

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN  
CCPG1043 - FUNDAMENTOS DE PROGRAMACIÓN  
PRIMERA EVALUACIÓN - I TÉRMINO 2023-2024/ Julio 7, 2023

**Nombre:** \_\_\_\_\_ **Matrícula:** \_\_\_\_\_ **Paralelo:** \_\_\_\_\_

Como participante de este examen, me comprometo a mantener altos estándares éticos y de integridad académica. Asumo la responsabilidad de realizar este examen de manera honesta, sin recurrir a prácticas de plagio, fraude o deshonestidad académica por medios físicos o electrónicos. Me comprometo a responder las preguntas con mis propias habilidades y conocimientos, sin recibir ayuda no autorizada. Además, garantizo que respetaré los derechos de propiedad intelectual y no divulgaré ni copiaré el contenido del examen. Entiendo que cualquier violación a este compromiso tendrá consecuencias disciplinarias y académicas. Acepto que el cumplimiento de este compromiso es esencial para mantener la integridad académica y la reputación de la institución.

\_\_\_\_\_  
Firma

**TEMA 1 (15 PUNTOS)**

Implemente la función `buscar_palindromos(mensaje)` que recibe un mensaje y retorna una lista con todas las palabras (**de 3 o más letras**) del mensaje que son palíndromos. Asuma que todas las palabras están en minúsculas y no existen signos de puntuación.

**Recuerde que un palíndromo es una cadena que se lee igual de izquierda a derecha o de derecha a izquierda. Por ejemplo: madam, ana, somos, reconocer, anilina.**

*Ejemplo de entrada*

`mensaje = "ana y yo somos amigos y trabajamos en la torre del radar"`

*Ejemplo de salida*

`['ana', 'somos', 'radar']`

**TEMA 2 (15 PUNTOS)**

Implemente un programa que:

1. Pida una palabra al usuario.
2. Pida al usuario una frase a la vez hasta que este ingrese la palabra "Final".
3. Elimine de cada frase las comas y los puntos. Asuma que las frases no tienen ningún otro signo de puntuación.
4. Al finalizar, muestre por pantalla cuántas veces se repitió la palabra ingresada en el paso 1 en las frases ingresadas en el paso 2. Al momento de contar asegúrese de considerar mayúsculas y minúsculas para que cuenten como la misma palabra. Ejemplo: 'hola' y 'Hola' son la misma palabra.

**Asegúrese de comparar palabras exactas. Ejemplo: "Hola" en "Estoy en Holanda" no cuenta.**

*Ejemplo de entrada*

Ingrese una palabra: hola

Ingrese una frase (Ingrese "Final" para terminar): Hola, estoy bien

Ingrese una frase (Ingrese "Final" para terminar): Hola. No me gusta saludar diciendo hola

Ingrese una frase (Ingrese "Final" para terminar): Final

*Ejemplo de salida*

La palabra "hola" se repitió 3 veces en las frases ingresadas.

### TEMA 3 (25 PUNTOS)

Implemente la función `siglas(frase, palabras_comunes)` que reciba una frase y una lista de palabras comunes. Asuma que tanto las palabras de la frase como de la lista están en minúsculas. La función debe generar una sigla utilizando las iniciales en mayúsculas de cada palabra. La función debe omitir de la sigla palabras comunes como "de", "y", "en", etc. (definidas en el parámetro `palabras_comunes`), a menos que sean la primera o la última palabra de la frase. Asuma que en la frase no existen signos de puntuación.

#### *Ejemplo de entrada*

```
frase = "el yin y el yang son conceptos filosóficos de la cultura china"
palabras_comunes = ["el", "y", "de", "la", "son"]
```

#### *Ejemplo de salida*

```
"EYYCFCC"
```

### TEMA 4 (25 PUNTOS)

Implemente la función `rotacion_parcial(numeros, k)` que recibe como parámetros una lista de números y un número entero positivo `k` mayor a cero y menor que el tamaño de la lista. La función realiza una rotación parcial hacia la izquierda de la lista, pero solo para los primeros `k` elementos y retorna la nueva lista.

#### *Ejemplo de entrada*

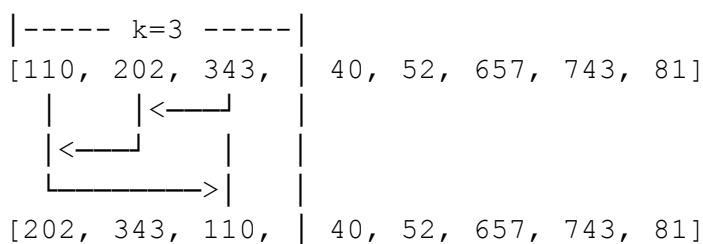
```
lista = [110, 202, 343, 40, 52, 657, 743, 81]
k=3
```

#### *Ejemplo de salida*

```
[202, 343, 110, 40, 52, 657, 743, 81]
```

#### *Explicación*

Para el ejemplo de `k=3`, solo los tres primeros elementos de la lista cambian su orden. El resto de elementos de la lista mantienen su orden original.



## TEMA 5 (20 PUNTOS)

Implemente la función `buscar_divisibles(numeros, divisor)` que recibe como primer parámetro una lista con números enteros y como segundo parámetro un número entero positivo mayor a 0. La función retorna una nueva lista con los números que son divisibles para el número recibido en el segundo parámetro.

Luego, implemente un programa principal y realice lo siguiente:

1. Genere una lista de 32 números aleatorios entre 5 y 342
2. Genere un número aleatorio entre 6 y 19.
3. Llame a la función con la lista y número generados en los pasos anteriores y muestre la lista resultante.

### Ejemplo de entrada

```
numeros_aleatorios = [228, 213, 277, 169, 230, 168, 110, 225, 58, 147, 104, 322, 249, 164, 178, 162, 192, 50, 61, 183, 198, 326, 102, 210, 90, 334, 30, 40, 331, 172, 337, 299]
divisor = 9
```

### Ejemplo de salida

```
[225, 162, 198, 90]
```

---//---

## Cheat Sheet. Funciones y propiedades de referencia en Python.

| para <b>listas</b> :                                                                                                                                                     | para <b>cadena</b> s:                                                                                                                                                                              | random as <b>rnd</b> :                                                                                                                                          |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>lista</b> .append(...)<br><b>lista</b> .extend(...)<br><b>lista</b> .count(...)<br><b>lista</b> .index(...)<br><b>lista</b> .pop()<br><b>elemento</b> in <b>lista</b> | <b>cadena</b> .lower()<br><b>cadena</b> .upper()<br><b>cadena</b> .split(...)<br><b>cadena</b> .index(...)<br><b>cadena</b> .find(...)<br><b>cadena</b> .count(...)<br><b>cadena</b> .replace(a,b) | <b>rnd</b> .randint(inicio,fin)<br><b>rnd</b> .choice(lista)<br><b>rnd</b> .choices(lista,cant)<br><b>rnd</b> .sample(lista,cant)<br><b>rnd</b> .shuffle(lista) |