P05 Cadenas (Strings)

1.- Strings

Una cadena se puede leer empleando la función `input()` o ser definida dentro del código empleando comillas simples o dobles (ojo, otros lenguajes de programación emplean para cosas distintas los dos tipos de comillas).

Como ya sabemos se pueden concatenar dos cadenas empleando el operador +. También se puede repetir varias veces una cadena multiplicándola por un entero.

```
print('>_< ' * 5)  # >_< >_< >_< >_< >_<
```

Python entiende una cadena como una secuencia de caracteres. La función devuelve cuantos caracteres hay en una cadena dada `len(some_string)`:

```
print(len('abcdefghijklmnopqrstuvwxyz'))  # 26
```

Prácticamente cualquier objeto en Python se puede convertir en una cadena usando la función `str(some_object)`.. Por ejemplo números:

```
s = str(2 ** 100)
print(s)  # 1267650600228229401496703205376
print(len(s))  # 31
```

2.- Slices (“fragmentos”)

Llamaremos slice de una cadena dada a un fragmento de ella, substring o subsecuencia.

2.1.- Un único carácter.

La forma más simple será un único carácter. `S[i]` devuelve el i-esimo símbolo de la cadena. Se empieza a contar por el 0, es decir si `S = 'Hello'`, `S[0] == 'H'`, `S[1] == 'e'`,`S[2] == 'l'`, `S[3] == 'l'`, `S[4] == 'o'`. El número `i` en `S[i]` se denomina índice

Es importante recordar que en Python no hay un tipo distinto para las letras, una letra suelta es una cadena con un único elemento.

Si se emplea un índice negativo se comienza a contar desde el final, empezando por el `-1`. es decir `S[-1] == 'o'`, `S[-2] == 'l'`, `S[-3] == 'l'`, `S[-4] == 'e'`, `S[-5] == 'H'`.

Veamos los ejemplos en una tabla para comparar

String S	H	e	l	l	o
Index	S[0]	S[1]	S[2]	S[3]	S[4]
Index	S[-5]	S[-4]	S[-3]	S[-2]	S[-1]

Si el índice en el fragmento `S[i]` es mayor o igual a `len(S)`, o menor que `-len(S)`, aparecerá el siguiente error `IndexError: string index out of range`.

2.2.- Substrings.

Un fragmento con dos parámetros `S[a:b]` devuelve una subcadena de longitud `b - a`, empezando por el símbolo de índice `a` y terminando por el de índice `b`, sin incluir el último. Por ejemplo, `S[1:4] == 'ell'`y se tendría el mismo resultado usando `S[-4:-1]`

Se pueden mezclar índices positivos y negativos, por ejemplo, es una substring que incluye todos los caracteres menos el primero y el último. , `S[1:-1]`

Los fragmentos con dos parámetros nunca devuelven `IndexError`. Por ejemplo si `S == 'Hello'` el fragmento `S[1:5]` devuelve `'ello'`, y el resultado es el mismo incluso si el segundo índice es muy grande como en `S[1:100]`.

Si se omite el segundo parámetro (pero se mantiene los dos puntos), el fragmento va hasta ese final de la cadena. Por ejemplo, para quitar el primer símbolo de la cadena se podría emplear `S[1:]`.. Si se omite el primero se comienza desde el principio de la cadena, es decir, para quitar el último símbolo se puede usar `S[:-1]`. El fragmento `S[:]` es la misma cadena

2.3.- Subsecuencias.

Si se determina un fragmento con tres parámetros `S[a:b:d]`, el tercero de ellos especifica el tamaño del paso, exactamente igual que en la función `range()`n. En este caso solo se toman los símbolos : `a` `a + d`, `a + 2 * d` y así, hasta llegar (pero sin incluir) al de índice . Si el tercer parámetro es igual a 2 se toman uno de cada dos símbolos. Si el paso se toma como `-1`, los símbolos se toman en orden inverso. Por ejemplo, se le puede dar una vuelta a una cadena como. `S[::-1]`.

Actividad: comprobar los resultados del siguiente código

```
s = 'abcdefg'
print(s[1])
print(s[-1])
print(s[1:3])
print(s[1:-1])
print(s[:3])
print(s[2:])
print(s[:-1])
print(s[::2])
print(s[1::2])
print(s[::-1])
```

Actividad: comprobar que el tercer parámetro del fragmento se comporta igual que el tercer parámetro de la función `range()`:

```
 s = 'abcdefghijklm'
print(s[0:10:2])
for i in range(0, 10, 2):
    print(i, s[i])
print(s[::2])
print(s[1::2])
print(s[::-1])
```

2.4.- Inmutabilidad de las cadenas.

En python los métodos o fragmentos que se generen de una cadena nunca modifican la cadena original, por lo que si se quiere usar más adelante es necesario almacenarla en una nueva variable

3.- Métodos para cadenas.

Un método es una función que esta ligada a un objeto. Cuando se ejecuta el método se aplica al objeto y realiza alguna computación relacionada con el. Siempre se llaman como `object_name.method_name(arguments)`.. Por ejemplo en `s.find("e")` el método de cadenas `find()` se aplica a la cadena `s` con un argumento `"e"`.

3.1.- Métodos find() y rfind().

El método `find()` busca una subcadena, pasada como argumento dentro de la cadena desde la que se llama. La función devuelve el índice de la primera aparición de la subcadena. Si no está incluida devuelve -1.

```
 s = 'Hello'
print(s.find('e'))
# 1
print(s.find('ll'))
# 2
print(s.find('L'))
# -1
```

De forma similar el método `rfind()` devuelve el índice de la última aparición de la subcadena.

```
 s = 'abracadabra'
print(s.find('b'))
# 1
print(s.rfind('b'))
# 8
```

Si se llama a `find()` con tres argumentos `s.find(substring, left, right)`, la búsqueda se realiza en el fragmento `s[left:right]`.. Si se especifican solo dos como en `s.find(substring, left)`, , la búsqueda se realiza en el fragmento `s[left:]`, , esto es, empezando en el símbolo con el índice `left` .

El método `s.find(substring, left, right)` siempre devuelve el índice relativo a la cadena `s`, no al fragmento de búsqueda.

```
s = 'my name is bond, james bond, okay?'
print(s.find('bond'))
# 11
print(s.find('bond', 12))
# 23
```

3.2.- El método replace()

El método `replace()` reemplaza todas las apariciones de una subcadena dada con otra. Se llama como `s.replace(old, new)` y coge la cadena `S` y reemplaza todas las veces que aparece `old` por `new` Recordemos que si queremos usarlo más adelante hace falta almacenarlo en otra variable

Veamos un ejemplo.

```
print('a bar is a bar, essentially'.replace('bar', 'pub'))
# 'a pub is a pub, essentially'
```

Se puede emplear un tercer argumento `count`, y quedaría asi `s.replace(old, new, count)`. Esto hace que se reemplace solo las primeras `count` veces que aparece `old` y se detenga después

```
print('a bar is a bar, essentially'.replace('bar', 'pub', 1))
# 'a pub is a bar, essentially'
```

3.3.- El método count()

Este método cuenta el número de veces que aparece una cadena dentro de otra cadena `s.count(substring)`.. Solamente se cuenta apariciones que no se solapen

```
print('Abracadabra'.count('a'))
# 4
print(('aaaaaaaaaa').count('aa'))
# 5
```

Si se especifican tres parámetros `s.count(substring, left, right)` la cuenta se realiza en el fragmento `s[left:right]`.

Ejercicios:

P1 Fragmentos

Dada una cadena
En la primera línea devolver el tercer símbolo de la cadena
En la segunda, imprimir el penúltimo símbolo.
En la tercera, imprimir los cinco primeros caracteres.
En la cuarta, imprimir la cadena sin sus dos últimos símbolos
En la quinta, imprimir los símbolos con índice par (recordar que se empieza en 0)
En la sexta imprimir los impares.
En la séptima imprimir la cadena del revés.
En la octava imprimir uno de cada dos símbolos en orden inverso, empezando por el último.
En la novena imprimir la longitud de la cadena.

P2 Número de palabras

Dada una cadena consistente en palabras separadas por espacios, determinar cuantas palabras tiene. Emplear el método count().

P3 Número de palabras

Dada una cadena cortarla en dos partes de la misma longitud, si la cadena tiene longitud impar, el símbolo de la posición central debe ir a la primera cadena resultado.

Imprimir una nueva cadena en una sola línea en la que se intercambien la primera y la segunda mitad.

No se puede usar if en este programa.

P4 Intercambiar palabras

Dada una cadena consistente en exactamente dos palabras separadas por un espacio, imprimir una nueva cadena con las dos palabras intercambiadas.

No se debe usar if en este ejercicio

P5 Primera y última aparición

Dada una cadena que puede o no contener una letra en concreto, imprimir el índice de la primera y última aparición de esta letra. Si la letra aparece solo una vez dar su índice solo una vez. Si dicha letra no aparece no imprimir nada.

No se deben usar bucles en este ejercicio

P6 Segunda aparición

Dada una cadena que puede o no contener una letra en concreto, imprimir el índice de la segunda aparición de esta letra. Si la letra aparece solo una vez devolver -1 . Si dicha letra no aparece devolver -2

P7 Quitar el fragmento

Dada una cadena en la que aparece la letra “h” al menos dos veces, devolver la cadena habiendo quitado todos los símbolos que hay entre la primera aparición de la “h” y la última (incluyendo ambas)

P8 Invertir el fragmento

Dada una cadena en la que aparece la letra “h” al menos dos veces, devolver invertir la secuencia de símbolos entre la primera y la última aparición de “h”.

P9 Reemplazar la subcadena

Dada una cadena reemplazar en ella todas las apariciones del número 1 por la palabra “uno”

P10 Reemplazar la subcadena

Dada una cadena eliminar de ella el símbolo @

P11 Reemplazar en un fragmento

Dada una cadena cambiar en ella todas las “h” por “H”, salvo la primera y la última.