



# ESCUELA POLITÉCNICA NACIONAL

## ESCUELA DE FORMACIÓN DE TECNÓLOGOS



### ALGORITMOS Y ESTRUCTURAS DE DATOS (TSDS)

ASIGNATURA:

ALGORITMOS Y ESTRUCTURAS DE DATOS

PROFESOR:

Ing. Lorena Chulde MSc.

PERÍODO ACADÉMICO:

2023-B

### TAREA

### Grupal

### TÍTULO:

### ARREGLOS

Nombres de los estudiantes:

**Guerra Lovato Josué**

**Pérez Orosco Carlos**

**Soria Ansa Richard**



**2023-B**

## PROPÓSITO DE LA TAREA

Reutilizar el código mediante funciones para una programación óptima.

### OBJETIVO GENERAL

Se tienen como objetivo profundizar en la capacidad de los arreglos para almacenar y manipular datos eficientemente, concentrándose en contextos específicos como la visualización de datos, el procesamiento de imágenes y la implementación de algoritmos avanzados de análisis numérico. Para evitar cualquier riesgo de similitud en sus trabajos, planea abordar este objetivo mediante el estudio autónomo, consultando fuentes académicas, participando activamente en foros especializados y, sobre todo, aplicando los conocimientos adquiridos en proyectos prácticos originales.

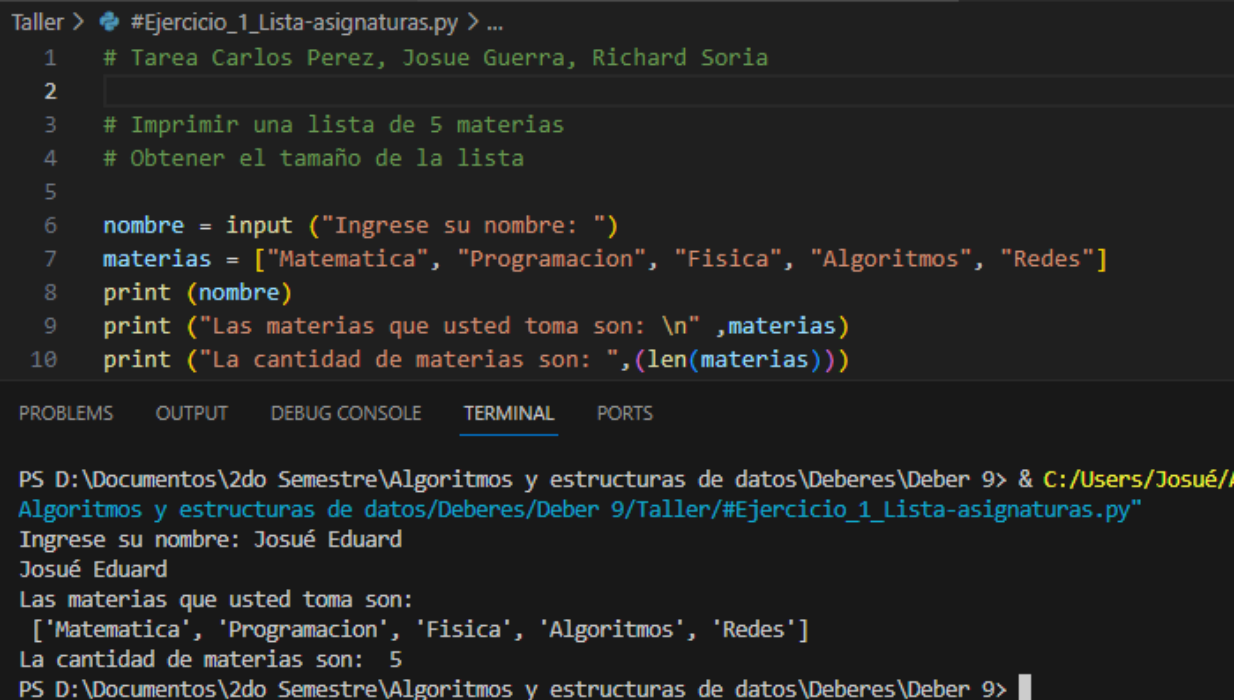
### OBJETIVOS ESPECÍFICOS

- Desarrollar habilidades avanzadas en la manipulación de arreglos.
- Explorar la aplicación de arreglos en la visualización de datos
- Profundizar en el uso de arreglos para el procesamiento de imágenes

### Parte I

#### TALLER:

1. Programa que imprime las asignaturas definidas en una lista.



```
Taller > #Ejercicio_1_Lista-asignaturas.py > ...
1  # Tarea Carlos Perez, Josue Guerra, Richard Soria
2
3  # Imprimir una lista de 5 materias
4  # Obtener el tamaño de la lista
5
6  nombre = input ("Ingrese su nombre: ")
7  materias = ["Matematica", "Programacion", "Fisica", "Algoritmos", "Redes"]
8  print (nombre)
9  print ("Las materias que usted toma son: \n" ,materias)
10 print ("La cantidad de materias son: ",(len(materias)))

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> & C:/Users/Josué/Algoritmos y estructuras de datos/Deberes/Deber 9/Taller/#Ejercicio_1_Lista-asignaturas.py
Ingrese su nombre: Josué Eduard
Josué Eduard
Las materias que usted toma son:
['Matematica', 'Programacion', 'Fisica', 'Algoritmos', 'Redes']
La cantidad de materias son: 5
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> |
```

## 2. Imprimir el tamaño de una lista dada.

```
Taller > #Ejercicio_2_tamaño-de-lista.py > ...
1  # Tarea Carlos Perez, Josue Guerra, Richard Soria
2
3  # Arreglos
4  # Imprimir el tamaño de una lista dada
5
6  frutas = ["manzana","fresa","pera","uva"]
7  print ("La lista es: ",frutas)
8  print ("El tamaño de la lista es: ",len (frutas))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> & C:/Users/
Algoritmos y estructuras de datos/Deberes/Deber 9/Taller/#Ejercicio_2_tamaño-de-lista.py"
La lista es: ['manzana', 'fresa', 'pera', 'uva']
El tamaño de la lista es: 4
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9>
```

## 3. Programa que imprime las asignaturas con for (elemento por elemento)

```
Taller > #Ejercicio_3_lista-elemento-por-elemento.py > ...
1  # Tarea Carlos Perez, Josue Guerra, Richard Soria
2
3  # Imprimir una lista de 5 materias
4  # Obtener el tamaño de la lista
5  # Imprimir asignatura por asignatura
6
7  nombre = input ("Ingrese su nombre: ")
8  materias = ["Matematica", "Programacion", "Fisica", "Algoritmos", "Redes"]
9  print (nombre)
10 print ("Las materias que usted toma son: \n",materias)
11 print ("La cantidad de materias son: ",(len(materias)))
12
13 for materia in materias:
14     print (materia)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> & C:/Users/Josué/AppD
Algoritmos y estructuras de datos/Deberes/Deber 9/Taller/#Ejercicio_3_lista-elemento-por-elemento.py"
Ingrese su nombre: Josué
Josué
Las materias que usted toma son:
['Matematica', 'Programacion', 'Fisica', 'Algoritmos', 'Redes']
La cantidad de materias son: 5
Matematica
Programacion
Fisica
Algoritmos
Redes
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> |
```

#### 4. Unir dos listas y mostrar en una tercera lista

```
Taller > #Ejercicio_4_Unir-dos-listas-en-tercera.py > ...
1  # Tarea Carlos Perez, Josue Guerra, Richard Soria
2
3  # Unir dos listas
4  # Imprimir en una tercera
5
6
7  estudiantes = ["Carlos","Josue","Adrian","Richard","Joshua","Julian"]
8  notas = [7,8,9,9,8,7]
9  union_listas = []
10
11  union_listas = estudiantes + notas
12  print (union_listas)
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> & C:/Users/Josué/AppData/Local/Programs/Python/Python39-6/Scripts/python.exe C:/Users/Josué/AppData/Local/Programs/Python/Python39-6/Scripts/python.exe D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9/Taller/#Ejercicio_4_Unir-dos-listas-en-tercera.py
['Carlos', 'Josue', 'Adrian', 'Richard', 'Joshua', 'Julian', 7, 8, 9, 9, 8, 7]
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9>
```

#### 5. Agregar un elemento quemado a una lista usando append()

```
Taller > #Ejercicio_5_Lista-Append.py > ...
1  # Tarea Carlos Perez, Josue Guerra, Richard Soria
2
3  # Agregar un elemento quemado a una lista usando append()
4
5  notas = [5,6,7,8,9]
6  nota_añadida = 10
7
8  print (notas)
9  notas.append (nota_añadida)
10
11  print (notas)
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> & C:/Users/Josué/AppData/Local/Programs/Python/Python39-6/Scripts/python.exe C:/Users/Josué/AppData/Local/Programs/Python/Python39-6/Scripts/python.exe D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9/Taller/#Ejercicio_5_Lista-Append.py
[5, 6, 7, 8, 9]
[5, 6, 7, 8, 9, 10]
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9>
```

## 6. Agregar un elemento pidiendo al usuario a una lista usando append()

```
Taller > #Ejercicio_6_Lista-Usuario.py > ...
1  # Tarea Carlos Perez, Josue Guerra, Richard Soria
2
3  # Agregar un elemento pidiendo al usuario a una lista usando append()
4  # Que el usuario ingrese los elementos
5
6  arreglo_usuario = []
7
8  for notas in range (5):
9      notas = int (input ("Ingrese la nota: "))
10     arreglo_usuario.append (notas)
11
12     print ("La lista de notas es: ",arreglo_usuario)
13     print ("El numero de notas ingresadas es: ", len(arreglo_usuario))
14
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> & C:/Users/Josue/AppData/Local/Programs/Python/Python39-64/Scripts/python.exe C:/Users/Josue/AppData/Local/Programs/Python/Python39-64/Scripts/python.exe D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9/Taller/#Ejercicio_6_Lista-Usuario.py
Ingrese la nota: 15
Ingrese la nota: 10
Ingrese la nota: 3
Ingrese la nota: 18
Ingrese la nota: 20
La lista de notas es: [15, 10, 3, 18, 20]
El numero de notas ingresadas es: 5
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> |
```

## 7. Insertar las asignaturas en la lista e imprimirlas.

```
Taller > #Ejercicio_7_Asignatura.py > ...
1  # Tarea Carlos Perez, Josue Guerra, Richard Soria
2
3  # Insertar las asignaturas en la lista e imprimirlas.
4
5  asignaturas = []
6
7  cantidad_asignaturas = int (input ("Ingrese la cantidad de asignaturas a añadir: "))
8
9  for materias in range (cantidad_asignaturas):
10     materias = (input ("Ingrese la asignatura: "))
11     asignaturas.append (materias)
12
13     print ("La lista de asignaturas es: ",asignaturas)
14     for materia in asignaturas:
15         print (materia)
16
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> & C:/Users/Josue/AppData/Local/Programs/Python/Python39-64/Scripts/python.exe C:/Users/Josue/AppData/Local/Programs/Python/Python39-64/Scripts/python.exe D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9/Taller/#Ejercicio_7_Asignatura.py
Ingrese la cantidad de asignaturas a añadir: 3
Ingrese la asignatura: Progrmación
Ingrese la asignatura: Algoritmos
Ingrese la asignatura: Frances
La lista de asignaturas es: ['Progrmación', 'Algoritmos', 'Frances']
Progrmación
Algoritmos
Frances
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> |
```

8. Crear una lista e inicialízala con 5 cadenas de caracteres. Copia los elementos de la lista en otra lista, pero en orden inverso, y muestra sus elementos por la pantalla.

```
Taller > #Ejercicio_8_Lista-inversa.py > ...
1  # Tarea Carlos Perez, Josue Guerra, Richard Soria
2
3  # Crear una lista e inicialízala con 5 cadenas de caracteres.
4  #Copia los elementos de la lista en otra lista pero en orden
5  #inverso, y muestra sus elementos por la pantalla.
6  # Imprimir la lista invversa
7
8  cadena_ingresada = []
9  cantidad = 5
10
11  for caracteres in range (cantidad):
12      caracteres = input ("Ingrese una palabra: ")
13      cadena_ingresada.append (caracteres)
14
15
16  print ("La lista original es: ",cadena_ingresada)
17  print ("La lista inversa es: ", cadena_ingresada [::-1])
18
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> & C:/Users/Josué/Algoritmos y estructuras de datos/Deberes/Deber 9/Taller/#Ejercicio_8_Lista-inversa.py
Ingrese una palabra: acompañe.
Ingrese una palabra: te
Ingrese una palabra: la FUERZA
Ingrese una palabra: que
Ingrese una palabra: Algoritmos
La lista original es: ['acompañe.', 'te', 'la FUERZA', 'que', 'Algoritmos']
La lista inversa es: ['Algoritmos', 'que', 'la FUERZA', 'te', 'acompañe.']
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9>
```

9. Con función **sort**: ordenar un arreglo de elementos de menor a mayor

```
Taller > #Ejercicio_9_funcion-sort.py > ...
1  # Tarea Carlos Perez, Josue Guerra, Richard Soria
2
3  # funcion de python para ordenar la lista
4  # Con función sort: ordenar un arreglo de elementos de menor a mayor
5
6  enteros = [5,4,3,2,1,6,7,8,9]
7  enteros.sort ()
8  print (enteros)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> & C:/Users/Josué/Algoritmos y estructuras de datos/Deberes/Deber 9/Taller/#Ejercicio_9_funcion-sort.py
[1, 2, 3, 4, 5, 6, 7, 8, 9]
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9>
```

10. Crear una fn para recorrer una lista.

**Método burbuja:** Crear una función para ordenar elementos de un array de menor a mayor.

Cree una función llamada ordenar() (estudiar para la lección)

```
def ordenar(array):
    tamaño = len(array)
    #for para recorrer las posiciones de 0 a 5
    for i in range(0,tamaño-1):
        #con estbucle comparamos
        for j in range(0,tamaño-1):
            if array[j] > array[j+1]:
                aux = array[j]
                array[j] = array[j+1]
                array[j+1] = aux
    return array
```

```
numeros = [6,3,8,2,7]
print("la lista original es: ", numeros)
#ordenar(numeros)
print("la lista ordenada es: ", ordenar(numeros))
```

The screenshot shows a Python IDE with a file named `#Ejercicio_10_Funcion-orden-elementos.py`. The code implements a bubble sort function and demonstrates its use with a list of integers. The terminal output shows the original list and the sorted list.

```
Taller > #Ejercicio_10_Funcion-orden-elementos.py > ...
1  # Tarea Carlos Perez, Josue Guerra, Richard Soria
2
3  # funcion de pytho para ordenar la lista
4
5  enteros = [5,4,3,2,1,6,7,8,9]
6  enteros.sort ()
7  print (enteros)
8
9  # Realizar un algoritmo que orde los elementos de la lista
10 n = len(enteros)
11 def ordenar_lista (enteros):
12     for i in range (n):
13         for j in range (0, n-i-1):
14             if enteros[j] > enteros[j+1]:
15                 enteros[j], enteros[j+1] = enteros[j+1], enteros[j]
16
17 ordenar_lista(enteros)
18 print (enteros)
19
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

```
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> & C:/Users/Josué/A
Algoritmos y estructuras de datos/Deberes/Deber 9/Taller/#Ejercicio_10_Funcion-orden-elementos.py"
[1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 2, 3, 4, 5, 6, 7, 8, 9]
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9>
```

## Parte II

### TAREA:

1. Programa que almacena las asignaturas de un curso en una lista y la muestre por pantalla pide al usuario las notas

```
#Ejercicio1_Notas.py > ...
1  # Tarea Carlos Perez, Josue Guerra, Richard Soria
2
3  # Programa que almacena las asignaturas de un curso
4  # en una lista y la muestre por pantalla pide al usuario las notas
5  print("--Bienvenido al programa-- ")
6  asignaturas = []
7  calificaciones = []
8
9  cantidad_asignaturas = int (input ("Ingrese la cantidad de asignaturas que desea ingresar: "))
10
11  for materia in range (cantidad_asignaturas):
12      materia = input ("Ingrese la asignatura: ")
13      nota = (float (input ("Ingrese su calificacion: ")))
14
15      asignaturas.append (materia)
16      calificaciones.append (nota)
17
18  asignaturas_notas = asignaturas + calificaciones
19  print ("Las asignaturas con sus notas correspondientes son: ",asignaturas_notas)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> & C:/Users/Josué/AppData/Local/Microsoft/
ritmos y estructuras de datos/Deberes/Deber 9/#Ejercicio1_Notas.py"
--Bienvenido al programa--
Ingrese la cantidad de asignaturas que desea ingresar: 3
Ingrese la asignatura: Programación
Ingrese su calificacion: 18.56
Ingrese la asignatura: Algoritmos
Ingrese su calificacion: 18.51
Ingrese la asignatura: Ingles
Ingrese su calificacion: 19.25
Las asignaturas con sus notas correspondientes son: ['Programación ', 'Algoritmos', 'Ingles', 18.56, 18.51, 19.25]
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> |
```

- Dentro del programa se crea dos arreglos vacíos de inicio para asignar tanto lo que son las asignaturas como las calificaciones, de ahí se solicita para que ingrese la cantidad que el usuario desea de asignaturas, dependiendo de eso se guardara dentro del arreglo pero solo la cantidad escrita, en la ejecución se puede ver como el arreglo mismo guarda lo que son los nombres de las asignaturas, y también las notas
- Tomando en cuenta que nosotros no quemamos el número si no es lo que el usuario decide poner dentro del programa, por ende el arreglo actuara de otra manera si se quemara el número inicialmente.



- Programa que inicialice una lista con 10 valores aleatorios (del 1 al 10) y posteriormente muestre en pantalla cada elemento de la lista junto con su cuadrado.

```
#Ejercicio2_Lista random.py > ...
1  # Tarea Carlos Perez, Josue Guerra, Richard Soria
2
3  # Programa que inicialice una lista con 10 valores aleatorios
4  #(del 1 al 10) y posteriormente muestre en pantalla cada
5  #elemento de la lista junto con su cuadrado.
6
7  import random
8
9  lista = [random.randint(1, 10) for i in range(10)]
10
11 for i in lista:
12     print(f"{i} al cuadrado es {i**2}")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> & C:/Users/Josué/
oritos y estructuras de datos/Deberes/Deber 9/#Ejercicio2_Lista random.py"
6 al cuadrado es 36
10 al cuadrado es 100
2 al cuadrado es 4
10 al cuadrado es 100
9 al cuadrado es 81
2 al cuadrado es 4
5 al cuadrado es 25
5 al cuadrado es 25
2 al cuadrado es 4
9 al cuadrado es 81
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9>
```

Al tener una lista quemada y en este caso con números random generados al azar en ocasiones el arreglo podrá ser más fácil programado como se muestra a continuación, por lo tanto solo se hace el cuadrado de un n número que se genera automáticamente y se presenta en pantalla.

- Crear una lista con 5 string, que el usuario ingrese los caracteres y los imprima de forma invertida.

```
#Ejercicio3_Cadena-string-inversa.py > ...
1  # Tarea Carlos Perez, Josue Guerra, Richard Soria
2
3  # Crear un lista con 5 string, que el usuario
4  # |ingrese los caracteres y los imprima de forma invertida.
5
6  lista = []
7
8  for i in range(5):
9      string = input("Ingresa un string: ")
10     lista.append(string)
11
12 print ("La lista ingresada es: ",lista)
13 print ("La lista inversa es: ". lista[::-1])
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> & C:/Users/
oritos y estructuras de datos/Deberes/Deber 9/#Ejercicio3_Cadena-string-inversa.py"
Ingresa un string: hola que tal que la fuerza te acompañe
Ingresa un string: a
Ingresa un string: b
Ingresa un string: h
Ingresa un string: j
La lista ingresada es: ['hola que tal que la fuerza te acompañe', 'a', 'b', 'h', 'j']
La lista inversa es: ['j', 'h', 'b', 'a', 'hola que tal que la fuerza te acompañe']
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> []
```

El string se lo puede considerar tanto como una frase como una sola palabra o letra, entonces lo que se hace en este programa es pedir al usuario 5 string y los mismos ya nos los muestra en orden si no desde abajo hacia arriba o inversamente.

4. Crea dos arrays unidimensionales que tengan el mismo tamaño (lo pedirá por teclado), en uno de ellos almacena nombres de personas como cadenas, en el otro array se almacena la longitud de los nombres.

```
#Ejercicio4_Nombres-y-longitud.py > ...
1  # Tarea Carlos Perez, Josue Guerra, Richard Soria
2
3  # Crea dos arrays unidimensionales que
4  #tengan el mismo tamaño (lo pedirá por teclado),
5  # en uno de ellos almacena nombres de personas
6  #como cadenas, en el otro array se almacena la longitud de los nombres.
7
8  tamaño = int (input ("Ingrese el tamaño que desea para el arreglo: "))
9
10 nombres = []
11 logintud_nombres = []
12
13 for nombre in range (tamaño):
14     nombre = input ("Ingresa un nombre: ")
15     nombres.append (nombre)
16     logintud_nombres.append (len(nombre))
17
18 print ("Los nombres son: ",nombres)
19 print ("La logintud de los nombres es: ",logintud_nombres)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> & C:/Users/oritmos y estructuras de datos/Deberes/Deber 9/#Ejercicio4_Nombres-y-longitud.py"
Ingrese el tamaño que desea para el arreglo: 3
Ingresa un nombre: Josué
Ingresa un nombre: Richard
Ingresa un nombre: Carlos
Los nombres son: ['Josué', 'Richard', 'Carlos']
La logintud de los nombres es: [5, 7, 6]
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> |
```

5. Inicialice el siguiente arreglo de números: [1, 2, 5, 8, 3, 4, 30, 9, 13]  
Imprima en pantalla programáticamente los números impares mayores a 3.

```
#Ejercicio5_impares-mayores-que-3.py > ...
1  # Tarea Carlos Perez, Josue Guerra, Richard Soria
2
3  # Inicialice el siguiente arreglo de números: [1, 2, 5, 8, 3, 4, 30, 9, 13]
4  # Imprima en pantalla programáticamente los números impares mayores a 3.
5
6  numeros = [1, 2, 5, 8, 3, 4, 30, 9, 13]
7
8  print ("La cadena es: ",numeros)
9  print ("Los numeros impares mayores a 3 son: ")
10
11 for numero in numeros:
12     if numero > 3 and numero % 2 != 0:
13         print(numero)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> & C:/Users/oritmos y estructuras de datos/Deberes/Deber 9/#Ejercicio5_impares-mayores-que-3.py"
La cadena es: [1, 2, 5, 8, 3, 4, 30, 9, 13]
Los numeros impares mayores a 3 son:
5
9
13
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9>
```

6. De las siguientes notas almacenadas en un arreglo: [20, 15, 12, 11, 8, 4, 1]  
Elimine la nota más baja programáticamente sin usar la función (min) y escriba en pantalla. Luego calcule el promedio de notas descontando la nota eliminada.

```
#Ejercicio6_Nota-baja.py > ...
1  # Tarea Carlos Perez, Josue Guerra, Richard Soria
2
3  # De las siguientes notas almacenadas en un arreglo:
4  #[20, 15, 12, 11, 8, 4, 1] Elimine la nota más baja
5  #programáticamente sin usar la función (min) y escriba
6  #en pantalla. Luego calcule el promedio de notas descontando la nota eliminada.
7
8  notas = [20, 15, 12, 11, 8, 4, 1]
9  tamaño = len(notas)
10 menor = notas[0]
11
12 for nota in range(0,tamaño):
13     if notas[nota]<menor:
14         menor = notas[nota]
15 notas.remove(menor)
16 tamaño_nuevo = len(notas)
17 promedio = (sum(notas)/tamaño_nuevo)
18
19 print("La notas menor es:",menor)
20
21 print("La notas finales son:", notas)
22
23 print("El promedio de las notas es:", round(promedio,2))
24
25
```

Ejecución del programa:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> & C:/Users/Josue/OneDrive/Desktop/Algoritmos y estructuras de datos/Deberes/Deber 9/#Ejercicio6_Nota-baja.py
La notas menor es: 1
La notas finales son: [20, 15, 12, 11, 8, 4]
El promedio de las notas es: 11.67
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9>
```

La lista que se ubica dentro del código ya viene predefinida por el problema en si entonces lo que se hace es comprara cual es menor quitar la nota como tal y eso sacar el promedio total.

7. Escribir un programa que almacene las asignaturas de un curso (por ejemplo: Base de Datos, Programación, Matemáticas, Algoritmos, Análisis y Lengua) en una lista, pregunte al usuario la nota que ha sacado en cada asignatura y elimine de la lista las asignaturas aprobadas. Al final el programa debe mostrar por pantalla las asignaturas que el usuario tiene que repetir.

```
#Ejercicio7_Materias-a-repetir.py > ...
1  # Tarea Carlos Perez, Josue Guerra, Richard Soria
2
3  # Escribir un programa que almacene las asignaturas
4  #de un curso (por ejemplo: Base de Datos, Programación,
5  #Matemáticas, Algoritmos, Análisis y Lengua) en una lista,
6  #pregunte al usuario la nota que ha sacado en cada asignatura
7  #y elimine de la lista las asignaturas aprobadas. Al final el
8  #programa debe mostrar por pantalla las asignaturas que el
9  #usuario tiene que repetir.
10
11  asignaturas = ["Base de Datos", "Programación", "Matemáticas", "Algoritmos", "Análisis", "Lengua"]
12  tamaño_asignaturas = len(asignaturas)
13  notas = []
14  nota = 0
15  materias_reprobadas = []
16  for asignatura in asignaturas:
17      while(True):
18          nota = float(input("Ingrese la nota de la asignatura de "+asignatura+" : "))
19          if (nota<=20 and nota>=1):
20              notas.append(nota)
21              break
22          else:
23              print("Ingrese una nota válida.")
24  tamaño_notas = len(notas)
25  for i in range(0,tamaño_notas):
26      if (notas[i]<14):
27          materias_reprobadas.append(asignaturas[i])
28
29  print("Las asignaturas a repetir por parte del estudiante son:", materias_reprobadas)
```

Ejecución del programa:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> & C:/Users/Josué
rítmos y estructuras de datos/Deberes/Deber 9/#Ejercicio6_Nota-baja.py"
Ingrese la nota de la asignatura de Base de Datos : 20
Ingrese la nota de la asignatura de Programación : 16
Ingrese la nota de la asignatura de Matemáticas : 14
Ingrese la nota de la asignatura de Algoritmos : 8
Ingrese la nota de la asignatura de Análisis : 14
Ingrese la nota de la asignatura de Lengua : 4
Las asignaturas a repetir por parte del estudiante son: ['Algoritmos', 'Lengua']
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> █
```

La nota para pasar se establece en 20 por ende el estudiante al ingresar una nota tendría que tomar en cuenta esta característica, además cabe recalcar que la nota mínima para pasar es de 14 así que si se inserta una menor a esta el arreglo lo detectara y por consiguiente se añadirá a la lista de materias a repetir que se muestra al final de toda la ejecución del programa.

8. Escribir un programa que almacene el abecedario en una lista, elimine de la lista las letras que ocupen posiciones múltiplos de 3, y muestre por pantalla la lista resultante.

```
#Ejercicio8_Abecedario.py > ...
1  # Tarea Carlos Perez, Josue Guerra, Richard Soria
2
3  # Almacenar el abecedario en una lista
4  print(" Bienvenido al programa ")
5  abecedario = list("ABCDEFGHIJKLMNOPQRSTUVWXYZ")
6
7  print("Abecedario original:")
8  print(abecedario)
9
10 abecedario_filtrado = [letra for i, letra in enumerate(abecedario) if (i + 1) % 3 != 0]
11
12 print("\nLista después de eliminar letras en posiciones múltiplos de 3:")
13 print(abecedario_filtrado)
14
```

Ejecución del programa:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> & C:/Users/Josué/AppData/Local/Microsoft/WindowsAp
ritmos y estructuras de datos/Deberes/Deber 9/#Ejercicio8_Abecedario.py"
Bienvenido al programa
Abecedario original:
['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']

Lista después de eliminar letras en posiciones múltiplos de 3:
['A', 'B', 'D', 'E', 'G', 'H', 'J', 'K', 'M', 'N', 'P', 'Q', 'S', 'T', 'V', 'W', 'Y', 'Z']
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9>
```

Dentro del código muestra el abecedario inicial y como el problema pide que se elimine cada letra que sea múltiplo de 3 entonces den el arreglo, se lo manda a que detecte la posición de cada letra y según eso vaya eliminando, con eso imprime en pantalla el nuevo abecedario el cual contienen las letras cuyas posiciones no son múltiplos de 3

9. Escribir un programa que pida al usuario una palabra y muestre por pantalla el número de veces que contiene cada vocal.

El código define las 5 vocales del abecedario que ya se las conoce dentro de un elif para que vaya contando según la palabra ingresada, añadiéndolas al contador como tal y eso se va acumulando hasta el final lo que muestra al usuario la cantidad de vocales dentro de la palabra ingresada.

### Ejecución

```
#Ejercicio9_vocales.py > ...
1  # Tarea Carlos Perez, Josue Guerra, Richard Soria
2
3  #Escribir un programa que pida
4  #al usuario una palabra y muestre
5  #por pantalla el número de veces
6  #que contiene cada vocal.
7
8  def contar_vocales(palabra):
9      contador_vocales = [0] * 5
10     for letra in palabra:
11         letra = letra.lower()
12
13         if letra in "aeiou":
14             if letra == "a":
15                 contador_vocales[0] += 1
16             elif letra == "e":
17                 contador_vocales[1] += 1
18             elif letra == "i":
19                 contador_vocales[2] += 1
20             elif letra == "o":
21                 contador_vocales[3] += 1
22             elif letra == "u":
23                 contador_vocales[4] += 1
24
25
26     print("-- La palabra repite la vocal 'a':", contador_vocales[0], "veces --")
27     print("++ La palabra repite la vocal 'e':", contador_vocales[1], "veces ++")
28     print("-/ La palabra repite la vocal 'i':", contador_vocales[2], "veces -/")
29     print("/- La palabra repite la vocal 'o':", contador_vocales[3], "veces -/")
30     print("** La palabra repite la vocal 'u':", contador_vocales[4], "veces **")
31
32     palabra_usuario = input("Ingrese una palabra: ")
33
34     contar_vocales(palabra_usuario)
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> & C:/Users/ritmos y estructuras de datos/Deberes/Deber 9/#Ejercicio9_vocales.py
Ingrese una palabra: Hola que hace, que la fuerza te acompañe Ankin
-- La palabra repite la vocal 'a': 7 veces --
++ La palabra repite la vocal 'e': 6 veces ++
-/ La palabra repite la vocal 'i': 1 veces -/
/- La palabra repite la vocal 'o': 2 veces -/
** La palabra repite la vocal 'u': 3 veces **
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> |
```



10. Una vez conocidas las 32 selecciones que participarán del próximo **mundial de fútbol** se necesita realizar el sorteo entre las 8 series o grupos de competencia. Las selecciones se encuentran numeradas del 1 al 32, las mejores han sido pre asignadas como “cabeza de serie”; una por cada grupo y no se sorteará su ubicación en la serie. Las selecciones restantes se sortearán la ubicación en cada serie (grupo) para completar los cuatro participantes por serie.

grupo	1	2	3	4	5	6	7	8
cabeza [grupo]	3	7	9	12	22	25	26	30

El sorteo de serie (luego de copiar los cabezas de grupo) se realizará en un vector como el mostrado:

selección	1	2	3	4	5	6	7	8	9	...	32
serie [selección]	0	0	1	0	0	0	2	0	3	...	0

Elabore un algoritmo que solicite cuáles son los 8 equipos que serán cabezas de serie, asigne aleatoriamente (y sin repeticiones) los 24 equipos restantes, al final muestre el listado de las series resultantes.

```
#Ejercicio10_Equipos.py X
#Ejercicio10_Equipos.py > ...
1 import random
2
3 def sorteo():
4     equipos = []
5
6
7     print("Ingrese los números de los equipos que serán cabezas de serie:")
8     equipos = [int(input("Equipo {}: ".format(i + 1))) for i in range(8)]
9
10    equipos_resto = [equipo for equipo in range(1, 33) if equipo not in equipos]
11    random.shuffle(equipos_resto)
12
13    print("\nListado de las series resultantes:")
14    for i in range(8):
15        cabeza_serie = equipos[i]
16        equipos_serie = [cabeza_serie] + equipos_resto[i * 3 : (i + 1) * 3]
17        print("Serie {}: {}".format(i + 1, equipos_serie))
18
19    sorteo()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - - - -

PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> & C:\Users\Josue\AppData\Local\Microsoft\WindowsApps\python3.11.exe "d:\Documentos\2do Semestre\Alg  
oritos y estructuras de datos\Deberes\Deber 9\#Ejercicio10\_Equipos.py"  
Ingrese los números de los equipos que serán cabezas de serie:  
Equipo 1: 12  
Equipo 2: 9  
Equipo 3: 8  
Equipo 4: 7  
Equipo 5: 15  
Equipo 6: 6  
Equipo 7: 11  
Equipo 8: 4  
  
Listado de las series resultantes:  
Serie 1: [12, 22, 24, 18]  
Serie 2: [9, 25, 10, 27]  
Serie 3: [8, 13, 32, 23]  
Serie 4: [7, 3, 30, 20]  
Serie 5: [15, 31, 26, 28]  
Serie 6: [6, 17, 19, 14]  
Serie 7: [11, 1, 2, 21]  
Serie 8: [4, 5, 16, 29]  
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 9> |

El algoritmo incluye una generación de números hasta el 32 una vez el usuario inserte los 8 primeros grupos cabezas automáticamente dará el resultado de la serie junto con los números random no repetidos dentro de is con los que se enfrentara.

**ENTREGABLES:**

- Una vez culminada tu tarea, capturar las pantallas de la ejecución del problema con tus datos y súbela en el apartado del aula virtual “S10-Tarea2
- Subir los ejercicios al git o al drive y entrega la url de los archivos .py o, a su vez, entregue el archivo.
- Recordar que el nombre del archivo deberá ser: S11\_Deber2\_Algoritmos\_2023B\_NApellido(de todos los integrantes)

**RECURSOS NECESARIOS**

- Acceso a Internet.
- Imaginación.
- VSC

**RECOMENDACIONES**

Se aconseja abordar el estudio de arreglos en Python con un enfoque integral y auténtico. Para evitar cualquier indicio de similitud, es fundamental explorar diversas fuentes de aprendizaje, como documentación oficial, tutoriales especializados y libros académicos. La participación activa en plataformas de aprendizaje en línea, donde se puedan resolver dudas específicas y discutir conceptos, también puede ser beneficioso.

Se recomienda aplicar los conocimientos adquiridos en proyectos prácticos, asegurándose de comprender cada línea de código y adaptándola a contextos propios. Este enfoque no solo fortalecerá las habilidades, sino que también garantizará la originalidad en cualquier trabajo relacionado con arreglos en Python.

**CONCLUSIONES**

En resumen, la comprensión profunda y la aplicación práctica de arreglos en Python son esenciales para cualquier estudiante. Al adoptar un enfoque genuino en el aprendizaje, evitando simplemente copiar y pegar códigos, se construye un conocimiento sólido y se minimiza el riesgo de similitud en trabajos académicos. La combinación de diversidad en las fuentes de información, participación activa en la comunidad educativa y la creación de proyectos originales no solo cumple con estándares éticos, sino que también contribuye a un aprendizaje más efectivo y duradero.

**ENLACE GITHUB(Dentro del enlace se encuentran los códigos escritos):**

<https://github.com/JosueGuerra2023B/Estructuras-Datos2023B/tree/master/Deberes/Deber%209>

**BIBLIOGRAFÍA**

Los recursos usados dentro del deber son los que nos dio la ingeniera en clase, solo se aplico lo aprendido en los diferentes ejercicios.