UVA6: Ejercicios propuestos

Objetivos:

- Resolver problemas que requieran del uso de strings, incluidos los que necesitan comparar strings lexicográficamente.
- Procesar strings (aplicando iteración con WHILE y FOR sobre los caracteres que los forman) para buscar patrones y/o construir otros strings.

Ejercicios de Andamiajes

1.- Dado el siguiente Texto:

```
frase = 'Solo vuela el que se atreve a hacerlo'
```

Responda las siguientes consultas:

a. ¿Cuál es la longitud del texto? ¿Cuál es el operador que nos permite obtenerlo?

b. Cuáles son los índices válidos para obtener los caracteres del texto anterior.

c. Obtenga el último caracter de cualquier frase, en términos de su longitud

d. Indique cómo podemos recorrer caracter a caracter un texto/frase dado. ¿Conoces alguna forma alternativa?

2.- Conversión de datos

a. Indique cómo podemos convertir a entero los siguientes alternativas:
   i. “12345”
   ii. “doce mil trescientos cuarenta y cinco”

3.- Orden lexicográfico

a. Indique los resultados que producen las siguientes comparaciones:

   i. `'perro' < 'gato'`
   ii. `'perro' == 'Perro'`
   iii. `'perro' < 'Perro'`

4.- Inmutabilidad

a. Indique qué es lo que imprimen las siguientes líneas de código

```
nombre = 'Juan Carlos Bodoque'
nombre.upper()
print(nombre)
```

b. `print(nombre[3])`

c. `nombre[3]='A'`

d. `'E' in 'Perro'`

e. `'pollo' in 'Vaca Pollo Cerdo Tomate Lechuga Repollo Zapallo'`

5.- Recuperación de Elementos

Considerando el siguiente texto:

```
texto = 'gato grande, negro y gordo'
```

Determine el resultado de las siguientes consultas

a. `texto[4]`

b. `len(texto[5:8])`

c. `texto[:4]`

d. `texto[-5:]`

6.- Recorrido de textos

Indique ¿Qué hace el siguiente programa?

```
texto = 'gato grande, negro y gordo'
for x in texto:
   if x in 'aeiou':
      print(x)
```

¿Cómo lo hacemos con while en vez de for?

7.- Ruteo

Rutear el siguiente programa para el texto: “En un lugar de la Mancha”. Indicar además, con pocas palabras, qué es lo que hace o cuál es el objetivo del programa.

```
texto = input("Texto: ")
inicio = True
convertido = ""
for c in texto:
    if inicio:
          convertido = convertido + c.upper()
          inicio = False
    elif c == "  ":
          inicio = True
    else:
          convertido = convertido + c
print(convertido)
```

8.-
Hacer un programa que saluda a los usuarios de la UTFSM. Para ello, solicita una dirección de email y extrae el nombre de usuario y el dominio (lo que va después del @). Si es un usuario del dominio “usm.cl”, salúdalo por su nombre de usuario. En caso que el dominio no corresponda al mencionado, indica que el nombre de usuario no está permitido.

Ejercicios a desarrollar (clase2)

1.- (20 ptos) Realice el ruteo del siguiente programa e indique qué es lo que imprime. Cada vez que el valor de una variable cambie, escríbalo en una nueva fila de la tabla. Recuerde que si una variable es de tipo STRING, debe colocar su valor entre comillas simples ‘ ‘. Importante: La tabla tiene suficientes filas.

```
a = "b"
b = 1
s = b + 2
while(b < s and a[-1] != "a"):
    if(b % 2 != 0):
        b += 1
    s += b
    if(s % 4 != 0):
        s -= 1
    else:
        a += "a"
    print s
```

A	b	S

2. [25 %] En el básquetbol existen tres diferentes tipos de anotaciones:

- el tiro libre (`L`), que vale un punto,
- el doble (`D`), que vale dos puntos, y
- el triple (`T`), que vale tres puntos.

Un partido de básquetbol está dividido en varios períodos.

Usted debe escribir un programa que reciba como entrada una única línea, que contenga todas las anotaciones realizadas por un equipo de básquetbol durante un partido. Las anotaciones de períodos distintos deben ir separadas por un espacio. Como salida, debe mostrar la cantidad de puntos obtenidos en cada período y los puntos totales, siguiendo el formato del ejemplo.

```
Anotaciones: DDTDLLDD DDLDT TDTLLD DDDDD
15 puntos en el periodo 1
10 puntos en el periodo 2
12 puntos en el periodo 3
10 puntos en el periodo 4
Total: 47 puntos
```

En el idioma de la tribu de los *Stringones*, la mayoría de las palabras tienen muchas letras que se repiten de manera consecutiva. El sabio de la tribu ideó un sistema para escribir las palabras de manera abreviada: cada letra aparece antecedida de un número, indicando cuántas veces está repetida.

Por ejemplo, la palabra `ppppprrrrrogggrraaa` se abrevia `4p5r1o3g2r3a`.

Desarrolle la función `original(abreviada)` que recibe como parámetro una palabra abreviada, y retorna la palabra original, antes de haber sido codificada. Suponga que ninguna letra aparece más de nueve veces seguidas.

Ejemplo:

```
>>> original('4p5r1o3g2r3a')
ppppprrrrrogggrraaa
```

10. Dado un string con el siguiente formato, pero del que desconocemos la cantidad de asignaturas: `"Progra=78;Mate=83;Física=68;Química=65"`. Escriba un programa que lea el string como entrada y calcule el promedio de calificaciones, indicando además la materia con mejor promedio. En caso de empate puede mostrar cualquiera de las que empatan.

```
"Progra=78;Mate=83;Física=68;Química=65"
```

11.- Realice el ruteo del siguiente programa e indique qué es lo que imprime. Cada vez que el valor de una variable cambie, escríbalo en una nueva fila de la tabla. Recuerde que si una variable es de tipo STRING, debe colocar su valor entre comillas simples ‘ ‘. Importante: La tabla tiene suficientes filas.

```
x = "2d4e1f"
i=0
t=""
n=0
while i < len(x):
    if i%2 == 0:
        n = int(x[i])
    else:
        t = t + x[i] * n
    i = i + 1
print(t)
```

2.- (30 ptos) Escriba un programa que le pida al usuario 2 palabras e indique si algunas de las letras de la primera palabra se pueden re-ordenar para escribir la segunda palabra. Para escribir la segunda palabra debe existir la misma cantidad de letras en la primera palabra, es decir si en la segunda palabra hay 3 letras "a", en la primera palabra también debe existir la misma cantidad de letras "a", de lo contrario no se podría escribir.

NOTA: Se le recomienda usar la función index().

```
Palabra 1: alpargata
Palabra 2: pata
Se puede
```

```
Palabra 1: h3llko
Palabra 2: hello
No se puede
```

2. Escriba la función `adaptar(carta, palabra)`, que recibe como parámetros dos *strings*. El primer parámetro corresponde al texto de una carta, en la que podría aparecer el caracter `%`. La función debe retornar un string que contenga la carta, habiendo reemplazado todas las ocurrencias de `%` con la palabra indicada en el segundo parámetro. Si la carta no contiene el caracter `%` la función retorna el string original.
Ejemplos:

```
>>> print(adaptar('Estimado %, ¿cómo estás?. Te extraño amigo %!','Edsger'))
Estimado Edsger, ¿cómo estás?. Te extraño amigo Edsger!
>>> print(adaptar('Estimado %, ¿cómo estás?. Te extraño amigo %!','Alan'))
Estimado Alan, ¿cómo estás?. Te extraño amigo Alan!
>>> print(adaptar('Hola, mundo!','USM'))
Hola, mundo!
```

```
print(adaptar('Estimado %, ¿cómo estás?. Te extraño amigo %!','Edsger'))
print(adaptar('Estimado %, ¿cómo estás?. Te extraño amigo %!','Alan'))
```

1. Los mensajes que se envían a través de la red social *Twitter*, que se denominan *tweets*, incorporan términos conocidos como *hashtags*, que son palabras claves utilizadas para relacionar un mensaje con ciertas temáticas. Los *hashtags* son palabras que comienzan con el caracter `#`. Por ejemplo: `#covid19`, `#juntossaldremosdeesta`, `#usm`, `#sansano`, etc.

(a) Escriba la función `extraer_temas(tweet)` que recibe como parámetro un *string* que contiene un *tweet*. La función debe retornar un *string* que contiene los *hashtags* del mensaje original, sin el caracter `#` y separados por espacios en blanco. Si el mensaje original no contenía ningún *hashtag*, la función debe retornar un *string* vacío.
Ejemplos:

```
>>> print(extraer_temas('Hoy saqué a pasear a mis perros'))

>>> print(extraer_temas('Lo mejor de pandemia #covid19 son las clases #usm'))
covid19 usm
```

(b) Escriba la función `aparece_como_hashtag(tweet, palabra)` que recibe como parámetros un *string* que contiene un *tweet* y un *string* que contiene una palabra cualquiera. La función debe retornar `True` si el *tweet* contiene como *hashtag* la palabra indicada en el parámetro, y `False` en caso contrario.
Ejemplos:

```
>>> print(aparece_como_hashtag('Saqué mis perros #doglover','doglover'))
True
>>> print(aparece_como_hashtag('Saqué mis perros doglover','doglover'))
False
```

Observe que en segundo ejemplo la función retornó `False` a pesar de que la palabra se encontraba en el *tweet*, pero no aparecía como *hashtag*.

1. Escriba una función que evalúe si una palabra es un palíndromo o no, retornando True o False según corresponda. Por ejemplo, la función debería retornar True cuando se llame con el string “arenera” como parámetro.

2. Mejore la función anterior para que evalúe palíndromos compuestos por varias palabras e ignorando diferencias de mayúsculas y minúsculas, así como los espacios en blanco que separan las palabras. Por ejemplo, la función debería retornar True cuando se llame con el string “Amad a la dama” como parámetro.

3. Escriba una función que retorne un string con las letras que coinciden (la misma letra en la misma posición) en dos strings pasados como parámetro. Por ejemplo, “amorosos” y “amortiza” coinciden en: “amor”; por otra parte, “conformidad” y “contorno” coinciden en “conor”. Observe que los strings pueden tener distintos largos.

4. Una cadena de ADN es válida si está compuesta únicamente por las bases Adenina (A), Citosina (C), Guanina (G) o Timina (T). Escriba una función para validar una cadena de ADN, retornando True si es válida y False en caso contrario.

Suponga que la cadena está compuesta por múltiples grupos de 4 letras separados por guiones. Adicionalmente escriba un programa que lea “n” cadenas, y vaya señalando las que son válidas. Al finalizar debe decir cuántas fueron válidas y cuántas no.

5. Fechas mágicas: Una fecha mágica es una fecha en la que el día multiplicado por el mes es igual a los últimos dos dígitos del año. Por ejemplo, el 10 de junio de 1960 es una fecha mágica, pues junio es el mes 6, y al multiplicarlo por 10 el resultado coincide con el año 60. Escriba una función que determine si una fecha es mágica o no, retornando True o False según corresponda. La fecha que recibe la función es un string en formato "mes dia, año", con un espacio separando el mes y el día, y una coma y un espacio separando el año. Por ejemplo, “Junio 10, 1960”. Puede utilizarse, una función que convierte un mes escrito en palabras a su equivalente en número entero.

6. Escriba una función que reciba dos strings a comparar y un nivel de tolerancia que es un número entero no negativo (cero o más). La función debe retornar True si los strings son iguales ignorando diferencias hasta la cantidad de tolerancia indicada. Por ejemplo, “perro” y “perXo” son iguales para tolerancia 1, pero son distintos para tolerancia 0.

7.- Escriba una función que retorne True si una contraseña particular es suficientemente segura y False en caso contrario. Una contraseña se considera suficientemente segura si contiene al menos una letra mayúscula, al menos una minúscula, al menos un dígito, al menos un caracter de puntuación (punto, coma, punto y coma o dos puntos), y debe tener al menos longitud 8.

8.- Modifique la función anterior para que reciba como parámetro adicional la contraseña anterior. Agregue como condición para ser una contraseña segura que la nueva contraseña no debe ser similar a la anterior con 3 o menos caracteres de tolerancia