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

CONSULTA 1
Grupal

TÍTULO:

Arreglos

Nombre de los estudiantes:
Guerra Lovato Josué
Pérez Orosco Carlos
Soria Ansa Richard

```
---

    https://parzibyte.me/blog

---

nombres = ["Luis", "María", "Pedro", "Mario", "Leon", "Claire"]
edades = [0, 23, 6, 1, 2, 3, 4]
print("Nombres originalmente: ")
print(nombres)
print("Edades originalmente: ")
print(edades)

# Los ordenamos; uno con sort y otro con sorted para demostrar el orden
nombres_ordenados = sorted(nombres, reverse=True)
edades.sort(reverse=True)

print("Nombres ordenados: ")
print(nombres_ordenados)
print("Edades ordenadas: ")
print(edades)
```

2023-B

PROPÓSITO DE LA TAREA

Reutilizar el código mediante funciones para una programación óptima [1].

OBJETIVO GENERAL

El objetivo general del aprendizaje de arreglos en Python, redactado como estudiante universitario, es comprender el uso de arreglos para almacenar y manipular conjuntos de datos de manera eficiente. Los arreglos en Python, especialmente con bibliotecas como NumPy, permiten realizar cálculos numéricos a gran escala, preprocesar datos para algoritmos de aprendizaje automático y trabajar con estructuras n-dimensionales de manera más eficiente que las listas convencionales. El objetivo es dominar el uso de arreglos para resolver problemas computacionales y científicos, aprovechando su velocidad y capacidad para operar con conjuntos de datos extensos.

Investigue sobre arreglos con Python
Consulta:

DEFINICIÓN ARREGLOS EN PYTHON

Las listas en Python son una estructura de datos fundamental que permite almacenar múltiples valores en una sola variable. Se asemejan a los vectores en otros lenguajes de programación y son esenciales para abordar problemas donde la cantidad exacta de datos no es conocida de antemano. Las listas son utilizadas para almacenar información como nombres de personas en un programa, donde la cantidad de datos puede variar. Su flexibilidad y capacidad para gestionar colecciones de elementos las convierten en herramientas poderosas para resolver una variedad de desafíos en programación [2].

CARACTERISTICAS

En Python, los arreglos se implementan principalmente a través de listas. Aquí hay algunas características clave de las listas, que funcionan como arreglos en Python:

- Mutable: Las listas en Python son estructuras de datos mutables, lo que significa que puedes modificar, agregar o eliminar elementos después de haber creado la lista.

- Heterogéneas: Las listas pueden contener elementos de diferentes tipos de datos, como enteros, cadenas, flotantes, u otros objetos, lo que las hace flexibles para almacenar información diversa.

- Indexación: Los elementos en una lista están indexados, comenzando desde 0 para el primer elemento. Puedes acceder a un elemento específico utilizando su índice, y también es posible acceder a elementos desde el final utilizando índices negativos.

    - mi_lista = [1, 2, 3, 4, 5]
      print (mi_lista [0]) # Imprime 1
      print(mi_lista[-1]) # Imprime 5 (último elemento)

- Longitud dinámica: No es necesario especificar la longitud de una lista al crearla, y puedes cambiar la longitud de la lista durante la ejecución del programa mediante la adición o eliminación de elementos.

    - mi_lista = [1, 2, 3]
      mi_lista. append (4) # Agrega un elemento al final
      mi_lista. remove (2) # Elimina el elemento 2

Funciones y Métodos incorporados: Python proporciona una variedad de funciones y métodos incorporados para trabajar con listas, como len () para obtener la longitud de la lista, sum () para sumar elementos, y métodos como append (), remove (), pop (), entre otros [3].

- mi_lista = [1, 2, 3, 4, 5]
  print(len(mi_lista)) # Imprime 5 (longitud de la lista)
  print(sum(mi_lista)) # Imprime 15 (suma de los elementos)

- Slicing: Puedes crear sublistas (rebanadas) de una lista más grande utilizando la notación de segmentación (slicing). Esto facilita la manipulación de partes específicas de una lista.

    - mi_lista = [1, 2, 3, 4, 5]
      sublista = mi_lista [1:4] # Crea una sublista desde el índice 1 hasta el 3

NUMPY

NumPy es una biblioteca de Python diseñada para realizar cálculos científicos, especialmente en el análisis de grandes conjuntos de datos. Su función principal es facilitar operaciones eficientes en arreglos multidimensionales. A través de este tutorial, puedes aprender a realizar diversas operaciones con arreglos NumPy, como agregar, eliminar, ordenar y manipular elementos de manera efectiva.

La ventaja clave de NumPy radica en su capacidad para trabajar con arreglos multidimensionales y matrices derivadas, como matrices enmascaradas o matrices multidimensionales enmascaradas. Esto permite el tratamiento eficiente de datos sin agotar la memoria RAM. Esta eficiencia es una característica destacada en comparación con lenguajes como Java o C#. No obstante, no se pretende menospreciar la potencia de estos últimos, ya que son lenguajes de programación robustos que han sido relevantes durante décadas [1].

LOS EJEMPLOS CORRESPONDIENTES SE ENCUENTRAN EN LA SIGUIENTE HOJA, JUNTO CON CPATURA Y UN POCO DE EXPLICACIÓN DE LOS MISMOS.

#Ejerciciop1_Ejemplo.py > ...

```python
# Ejercicio 1
# Solicitar al usuario el tamaño del arreglo
# Llenar el arreglo con los datos del usuario
# Sumar los pares del arreglo
# Contar los impares
#Ejercicio echo por Josué Guerra
import random
print("---Bienvenido al sistema---")
print(" Muestra de números pares e impares ")
tamanio = int(input("Ingrese el tamaño del arreglo: "))
arreglo = []

for i in range(tamanio):
    arreglo.append(random.randint(1, 100))

print("El arreglo generado es:")
print(" | ".join(str(num) for num in arreglo))

suma_pares = sum(num for num in arreglo if num % 2 == 0)
impares = sum(1 for num in arreglo if num % 2 != 0)

print("La suma de los pares es:", suma_pares)
print("La cantidad de impares es:", impares)

```

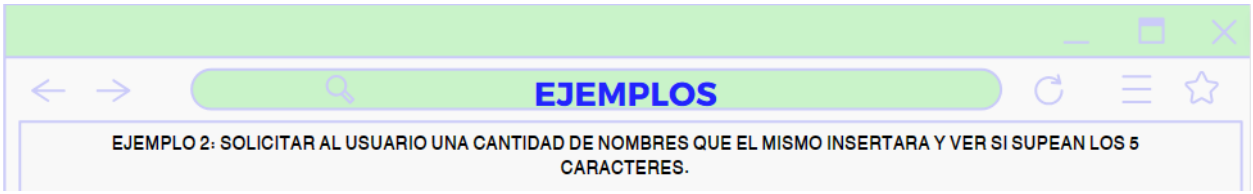
Ejecución

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber > & C:/Users/Josué/AppData/Local/
tmos y estructuras de datos/Deberes/Deber 8/#Ejerciciop1_Ejemplo.py"
---Bienvenido al sistema---
 Muestra de números pares e impares
Ingrese el tamaño del arreglo: 10
El arreglo generado es:
3 | 6 | 61 | 36 | 14 | 56 | 62 | 98 | 74 | 14
La suma de los pares es: 360
La cantidad de impares es: 2
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 8>
```

Explicacion sencilla:

El programa comienza solicitando al usuario que ingrese el tamaño del arreglo. Luego, llena el arreglo con números aleatorios, los muestra, calcula la suma de los números pares y cuenta la cantidad de números impares. Finalmente, muestra los resultados.



#Ejercicio2_Ejemplo.py > ...

```python
# Ejercicio 2
#  se solicita al usuario que ingrese la cantidad de nombres a ingresar.
# Para luego analizar si tiene más de 5 caracteres
#Ejercicio echo por Josué Guerra

print("         ---Bienvenido al sistema---           ")
print(" Solicitud de nombres ingresados por le usuario ")

n = int(input("Inserte la cantidad de nombres que desea: "))
# Inicializar un arreglo vacío
nombres = []
for i in range(n):
    nombre = input(f"Ingrese el nombre {i+1}: ")
    if len(nombre) > 5:
        nombres.append(nombre)

# Imprimir los nombres con más de 5 caracteres del arreglo
print("Nombres con más de 5 caracteres ingresados:")
for nombre in nombres:
    print(nombre)

```

Ejecución

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 8> & C:/
d:/Documentos/2do Semestre/Algoritmos y estructuras de datos/Deberes/Deber 8/#Ejercici
         ---Bienvenido al sistema---
 Solicitud de nombres ingresados por le usuario
Inserte la cantidad de nombres que desea: 6
Ingrese el nombre 1: Josué
Ingrese el nombre 2: Carlos
Ingrese el nombre 3: Richard
Ingrese el nombre 4: Mauricio
Ingrese el nombre 5: David
Ingrese el nombre 6: Eduard
Nombres con más de 5 caracteres ingresados:
Carlos
Richard
Mauricio
Eduard
```

Explicación sencilla:

En este ejemplo, el programa solicita al usuario que ingrese la cantidad de nombres a ingresar, luego inicializa un arreglo vacío llamado "nombres". Utiliza un bucle for para pedir al usuario que ingrese cada nombre, y si el nombre tiene más de 5 caracteres, lo agrega al arreglo "nombres". Finalmente, imprime la lista de nombres con más de 5 caracteres. El código detallado incluye la solicitud de entrada al usuario, la inicialización del arreglo, el bucle de iteración, la verificación de la longitud del nombre y la impresión de la lista resultante.

EJEMPLOS

EJEMPLO 3: SOLICITAR AL USUARIO LA CANTIDAD DEL ARREGLO, Y SE VE EL NÚMERO MAYOR Y SI ESTE ES LA SUMA DE LOS OTROS

#Ejercicio3_Ejemplo.py > ...

```python
# Ejercicio 3
# Se solicita al usuario que ingrese la cantidad deL ARREGLO
# Se ve si al número mayor es la suma de los otros o no
#Ejercicio echo por Josué Guerra
import random
print("  ---Bienvenido al sistema---    | ")
print(" Solicitud del rango del arreglo  ")
tamanio = int(input("Ingrese el tamaño del arreglo: "))
arreglo = []
suma = 0

# Llenar el arreglo con datos aleatorios
for i in range(tamanio):
    num = random.randint(1, 100)
    arreglo.append(num)
    suma += num
# Imprimir el arreglo generado
print("El arreglo generado es:")
print("|", end=" ")
for num in arreglo:
    print(num, "|", end=" ")
print()
# Encontrar el número mayor y menor en el arreglo
mayor = max(arreglo)
menor = min(arreglo)
print("El número mayor es:", mayor)
print("El número menor es:", menor)
# Verificar si existe un número que es la suma del resto
suma_encontrada = False
for num in arreglo:
    if num == suma - num:
        suma_encontrada = True
        print("El número que es la suma del resto es:", num, "y", suma - num)
        break

if not suma_encontrada:
    print("No existe un número que sea la suma del resto.")
```

Ejecución

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 8> & C
oritmos y estructuras de datos/Deberes/Deber 8/#Ejercicio3_Ejemplo.py"
  ---Bienvenido al sistema---
 Solicitud del rango del arreglo
Ingrese el tamaño del arreglo: 12
El arreglo generado es:
| 40 | 10 | 1 | 22 | 45 | 4 | 88 | 18 | 76 | 83 | 18 | 50 |
El número mayor es: 88
El número menor es: 1
No existe un número que sea la suma del resto.
PS D:\Documentos\2do Semestre\Algoritmos y estructuras de datos\Deberes\Deber 8>
```

Explicación sencilla:

El código proporcionado, se solicita al usuario que ingrese el tamaño del arreglo, luego se genera un arreglo de números aleatorios con el tamaño especificado. Después, se calcula la suma de todos los números en el arreglo y se encuentran el número mayor y el número menor en el arreglo. Finalmente, se verifica si existe un número en el arreglo que sea igual a la suma del resto de los números, y en caso afirmativo, se imprime dicho número y su complemento que completa la suma; de lo contrario, se imprime que no existe un número que sea la suma del resto. Este ejercicio demuestra el uso de arreglos, generación de números aleatorios, cálculo de la suma, y búsqueda de ciertas condiciones dentro del arreglo.

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 “S9-Consulta-1
- Recordar que el nombre del archivo deberá ser: **S9-Consulta-1-Algoritmos_2023B_NApellido**.(de todos los integrantes)

RECURSOS NECESARIOS

- Acceso a Internet.
- Imaginación.
- VSC

CONCLUSIONES

En conclusión, el aprendizaje de arreglos en Python es fundamental para comprender y utilizar eficientemente estructuras de datos que son esenciales en el ámbito computacional y científico. La capacidad de almacenar y manipular conjuntos de datos de forma eficiente, junto con la posibilidad de realizar cálculos numéricos a gran escala y trabajar con estructuras n-dimensionales, brinda a los estudiantes universitarios las herramientas necesarias para abordar problemas complejos en campos como la ciencia de datos, la ingeniería y la investigación computacional.

RECOMENDACIONES

Como recomendación, es crucial practicar de manera constante la implementación y manipulación de arreglos en Python, así como explorar bibliotecas especializadas como NumPy para ampliar las capacidades en el manejo de datos numéricos. Además, buscar aplicaciones prácticas en proyectos y ejercicios relacionados con el área de estudio o interés, permitirá fortalecer el entendimiento y dominio de los arreglos, preparándonos para enfrentar desafíos reales en el campo profesional.

ENLACES:

Enlace GitHub con los ejecutables:
https://github.com/JosueGuerra2023B/Estructuras-Datos2023B/tree/master/Deberes/Deber%208

Enlace diapositivas:
https://www.canva.com/design/DAF6RQgV6O0/KwwrAEP2VTP-ox8_SFqlJA/view?utm_content=DAF6RQgV6O0&utm_campaign=designshare&utm_medium=link&utm_source=editor

1 Bibliografía

[I. L. Chulde, «Aulas virtuales EPN,» [En línea]. Available: https://epnecuador-
1 my.sharepoint.com/personal/lorena_chulde_epn_edu_ec/_layouts/15/onedrive.aspx?fromShare
] =true&ga=1&id=%2Fpersonal%2Florena%5Fchulde%5Fepn%5Fedu%5Fec%2FDocuments%
2FAlgoritmos%20y%20Estructuras%20de%20Datos%2F2023%2DB%2FSemana%2D02%2F
Clase.

[J. D. Meza, «ProgramarYA,» 16 05 2012. [En línea]. Available:
2 https://www.programarya.com/Cursos/Python/estructuras-de-datos/listas. [Último acceso: 14
] 01 2024].

[Codesi, «Codesi,» 2020. [En línea]. Available: https://www.buscaminegocio.com/cursos-de-
3 python/como-utilizar-los-arreglos-en-python-para-
] programar.html#:~:text=%C2%BFQue%20es%20un%20arreglo%20en,representar%20sistema
s%20del%20mundo%20real. [Último acceso: 14 01 2024].

[NumPy, «NumPy.org,» 16 09 2023. [En línea]. Available: https://numpy.org/. [Último acceso:
4 14 01 2024].
]