GUÍA N°5: STRINGS

FACULTAD DE
INGENIERÍA
UNIVERSIDAD DE SANTIAGO DE CHILE

1. Construya un programa en Python que cuente la cantidad de vocales y consonantes en un texto.

In [ ]:
```
# Complete aquí con su solución 😃
```

2. Construya un programa en Python que indique si un texto entregado es palíndromo o no, considere que un palíndromo es una palabra que se lee igual de derecha a izquierda y de izquierda a derecha.

In [ ]:
```
# Complete aquí con su solución 😃
```

3. Construya un programa que encuentre la palabra más larga en un texto, considere que todas las palabras estarán separadas por espacio. Una vez resuelto el ejercicio, intente resolverlo sin utilizar listas.

In [ ]:
```
# Complete aquí con su solución 😃
```

4. Construya un programa que reciba como entrada una oración y una palabra e indique si la palabra está en el texto o no. No puede usar el operador in para este ejercicio.

In [ ]:
```
# Complete aquí con su solución 😃
```

5. Construya un programa en Python que reemplace un caracter (A) por otro (B) en un texto, para ello solicite las entradas:

- Un string, representando el texto.
- El caracter a buscar (A).
- El caracter a reemplazar (B).

Indicación: No puede usar `.replace()` o similes para su solución.

In [ ]:
```
# Complete aquí con su solución 😄
```

6. **(EJERCICIO PEP)** Cuándo escribimos algo erróneo en nuestros teléfonos, normalmente este tiende a corregirnos, sugiriéndonos la palabra que cree, se acerca más a nuestra palabra errónea. Una forma de determinar qué palabra es más cercana a la que hemos escrito, es la distancia Hamming. Esta métrica se obtiene calculando cuántos caracteres de diferencia tienen palabras de un mismo largo, por ejemplo:

- La distancia Hamming entre 101**1**01**00**1 y 101**0**01**11**1 es 3, pues hay 3 caracteres que difieren.
- La distancia Hamming entre **t**ener y **T**ener es 1, pues hay 1 caracter que difiere.
- La distancia Hamming entre **v**al**o**r**a**r y **r**ar**o**l**a**v, es 4, pues hay 4 caracteres que difieren.

Utilizando la distancia Hamming, un teléfono compara la palabra que escribimos con las del diccionario, y obtiene todas las palabras con la menor distancia Hamming posible y luego, basándose en nuestro estilo de escritura, y la posición de las letras en el teclado, nos sugiere la palabra más cercana.

A fin de implementar un sistema rudimentario de autocorrección, se le solicita a usted que construya una función en Python que reciba como entrada dos strings del mismo largo y entregue como resultado la distancia de Hamming entre ambos.

Restricción: No se pueden usar las palabras reservadas `for` `e` `in` para el desarrollo del ejercicio.

In [ ]:
```
# Complete aquí con su solución 😄
```

7. Construya un programa en Python que reciba como entrada un número y entregue su resultado en:

- Base binaria. (Símbolos 0 y 1)
- Base hexadecimal. (Símbolos 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A (10), B (11), C (12), D (13), E (14), F (15))

Por ejemplo, el número 124 en Base binaria se representaría como `1111100 = 1 * 2**6 + 1 * 2**5 + 1 * 2**4 + 1 * 2**3 + 1 * 2**2 + 0 * 2**1 + 0 * 2**0 = 64 + 32 + 16 + 8 + 4 + 0 + 0` y en base hexadecimal como: `7C = 7 * 16**1 + C * 16**0 = 7 * 16**1 + 12 * 16**0 = 112 + 12`

In [ ]:
```
# Complete aquí con su solución 😄
```

```
# Complete aquí con su solución 😉
```

8. **(EJERCICIO PEP)** Juanito es un lingüista que está investigando las redundancias en el lenguaje, para ello está intentando reducir las palabras a su mínima expresión, para ello ha consultado en sus referencias y un algoritmo que podría servirle es el de reducción de strings, el cuál es un algoritmo sencillo que funciona sólo con un par de reglas:

- Se pueden eliminar cualquier par de letras adyacentes, siempre y cuando estas sean iguales.
- Mientras existan dos letras iguales adyacentes, se deben seguir eliminando los pares hasta que no quede ningún par de letras iguales adyacentes.

Por ejemplo:

- Si la entrada fuese “aabcc”, el resultado sería “b”, pues se pueden eliminar las “aa” del inicio y las “bb” del final.
- Si la entrada fuese “murcielago”, el resultado sería “murcielago”, pues no hay letras iguales adyacentes para eliminar.
- Si la entrada fuese “aaabccddd”, el resultado sería “abd” pues:

  > - Se eliminan las “aa” del inicio, quedando “abccddd”
  >
  > - Se eliminan las “cc” del medio, quedando “abddd”
  > - Se elimina un par de “d”s, quedando finalmente “abd” que no puede reducirse más

- Si la entrada fuese “aabccbaa”, el resultado sería un string vacío pues.

  > - Se eliminan las “aa” del inicio, quedando “bccbaa”

  ```
  > * Se eliminan las “cc” del medio, quedando “bbaa”
  > * Se eliminan las “bb” del inicio, quedando “aa”
  > * Se elimina las “aa” restantes, quedando un string vacío
  ```

Construya el programa en Python que implementa la reducción de strings para cualquier palabra ingresada. Considerando que:

- Si el resultado es un string vacío el programa debería informarlo.
- El programa debe funcionar para cualquier caso que respete el formato de entrada, no solamente para los ejemplos aquí dados

In [ ]:
```
# Complete aquí con su solución 😃
```

9. **(EJERCICIO PEP)** Un panagrama es una oración que contiene todas las letras del alfabeto, por ejemplo, en español, la oración:

"Whisky bueno: ¡excitad mi pequeña y frágil vejez!"

Es un panagrama, por otro lado, la oración:

"Por un momento estuve en un extraño sillón de felpa chirriante"

No es un panagrama, pues no están presentes las letras b, g, h, j, k, q, w, x, y, z.

Construya un programa en Python que reciba como entrada un string e informe al usuario si este es un panagrama o no. Considere para su implementación que: • El texto recibido puede venir con mayúsculas y minúsculas • El programa recibirá sólo oraciones en inglés, como por ejemplo *"The quick brown fox jumps over the lazy dog"*, por lo que ni los tildes, ni la letra ñ deben considerarse.

In [ ]:
```
# Complete aquí con su solución 😃
```

10. Construya un programa en Python que muestre un triángulo de Pascal con n filas, dónde el valor de n sea un valor entregado por el usuario. Por ejemplo, si la entrada fuese n = 10, entonces la salida debería ser:

```
                    1
                  1   1
                1   2   1
              1   3   3   1
            1   4   6   4   1
          1   5  10  10   5   1
        1   6  15  20  15   6   1
      1   7  21  35  35  21   7   1
    1   8  28  56  70  56  28   8   1
1           9  36  84 126 126  84  36   9   1
```

In [ ]:
```
# Complete aquí con su solución 😃
```

12. **(EJERCICIO PEP)** Un proceso común en las matemáticas es el redondeo, mediante el cual se eliminan cifras significativas de un número representado de manera decimal para obtener un valor aproximado. El fin del redondeo es facilitar los cálculos matemáticos, aunque dicho proceso introduce error en los cálculos finales.

En la actualidad, diversas herramientas de cálculo ofrecen la función de redondear, las que reciben el valor a redondear y el número de cifras decimales deseadas para entregar como resultado el número redondeado.

Construya un programa en Python que reciba como entrada:

- un string representando un número decimal.
- la cantidad de cifras decimales deseadas.

Y entregue como resultado el valor redondeado por pantalla.

Considere que el número entregado siempre tendrá más cifras decimales que el valor de cifras decimales deseadas, y que NO puede usar la función round() en este ejercicio .

Considere los siguientes valores como ejemplos de cómo debería funcionar el programa:

Si el valor a redondear es 45.141592632889

- con 4 cifras decimales resultaría: 45.1416
- con 5 cifras decimales resultaría: 45.14159
- con 2 cifras decimales resultaría 45.14
- con 0 cifras decimales resultaría 45

In [ ]:
```

```

In [ ]:
```
# Complete aquí con su solución 😃
```

12. **(EJERCICIO PEP)** Pablo es un programador aficionado que está buscando trabajo, Pablo sabe que, para este tipo de trabajos, muchas veces es más importante el ingenio que la capacidad de escribir código, por lo tanto para asegurar ser

uno de los mejores aspirantes al puesto, ha decidido crear un programa que le permita codificar mensajes, para ello, ha inventado el siguiente algoritmo:

- Reemplazar cada 'a', por una 'b', cada 'b' por una 'c', cada 'c' por una 'd' y así sucesivamente, hasta reemplazar cada 'z' por una 'a', en el mensaje obtenido.
- Para cada caracter que esté ubicado en una posición prima en el mensaje del paso anterior, el caracter codificado debe duplicarse, es decir, si en la posición 3 del mensaje hay una 'b', en el mensaje codificado debería escribirse 'bb'.

Ayuda a Pablo, implementando su algoritmo en Python, considerando que:

- Las palabras siempre estarán escritas en inglés, es decir, no existen ñ's ni tildes y siempre las letras serán minúsculas.
- El programa aceptará como entrada un string, que puede contener espacios y símbolos especiales (los cuáles no se codifican) y entregará como salida el mensaje codificado.

In [ ]:
```
# Complete aquí con su solución 😃
```

13. **(EJERCICIO PEP)** Construya una función en Python que reciba como entrada un string, representando una palabra en minúscula y entregue como salida una lista de strings representando los elementos de dicha palabra haciendo la "la ola". Para generar "la ola" del string se debe generar en cada posición de la lista una copia del string, con la letra en dicha posición capitalizada.

Por ejemplo, si la entrada de la función fuese 'perro', la ola sería: `['Perro', 'pErro', 'peRro', 'perRo', 'perrO']`. Pues se genera una copia del string 'perro' con cada letra capitalizada en orden.

Considere para su solución que la entrada puede ser cualquier string compuesto por letras en minúsculas.

In [ ]:
```
# Complete aquí con su solución 😃
```

14. Construya un programa en Python que determine cuál es la letra que más se repite en un texto. Evite usar el método `.count()` para su solución.

In [ ]:
```
# Complete aquí con su solución 😃
```

15. Construya un programa en Python que reciba como entrada un número n y genere el patrón solicitado en cada caso, considere que, en todos los casos, se asume para el ejemplo que n = 5.

Intente imprimir cada patrón utilizando solamente un print()

```
a)
*****
*****
*****
*****
*****

b)
    *****
  +++++
  *****
+++++
*****

c)
*****
*   *
*   *
*   *
*****

d)
*
*0
*0*
*0*0
*0*0*

e)
    *
  **
  * *
*  *
*****
```

```
f)
*****
0000
  ***
  00
    *

g)
*********
*******
  *****
  ***
    *

h)
*********
*     *
  *   *
  * *
    *

i)
    *
  ***
  *****
*******
*********
*******
  *****
  ***
    *

j)
    +
    +
    +
    +
+++++++++
    +
```

```
    +
    +
    +

k)
*****
  ****
    ***
      **
        *
      **
    ***
  ****
*****

l)
```