Universidad Peruana Cayetano Heredia
Facultad de Ciencias e Ingeniería
Departamento de Ciencias Exactas

Programación Avanzada 2025

Lab. 1.1. Procesamiento de String en Python

Agosto 20, 2025

Cree una carpeta de trabajo en el disco D, nómbrela con su apellido paterno seguido de su DNI. Ejemplo: LOPEZ12345

Tabla 1.1 Operaciones de Secuencias aplicables a String

Nombre	Operación	Significado para S="¡Hola Como Estas!"
Length	`len(str)`	`len(S) ---> 17`
Select	`S[i]`	`S[6] ---> 'C'`
Slice	`S[i:j]`	`S[11:16]---> "Estas"(desde i hasta j-1)`
	`S[i:j:k]`	`S[1:12:5]---> "HCE"`
Count	`S.count(chr)`	`S.count('o') ---> 3`
Index	`S.index(chr)`	`S.index('E') ---> 11`
Membership	`'chr' in S`	`'p' in S ---> False`
	`'chr' not in S`	`'p' not in S ---> True`
Concatenation	`S + W`	`S + "!!" ---> "¡Hola Como Estas!!!"`
	`n*S (S*n)`	`2*S-->"¡Hola Como Estas!¡Hola Como Estas!"`
Minimum Value	`min(S)`	`min(S) ---> " " (espacio)`
Maximum Value	`max(S)`	`max(S) ---> 't'`
Comparison	`S == W`	`S == "Hola" ---> False`

Tabla 1.2 Principales Métodos de String

Métodos para verificar el contenido de un String			
`str.isalpha()`	Devuelve True si `str` contiene solo letras	`S="Hello"`	`S.isalpha()--> True`
		`S="Hello!"`	`S.isalpha()--> False`
`str.isdigit()`	Devuelve True si `str` contiene solo dígitos	`S="789"`	`S.isdigit()--> True`
		`S="789W"`	`S.isdigit()--> False`
`str.islower()` `str.isupper()`	Devuelve True si `str` contiene solo letras minúsculas (mayúsculas)	`S="hello"`	`S.islower()--> True`
		`S="Hello"`	`S.isupper()--> False`
`str.lower()` `str.upper()`	Devuelve la versión minúsculas (mayúsculas) de `str`	`S="Hello!"`	`S.lower()-->"hello!"`
		`S="hello!"`	`S.upper()-->"HELLO!"`
Método de búsqueda en un String			
`str.find(w)`	Retorna el índice de la primera ocurrencia de `w` en `str`. Retorna -1 si no lo encuentra	`S="Hello!"`	`S.find('l')--> 2`
		`S="Goodbye"`	`S.find('l')--> -1`
Método para reemplazar el contenido de un String			
`str.replace(w,t)`	Retorna una copia de `str` con todas las ocurrencias de `w` reemplazadas con `t`	`S="Hello!"`	`S.replace('H','J')--> "Jello"`
		`S="Hello"`	`S.replace('ll','r')--> "Hero"`
Método para remover el contenido de un String			
`str.strip(w)`	Retorna una copia de `str` con todos los caracteres iniciales y finales que aparecen en `w` eliminados	`S=" Hello! "`	`S.strip(' !') --> "Hello"`
		`S="Hello\n"`	`S.strip('\n') --> "Hello"`
Método para dividir el contenido de un String			
`str.split(w)`	Devuelve una lista que contiene todas los string de `str` delimitado por `w`	`S="Lu,Chao"`	`S.split(',') --> ["Lu","Chao"]`

Recorrido de un string (cadena de caracteres)

Recorrer un string implica visitar uno a uno los elementos del string. Podemos recorrer un string con la ayuda de un bucle, de preferencia el bucle for, el cual nos permite recorrer de dos maneras. Por ejemplo, si deseamos determinar el número de caracteres espacio (" ") en una frase, podemos hacer lo siguiente:

```
space = " "
num_spaces = 0
phrase = input("Enter a sentence: ")
for k in range(0,len(phrase)):
    if phrase[k] == space:
        num_spaces = num_spaces + 1
print("El número de espacios en",'"'+phrase+'"',"es:",num_spaces)
```

El recorrido también se puede realizar sin usar los índices (recorrido por valor), usando la forma: `for chr in string`

```
space = " "
num_spaces = 0
phrase = input("Enter a sentence: ")
for chr in phrase:
    if chr == space:
        num_spaces = num_spaces + 1
print("El número de espacios en",'"'+phrase+'"',"es:",num_spaces)
```

Supongamos que `s1`, `s2`, `s3` y `s4` son cuatro string con los valores siguientes:

```
s1 = "Welcome to Python"
s2 = s1
s3 = "Welcome to Python"
s4 = "to"
```

¿Cuál es el resultado de las siguientes expresiones?

a. s1 == s2
b. s2.count('o')
c. id(s1) == id(s2)
d. id(s1) == id(s3)
e. s1 <= s4
f. s2 >= s4
g. s1 != s4
h. s1.upper()
i. s1.find(s4)
j. s1[:7]
k. s1[4 : 8]
l. 4 * s4
m. s1[::2]
n. max(s1)
o. min(s1)
p. s1[-4]
q. s1.lower()
r. s1.rfind('o')
s. s1.startswith("o")
t. s1.endswith("o")
u. s1.isalpha()
v. s1 + s1

Para cada uno de los siguientes ejercicios, elabore un programa en Python. Use los métodos de String según corresponda.

1. *Generando cadena de ADN*. Una cadena ADN es una cadena que contiene solamente los símbolos 'A', 'C', 'G' y 'T'. Así la cadena "ATGCTTCAGAAAGGTCTTACG" es una cadena ADN de longitud 21.
   Implemente una función que reciba el tamaño de una cadena y genere de manera aleatoria una cadena ADN del tamaño especificado. La función debe verificar que el tamaño de la cadena ADN debe estar entre 20 y 1000 ambos inclusive. Use las funciones `choice()`, `join()`.

```
Enter the string length [20, 1000]: 25
The DNA string is: CTCGATGCTTCAGAAAGGTCTTACG

Enter the string length [20, 1000]: 10
¡ERROR!, the size must be between 20 and 1000
```

2. (*Bioinformática: encontrar genes*) Los biólogos usan secuencias de las letras **A**, **C**, **T** y **G** para modelar un *genoma*. Un *gen* es una subcadena de un genoma que comienza después de una tripleta **ATG** y termina antes de una tripleta **TAG**, **TAA** o **TGA**. Además, la longitud de una cadena gen es un múltiplo de 3 y el gen no contiene ninguna de las tripletas **ATG**, **TAG**, **TAA** y **TGA**. Implemente una función que reciba un genoma y muestre todos los genes del genoma. Si no se encuentra ningún gen en la secuencia de entrada, la función muestra el mensaje **`no se encuentra ningún gen`**.

```
Enter a genome string: TTATGTTTTAAGGATGGGGCGTTAGTT
TTT
GGGCGT
```

3. Una frase en el idioma español puede contener vocales acentuadas, la letra ñ y la u con diéresis. Implemente una función que reciba una frase en español y devuelva la frase con las vocales acentuadas reemplazadas por las mismas vocales sin acentuar, la ñ por n, y la u con diéresis por u.

```
Ingrese una frase:   En España no hay pingüinos, en Bélgica tampoco
La frase modificada: En Espana no hay pinguinos, en Belgica tampoco
```

4. Un simple y muy antiguo método de enviar mensajes secretos es la ***encriptación por sustitución***. Básicamente, cada letra del alfabeto se reemplaza por otra letra del alfabeto, para lo cual se tiene dos conjuntos de letras, un conjunto es el ***alfabeto*** con sus 26 letras ordenadas y el otro conjunto es la ***llave***, que es el mismo alfabeto con sus letras **reordenadas al azar**.
   Para realizar la encriptación, cada letra del mensaje original se ubica en el alfabeto ordenado y se sustituye por la letra correspondiente en la llave. Ejemplo:

```
Alfabeto: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Llave:    J V K M L O N I B H D C R E F Z U P Y G X A S Q W T
Mensaje original:   José, La Respuesta es SÍ.
Mensaje encriptado: HFYL, CJ PLYZXLYGJ LY YB.
```

   Implemente una función que reciba un texto original y una llave, y devuelva el texto encriptado.

OPCIONAL:

5. Resuelva el ejercicio N° 1, sin usar las funciones choice() y join(). Implemente un bucle que de la cantidad de vueltas necesarias para formar la cadena. En cada vuelta del bucle, genere un número entero aleatorio que corresponda a una de las letras 'A', 'C', 'G', 'T', concatene la letra generada en un acumulador.

6. Resuelva el ejercicio N° 3 sin usar funciones de transformación de caracteres. Use un bucle que recorra la frase en español, a medida que va visitando cada letra va generando la cadena de salida.

7. Escriba una función que reciba una frase cuyas palabras estén separadas por un espacio y muestre la frecuencia de cada carácter en la cadena.

Guarde todos vuestros programas en una carpeta con el nombre su **Apellido** paterno seguido de vuestro **DNI**, luego comprima esta carpeta. Envíe este archivo a:
Katherine Navarro katherine.navarro@upch.pe especificando como asunto **Lab1.1**.