Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas

Programación Avanzada 2MV7

Practica 1

Autor:
Barrios Mendez Jose Alberto
Boleta: 2022640111

Profesor:
Cruz Mora Jose Luis

Ing. Mecatrónica

27 de febrero de 2024

Índice

1. Objetivo.

Conocer y familiarizarse con el lenguaje de programación Python, el manejo de entradas y salidas y operaciones basicas.

2. Introduccion.

El objetivo de esta practica es familiarizarse con el lenguaje de programacion python,mediante una serie de ejercicios los cuales comparados a otros lenguajes con C/C++, en este lenguaje es de cierta forma mas "sencilloz mas practicos de desarrollar, uno de estos ejemplos es la resolucion de un sistema de ecuaciones 3x3, lo cual en lenguaje python es muy rapido, tambien veremos un ejemplo de como la media aritmetica y la desviacion estandar, lo cual en algunos otros lenguajes es sumamente complicado, sin emabargo en python, usando la libreria Numpy, esto se puede llegar a resumir en una sola linea de comando
Durante esta practiva nos adentraremos mas dentro del lenguaje python, y notaremos la simplicidad que lo caracteriza, ya que debemos recordar que es un lenguaje de alto nivel, es decir que trata de parecerse al lenguaje natural

3. Desarrollo.

3.1. Entrada y Salida de Arreglos

Para esta preactica se necesito considerar el siguiente diagrama de flujo. A continuacion detallaremos cada elemento y el proceso que se uso para que el reesultado sea eficiente

- Media aritmetica
  El proposito de esta funcion es como el mismo nombre puede indicar, preguntar al usuario un numero n de elementos y calcular la media aritmetica de estos elementos

  1. Solicitar el numero de elementos: Para este paso es de forma indispensable que el usuario ingrese un valor de tipo entero, esto se puede conseguir gracias al siguiente bloque de codigo:

```python
def solicitar_elementos():
    while True:
        try :
            num_elementos=int(input("Ingrese el numero de elementos a  ingresar: "))

            break
        except ValueError:
            os.system("cls")
            print("Ingresa un valor valido: ")
    return num_elementos
```

Figura 1: Funcion para forzar al usuario a ingresar un numero entero

Esta funcion es una que se va a compartir en varias funciones, ya que es fundamental que el usuario ingrese un valor de forma int, por lo cual, en el codigo esto es posible con un ciclo while, el cual se ejecuta mientras sea "True", es notable el uso de un "try.el cual tiene la funcion de Intentar.obtener un valor entero, en caso de ingresar un valor de otro tipo, hace una limpieza del sistema con el comando çls", y nuevamente se vuelve a pedir un valor valido, la forma de salir de este ciclo es ingresando un valor entero, para que se ejecute la sentencia "breaz asi salir del ciclo while, para posteriormente devolver el numero de elementos

2. Ingresar datos: Para llevar a cabo el proceso de solicitar datos, se implemento la Figura 2, la cual contiene la variable, "num_elementos"donde usando la funcion del paso 1, esta almacena el numero de elementos a ingresar, posteriormente se declara la variable suma=0, e iniciamos un ciclo "for", el cual nos sirve para almacenar los n elementos que el usuario desea ingresar, notese el uso de un ciclo "while", muy similar al del paso 1, el cual se ejecutara mientras el usuario no ingrese un numero valido, este puede ser un valor de tipo entero o tipo flotante, en caso de ingresar un numero valido, el ciclo while se detendra para el valor de iteracion ï", ademas de que se asignara el espacio i-esimo de la matriz, tambien vamos a llevar a cabo la suma de cada valor ingresado con el comando suma=mat[i], el cual sumara cada elemento i-esimo que ingresemos.
Algo mas a destacar es la forma de creacion de un array, donde gracias a la libreria Numpy, la cual tenemos de forma abreviada "num", y usando uno de sus metodos ".empty.el cual nos crea una matris de n elementos iniciada en zeros

```
17  def media():
18
19      num_elementos = solicitar_elementos()
20      mat=num.empty(num_elementos)
21
22      suma=0
23
24      for i in range(num_elementos):
25          while True:
26              try :
27                  mat[i]=float(input(f"Ingrese el elemento {i+1}: "))
28                  suma+=mat[i]
29                  break
30              except ValueError:
31                  os.system("cls")
32                  print("Elemento no valido: ")
33      print(suma/num_elementos)
```

Figura 2: Funcion de la opcion media dentro del programa

3. Calcular la media aritmetica: Para este ultimo paso es importante considererar la forma en la que implementamos el calculo, la cual a mi parecer es algo abstracta, debido que usamos la definicion de media aritmerica: $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$

Es probable que Numpy tenga una funcion que haga esto de forma mas rapida. Una vez calculada la media aritmetica lo que resta es imprimir en pantalla el resultado.

- Desviacion Estandar
  En esta funcion se busca preguntar al usuario una coleccion de datos y calcular su desviacion estandar usando la libreria Numpy

  1. Ingreso numero de elementos: Este proceso se hace de la misma forma que en el caso anterior de la media aritmetica, recordando que partimos de la misma funcion, indicada en la Figura 1, de la misma forma necesitamos que ingrese un numero de tipo entero, por lo cual espero este paso no tiene ninguna complicacion
  2. Ingreso de datos: En la Figura 3, podemos observar la funcion desviacion_estandar, la cual contiene similitud con el ejemplo anterior, creamos la matriz con ayuda de la libreria Numpy (num.empty()) la cual nos crea una matriz de n, elementos iniciados en ceros, posteriormente tenemos un ciclo for, donde se obliga al usuario a ingresar un elemento valido, ya sea de tipo entero (int) o de tipo flotante (float), esto gracias al ciclo while, el cual se terminara hasta ingresar un elemento valido
  3. Calculo desviacion estandar: Con ayuda de la libreria Numpy, podemos hacer uso de la funcion std(), la cual toma como argumento, un array y devuelve la desviacion estandar de esta coleccion de datos. Esto es de gran ayuda, debido a que la forma de calcular la desviacion estandar para n elementos es de la forma $\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$. La cual resultaria muy tediosa de programar sin el uso de esta libreria. Para terminar mostramos en pantalla el resultado

```
36  def desviacion_estandar():
37
38      elementos=solicitar_elementos()
39      mat_elementos=num.empty(elementos)
40      for i in range(elementos):
41
42          while True:
43              try:
44                  mat_elementos[i]=int(input(f"Ingrese el elemento {i+1}: "))
45                  break;
46              except ValueError:
47                  os.system("cls")
48                  print("Elemento no valido, prueva con UN NUMERO: ")
49
50      print("La desviacion estandar es: ",num.std(mat_elementos))
51
```

Figura 3: Funcion de la opcion desviacion estandar

- Burbuja:
  En esta funcion se busca que el usuario ingrese el numero de elementos que desea obtener, usando la librerua Numpy, generar un arreglo semi aleatorio, del mismo tamaño de la entrada n, ademas de ordenar estos elementos usando el metodo burbuja

1. Solicitar el numero de elementos: En este paso seguimos usando la funcion indicada en la Figura 1, la cual nos proporcionara un numero entero de elementos
2. Generar elementos: En esta parte del programa, nuevamente utilizaremos la libreria Numpy, usando el metodo rando.rand(num_elementos), el cual devuelve un arrego con numeros aleatorios del tamaño indicado e imprime la matriz de elemetnos generados de forma aleatoria
3. Ordenamiento: El ordenamiento burbuja como tal no lo encotre en la libreria, debido a que es uno de los menos efectivos y mas tardados, cuando hablamos de grandes cantidades de datos, para el programa se implento el ordenamiento burbuja, el cual consiste en iterar desde el primer elemento del array [i] iniciando los ciclos en i=0, y si este cumple que es mayor al elemento siguiente [i+1] se hace un intercambio de posiciones, es decir cambiamos el elemento mayor a la siguiente posicion, una vez que iteramos sobre todos los pares de elementos, el elemento mayor habra quedado en la parte final del array, de esta forma se puede iterar hasta ordenar todos los elementos, observe la Figura 5

```
53  def burbuja():
54      num_elementos=solicitar_elementos()
55      mat=num.random.rand(num_elementos)
56      print(f"La matriz de {num_elementos} elementos aleatorios es:\n{mat}")
57
58      for i in range(num_elementos):
59          for j in range(num_elementos-i-1):
60              if mat[j]>mat[j+1]:
61                  temp=mat[j+1]
62                  mat[j+1]=mat[j]
63                  mat[j]=temp
64      print(f"El arreglo ordenado es: {mat}")
65
```

Figura 4: Funcion de la opcion burbuja

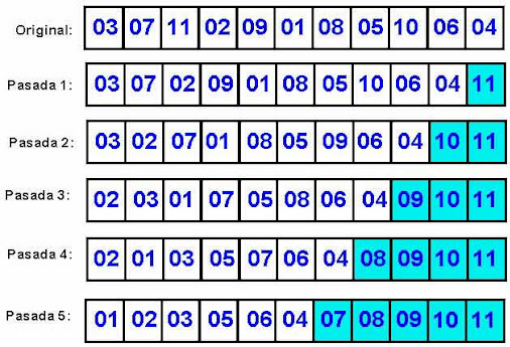

Figura 5: Ejemplo de un ordenamiento burbuja

- Salir:
  Para esta opcion dentro del menu, solo basta con imprimir un mensaje de despedida y finalizar el programa, se puede hacer de la siguiente manera:

```
26  if opcion==4:
27      os.system("cls")
28      print("Hata luego, vuelva pronto!!!!")
29      exit()
```

Figura 6: Funcion de la opcion salir

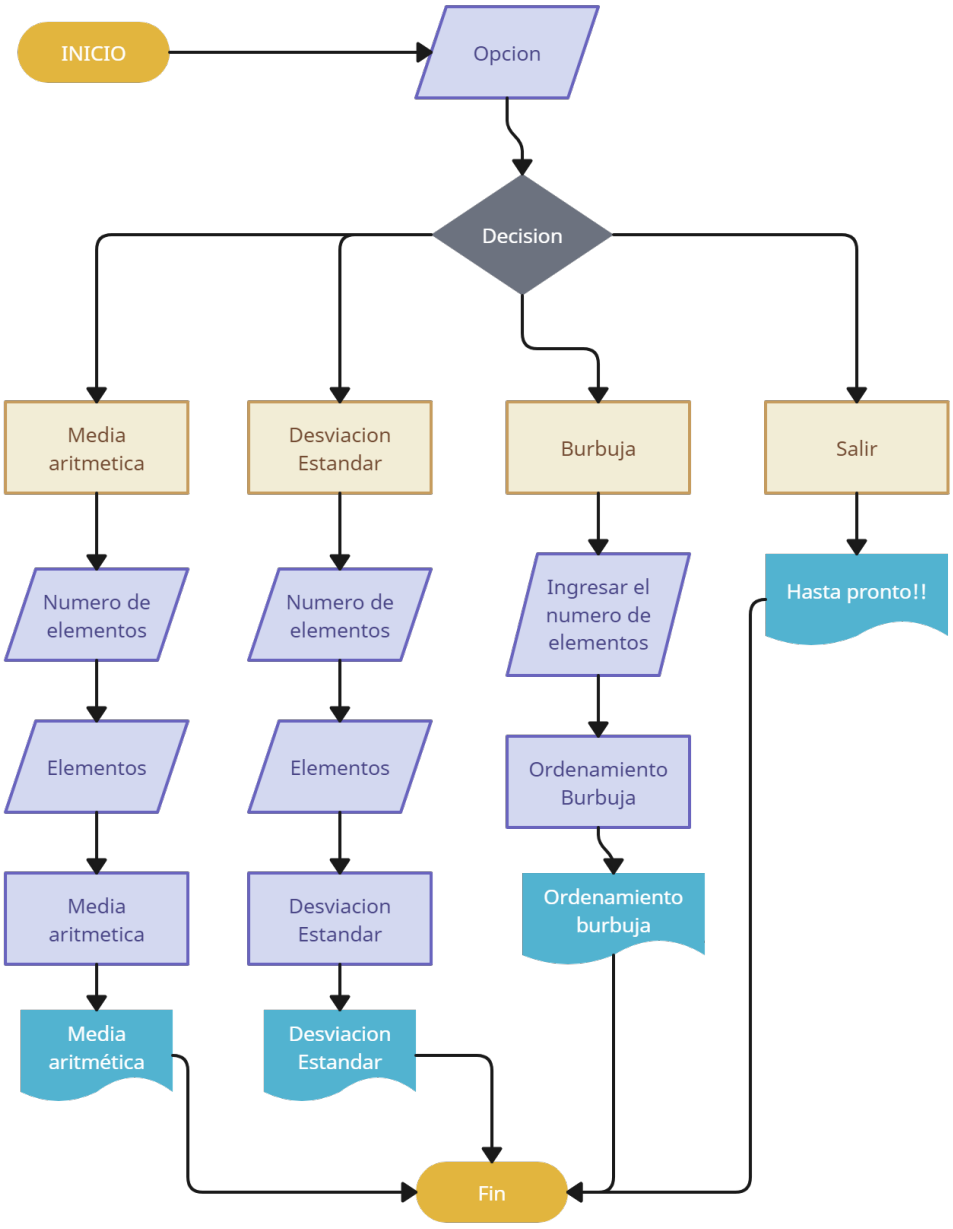

Figura 7: Diagrama de flujo utilizado para el primer programa

3.2. Sistemas de ecuaciones

- Solucion al problema propuesto:

1. Analisis del problama:
   a partir del sistema de ecuaciones proporcionada en el problema, el cual es el siguiente:

$$\begin{aligned} 2x - 4y - 3z &= 15 \\ x + 5y - 5z &= 5 \\ 4x + 2y + 67z &= 20 \end{aligned}$$

   Donde podemos observar que

$$A = \begin{bmatrix} 2 & -4 & -3 \\ 1 & 5 & -5 \\ 4 & 2 & 67 \end{bmatrix}, \quad B = \begin{bmatrix} 15 \\ 5 \\ 20 \end{bmatrix}$$

2. Solucion:
   Recordando que se esto se puede solucionar de la forma $X = A^{-1} * B$
   En la Figura 8, podemos encontrar que utilizando la libreria Numpy, podemos declarar la matriz A, el vector B, con el metodo .array([]). Tambien notemos que para obtener la inversa de la matriz A, se usa otro metodo, el cual es .linalg.inv(A), el cual nos devuelve la matriz inversa
   Finalmente para la solucion basta con usar la formula antes descrita, ya que contamos con todos los elementos necesarios para realizar el producto punto de $A^{-1} * B$, con el metodo .dot(A^{-1},B) es suficiente para que nos devuelva los resultados para cada variabke, los cuales se imprimen en pantalla

```python
# resolver el sistema de ecuacaiones
import numpy as num
import os
print("2x-4y-3z=15\n1x+5y-5z=5\n3x+2y+67z\n")
A=num.array([[2,-4,-3],[1,5,-5],[4,2,67]])
B=num.array([15,5,20])

A_inversa=num.linalg.inv(A)
X=num.dot(A_inversa,B)
print(f"Los resultados son:\nX={X[0]}\nY={X[1]}\nZ={X[2]}")

#haciendo la comprobacion
print("\nHaciendo la comprobacion A*X, lo que nos debe arrojar B")
print(num.dot(A,X))
```

Figura 8: Solucion al sistema de ecuaciones propuesto

- Modificacion para resolucion de sistemas 3x3:

1. Comenzamos preguntando al usuario si desea resolver un sistema de ecuaciones de 3x3, similar al ejemplo resuelto, obligando a responder con s/n gracias al ciclo while, de esta forma garantizamos que no "truene.[el] programa, en la figura 9, vemos que se usa la funcion de convertir a minusculas las opciones, con el fin de acepar tambien S/N

```
while True:
    opcion=input("Quieres resolver un sistema de ecuaciones de 3*3 (s/n)?: ")
    if opcion.lower()=="s":
        break
    elif opcion.lower()=="n":
        exit()
    else:
        os.system("cls")
```

Figura 9: funcion que solicita elementos en un arreglo 3x3

2. Solicitar elementos del sistema: En este paso lo primero que hacemos es crear una matriz 3x3 de ceros, con ayuda de la libreria Numpy, usando el metodo .zeros((3,3)), la cual crea la matriz antes descrita. Iniciamos un ciclo for, el cual tiene la finalidad de solicitar los elementos, ademas de que estos elementos tienen que ser validos, esto es posible gracias al ciclo while y un try, el cual obliga al usuario a ingresar un elemento de tipo entero (int) o tipo float), en caso de ingresar otro tipo de dato el ciclo while se va a repetir hasta ingresar en elemento valido.

```
matriz=num.zeros((3,3))
for i in range(3):
    for j in range(3):
        while True:
            try:
                matriz[i][j]=float(input(f"Ingrese el elemento [{i+1}][{j+1}] de la matriz: "))
                break
            except ValueError:
                os.system("cls")
                print("Ingresa un valor VALIDO!!!!")
```

Figura 10: funcion para solicitar elementos de un vector con 3 elementos

Esto fue para los valores de las variables (x,y,z).
Para ingresar los valores del vector B, observemos la figura 11. Nuevamente usamos la el metodo .zeros(3) de la libreria numpy, ahora solo es un vector con 3 elemetnos, el cual guarda los resultados de cada ecuacion

```
resultados=num.zeros(3)

for i in range(3):
    while True:
        try:
            resultados[i]=float(input(f"Ingrese el resultado de la ecuacion {i+1}: "))
            break
        except ValueError:
            os.system("cls")
            print("Ingrese un valor valido!!! ")
```

Figura 11: funcion que pregunta al usuario si desea resolver un sistema

3. Calculo resultados:
   En la figura 12, se observa que, el proceso es el mismo que en el ejemplo propuesto donde se busca la solucion por medio de la formula $X = A^{-1} * B$, asi usando la libreria Numpy, para calcular la matriz inversa, y posteriormente imprimir el resultado en consola

```
mat_inversa=num.linalg.inv(matriz)
resul=num.dot(mat_inversa,resultados)
print(f"La solucion a la ecuacion es:\nx={resul[0]}\ny={resul[1]}\nz={resul[2]}")
```

Figura 12: funcion para calcular el resultado de la matriz ingresada

4. Ejecucion.

4.1. Entrada y Salida de Arreglos

Ahora analizaremos el funcionamiento del programa
En la figura 13, se observa que nos muestra el menu, y nos pide que ingresemos

```
Mi programa
1.Media Aritmetica
2. Desviacion estandar
3. Burbuja
4. Salir
Ingrese la opcion que desea:
```

Figura 13: Ejecucion del programa

Si intentamos ingresar algun elemento no valido, en consola, como se muestra en la figura 14. El programa volvera a mostrar el menu y nuevamente pedira un elemento

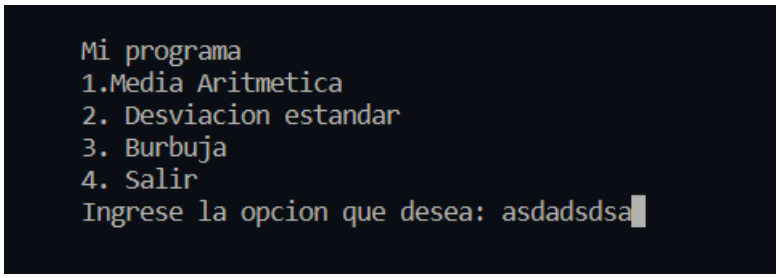

Figura 14: Ingreso de valores erroneos

Esto ocaciona que el programa se vuelva a ejecutar hasta obtener una respuesta valida.

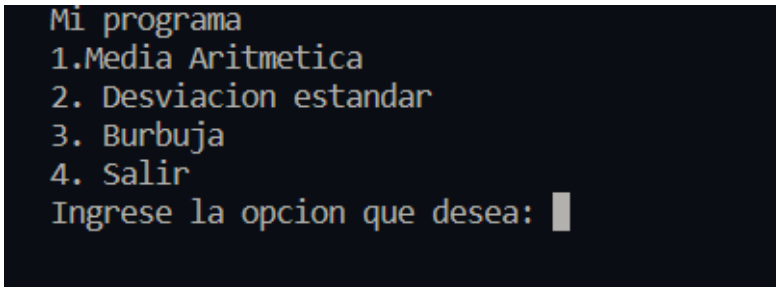

Figura 15: Retorno del programa al ingresar un valor erroneo

Una vez que ingresemos un elemento valido, por ejempllo, al ingresar el elemento 1, el cual nos deberia dirigir hacia la media aritmetica, veamos que sucede: Paa fines practicos vamos a ingresr 4 elementos, considerando que de la misma forma no nos dejara ingresar algun otro elemento que sea invalido. Vease la figura 16

Ahora vamos a analizar la forma de ingresar los 4 elementos, tambien ingrese un valor no valido para ver la respuesta del programa, veamos en las siguientes imagenes el resultado

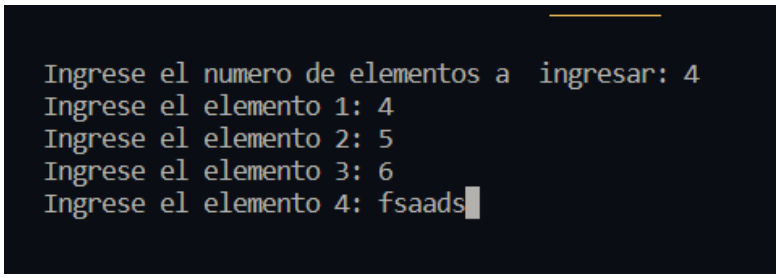

Figura 16: Ingreso de los 4 elementos

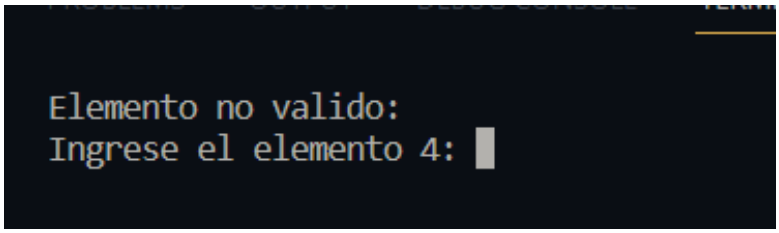

Figura 17: Respuesta al ingreso de un valor no valido

Notamos que nos regresa un mensaje de elemento no valido. Ademas de nuevamente solicitar el elemento 4.
Cuando ya ingresamos los datos correctos el programa nos arroja lo siguiente:

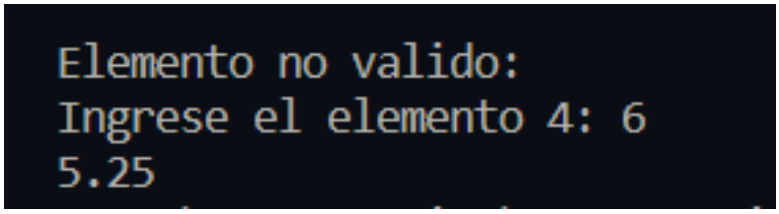

Figura 18: Resultado de la media aritmetica

//Ahora solo mostraremos la ejecucion del programa, en cada una de las 3 opciones restantes, consideraremos las entradas de forma correcta, ya que en caso de ingresar un elemento no valido, simplemente nos volvera a pedir el elemento.

- Desviacion estandar

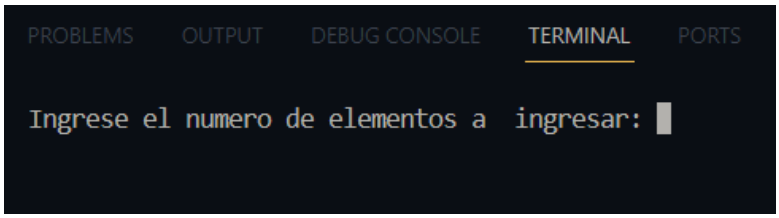

Figura 19: Nos pide el numero de elementos

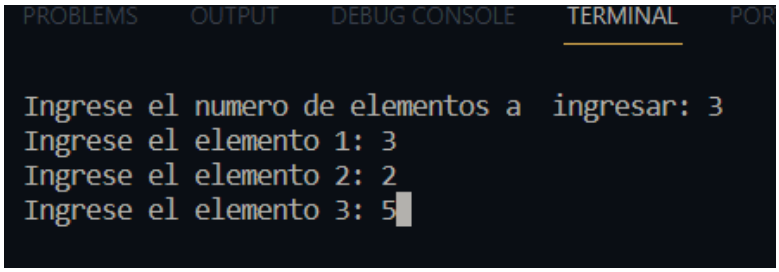

Figura 20: Ingreso de los elementos

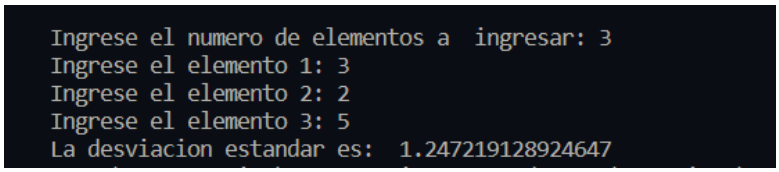

Figura 21: Resultado de la desviacion estandar

- Burbuja:
  Para esta funcion solo requiere el numero de elementos, ya que el resto lo realiza el programa (generar elementos y ordernarlos)

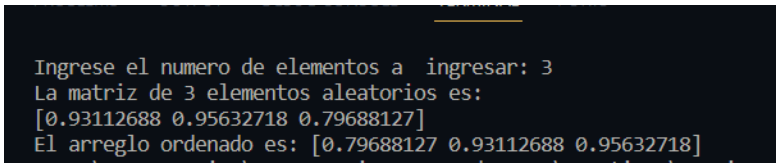

Figura 22: Metodo burbuja

- Salir:

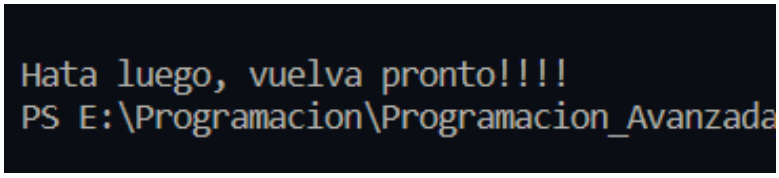

Figura 23: Opcion Salir

4.2. Sistema de ecuaciones

Al ejecutar directamante el programa nos arroja lo siguiente en consola:
Veamos que tambien nos pregunta si queremos resolver un sistema independiente, vamos a poner que si y seguir con el programa

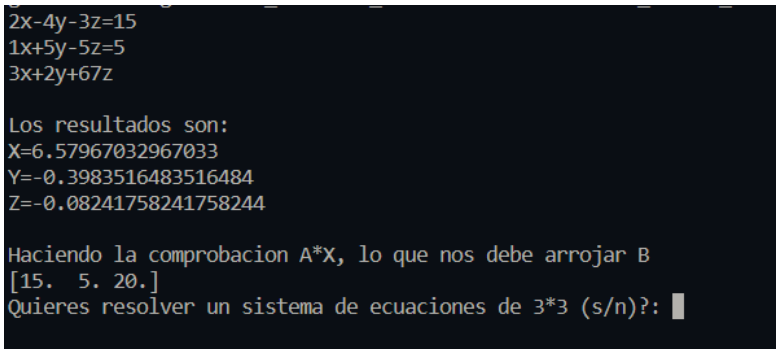

Figura 24: Resolucion del sistema de ecuaciones

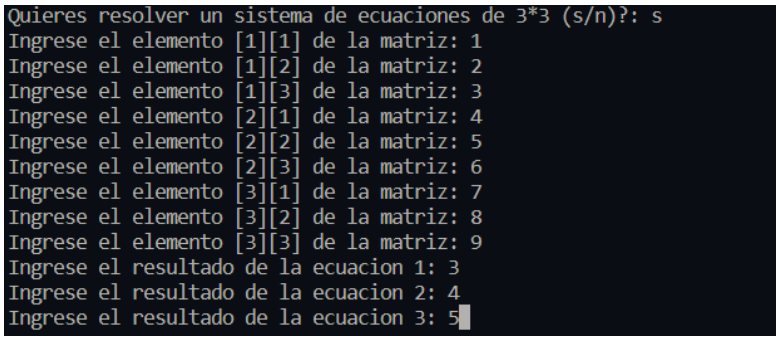

Figura 25: Ingresamos los valores del sistema de ecuaciones o la matriz, tambien ingresamos los resultados de cada ecuacion

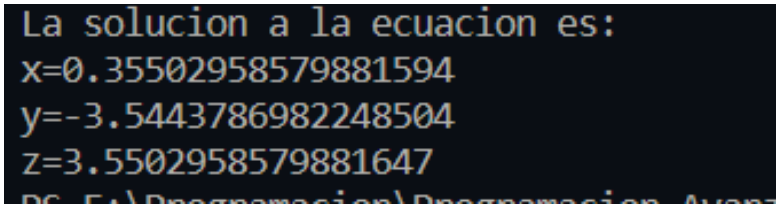

Figura 26: Resultado del sistema de ecuaciones

5. Conclusiones.

Al concluir esta practica e considerado que realizar este programa en python es bastante "sencillo.ª comparacion de algun otro lenguaje como lo es C ó C++, que son los que ya que gracias a las librerias como Numpy se puede calcular la matriz inversa de forma inmediata.Asi que dando una opinion de forma generalizada python contiene herramientas que son muy poderosas, solo debemos encontrar la forma correcta de realizar una tarea. Esta practica sirve como introduccion para lograr tener una mejor comprensio sobre este lenguaje de programacion, ya que una vez comprendido la logica de codificacion lo que resta es aprender la sintaxis, asi como

nos sirve de introduccion para los siguientes temas, dandonos una muestra del poder de python junto con sus modulos.