

# **EMTECH INSTITUTE**

## **Learning Path: Data Science**

**Tutor: Alonso Sánchez Jaime Saúl**

**Miembro: Resendiz de la Cruz Jose de Jesus**

**Grupo: 04**



**Proyecto 01 – Introducción a Python: Caso LifeStore**



# ÍNDICE

1.- Introducción

2.- Objetivo General

2.1.- Objetivos Particulares

3.- Desarrollo de la Problemática: Definición del Código.

3.1.- Problemática

3.2.- Reglas a Seguir

3.3.- Base de Datos

3.4.- Identificación del Perfil

4.- Solución de la Problemática: Creación de Listas.

4.1.- Lista para Visualización del Administrador.

4.1.- Listas para Visualización del Usuario

5.- Conclusiones:

5.1.- Propuesta de Retiro de Productos.

# 1.- INTRODUCCIÓN

En el presente trabajo, abordaremos la Problemática de la Tienda LifeStore, como una forma de poner en práctica los conocimientos adquiridos durante el curso “Fundamentos de Programación con Python” de EmTech Institute.

## 2.- OBJETIVO

Poner en práctica las bases de Programación en Python para análisis y clasificación de datos mediante la creación de *Programas de Estrada de Usuario y validaciones*, uso y definición de variables y lista, operadores lógicos y condicionales para la *clasificación de información*. Para lo cual tomaremos como objeto de estudio el caso LifeStore y, realizar un análisis de rotación de productos.

### 2.1.- Objetivos Particulares:

- Encontrar los Productos más vendidos y productos rezagados.
- Encontrar los Productos por reseña en el servicio.
- Sugerir una estrategia.

## 3.- Desarrollo: Definición del Código

### 3.1.- Problemática:

“LifeStore es una Tienda virtual que maneja una amplia gama de artículos, recientemente, la Gerencia de ventas, se percató que la empresa tiene una importante acumulación de inventario. Asimismo, se ha identificado una reducción en las búsquedas de un grupo importante de productos, lo que ha redundado en una disminución sustancial de sus ventas el último semestre”.

### 3.2.- Reglas a seguir:

*Es de suma importancia* aclarar que el desarrollo del código será elaborado únicamente con funciones Fundamentales en Python. Las cuales incluyen:

- Declaración de variables
- Creación de listas y funciones que interactúan en las mismas (len, append, remove, etc.)
- Funciones Int e Input
- Operadores Booleanos, Racionales, Lógicos y de Asignación.
- Herramientas de Control de Flujo (Sentencias de control condicional e Iterativo).
- Bucles
- Control de Bucles

Por lo cual, las instrucciones que no entran dentro de estos temas no podrán ser implementados. De igual forma, queda estrictamente prohibida la implementación de Funciones, Clases y librerías. Todo esto como una forma de implementar únicamente los conocimientos adquiridos durante el primer bloque.

Nota: Una función permitida pero no incluida dentro del programa es la **Función Sort**.

### 3.3.- Base de Datos:

Para el análisis utilizaremos el archivo “**lifestore-file.py**”, el cual contiene los registros de comprar, búsquedas y productos manejados por la tienda. Este archivo fue copiado dentro de nuestro propio Repositorio en GitHub, conservando el mismo nombre. Pero al no poder crear funciones que llamen un Data Set fuera de nuestro Script, desarrolle el código dentro del mismo archivo que incluye a todas las Listas.

En un primer momento se desarrollo el código dentro del **IDE repl.it**, pero por facilitar la ejecución y visualización de algunos resultados, las siguientes imágenes van a ser del IDE de Spyder. Mencionado lo anterior, podemos visualizar nuestra Base de Datos y, comprender lo que representa cada una de las listas:

```
C:\Users\casa\spvder-pv3\Lifystore\lifestore_file.py
lifestore_file.py
1 """
2 This is the LifeStore-SalesList data:
3
4 lifestore-searches = [id_search, id product]
5 lifestore-sales = [id_sale, id_product, score (from 1 to 5), date, refund (1 for true or 0 to false)]
6 lifestore-products = [id_product, name, price, category, stock]
7 """
```

Como podemos visualizar el nombre del archivo va ser “**lifestore\_file.py**” y dentro del archivo se incluyen los Datos de **Búsquedas**, **Ventas** y **Productos**. Si podemos atención a esto, podemos comprender mejor que representa cada uno de los valores, ya que a primera vista esto puede parecer que no tiene ningún contexto, eh aquí la importancia de esta primera nota.

```
lifestore_products = [
    [1, 'Procesador AMD Ryzen 3 3300X S-AM4, 3.80GHz, Quad-Core, 16MB L2 Cache', 3019, 'procesadores', 16],
    [2, 'Procesador AMD Ryzen 5 3600, S-AM4, 3.60GHz, 32MB L3 Cache, con Disipador Wraith Stealth', 4209, 'proces
```

```
lifestore_sales = [
    [1, 1, 5, '24/07/2020', 0],
    [2, 1, 5, '27/07/2020', 0],
    [3, 2, 5, '24/02/2020', 0],
lifestore_searches = [
    [1, 1],
    [2, 1],
    [3, 1],
```

En la imagen de la nota, se nos indican los atributos de cada una de las columnas de las listas que forman parte de nuestras bases de datos.

- **Productos** está conformada por: ID, nombre, precio, categoría y stock.
- **Ventas** está conformada por: ID de la venta y del producto, calificación, fecha y devolución.
- **Búsquedas** está conformada por: ID de la búsqueda y del producto.

Identificarlos en este momento nos ayudará a encontrar los valores que estamos buscando.

### 3.4.- Identificación del Perfil:

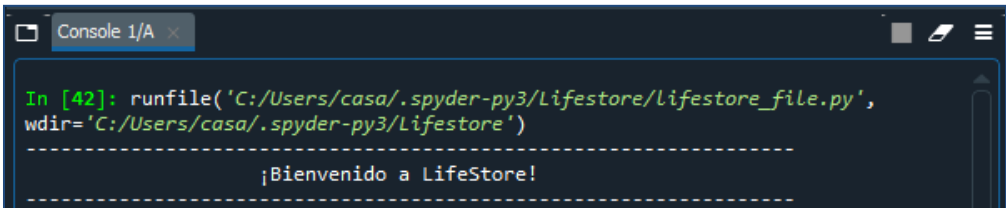
Como hemos visto hasta ahora, nuestro primer reto consiste en elaborar una instrucción que detecte la entrada de un **usuario – administrador** y, muestre los respectivos Reportes que se mencionaron con anterioridad. Para este paso, primero debemos declarar las variables que van a incluir el ID del usuario, ID del administrador y sus respectivas contraseñas.

```
# Variables - Perfiles y Contraseñas
administradores = ["Administrador1", "Administrador2"]
password_admin = ["jesus1", "admin2"]
usuarios = ["user1", "user2"]
password_user = ["user123", "user1234"]
```

En este paso también debemos declarar valores **Booleanos** que nos van a ayudar a detener la ejecución del código cuando se cumplan las condiciones de un Administrador o usuario, es decir, si el ID y la contraseña ingresados son correctos (los mismo valores que en las variables) se va detener la ejecución del código, continuando con la **visualización del Menú**.

```
# Variables de Entrada que detienen o repiten while del Mensaje de Entrada
entrada = True
admins = False
users = False

#-----Mensaje de Bienvenida-----
print("-----")
print("                ¡Bienvenido a LifeStore!")
print("-----")
```



El **Mensaje de Bienvenida** solo indica que estamos entrando a LifeStore, pero también nos ayuda tener una mejor visualización en la parte de la **Consola** cuando se ejecutan las líneas de código.

En la **Identificación del Perfil** pondremos a prueba las variables de administrador, usuario y contraseñas a prueba, sumado a las variables con valores Booleanos que declaramos en un principio. Todo esto iniciara con un **bucle While** y variables con una Función Input, que validaran que el valor ingresado en la Consola sea similar a las variables de Administrador y Usuario.

Si los valores ingresados son los correctos se arrojara el respectivo **mensaje de Bienvenida**, esto de acuerdo al usuario o administrador que haya ingresado.

Si la contraseña es incorrecta se volverá a solicitar que ingrese la Contraseña correcta, si el usuario ingresado y contraseña son erróneos se devolverá el mensaje que no se encuentra en el registro. Esto es posible gracias a la serie de **bucles If, Elif y Else**.

```
#-----Identificación de Perfil-----

while entrada:
    usuario_entrada = input("Ingrese su perfil (usuario/administrador): ")
    password_entrada =input("Ingree su contraseña: ")
    print("-----")
```

```

if usuario_entrada in administradores:
    if usuario_entrada == administradores[0] and password_entrada == password_admin[0]:
        entrada = False
        admins = True
        print("                ¡Bienvenido Administrador Jesus!")
    elif usuario_entrada == administradores[1] and password_entrada == password_admin[1]:
        entrada = False
        admins = True
        print("                ¡Bienvenido Administrador 2!")
    else:
        print("Por favor intente de nuevo...")
elif usuario_entrada in usuarios:
    if usuario_entrada == usuarios[0] and password_entrada == password_user[0]:
        entrada = False
        users = True
        print("                ¡Bienvenido Usuario1!")
    elif usuario_entrada == usuarios[1] and password_entrada == password_user[1]:
        entrada = False
        users = True
        print("                ¡Bienvenido Usuario 2!")
    else:
        print("Por favor intente de nuevo...")
else:
    print("Lo sentimos no se encontro registro...")
    entrada = True

```

El resultado obtenido es el siguiente:

```

-----
Ingrese su perfil (usuario/administrador): Administrador1
Ingree su contraseña: jesus1
-----
                ¡Bienvenido Administrador Jesus!
-----

```

Si los datos ingresados fueran erróneos, el resultado sería:

```

-----
Ingrese su perfil (usuario/administrador): Administrador1
Ingree su contraseña: 111
-----
Por favor intente de nuevo...
Ingrese su perfil (usuario/administrador): |

```

En esta ocasión al ingresar datos erróneos, se nos regresa al primer Input, para volver a introducir los datos. Como en el primer caso si obtuvimos el mensaje de Bienvenida, los datos ingresados fueron los correctos, ahora se nos debe desplegar el *Menú* que corresponda para el perfil ingresado. En este caso solo declare un **Bucle If** que ayudado de **valores Booleanos** detecta si el perfil ingresado es un Administrador o un Usuario.

Para determinar el número de opciones que deseo mostrar en mi código, organice la información proporcionada en una lluvia de ideas, separando cada una de los requisitos que planteo el problema en diferentes opciones para el Menú.

El problema nos pide crear:

- ✚ Generar un Listado de los 15 Productos con MAYORES VENTAS.
- ✚ Generar un Listado de los 20 Productos con MAYORES BÚSQUEDAS.
- ✚ Generar un Listado de los 5 Productos con MENORES VENTAS.
- ✚ Generar un Listado de los 20 Productos con MENORES BÚSQUEDAS.
  
- ❖ Mostrar 1 Lista de 10 Productos con las MEJORES RESEÑAS.
- ❖ Mostrar 1 Lista de 10 Productos con las PEORES RESEÑAS.  
Consideras Productos con Devolución.
  
- Mostrar Ventas con MENSUALES.

PLUS:

- Sugerir una estrategia de PRODUCTOS A RETIRAR.
- Sugerencia de cómo REDUCIR LA ACUMULACIÓN DE INVENTARIO  
Considerando datos de Ingresos y Ventas Mensuales.

Estas opciones vistas desde un desarrollador de código son las que solicitaría la parte interesada, por lo cual, son opciones que decidí incluir para el *perfil del Administrador*. Un Usuario no tendría porque ver toda esta información, para el creamos un Menú diferente, pero que vamos a incluir en el **Bucle Elif**.

```
#-----Lista de Opciones -----
if admins == True:
    print("-----")
    print("Las opciones disponibles son Las siguientes: ")
    print("1 Lista de MAYORES Ventas.")
    print("2 Lista de MAYORES Búsquedas.")
    print("3 Lista con MENORES Ventas.")
    print("4 Lista de MENORES Búsquedas.")
    print("5 Lista de MEJORES Reseñas.")
    print("6 Lista de PEORES Reseñas.")
    print("7 Ventas MENSUALES.")
    print("-----")
elif users == True:
    print("-----")
    print("Las opciones disponibles son Las siguientes: ")
    print("A Lista de Productos.")
    print("B Lista de Productos MÁS VENDIDOS.")
    print("C Lista con Productos MÁS BUSCADOS.")
    print("-----")
```



Cuando el Perfil detectado es el de un *Administrador*, el mensaje del menú sería el siguiente:

```
Console 1/A x
-----
¡Bienvenido Administrador Jesus!
-----
Las opciones disponibles son las siguientes:
1 Lista de MAYORES Ventas.
2 Lista de MAYORES Búsquedas.
3 Lista con MENORES Ventas.
4 Lista de MENORES Búsquedas.
5 Lista de MEJORES Reseñas.
6 Lista de PEORES Reseñas.
7 TOTAL de Ingresos.
8 Ventas promedio MENSUALES.
9 Ventas promedio ANUALES.
10 Meses con MÁS Ventas.
-----
```

Por otro lado, si el perfil ingresado corresponde al de un *Usuario*, el menú sería diferente:

```
Console 1/A x
Ingrese su perfil (usuario/administrador): user1
Ingres su contraseña: user123
-----
¡Bienvenido Usuario1!
-----
Las opciones disponibles son las siguientes:
A Lista de Productos.
B Lista de Productos MÁS VENDIDOS.
C Lista con Productos MÁS BUSCADOS.
-----
```

Como podemos observar, las opciones del Menú están señaladas por *número* en el *perfil del Administrador* y por *Letras* en el *perfil del Usuario*. Lo cual va evitar confundirnos en el desarrollo y visualización de Listas por las que tenemos interés.

## 4.- Solución: Creación de Listas.

### 4.1 - Listas para Visualización del Administrador.

Con el Menú definido, podemos pasar a la creación de Listas que incluirán a los datos que son de nuestro interés. Donde ahora nuestro código va interactuar directamente con la Base de datos que tenemos desde el inicio. Las primeras 7 opciones de nuestro menú tienen valores numéricos y hacen referencia a las opciones disponibles para un Administrador.

La primera opción es: *Lista de Mayores Ventas*.

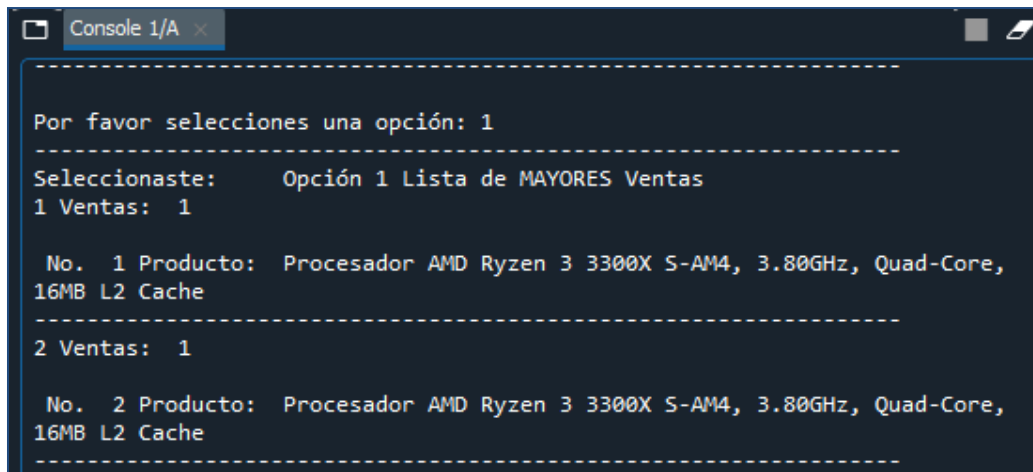
Como podemos ver a continuación, el código inicia con la **Función Int** que se encargara de solicitar el valor u opción que se desea visualizar. Y con cada **sentencia If** vamos a interactuar con las opciones disponibles.

**NOTA:** El código tiene comentarios línea por línea, para comprender un poco mejor qué papel desempeña cada una dentro del mismo. De igual forma, esta primera iteración es la más tardada.

Creamos listas vacías (**ventas\_totales** y **ventas\_ordenadas**) que posteriormente se irán ocupando con las iteración entre la Lista de Productos y de Ventas. Buscando primero por ID del producto y posteriormente por número de ventas.

```
if select == "1":
    print("Seleccionaste: Opción 1 Lista de MAYORES Ventas")
    #Declaramos la variable que se encargara de ir contando las iteraciones con la B
    contador = 0 #En el último bucle for nos ayudara a detener la búsqueda cuando se
    ventas_totales = [] #Declaramos nuestras listas que en un primer momento se enco
    ventas_ordenadas = [] #Con los bucles for, While y la función append se añadirán
    #Con este primer bucle for interactuamos directamente con las listas que son de
    for producto in lifestore_products:
        for venta in lifestore_sales:
            if producto[0] == venta[1]: # Buscamos el índice del ID del Producto, el
                contador +=1
            formato = [producto[0], contador, producto[1]] #Buscamos índices especifi
            ventas_totales.append(formato) #Se añaden a la Lista de Ventas Totales
            contador = 0
        while ventas_totales:
            maximo = ventas_totales[0][1] #Con esta variable indicamos el índice que des
            lista_actual = ventas_totales[0]
            for grupo in ventas_totales: #Buscamos dentro de ventas totales.
                if grupo[1]>maximo: #Y obtendremos los que cumplan solo esta condición.
                    maximo = grupo[1] #Declaramos el índice que deseamos de la interacción
                    lista_actual = grupo
            ventas_ordenadas.append(lista_actual) #Agregamos los valores obtenidos a la
            ventas_totales.remove(lista_actual) #Si alguna de esta
        for i in range(0,15): #Obtendremos solo los primeros 15 productos
            contador +=1 #Nuestro contador va ir sumando con cada producto que se añade
            print(contador, "Ventas: ", ventas_ordenadas[i][1]) #Le damos formato a los
            print("Número ", contador, "Producto: ", ventas_ordenadas[i][2])
            print("-----")
```

Como resultado obtendremos los siguientes resultados dentro de nuestra consola:



```
Por favor selecciones una opción: 1
-----
Seleccionaste:      Opción 1 Lista de MAYORES Ventas
1 Ventas: 1

No. 1 Producto:  Procesador AMD Ryzen 3 3300X S-AM4, 3.80GHz, Quad-Core,
16MB L2 Cache
-----
2 Ventas: 1

No. 2 Producto:  Procesador AMD Ryzen 3 3300X S-AM4, 3.80GHz, Quad-Core,
16MB L2 Cache
-----
```

Como podemos observar el conteo se va detener en el número que hayamos especificado dentro del último bucle For, de ahí la importancia de la variable contador que se detiene llegado a ese límite.

Para nuestra **Lista de Mayores Búsquedas** creamos las listas vacías “*búsquedas\_totales*” y “*búsquedas\_ordenadas*” que posteriormente se ocuparan con la iteración del Bucle for que buscara dentro de la Lista de Productos y la Lista de Búsquedas el ID del Producto. Como buscamos solo los primero 20 Productos más buscados, el contador se va detener cuando llegue a ese número.

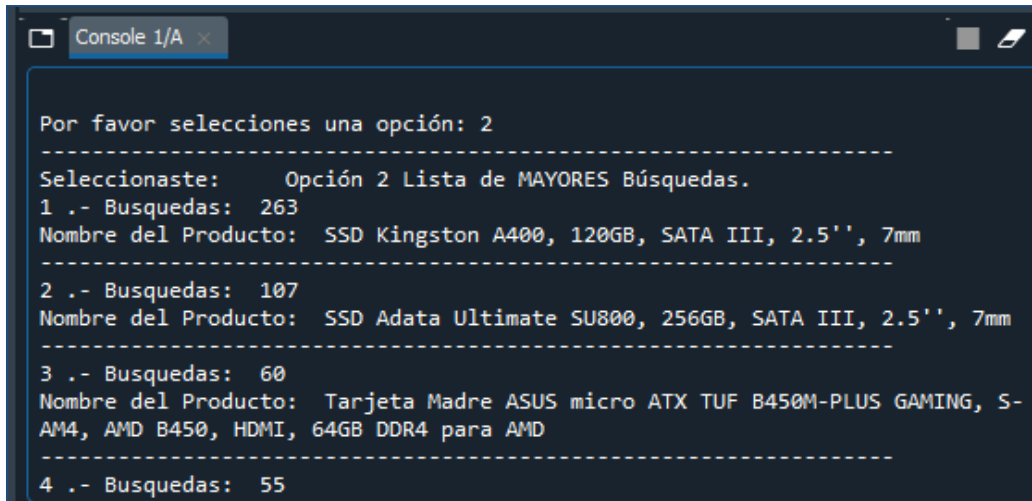
```
elif select == "2":
    print("Seleccionaste:      Opción 2 Lista de MAYORES Búsquedas.")
    contador = 0 # Variable que nos va servir para contar el número de iteraciones.
    busquedas_totales = [] #Declaramos las Listas vacias
    busquedas_ordenadas = []

    for producto in lifestore_products:
        for buscar in lifestore_searches:
            if producto[0] == buscar[1]: #Busqueda del ID de los productos dentro de
                contador += 1
            formato_busquedas = [producto[0], contador, producto[1]] #Formato de las bus
            busquedas_totales.append(formato_busquedas)
            contador = 0

    while busquedas_totales:
        maximo = busquedas_totales[0][1]
        lista1 = busquedas_totales[0]
        for grupo in busquedas_totales:
            if grupo[1] > maximo:
                maximo = grupo[1]
                lista1 = grupo
        busquedas_ordenadas.append(lista1)
        busquedas_totales.remove(lista1)

    for i in range (0,20):
        contador += 1
        print(contador, "- ", "Busquedas: ", busquedas_ordenadas[i][1]) #Agregamos f
        print("Nombre del Producto: ", busquedas_ordenadas[i][2])
        print("-----")
```

Como resultado obtendremos:



```
Console 1/A x

Por favor selecciones una opción: 2
-----
Seleccionaste:      Opción 2 Lista de MAYORES Búsquedas.
1 .- Búsquedas:  263
Nombre del Producto: SSD Kingston A400, 120GB, SATA III, 2.5'', 7mm
-----
2 .- Búsquedas:  107
Nombre del Producto: SSD Adata Ultimate SU800, 256GB, SATA III, 2.5'', 7mm
-----
3 .- Búsquedas:   60
Nombre del Producto: Tarjeta Madre ASUS micro ATX TUF B450M-PLUS GAMING, S-
AM4, AMD B450, HDMI, 64GB DDR4 para AMD
-----
4 .- Búsquedas:   55
```

Como podemos observar, el contador indica la posición del Producto conforme a su búsqueda, añadiendo el Nombre del Producto.

Con **las Listas de Menores Ventas y Menores Búsquedas** es prácticamente lo mismo, reciclamos gran parte del código. Pero en la iteración dentro de For en lugar de buscar lo que cumplan un con “**mayor que**” indicamos que debe ser “**menor que**”.

```
elif select == "3":
    print("Seleccionaste:      Opción 3 Lista con MENORES Ventas.")
    contador = 0
    ventas_totales = []
    ventas_ordenadas = []

    for producto in lifestore_products:
        for venta in lifestore_sales:
            if producto[0] == venta[1]:
                contador += 1
            formato = [producto[0], contador, producto[1]]
            ventas_totales.append(formato)
            contador = 0

    while ventas_totales:
        minimo = ventas_totales[0][1]
        lista_actual = ventas_totales[0]
        for grupo in ventas_totales:
            if grupo[1] < minimo:
                minimo = grupo[1]
                lista_actual = grupo
        ventas_ordenadas.append(lista_actual)
        ventas_totales.remove(lista_actual)

    for i in range(0, 15):
        contador += 1
        print(contador, "Ventas: ", ventas_ordenadas[i][1])
        print(contador, " Nombre: ", ventas_ordenadas[i][2])
        print("-----")
```

Como resultado obtenemos:

```
Por favor selecciones una opción: 3
-----
Seleccionaste:      Opción 3 Lista con MENORES Ventas.
1 Ventas:  0
1  Nombre:  Procesador Intel Core i3-8100, S-1151, 3.60GHz, Quad-Core, 6MB
Smart Cache (8va. Generación - Coffee Lake)
-----
2 Ventas:  0
2  Nombre:  Tarjeta de Video EVGA NVIDIA GeForce GT 710, 2GB 64-bit GDDR3,
PCI Express 2.0
```

Los productos son completamente diferentes a lo que obteníamos con la primera Lista. Con la Lista de **MENORES Búsquedas** tenemos:

```
elif select == "4":
    print("Seleccionaste:      Opción 4 Lista de MENORES Búsquedas")
    agregar = 1
    contador = 0
    busquedas_totales = []
    busquedas_ordenadas = []

    for producto in lifestore_products:
        for buscar in lifestore_searches:
            if producto[0] == buscar[1]:
                contador += 1
            formato = [producto[0], contador, producto[1]]
            busquedas_totales.append(formato)
            contador = 0

    while busquedas_totales:
        minimos = busquedas_totales[0][1]
        lista_actual1 = busquedas_totales[0]
        for grupo in busquedas_totales:
            if grupo[1] < minimos:
                minimos = grupo[1]
                lista_actual1 = grupo
        busquedas_ordenadas.append(lista_actual1)
        busquedas_totales.remove(lista_actual1)

    for i in range(0,20):
        contador += 1
        print(contador,": ", "Busquedas: ", busquedas_ordenadas[i][1])
        print("Producto: ", busquedas_ordenadas[i][2])
        print("-----")
```

```
Console 1/A x
Por favor selecciones una opción: 4
-----
Seleccionaste:      Opción 4 Lista de MENORES Búsquedas
1 : Busquedas:  0
Producto:  Tarjeta de Video EVGA NVIDIA GeForce GT 710, 2GB 64-bit GDDR3,
PCI Express 2.0
-----
2 : Busquedas:  0
Producto:  Tarjeta de Video EVGA NVIDIA GeForce RTX 2060 SC ULTRA Gaming,
6GB 192-bit GDDR6, PCI Express 3.0
-----
3 : Busquedas:  0
Producto:  Tarjeta de Video Gigabyte NVIDIA GeForce GTX 1650 OC Low Profile,
4GB 128-bit GDDR5, PCI Express 3.0 x16
-----
```

En la **Lista de Mejores Reseñas**, creamos una iteración entre la Lista de Productos y la Lista de Ventas, pero le daremos importancia al **Índice 2**, el cual hace referencia a “**Score**” o la **calificación** que recibieron los productos vendidos. Las reseñas serán mostradas junto con el Nombre del producto, al devolver los valores de toda la función.

```
elif select == "5":
    print("Seleccionaste:      Opción 5 Lista de MEJORES Reseñas.")
    agregar = 1
    contador = 0
    suma = 0
    mejores_resenas = []
    resenas_ordenadas = []

    for producto in lifestore_products:
        for mejores in lifestore_sales:
            if producto[0] == mejores[1]:
                contador += 1
                suma += mejores[2]
        media = suma / contador
        formato = [media, producto[1], producto[0],]
        mejores_resenas.append(formato)

    while mejores_resenas:
        mejor = mejores_resenas[0][2]
        lista_actual1 = mejores_resenas[0]
        for grupo in mejores_resenas:
            if grupo[0] < mejor:
                mejor += grupo[2]
                lista = grupo
        resenas_ordenadas.append(lista)
        mejores_resenas.remove(lista)

    for i in range(0,10):
        print("Producto: ",resenas_ordenadas[i][1])
        print("Calificación en Reseña: ",resenas_ordenadas[i][0])
        print("-----")
```

```

Por favor selecciones una opción: 5
-----
Seleccionaste:      Opción 5 Lista de MEJORES Reseñas.
Producto:  Procesador AMD Ryzen 3 3300X S-AM4, 3.80GHz, Quad-Core, 16MB L2
Cache
Calificación en Reseña:  5.0
-----
Producto:  Procesador AMD Ryzen 5 3600, S-AM4, 3.60GHz, 32MB L3 Cache, con
Disipador Wraith Stealth
Calificación en Reseña:  4.333333333333333
-----
Producto:  Procesador AMD Ryzen 5 2600, S-AM4, 3.40GHz, Six-Core, 16MB L3
Cache, con Disipador Wraith Stealth
Calificación en Reseña:  4.684210526315789

```

Como podemos observar los productos son listados de acuerdo a su Reseña, pero también podemos visualizar los datos del Producto que fue vendido y calificado.

Por último, tenemos la **Lista de Ventas PROMEDIO Mensuales**. En donde vamos a crear una Lista de las Fechas que va a ir añadiendo con la **Función Append** las Ventas de acuerdo a la Lista de Ventas de LifeStore. Dentro de esta Iteración crearemos una nueva lista que se encargara de recolectar los valores de LifeStore (en este caso Venta) en el **Índice 3** que pertenece a las **Fechas**. Como Fechas puede ser tratado como un **Subíndice**, podemos indicarle a la función que se encargue de trabajar con los valores de **Mes** y **Año**.

Logrando que se contabilicen las Ventas de cada uno de los Meses (uno por uno).

```
elif select == "7":
    print("Seleccionaste: Opción 7 Ventas promedio MENSUALES.")
    lista_fecha= []
    for venta in lifestore_sales:
        lista_fecha.append(venta[3])
        lista_anos = []
    for ano in lista_fecha:
        lista_anos.append([ano[3:5], ano[6:10]])
    for venta_mes in lista_anos:
        enero = lista_anos.count(['01','2020'])
    print("Enero: ", enero)
    for venta_mes in lista_anos:
        febrero = lista_anos.count(['02','2020'])
    print("Febrero: ", febrero)
    for venta_mes in lista_anos:
        marzo = lista_anos.count(['03','2020'])
    print("Marzo: ", marzo)
    for venta_mes in lista_anos:
        abril = lista_anos.count(['04','2020'])
    print("Abril: ", abril)
    for venta_mes in lista_anos:
        mayo = lista_anos.count(['05','2020'])
    print("Mayo: ", mayo)
    for venta_mes in lista_anos:
        junio = lista_anos.count(['06','2020'])
    print("Junio: ", junio)
    for venta_mes in lista_anos:
        julio = lista_anos.count(['07','2020'])
    print("Julio: ", julio)
    for venta_mes in lista_anos:
        agosto = lista_anos.count(['08','2020'])
    print("Agosto: ", agosto)
    for venta_mes in lista_anos:
        septiembre = lista_anos.count(['09','2020'])
    print("Septiembre: ", septiembre)
```

```
In [106]: venta
Out[106]: [283, 94, 4, '10/04/2020', 0]

In [107]: ano
Out[107]: '10/04/2020'

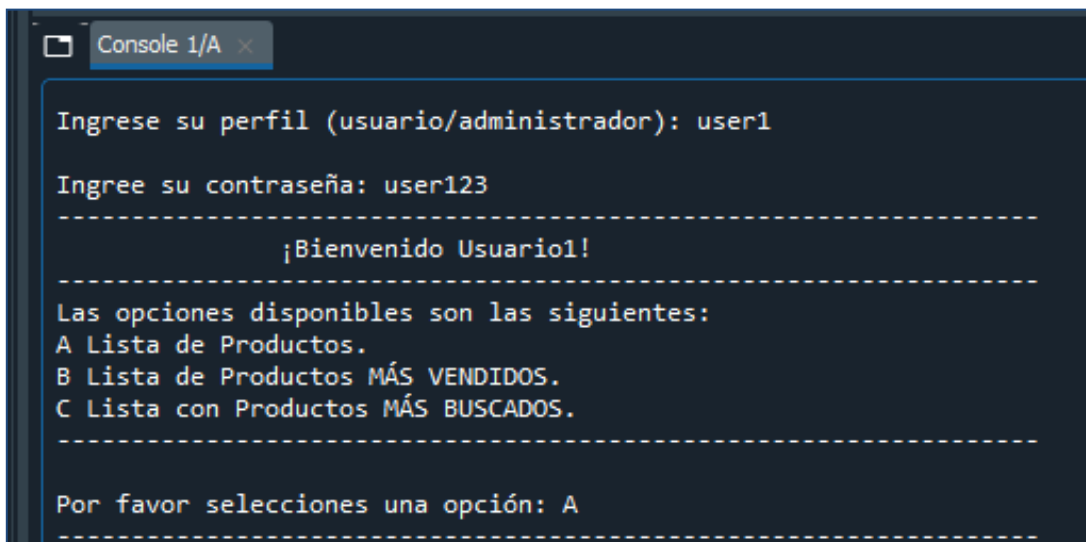
In [108]:
```

```
-----
Seleccionaste: Opción 7 Ventas promedio MENSUALES.
Enero: 53
Febrero: 41
Marzo: 51
Abril: 75
Mayo: 35
Junio: 11
Julio: 11
Agosto: 3
Septiembre: 1
Octubre: 0
Noviembre: 0
Diciembre: 0
```

## 4.2 - Listas para Visualización del Usuario.

Así como obtuvimos un Menú que es único para los Administradores, podemos obtener un Menú que lo sea para los usuarios. Esto para evitar que no puedan tener acceso a “información privilegiada”.

El **Menú de los Usuarios** es el siguiente:



```
Console 1/A x
Ingrese su perfil (usuario/administrador): user1
Ingree su contraseña: user123
-----
¡Bienvenido Usuario1!
-----
Las opciones disponibles son las siguientes:
A Lista de Productos.
B Lista de Productos MÁS VENDIDOS.
C Lista con Productos MÁS BUSCADOS.
-----
Por favor selecciones una opción: A
-----
```

Como ya se había mencionado con anterioridad, tenemos un menú más reducido y señalado por letras. Esto como una medida de estética y evitar complicaciones con el Menú de Usuario. Parte del Menú corresponde a código trabajado con anterioridad, pero modificado ligeramente para evitar la visualización de algunas variables que no sean de interés para el Usuario.

La Lista de Producto llama a la Lista del mismo nombre que se encuentra dentro de nuestra Base de Datos el “**lifestore\_products**”

```
elif select == "A":
    print("Seleccionaste: Opción A Lista de Productos.")
    print(lifestore_products)
```

Como podemos ver, seguimos trabajando dentro de la misma sentencia If de un inicio, ya que el menú cuenta con diferentes variables de entrada, podemos trabajar sobre la misma Función.

En la **Lista B de Productos Más vendidos** tenemos una lista más reducida de la que puede observar el Administrador, así como la omisión del **Contador**, sonde la Consola solo muestra los nombres de los 10 Productos más vendidos. El cual queda de la siguiente forma:



```

elif select == "B":
    print("Seleccionaste:      Opción B Lista de Productos MÁS Vendidos.")
    contador = 0
    ventas_totales = []
    ventas_ordenadas = []

    for producto in lifestore_products:
        for venta in lifestore_sales:
            if producto[0] == venta[1]:
                contador +=1
                formato = [producto[0], contador, producto[1]]
                ventas_totales.append(formato)
                contador = 0
    while ventas_totales:
        maximo = ventas_totales[0][1]
        lista_actual = ventas_totales[0]
        for grupo in ventas_totales:
            if grupo[1]>maximo:
                maximo = grupo[1]
                lista_actual = grupo
        ventas_ordenadas.append(lista_actual)
        ventas_totales.remove(lista_actual)
    for i in range(0,10):
        contador +=1
        print("\n No. ", contador, "Producto: ", ventas_ordenadas[i][2])
        print("-----")

```

De igual forma, en la **Lista C de Productos Más Buscados** también contamos con una versión del código más sencilla. Ocultando así mismo el contador y reduciendo la Lista de Productos que se pueden visualizar.

```

elif select == "C":
    print("Seleccionaste:      Opción C Lista con Productos MÁS BUSCADOS.")
    contador = 0
    busquedas_totales = []
    busquedas_ordenadas = []

    for producto in lifestore_products:
        for buscar in lifestore_searches:
            if producto[0] == buscar[1]:
                contador += 1
                formato_busquedas = [producto[0], contador, producto[1]]
                busquedas_totales.append(formato_busquedas)
                contador = 0

    while busquedas_totales:
        maximo = busquedas_totales[0][1]
        lista1 = busquedas_totales[0]
        for grupo in busquedas_totales:
            if grupo[1] > maximo:
                maximo = grupo[1]
                lista1 = grupo
        busquedas_ordenadas.append(lista1)
        busquedas_totales.remove(lista1)

    for i in range (0,10):
        contador += 1
        print("Nombre del Producto: ", busquedas_ordenadas[i][2])
        print("-----")

```

Para finalizar, nuestro código contara con un mensaje de “**Selección Invalida**”, cual la opción que se ingrese no sea ninguna de las opciones que tenemos en el menú.

```
else:  
    print("La opción que selecciono no es valida...")  
    print("Por favor, intente nuevamente")
```

```
Por favor selecciones una opción: 11  
-----  
La opción que selecciono no es valida...  
Por favor, intente nuevamente
```

Si el mensaje que arroje la consola es el que se muestra en la imagen anterior, solo basta con volver a ejecutar el código y las opciones para ingresar el perfil y la visualización del menú podrán verse nuevamente.

## 4.- Solución: Creación de Listas.

Llegados a este punto, podemos concluir que la visualización de Bases de Datos sin la utilización de Librerías o funciones más avanzadas puede volver un poco más complicado el análisis de datos. Pero al trabajar con funciones Fundamentales en Python, podemos entender de mejor manera el proceso de creación de algoritmos que nos ayuden en el análisis de Bases de datos, es decir, entendemos de una manera más clara el uso de listas, índices y Herramientas de Control de Flujo.

Conforme vas avanzando en la creación de nuevas listas, llegan ideas de la creación de nuevas listas que nos faciliten aún más el análisis.

### 4.1 - Listas para Visualización del Administrador.

Como propuesta para la Tienda LifeStore en sus productos a retirar, podemos basarnos en las Listas de los Productos que tienen una calificación más baja en sus ventas o inclusive en las que tienen más devoluciones. Pero como contamos con muchos productos que tienen ventas muy bajas o nulas, podemos comenzar con retirar esos productos.

El Administrador puede visualizar la Lista de Productos MENOS Vendidos, así como los Productos que tienen MENOS Búsquedas, y por ende, no tienen ninguna venta realizada. Algunos de los productos que podemos por comenzar a retirar son:

```
Seleccionaste: Opción 3 Lista con MENORES Ventas.
1 Ventas: 0
1 Nombre: Procesador Intel Core i3-8100, S-1151, 3.60GHz, Quad-Core, 6MB Smart
Cache (8va. Generación - Coffee Lake)
-----
2 Ventas: 0
2 Nombre: Tarjeta de Video EVGA NVIDIA GeForce GT 710, 2GB 64-bit GDDR3,
PCI Express 2.0
-----
3 Ventas: 0
3 Nombre: Tarjeta de Video EVGA NVIDIA GeForce GTX 1660 Ti SC Ultra
Gaming, 6GB 192-bit GDDR6, PCI 3.0
-----
4 Ventas: 0
4 Nombre: Tarjeta de Video EVGA NVIDIA GeForce RTX 2060 SC ULTRA
Gaming, 6GB 192-bit GDDR6, PCI Express 3.0
-----
5 Ventas: 0
5 Nombre: Tarjeta de Video Gigabyte NVIDIA GeForce GTX 1650 OC Low Profile,
4GB 128-bit GDDR5, PCI Express 3.0 x16
-----
```

6 Ventas: 0

6 Nombre: Tarjeta de Video Gigabyte NVIDIA GeForce RTX 2060 SUPER WINDFORCE OC, 8 GB 256 bit GDDR6, PCI Express x16 3.0

7 Ventas: 0

7 Nombre: Tarjeta de Video MSI Radeon X1550, 128MB 64 bit GDDR2, PCI Express x16

8 Ventas: 0

8 Nombre: Tarjeta de Video PNY NVIDIA GeForce RTX 2080, 8GB 256-bit GDDR6, PCI Express 3.0

9 Ventas: 0

9 Nombre: Tarjeta de Video VisionTek AMD Radeon HD 5450, 1GB DDR3, PCI Express x16 2.1

10 Ventas: 0

10 Nombre: Tarjeta de Video VisionTek AMD Radeon HD5450, 2GB GDDR3, PCI Express x16

11 Ventas: 0

11 Nombre: Tarjeta Madre AORUS ATX Z390 ELITE, S-1151, Intel Z390, HDMI, 64GB DDR4 para Intel

12 Ventas: 0

12 Nombre: Tarjeta Madre ASRock Z390 Phantom Gaming 4, S-1151, Intel Z390, HDMI, 64GB DDR4 para Intel

13 Ventas: 0

13 Nombre: Tarjeta Madre ASUS ATX ROG STRIX B550-F GAMING WI-FI, S-AM4, AMD B550, HDMI, max. 128GB DDR4 para AMD

14 Ventas: 0

14 Nombre: Tarjeta Madre Gigabyte micro ATX Z390 M GAMING, S-1151, Intel Z390, HDMI, 64GB DDR4 para Intel

15 Ventas: 0

15 Nombre: Tarjeta Madre Gigabyte micro ATX Z490M GAMING X (rev. 1.0), Intel Z490, HDMI, 128GB DDR4 para Intel