Python

Sistemas de gestión empresarial – 148FA (DAM2)

Strings

Cualquier variable que contenga un valor de tipo String (str) será tratada como un subtipo del objeto String. Este objeto contiene varios métodos, como hemos visto con listas.

```
a = “Hola mund0”

b = “3”

c = “Hola” + “ qué tal”
```

Strings

- Como si fuera una lista, podemos acceder a un “subelemento” del string o cadena de caracteres utilizando [ ] y un índice:

```
mi_cadena = “Hola mundo”
print (mi_cadena[0:4])
>>> “Hola”
```

- Podemos poner o no poner el límite a la izquierda o a la derecha. Si no ponemos a la izquierda, comenzará desde la izquierda del todo. Si no ponemos en la derecha, llegará hasta el final:

```
print (mi_cadena[1:])
>>> “ola Mundo”
print (mi_cadena[:4])
>>> “Hola”
```

Concatenación

- Para concatenar dos cadenas de caracteres podemos utilizar el símbolo + entre las dos:

```
a = “Hola”

b = “Mundo”

c = a+” ”+b  #Con espacio en medio

print (c)

>>> “Hola Mundo”
```

Métodos

- **upper():** Convierte todo a mayusculas.

```
mi_string = “Hola Mundo”
print (mi_string.upper())  # HOLA MUNDO
```

- **lower():** Devuelve una copia de la cadena en minúsculas.

```
mi_string = “Hola Mundo”
print (mi_string.lower())  # hola mundo
```

- **capitalize():** Devuelve una copia de la cadena con la primera letra en mayúsculas.

```
mi_string = “hola mundo”
print (mi_string.capitalize())  # Hola mundo
```

- **swapcase():** Devuelve una copia de la cadena con las mayúsculas convertidas en minúsculas y viceversa.

```
mi_string = “Hola Mundo”
print (mi_string.swapcase())  # hOLA mUNDO
```

Métodos

- **title():** devuelve una copia de la cadena convertida: la primera letra de cada palabra en mayúsculas.

```
mi_string = “hola mundo”

print (mi_string.title())  # Hola Mundo
```

- **center(longitud, "carácter de relleno”):** Devuelve una copia de la cadena centrada, con un carácter de relleno a los lados.

```
mi_string = “esta es mi cadena de texto”.capitalize()
print (mi_string.center(50, “=“))

==========Esta es mi cadena de texto==========

print (mi_string.center(50, “ “))

          Esta es mi cadena de texto
```

Métodos

- **ljust(longitud, "caracter de relleno"):** devuelve una copia de la cadena alineada a la izquierda, con un carácter de relleno a su derecha.

```
mi_string = “esta es mi cadena de texto”.capitalize()
print (mi_string.ljust(50, “=“))

Esta es mi cadena de texto==========================
```

- **rjust(longitud, "caracter de relleno”):** devuelve una copia de la cadena alineada a la derecha, con un carácter de relleno a su izquierda.

```
mi_string = “esta es mi cadena de texto”.capitalize()
print (mi_string.rjust(50, “=“))

==========================Esta es mi cadena de texto

print (mi_string.rjust(50, “ “))

                         Esta es mi cadena de texto
```

Métodos

- **zfill(longitud):** devuelve una copia de la cadena con ceros a la izquierda hasta llegar a la longitud que se indica. Si el string es más largo, se muestra entero, sin ningún 0 adicional.

```
numero_agente= “7” #Tiene que ser string

print (numero_socio.zfill(3))


>>> 007
```

- **count("subcadena" [, posicion_inicio, posicion_fin]):** devuelve un número entero que indica la cantidad de apariciones de la subcadena.

```
mi_cadena= “la palabra azul tiene una a”

print (mi_cadena.count(“a”))


>>> 7
```

Métodos

- **find("subcadena" [, posicion_inicio, posicion_fin]):** •devuelve un entero representando la posición donde se inicia la subcadena dentro de cadena (case sensitive). Si no la encuentra, devuelve -1.

```
mi_cadena= “Hola mundo, estoy estudiando Python”
print (mi_cadena.find(“Python”))
>>> 29

print (mi_cadena.find(“Python”, 0, 20))
>>> -1
```

Métodos de validación

- **startswith**("subcadena" [, posicion_inicio, posicion_fin]): saber si una cadena empieza con una subcadena determinada
- **endswith**("subcadena" [, posicion_inicio, posicion_fin]): saber si una cadena termina con una cadena determinada
- **isalnum**(): saber si una cadena es alfanumérica
- **isalpha**(): saber si una cadena es alfabética. Devuelve falso si encuentra algún número.
- **isdigit**(): saber si una cadena es numérica. Devuelve falso si encuentra alguna letra.
- **islower**(): saber si una cadena sólo tiene minúsculas
- **isupper**(): saber si una cadena sólo tiene mayúsculas
- **isspace**(): saber si una cadena sólo tiene espacios en blanco
- **istitle**(): saber si una cadena tiene formato de título

Métodos de sustitución

- **replace("subcadena a buscar", "subcadena por la cual reemplazar"):** devuelve la cadena resultante.

```
buscar = “Java”

reemplazar=“Python”

print (“Estoy estudiando Java”.replace(buscar, reemplazar))

>>> Estoy estudiando Python
```

Métodos de sustitución

- **strip([“caracter"]):** devuelve la cadena resultante sin espacios ni por la derecha ni por la izquierda.

```
mi_cadena = “       Esta es una cadena de texto       ”

print (mi_cadena.strip())

>>> Esta es una cadena de texto
```

- **lstrip([“caracter”]):** elimina el carácter o caracteres indicados (o espacios en blanco si no se especifica) desde el inicio de la cadena.

- **rstrip([“caracter”]):** elimina el carácter o caracteres indicados (o espacios en blanco si no se especifica) desde el final de la cadena.

```
texto = "   hola mundo   "

print(texto.lstrip())  # "hola mundo   "

print(texto.rstrip())  # "   hola mundo"
```

Métodos de sustitución

- **format():** permite insertar valores en una cadena usando llaves {} como marcadores de posición, que luego se rellenan con los valores especificados en el método.
  - Posicional:
```
"{} y {}".format( *args: "manzanas", "naranjas")  # "manzanas y naranjas"
```
  - Por nombre:
```
"{fruta1} y {fruta2}".format(fruta1="manzanas", fruta2="naranjas")  # "manzanas y naranjas"
```
  - Formato avanzado:
```
"Precio: {:.2f}".format(5)  # "Precio: 5.00"
```

Métodos de unión-división

- **separador.join(iterable):** une una secuencia de elementos (como una lista de cadenas) en una sola cadena, usando una cadena separadora especificada.

```
palabras = ["Hola", "mundo", "Python"]
resultado = " ".join(palabras)
print(resultado)  # "Hola mundo Python"
```

- **partition(separador):** divide una cadena en tres partes, utilizando un separador específico.

```
mi_cadena = 'Python es divertido'
print(mi_cadena.partition('es'))
>>> ('Python', 'es', 'divertido')
print(mi_cadena.partition('no'))
>>> ('Python es divertido', '', '')
```

Métodos de unión-división

- **split("separador"):** Separa una cadena en elementos de una lista.

```
mi_cadena = ‘Python es divertido’

print(mi_cadena.split())

>>> [‘Python’,’es’,’divertido’]

mi_cadena2 = ‘Python es, divertido’

print(mi_cadena.split(‘,’))

>>> [‘Python es’,’divertido’]
```

Métodos de unión-división

- **Longitud de Strings (len):** Para saber la longitud de un string usamos la función “len”.

```
len('abcde') #devolverá 5
```

- **Recorrer parte de un string:**

```
for i in mi_cadena[0:len(mi_cadena)-3]:
```

- **Recorrer un string:**

```
for i in mi_cadena:
```