta de Andalucía

FPA
FORMACIÓN
PROFESIONAL
ANDALUZA

Fondo Social Europeo

Índice | Cadenas | Listas | Tuplas

Cadenas:

- Introducción.
- Operaciones.
- Métodos.

Listas:

- Introducción.
- Operaciones.
- Métodos.

Tuplas:

- Introducción.
- Operaciones.
- Métodos.

- Se trata de un tipo de secuencia de caracteres, cuyo formato es *unicode*.
- Para denotar una cadena de caracteres usamos comillas dobles *"texto"*, comillas simples *'texto'* o 3 comillas simples cuando queremos escribir en varias líneas *'''texto'''*.

```
"Hola"
'Hola'
'''Hola.
¿Qué tal
estás?'''
```

- Cada vez que creamos una cadena de caracteres estamos creando un objeto de la clase cadena de caracteres o str.
- Los objetos creados son inmutables, es decir, no los podemos modificar.
- Hay una serie de métodos de dicha clase que podemos usar. A continuación, comentaremos los más importantes, pero en este enlace se pueden ver todos.

Introducción
Operaciones
Métodos

- Se pueden concatenar dos cadenas mediante el +.
- Se puede repetir una cadena añadiendo * y el número de repeticiones que se quieren.
- Se puede obtener un carácter en una posición concreta mediante *[i]*, siendo *i* dicha posición.
- Se puede obtener una subcadena mediante *[n:m]*, donde *n* es el primer carácter a tomar y *m* el primer carácter que no se desea. Pueden tomar cualquier valor desde 0 hasta el tamaño de la cadena, y el valor vacío. Si se deja sin ningún valor indicamos que se quiere desde el principio de la cadena (si es *n*) o hasta el final (si es *m*).
- Se pueden usar el operador módulo (*%*) para dar formato a una cadena. Se usará *%s* para insertar cadenas de caracteres, *%d* para enteros o *%f* para números reales, entre otros.

```
c = 'Erase una vez '; d = 'un cuento chino.'
print(c + d)
print(c * 4)
print(c[7])
print(c[:5])
e = 'chino'; n = 10
print("Erase una vez un cuento %s, que era el número %d" %(e, n))
```

- capitalize(): devuelve una copia de la cadena con la primera letra en mayúscula.
- count("texto"): devuelve cuantas veces aparece en la cadena el texto pasado por parámetro.
- find/index("texto"): devuelve la posición de la cadena donde empieza el texto pasado por parámetro. En caso de no encontrarlo devuelve -1.
- lower(): devuelve una copia de la cadena en minúscula.
- replace("busca", "reemplaza"): devuelve una copia de la cadena en la que sustituye el parámetro busca por reemplaza.
- splitlines(): devuelve una lista en la que se ha separado la cadena por los saltos de línea (\n).
- strip(): devuelve una copia de la cadena en la que se eliminan los espacios del principio y del final.
- title(): devuelve una copia de la cadena con la primera letra de cada palabra en mayúscula.
- upper(): devuelve una copia de la cadena en mayúscula.

- Son una secuencia de datos pero que no tienen por qué ser del mismo tipo. En una misma lista podemos tener cadenas, enteros y booleanos (entre otros).
- Las datos de una lista se denotan entre corchetes y separados por comas.

```
lista = [5, 'perro', True]; print(lista)
```

- El contenido de una lista se puede modificar (son mutables).

```
lista[2] = 3.2; print(lista)
```

- Se pueden concatenar dos listas mediante el símbolo *+*. También se puede restar a una lista otra mediante -.
- De igual modo, se puede crear una lista en la que se repite su contenido *n* veces usando el *.
- Se puede obtener un carácter en una posición concreta mediante *[i]*, siendo *i* dicha posición.
- De igual modo, se puede modificar un elemento de la lista indicando su posición y asignándole un nuevo valor.
- Se puede obtener una sublista mediante *[n:m]*, donde *n* es el primer elemento de la lista que se desea tomar y *m* el primer elemento que queda fuera de la sublista. Pueden tomar cualquier valor desde 0 hasta el tamaño de la lista, y el valor vacío. Si se deja sin ningún valor indicamos que se quiere desde el principio de la lista (si es *n*) o hasta el final (si es *m*).
- valor in lista: indica si un valor se encuentra en la lista. También se puede usar *not in* para ver si un elemento no está en la lista.

```
lista1 = ['Nicolás Salmerón', 3, True];
lista2 = [5, 8.2]
listaFinal = lista1 + lista2; print(listaFinal)
listaFinal = lista1 * 3; print(listaFinal)
```

```
print(lista1[0]); lista1[2] = 'tercero'
listaFinal = lista1[:1]; print(listaFinal)
print(3 in lista1)
```

- append(valor): añade un elemento al final de la lista.
- clear(): elimina todos los elementos que contiene la lista.
- copy(): permite crear una copia de la lista.
- count(valor): devuelve las veces que aparece un elemento en la lista.
- del(lista[i]): permite eliminar el elemento *i* de la lista.
- extend(lista2): añade la segunda lista a la primera.
- index(valor): posición que ocupa el valor dentro de la secuencia.
- insert(pos, valor): añade un valor en la posición indicada.

```
lista = [5, 4, 17, 13, -4]
lista.append(0); print(lista)
lista.clear(); print(lista)
lista = [5, 4, 17, 13, -4]; l = lista.copy(); print(l)
print(lista.count(4))
del(lista[1]); print(lista)
lista.extend([20, 30,40]); print(lista)
print(lista.index(20))
lista.insert(1, 80); print(lista)
```

- len(lista): devuelve el tamaño de la lista.
- pop(): saca el primer elemento de la lista.
- remove(valor): elimina un valor de la lista.
- reverse(): ordena una lista en sentido inverso.
- sort(): ordena una lista de mayor a menor.

```
lista = [5, 4, 17, 13, -4]
print(len(lista))
print(lista.pop()); print(lista)
lista.remove(13); print(lista)
lista.reverse(); print(lista)
lista.sort(); print(lista)
```

- Conjunto no modificable de datos de datos.
- Las datos de una tupla se denotan entre paréntesis y separados por comas.

```
tupla = ('Nicolás Salmerón', 3, True); print(tupla)
```

- Como se ha comentado en el primer punto, el contenido de una tupla no se puede modificar.

```
tupla[1] = 3.2 # No se puede modificar un dato
```

- Se pueden concatenar dos listas mediante el símbolo *+*.
- De igual modo, se puede crear una lista en la que se repite su contenido *n* veces usando el *.

- Se pueden concatenar dos tuplas mediante el símbolo *+*. También se puede restar a una tupla otra mediante -.
- De igual modo, se puede crear una tupla en la que se repite su contenido *n* veces usando el *.
- Se puede obtener un carácter en una posición concreta mediante *[i]*, siendo *i* dicha posición.
- Se puede obtener una sublista mediante *[n:m]*, donde *n* es el primer elemento de la lista que se desea tomar y *m* el primer elemento que queda fuera de la sublista. Pueden tomar cualquier valor desde 0 hasta el tamaño de la lista, y el valor vacío. Si se deja sin ningún valor indicamos que se quiere desde el principio de la lista (si es *n*) o hasta el final (si es *m*).
- valor in lista: indica si un valor se encuentra en la lista. También se puede usar *not in* para ver si un elemento no está en la lista.

```
tupla1 = ('Nicolás Salmerón', 3, True);
tupla2 = (5, 8.2)
tuplaFinal = tupla1 + tupla2; print(tuplaFinal)
tuplaFinal = tupla1 * 3; print(tuplaFinal)
print(tupla1[0])
tuplaFinal = tupla1[:2]; print(tuplaFinal)
print(3 in tupla1)
```

Al no poderse modificar las tuplas solo tenemos dos métodos disponibles:

- count(valor): devuelve cuantas veces está el valor en la tupla.
- index(valor): devuelve la primera posición en la que aparece el valor dentro de la tupla.

```
tupla = (2, 5, -4, 43, 5)
print(tupla.count(5))
print(tupla.index(43))
```

Instituto de Educación Secundaria PUEBLA DE VICAR

FPA
FORMACIÓN PROFESIONAL ANDALUZA

Fondo Social Europeo

Imágenes: Flaticon.com, IES La Puebla, Junta de Andalucía