Ejercicios con Funciones - Desarrolle el código necesario para solucionar la premisa de cada ejercicio.

1. Calcular el máximo de dos números:

- Función: `maximo(a, b)`
- Parámetros: a y b (números)
- Devuelve: El máximo de a y b

```
def maximo(a, b):
    return max(a, b)

print(f' El valor mas grande es: {maximo(4, 10)} ')
```

2. Calcular el área de un triángulo:

- Función: `area_triangulo(base, altura)`
- Parámetros: `base` y `altura` (números)
- Devuelve: El área del triángulo

```
def area_triangulo(base, altura):
    area = (base * altura) / 2
    return area


base = 4
altura = 7
resultado = area_triangulo(base, altura)
print("El area del triangulo con base", base, "y altura", altura, "es:", resultado)
```

3. Convertir una temperatura de Fahrenheit a Celsius:

- Función: `fahrenheit_a_celsius(fahrenheit)`
- Parámetro: `fahrenheit` (temperatura en Fahrenheit)
- Devuelve: La temperatura en Celsius

```
def fahrenheit_a_celsius(fahrenheit):
    celsius = (fahrenheit - 32) * 5/9
    return celsius


temperatura_fahrenheit = 69
temperatura_celsius = fahrenheit_a_celsius(temperatura_fahrenheit)
print("La temperatura", temperatura_fahrenheit, "Fahrenheit equivale a", temperatura_celsius, "Celsius.")
```

4. Promediar una lista de números:

- Función: `promedio(lista_numeros)`
- Parámetro: `lista_numeros` (lista de números)
- Devuelve: El promedio de la lista de números

```
# Python program to get average of a list
def Average(lst):
    return sum(lst) / len(lst)
```

```
def promedio(lista_numeros):
    suma = 0
    array_length = 0
    for valor in lista_numeros:
        suma = suma + valor #   suma += valor
        array_length = array_length + 1       #   ctr++
    return (suma / array_length)


# Version ChatGPT
lst = [15, 9, 55, 41, 35, 20, 62, 49]
average = Average(lst)
print(lst)
print(f'El promedio de la lista es {average}')


# Version del Curso
print(lst)
print(f'El valor promedio de la lista es {promedio(lst)}')
```

5. Encontrar el índice de la primera aparición de un elemento en una lista:

- Función: `encontrar_indice(lista, elemento)`
- Parámetros: `lista` (lista) y `elemento` (valor a buscar)
- Devuelve: El índice de la primera aparición del elemento en la lista, o -1 si no se encuentra

```
def encontrar_indice(lista, elemento):
    try:
        indice = lista.index(elemento)
        return indice
    except ValueError:
        return -1


mi_lista = [10, 20, 30, 40, 50]
elemento_buscado = 30
resultado = encontrar_indice(mi_lista, elemento_buscado)
if resultado != -1:
    print("El elemento", elemento_buscado, "se encuentra en el índice:", resultado)
else:
    print("El elemento", elemento_buscado, "no se encuentra en la lista.")
```

6. Invertir una cadena de texto:

- Función: `invertir_cadena(cadena)`
- Parámetro: `cadena` (texto a invertir)
- Devuelve: La cadena de texto invertida

```
import array


def invertir_cadena(cadena):
    array_lenght = len(cadena)
    nueva_cadena = ''
    for i in range(array_lenght):
        nueva_cadena += (cadena[array_lenght-1-i])
    return nueva_cadena


mensaje = "el oso perezoso "
print(mensaje)
print(invertir_cadena(mensaje))
```

7. Validar si una cadena de texto es un número entero:

- Función: `es_numero_entero(cadena)`
- Parámetro: `cadena` (texto a validar)
- Devuelve: True si la cadena es un número entero, False si no lo es

```
def es_numero_entero(cadena):
    try:
```

```
        int(cadena)
        return True
    except ValueError:
        return False

cadena1 = "123"
cadena2 = "12.34"
cadena3 = "abc"
print("¿La cadena '{}' es un número entero? {}".format(cadena1, es_numero_entero(cadena1)))
print("¿La cadena '{}' es un número entero? {}".format(cadena2, es_numero_entero(cadena2)))
print("¿La cadena '{}' es un número entero? {}".format(cadena3, es_numero_entero(cadena3)))
```

8. Contar la cantidad de vocales en una cadena de texto:

- Función: `contar_vocales(cadena)`
- Parámetro: `cadena` (texto a analizar)
- Devuelve: La cantidad de vocales en la cadena de texto

```
def contar_vocales(cadena):
    cadena = cadena.lower()
    vocales = ['a', 'e', 'i', 'o', 'u']
    contador = 0
```

```
    for caracter in cadena:
        if caracter in vocales:
            contador += 1
    return contador

texto = "Hello World"
cantidad_vocales = contar_vocales(texto)
print("La cantidad de vocales en la cadena '{}' es: {}".format(texto, cantidad_vocales))
```

9. Generar una lista de números aleatorios:

- Función: `generar_numeros_aleatorios(cantidad, minimo, maximo)`
- Parámetros: `cantidad` (de numeros), `minimo` y `maximo` (valores)
- Devuelve: Una lista con la cantidad de números aleatorios generados

```
import random

def generar_numeros_aleatorios(cantidad, minimo, maximo):
    numeros_aleatorios = []
    for _ in range(cantidad):
        numero = random.randint(minimo, maximo)
        numeros_aleatorios.append(numero)
    return numeros_aleatorios
```

cantidad_numeros = 5

minimo = 1

maximo = 100

numeros_generados = generar_numeros_aleatorios(cantidad_numeros, minimo, maximo)

print("Lista de números aleatorios generados:", numeros_generados)

10. Contar la cantidad de apariciones de una letra en una cadena de texto:

- Función: `contar_apariciones_letra(cadena, letra)`
- Parámetros: `cadena` (texto a analizar) y `letra` (letra a buscar)
- Devuelve: La cantidad de apariciones de la letra en la cadena de texto
  def contar_apariciones_letra(cadena, letra):

```
    contador = 0

    for caracter in cadena:
        if caracter == letra:
            contador += 1
    return contador
```

print(f'La cantidad de veces que aparece esta letra es: {contar_apariciones_letra("hola mundo", "o")}')

```
print(f'La cantidad de veces que aparece esta letra es:
{contar_apariciones_letra("John Doe likes Jane Doe", "e")}' )
```

Los siguientes ejercicios requieren el uso de arreglos. Desarrolle el código necesario para solucionar cada premisa. No puede utilizar funciones especiales del lenguaje Python.

1. Sumar los elementos de un arreglo:

- Crea un arreglo con números.
- Recorre el arreglo y suma cada elemento.
- Imprime la suma total.

```
arreglo = [1, 2, 3, 4, 5]

suma_total = 0

for elemento in arreglo:
    suma_total += elemento

print("La suma total es:", suma_total)
```

2. Encontrar el máximo elemento de un arreglo:

- Crea un arreglo con números.
- Recorre el arreglo y guarda el máximo elemento encontrado.
- Imprime el máximo elemento.

```
def mayor(lista):
    valor_mayor = lista[0]
    for n in lista:
        if (n > valor_mayor):
```

```
            valor_mayor = n
    return valor_mayor


lst = [10, 4, 3, 15, 9, 22, 1]
print(lst)
print(f'El valor mayor es {mayor(lst)}')
 #version chat gpt
def encontrar_maximo(arreglo):
    maximo_elemento = arreglo[0]
    for elemento in arreglo:
        if elemento > maximo_elemento:
            maximo_elemento = elemento
    return maximo_elemento

def imprimir_maximo(maximo):
    print(arreglo)
    print("El máximo elemento del arreglo es:", maximo)
arreglo = [5, 20, 2, 10, 3]
maximo = encontrar_maximo(arreglo)
imprimir_maximo(maximo)
```

3. Invertir el orden de los elementos de un arreglo:

- Crea un arreglo con números.
- Invierte el orden de los elementos del arreglo.
- Imprime el arreglo con el orden invertido.

```
arreglo = [1, 2, 3, 4, 5]
```

```
longitud = len(arreglo)

for i in range(longitud // 2):
    temp = arreglo[i]
    arreglo[i] = arreglo[longitud - i - 1]
    arreglo[longitud - i - 1] = temp

print("Arreglo con el orden invertido:", arreglo)
```

4. Buscar un elemento en un arreglo:

- Crea un arreglo con números.
- Busca un elemento específico en el arreglo.
- Imprime la posición del elemento encontrado o un mensaje si no se encuentra.

```
arreglo = [5, 10, 15, 20, 25]

elemento_buscado = 15

posicion = None

for i in range(len(arreglo)):
    if arreglo[i] == elemento_buscado:
        posicion = i
        break

if posicion is not None:
    print("El elemento", elemento_buscado, "se encuentra en la posición:", posicion)
```

```
else:
    print("El elemento", elemento_buscado, "no se encuentra en el arreglo.")
```

5. Eliminar un elemento de un arreglo:

- Crea un arreglo con números.
- Elimina un elemento específico del arreglo.
- Imprime el arreglo con el elemento eliminado.

```
arreglo = [10, 20, 30, 40, 50]

elemento_eliminar = 30

nuevo_arreglo = []

for elemento in arreglo:
    if elemento != elemento_eliminar:
        nuevo_arreglo.append(elemento)

print("Arreglo con el elemento", elemento_eliminar, "eliminado:",
nuevo_arreglo)
```

6. Contar la cantidad de apariciones de un elemento en un arreglo:

- Crea un arreglo con números.
- Cuenta la cantidad de veces que aparece un elemento específico en el arreglo.
- Imprime la cantidad de apariciones del elemento.

```
arreglo = [1, 2, 3, 4, 5, 2, 3, 2, 1]
```

```
elemento_buscar = 2

contador = 0

for elemento in arreglo:
    if elemento == elemento_buscar:
        contador += 1

print("El elemento", elemento_buscar, "aparece", contador, "veces en el arreglo.")
```

7. Calcular el promedio de los elementos de un arreglo:

- Crea un arreglo con números.
- Calcula el promedio de los elementos del arreglo.
- Imprime el promedio.

```
arreglo = [10, 20, 30, 40, 50]

suma = 0
cantidad_elementos = 0

for elemento in arreglo:
    suma += elemento
    cantidad_elementos += 1

if cantidad_elementos != 0:
    promedio = suma / cantidad_elementos
    print("El promedio de los elementos del arreglo es:", promedio)
```

```
else:
    print("No se puede calcular el promedio. El arreglo está vacío.")
```

8. Ordenar los elementos de un arreglo de menor a mayor:

- Crea un arreglo con números.
- Ordena los elementos del arreglo de menor a mayor.
- Imprime el arreglo ordenado.

```
for i in range(1, len(arreglo)):
    valor_actual = arreglo[i]
    j = i - 1
    while j >= 0 and arreglo[j] > valor_actual:
        arreglo[j + 1] = arreglo[j]
        j -= 1
    arreglo[j + 1] = valor_actual

print("Arreglo ordenado de menor a mayor:", arreglo)
```

9. Combinar dos arreglos en uno solo:

- Crea dos arreglos con números.
- Combina los dos arreglos en uno solo.
- Imprime el arreglo combinado.

```
arreglo1 = [1, 2, 3]
arreglo2 = [4, 5, 6]

arreglo_combinado = []

for elemento in arreglo1:
```

```
    arreglo_combinado.append(elemento)

for elemento in arreglo2:
    arreglo_combinado.append(elemento)

print("Arreglo combinado:", arreglo_combinado)
```

10. Rotar los elementos de un arreglo:

- Crea un arreglo con números.
- Rota los elementos del arreglo una posición a la derecha.
- Imprime el arreglo con los elementos rotados.

```
arreglo = [1, 2, 3, 4, 5]

ultimo_elemento = arreglo[-1]

for i in range(len(arreglo) - 1, 0, -1):
    arreglo[i] = arreglo[i - 1]

arreglo[0] = ultimo_elemento

print("Arreglo con los elementos rotados:", arreglo)
```