Investiga acerca de las cadenas o Strings y sus métodos en Python.

Genere una tabla resumen donde incluya cada método encontrado y un ejemplo de su aplicación

|  |  |  |
| --- | --- | --- |
| NOMBRE | QUE HACE | EJEMPLO |
| **x.upper()** | Este método devuelve una copia de la cadena convirtiendo las letras minúsculas a mayúsculas. El hecho de que devuelva una copia significa que la variable original no se ve afectada por la operación. | >>> x = "Programa en Python 3"  >>> x.upper()  'PROGRAMA EN PYTHON 3'  >>> x  'Programa en Python 3' |
| **x.lower()** | Devuelve una copia de la cadena convirtiendo las letras mayúsculas a minúsculas. | >>> x = "PROGRAMA EN PYTHON 3"  >>> x.lower()  'programa en python 3' |
| **x.title()** | Devuelve una copia de la cadena usando la notación de título. | >>> x = "programa en python 3"  >>> x.title()  'Programa En Python 3' |
| **x.replace(viejo, nuevo)** | Devuelve una copia de la cadena a la cual se le ha cambiado la primera ocurrencia del carácter especificado en viejo por el especificado en nuevo. | >>> x = "Programa En Python 3"  >> x.replace("E", "e")  'Programa en Python 3' |
| **x.lstrip()**: | Devuelve una copia de la cadena a la cual se le han eliminado los espacios del principio. |  |
| **x.rstrip()** | Se comporta igual que el anterior pero para los espacios del final. | >>> x = " Programa en Python "  >>> x.lstrip()  'Programa en Python '  >>> x.rstrip()  ' Programa en Python' |
| **x.split(sep=None)** | Devuelve una lista de las palabras de la cadena separadas acorde el parámetro sep. Si este parámetro no se especifica o es None, la cadena se separa teniendo en cuenta los espacios en blanco. Si usamos el símbolo arroba (@) como separador y nuestra cadena es una dirección de correo electrónico, podemos separar los nombres de usuario y servidor. | >>> x = "Programa en Python"  >>> x.split()  ['Programa', 'en', 'Python']  >>> email = "nombre.apellido@ejemplo.tld"  >>> email.split('@')  ['nombre.apellido', 'ejemplo.tld'] |
| **Formatear Strings** | Este método devuelve una copia de la cadena a la que se le han sustituido las posiciones que contienen llaves ({}) por los argumentos del método. Esta sustitución se realiza por defecto en el mismo orden de los argumentos. | >>> "Mi {} se llama {} y tiene {} años".format("perro", "Ashley", "tres")  'Mi perro se llama Ashley y tiene tres años' |
| **count()** | retorna el número de veces que se repite un conjunto de caracteres especificado. | >>> s = "Hola mundo"  >>> s.count("Hola")  1 |
| **find() e index()** | retornan la ubicación (comenzando desde el cero) en la que se encuentra el argumento indicado. | 1. >>> s.find("mundo") 2. 5 3. >>> s.index("mundo") 4. 5 |
| **startswith() y endswith()** | indican si la cadena en cuestión comienza o termina con el conjunto de caracteres pasados como argumento, y retornan True o False en función de ello. | 1. >>> s = "Hola mundo" 2. >>> s.startswith("Hola") 3. True 4. >>> s.endswith("mundo") 5. True 6. >>> s.endswith("world") 7. False |
| **isdigit(), isnumeric() e isdecimal()** | determinan si todos los caracteres de la cadena son dígitos, números o números decimales. | 1. >>> "1234".isnumeric() 2. True 3. >>> "1234".isdecimal() 4. True 5. >>> "abc123".isdigit() 6. False |
| **capitalize()** | retorna la cadena con su primera letra en mayúscula. | >>> "hola mundo".capitalize()  'Hola mundo' |
| **encode()** | codifica la cadena con el mapa de caracteres especificado y retorna una instancia del tipo bytes | 1. >>> "Hola mundo".encode("utf-8") 2. b'Hola mundo' |
| **center(), ljust() y rjust()** | alinean una cadena en el centro, la izquierda o la derecha respectivamente. Toman un argumento, la cantidad de caracteres respecto de la cual se producirá la alineación. | 1. >>> "Hola".center(10) 2. ' Hola ' 3. >>> "Hola".ljust(10) 4. 'Hola ' 5. >>> "Hola".rjust(10) 6. ' Hola' |
| **lower() y upper()** | retornan una copia de la cadena con todas sus letras en minúsculas o mayúsculas según corresponda. | 1. >>> "Hola Mundo!".lower() 2. 'hola mundo!' 3. >>> "Hola Mundo!".upper() 4. 'HOLA MUNDO!' |
| **swapcase()** | por su parte, cambia las mayúsculas por minúsculas y viceversa. | 1. >>> "Hola Mundo!".swapcase() 2. 'hOLA mUNDO!' |
| **strip(), lstrip() y rstrip()** | remueven los espacios en blanco que preceden y/o suceden a la cadena. | 1. >>> s = " Hola mundo! " 2. >>> s.strip() 3. 'Hola mundo!' 4. # Remueve los de la derecha. 5. >>> s.rstrip() 6. ' Hola mundo!' 7. # Remueve los de la izquierda. 8. >>> s.lstrip() 9. 'Hola mundo! ' |
| **split()** | cuyo separador por defecto son espacios en blanco y saltos de línea. | 1. >>> "Hola mundo!\nHello world!".split() 2. ['Hola', 'mundo!', 'Hello', 'world!'] |
| **partition()** | retorna una tupla de tres elementos: el bloque de caracteres anterior a la primera ocurrencia del separador, el separador mismo, y el bloque posterior. | 1. >>> s = "Hola mundo. Hello world!" 2. >>> s.partition(" ") 3. ('Hola', ' ', 'mundo. Hello world!') |