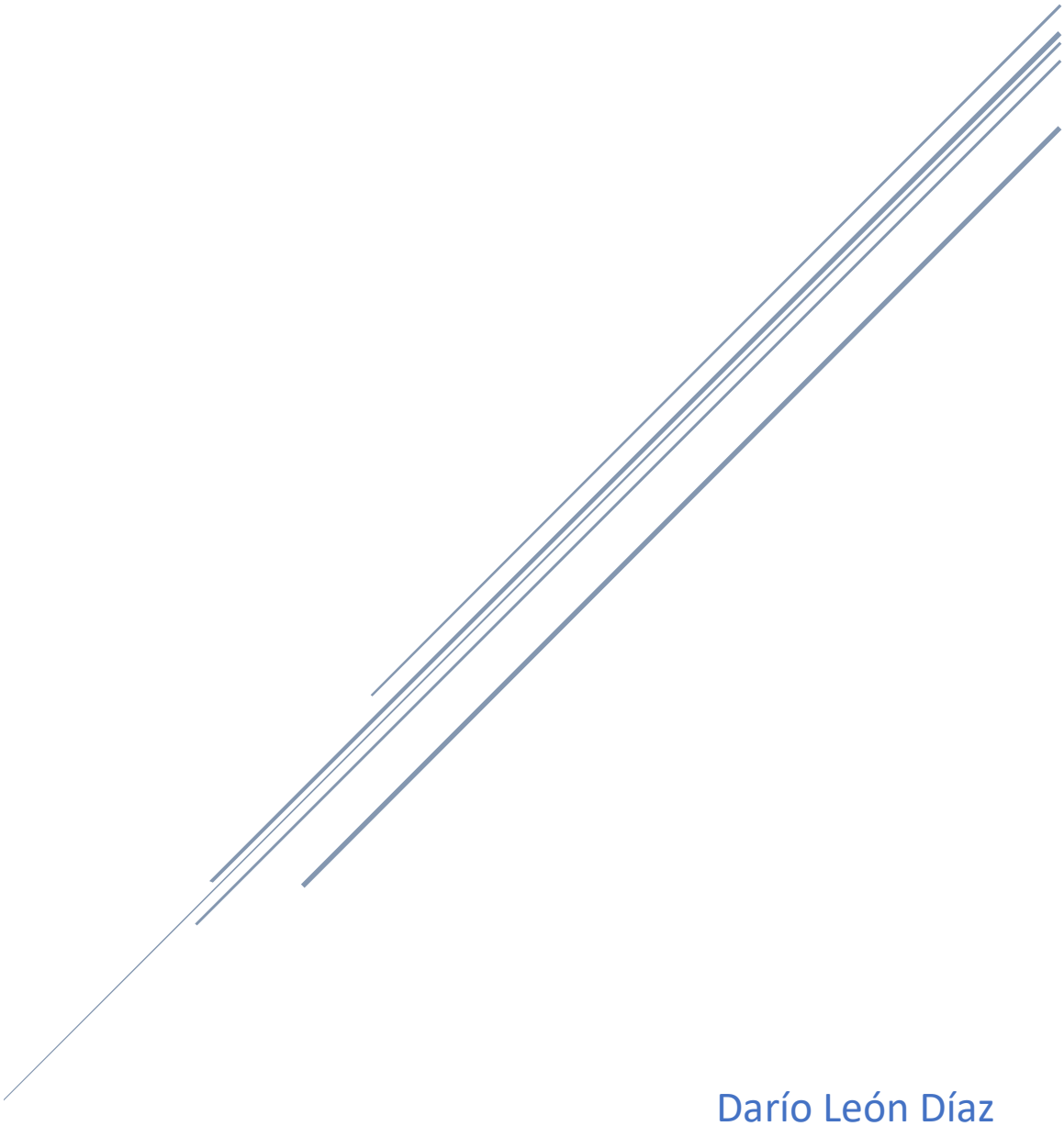
PROYECTO FINAL

Fundamentos de programación con Python



Darío León Díaz
EMTECH

Índice

Introducción

El presente documento contiene una explicación detallada de la ejecución del proyecto final del curso Fundamentos de Programación con Python proporcionado por EMTECH, con el objetivo de reafirmar los conocimientos aprendidos en todo el curso como lo son el uso de variables, índices, listas, ingreso de datos, booleanos, operadores relacionales, operadores lógicos, operadores de asignación, sentencias if (if, elif, else), bucles while y for, y control de bucles.

El software fue creado en el lenguaje de programación Python en la plataforma online Repl.it.

El objetivo de este software radica en el análisis de la rotación de los productos identificando cuales son los productos más vendidos y buscados, los productos menos vendidos y buscados por categoría, los productos con mejores y peores reseñas, total de ingresos y ventas promedio mensuales, total anual y meses con más ventas.

Por último, se sugiere una estrategia a partir del análisis de los resultados, así como una solución a la acumulación del inventario y que productos deben salir del mercado.

Definición del código

En este apartado se describe el código, el proceso y las variables.

Importación de librerías, archivos necesarios

1. En un inicio, se realizan las importaciones correspondientes del archivo lifestore_file.py, además de otras librerías como datetime para manejar fechas, time para realizar esperas y locale para formatear los ingresos en moneda.
2. Posteriormente se ocupa un primer ciclo while para poder regresar al ingreso de usuario y contraseña.

```
lista_usuarios=[["dario","progpython"],["juan","python123"],["david","admin1234"]
```

3. Se genera una lista de usuarios y contraseñas llamada **lista_usuarios** para poder ingresar al sistema.
4. La variable **acceso** se inicializa en 0

Interfaz de Bienvenida y validación de usuarios

5. Se crea una interfaz de Bienvenida, además de inputs para guardar en las variables user y contra el usuario y contraseña ingresados y se valida y compara con la **lista_usuarios** para que la variable **acceso** pueda cambiar a 1 si coincide. Si **acceso = 1** entonces se realizan las instrucciones de todas las opciones del programa

```
#Se crea una interfaz de Bienvenida ademas de inputs para guardar en las variables user y
contra el usuario y contraseña ingresados
print("\n::::::::::Bienvenido:::::::::: \n ")
user=input("Ingrese Usuario: ")
contra=input("Ingrese Contraseña: ")

#El ciclo for entra a la lista_usuarios para revisar cada elemento y la sentencia if valid
si el usuario y contraseña coinciden con los de la lista_usuarios
for usuario in lista_usuarios:
  if usuario[0] == user and usuario[1] == contra:
    #Si coincide, la variable acceso cambia a 1
    acceso=1

#Si acceso es igual a 1, entra a todas las opciones del programa
if acceso == 1:
  #El ciclo while funciona para crear un ciclo que permitirá regresar al menú principal
  while True:
```

Menú Principal

6. En primer lugar, el ciclo while funciona para crear un ciclo que permitirá regresar al menú principal
7. Se crea una interfaz para navegar por las opciones del programa, además de una opción de salir a la selección de usuarios. El input recibe un valor en la variable **seleccion** para poder capturar la opción del usuario que serán validadas mediante sentencias if.

```
print("\n::::::::::Menú::::::::::")
print("1. Top 10 productos con mayores ventas\n 2. Top 10 productos con mayores
búsquedas\n 3. Top 10 productos con menores ventas (Por categoría)\n 4. Top 10
productos con menores busquedas (Por categoría)\n 5. Top 20 Productos con mejores
reseñas\n 6. Top 20 Productos con peores reseñas\n 7. Total de Ingresos Mensuales y
Ventas promedio Mensuales\n 8. Total Anual y Meses con más ventas\n 9. Salir...")
seleccion=input("\nSeleccion: ")
```

Opción 1. Top 10 productos con mayores ventas

8. Se inicializa una variable **n** en 0 que servirá de acumulador en el conteo de ventas totales.
9. Se inicializa una lista llamada **lifestore_productoventa = [ ]** para posteriormente añadir los valores finales.
10. Se crea un ciclo for para itera la lista **lifestore_products** por medio de índices.
11. Nuevamente se crea un ciclo for para itera la lista **lifestore_sales** por medio de índices.

12. Se crea una sentencia if con la que se realiza una validación en donde la condición indica que si en cada elemento 1 de cada sublista de la lista **lifestore_sales** (id_producto) coincide con el elemento 0 de cada sublista de la lista **lifestore_products** (id_producto) ejecutara las siguientes instrucciones. Esto para relacionar ambas listas con cada producto y cada venta.

```
for i in range(len(lifestore_products)):
  #El siguiente ciclo for itera la lista lifestore_sales por medio de indices
  for j in range(len(lifestore_sales)):
    #Con la sentencia if se realiza una validacion en donde la condicion indica que
    si en cada elemento 1 de cada sublista de la lista lifestore_sales (id_producto)
    coincide con el elemento 0 de cada sublista de la lista lifestore_products
    (id_producto) ejecutara las siguientes instrucciones. Esto para relacionar ambas
    listas con cada producto y cada venta
    if lifestore_sales[j][1] == lifestore_products[i][0]:
      #El contador n de ventas totales
      n=n+1
```

13. Se añade a la lista **lifestore_productoventa** los datos que se van a mostrar (Id de cada producto, Nombre de cada producto y su contador de ventas totales) por medio de la función *append.*

```
lifestore_productoventa.append([lifestore_products[i][0],lifestore_products[i][1],n])
```

14. En una nueva lista llamada **lifestore_masvendidos**, se realiza una función *sorted*, con un valor de reverse=True para ordenar los datos de la lista **lifestore_productoventa** con filtro en las ventas para obtener los productos con mayores ventas.

```
lifestore_masvendidos=sorted(lifestore_productoventa, key = lambda x: int(x[2]),reverse=True)
```

15. Se crea un nuevo ciclo for que permite recorrer la lista lifestore_masvendidos previamente ordenada en un rango de 0 a 10 para mostrar solo los 10 productos con mayores ventas y por último se hace una espera de 3 segundos.

```
for x in lifestore_masvendidos[0:10]:
  #Con el print y los indices de la lista, se organiza de manera que el usuario pueda
  ver los datos con una mejor representacion
  print("Id Producto:",x[0],"| Nombre: ",x[1]," |No. Ventas: ",x[2])
#Con esta funcion se realiza una espera de 3 segundos.
time.sleep(3)
```

Opción 2. Top 10 productos con mayores búsquedas.

16. De igual forma se inicializa una variable n en 0 que servirá de acumulador en el conteo de búsquedas totales, se inicializa una lista llamada **lifestore_productobusqueda = [ ]** para posteriormente añadir los valores finales. Se crea un ciclo for para itera la lista lifestore_products por medio de índices. Nuevamente se crea un ciclo for para itera la lista lifestore_searches por medio de índices. Se crea una sentencia if con la que se realiza una validación en donde la condición indica que si en cada elemento 1 de cada sublista de la lista lifestore_searches (id_producto) coincide con el elemento 0 de cada sublista de la lista lifestore_products (id_producto) ejecutara las siguientes instrucciones. Esto para relacionar ambas listas con cada producto y cada venta.

17. Se añade a la lista los datos que se van a mostrar (Id de cada producto, Nombre de cada producto y su contador de busquedas totales).

```
lifestore_productobusqueda.append([lifestore_products[i][0],lifestore_products[i][1],n])
```

18. En una nueva lista llamada **lifestore_masvendidos**, se realiza una función *sorted*, con un valor de reverse=True para ordenar los datos de la lista **lifestore_productoventa** con filtro en las ventas para obtener los productos con mayores ventas.

```
lifestore_masvendidos=sorted(lifestore_productobusqueda, key = lambda x: int(x[2]),reverse=True)
```

19. Se crea un nuevo ciclo for que permite recorrer la lista lifestore_masvendidos previamente ordenada en un rango de 0 a 10 para mostrar solo los 10

productos con mayores ventas y por último se hace una espera de 3 segundos.

```
for x in lifestore_masbuscados[0:10]:
  #Con el print y los indices de la lista, se organiza de manera que el usuario pueda
  ver los datos con una mejor representacion
  print("Id Producto:",x[0],"| Nombre: ",x[1]," |No. de Búsquedas: ",x[2])
#Con esta funcion se realiza una espera de 3 segundos.
time.sleep(3)
```

Opción 3. Top 10 productos con menores ventas (Por categoría)

20. En primer lugar, el ciclo while funciona para crear un ciclo que permitirá regresar al menú principal
21. Se crea una interfaz para navegar por las categorías de los productos, además de una opción para regresar al menú principal. El input recibe un valor en la variable selección para poder capturar la opción del usuario que serán validadas mediante sentencias if.

```
print("\nCategorías:\n 1. Procesadores\n 2. Tarjetas de video\n 3. Tarjetas madre\n
4. Discos duros\n 5. Memorias USB\n 6. Pantallas\n 7. Bocinas\n 8. Audifonos\n 9.
Regresar")
seleccioncat=input("\nElegir Categoría (1-9): ")
#Se inicializa una variable n en 0 que servira de acumulador en el conteo de ventas
totales
n=0
```

22. Con la sentencia if se valida si la selección del usuario es igual a 1 al 8 y la variable "var" toma el valor según sea el caso, además de utilizar una opción adicional para regresar al menú principal.

```
if seleccioncat == "1":
  #Si la seleccion es 1 la variable "var" toma el valor de "procesadores"
  var="procesadores"
elif seleccioncat == "2":
  #Si la seleccion es 2 la variable "var" toma el valor de "tarjetas de video"
  var="tarjetas de video"
elif seleccioncat == "3":
  #Si la seleccion es 3 la variable "var" toma el valor de "tarjetas madre"
  var="tarjetas madre"
elif seleccioncat == "4":
  #Si la seleccion es 4 la variable "var" toma el valor de "discos duros"
  var="discos duros"
elif  seleccioncat == "5":
  #Si la seleccion es 5 la variable "var" toma el valor de "memorias usb"
  var="memorias usb"
elif  seleccioncat == "6":
  #Si la seleccion es 6 la variable "var" toma el valor de "pantallas"
  var="pantallas"
elif seleccioncat == "7":
  #Si la seleccion es 7 la variable "var" toma el valor de "bocinas"
  var="bocinas"
elif  seleccioncat == "8":
  #Si la seleccion es 8 la variable "var" toma el valor de "audifonos"
  var="audifonos"
elif seleccioncat == "9":
  #Si la seleccion es 9 se regresa al menu principal
  print("Regresando al menú anterior...")
  time.sleep(3)
```

23. Se crea un ciclo while funciona para crear un ciclo que permitirá regresar al menú de selección de categoría.
24. Se inicializa una lista llamada **lifestore_productoventacat = []** para posteriormente añadir los valores finales.
25. Se crea un ciclo for para itera la lista **lifestore_products** por medio de índices.
26. Se crea una sentencia if valida sí cada sublista en su posición 3 es idéntico a la variable **var**. Si la categoría es idéntica a la que el usuario quiere visualizar.
27. Nuevamente se crea un ciclo for para itera la lista lifestore_sales por medio de índices.

28. Con la sentencia if se realiza una validación en donde la condición indica que si en cada elemento 1 de cada sublista de la lista lifestore_sales(id_producto) coincide con el elemento 0 de cada sublista de la lista **lifestore_products** (id_producto) ejecutara las siguientes instrucciones. Esto para relacionar ambas listas con cada producto y cada venta
29. Se añade a la lista **lifestore_productoventacat** los datos que se van a mostrar (Id de cada producto, Nombre de cada producto y su contador de ventas totales) por medio de la función *append.*

```
lifestore_productoventacat.append([lifestore_products[i][0],lifestore_products[i][1],n])
```

30. En una nueva lista llamada lifestore_menosvendidoscat, se realiza una función sorted, con un valor de reverse=True para ordenar los datos de la lista lifestore_productoventa con filtro en las ventas para obtener los productos con mayores ventas.

```
lifestore_menosvendidoscat=sorted(lifestore_productoventacat, key = lambda x: int(x[2]),reverse=True)
```

31. Se crea un nuevo ciclo for que permite recorrer la lista lifestore_menosvendidoscat previamente ordenada en un rango de 0 a 10 para mostrar solo los 10 productos con mayores ventas y por último se hace una espera de 3 segundos.

Opción 4. Top 10 productos con menores búsquedas (Por categoría)

32. En primer lugar, el ciclo while funciona para crear un ciclo que permitirá regresar al menú principal
33. Se crea una interfaz para navegar por las categorías de los productos, además de una opción para regresar al menú principal. El input recibe un valor en la variable selección para poder capturar la opción del usuario que serán validadas mediante sentencias if.

```
print("\nCategorías:\n 1. Procesadores\n 2. Tarjetas de video\n 3. Tarjetas madre\n
4. Discos duros\n 5. Memorias USB\n 6. Pantallas\n 7. Bocinas\n 8. Audifonos\n 9.
Regresar")
seleccioncat=input("\nElegir Categoría (1-9): ")
#Se inicializa una variable n en 0 que servira de acumulador en el conteo de ventas
totales
n=0
```

34. Con la sentencia if se valida si la selección del usuario es igual a 1 al 8 y la variable "var" toma el valor según sea el caso, además de utilizar una opción adicional para regresar al menú principal.

```
if seleccioncat == "1":
  #Si la seleccion es 1 la variable "var" toma el valor de "procesadores"
  var="procesadores"
elif seleccioncat == "2":
  #Si la seleccion es 2 la variable "var" toma el valor de "tarjetas de video"
  var="tarjetas de video"
elif seleccioncat == "3":
  #Si la seleccion es 3 la variable "var" toma el valor de "tarjetas madre"
  var="tarjetas madre"
elif seleccioncat == "4":
  #Si la seleccion es 4 la variable "var" toma el valor de "discos duros"
  var="discos duros"
elif  seleccioncat == "5":
  #Si la seleccion es 5 la variable "var" toma el valor de "memorias usb"
  var="memorias usb"
elif  seleccioncat == "6":
  #Si la seleccion es 6 la variable "var" toma el valor de "pantallas"
  var="pantallas"
elif seleccioncat == "7":
  #Si la seleccion es 7 la variable "var" toma el valor de "bocinas"
  var="bocinas"
elif  seleccioncat == "8":
  #Si la seleccion es 8 la variable "var" toma el valor de "audifonos"
  var="audifonos"
elif seleccioncat == "9":
  #Si la seleccion es 9 se regresa al menu principal
  print("Regresando al menú anterior...")
  time.sleep(3)
```

35. Se crea un ciclo while funciona para crear un ciclo que permitirá regresar al menú de selección de categoría.

36. Se inicializa una lista llamada lifestore_productobuscat = [] para posteriormente añadir los valores finales.
37. Se crea un ciclo for para itera la lista lifestore_products por medio de índices.
38. Se crea una sentencia if valida sí cada sublista en su posición 3 es idéntico a la variable var. Si la categoría es idéntica a la que el usuario quiere visualizar.
39. Nuevamente se crea un ciclo for para itera la lista lifestore_searches por medio de índices.
40. Con la sentencia if se realiza una validación en donde la condición indica que si en cada elemento 1 de cada sublista de la lista lifestore_searches (id_producto) coincide con el elemento 0 de cada sublista de la lista lifestore_products (id_producto) ejecutara las siguientes instrucciones. Esto para relacionar ambas listas con cada producto y cada venta
41. Se añade a la lista lifestore_productobuscat los datos que se van a mostrar (Id de cada producto, Nombre de cada producto y su contador de ventas totales) por medio de la función append.
42. En una nueva lista llamada lifestore_menosbuscat, se realiza una función sorted, con un valor de reverse=True para ordenar los datos de la lista lifestore_productoventa con filtro en las ventas para obtener los productos con mayores ventas.
43. Se crea un nuevo ciclo for que permite recorrer la lista lifestore_menosbuscat previamente ordenada en un rango de 0 a 10 para mostrar solo los 10 productos con mayores ventas y por último se hace una espera de 3 segundos.

Opción 5. Top 20 Productos con mejores reseñas

44. Se inicializa una variable n en 0 que servirá de acumulador en el conteo de ventas totales.
45. Se inicializa una variable suma en 0 que servirá de acumulador en el conteo de las calificaciones de cada producto que servirá de acumulador en el conteo de ventas totales.
46. Se inicializa una lista llamada **lifestore_ productores = [ ]** para posteriormente añadir los valores finales.
47. Se crea un ciclo for para itera la lista lifestore_products por medio de índices, Nuevamente se crea un ciclo for para itera la lista lifestore_sales por medio de índices. Se crea una sentencia if con la que se realiza una validación en donde la condición indica que si en cada elemento 1 de cada sublista de la lista lifestore_sales (id_producto) coincide con el elemento 0 de cada sublista de la lista lifestore_products (id_producto) ejecutara las siguientes instrucciones. Esto para relacionar ambas listas con cada producto y cada venta.
48. Se suma 1 al acumulador n y se comienza a hacer la sumatoria de las calificaciones de cada producto.

```
n=n+1
suma=suma + lifestore_sales[j][2]
```

49. Para sacar el promedio, la sumatoria se divide entre el número de ventas del producto y se realiza una validación con un if para asegurarse que no se hagan divisiones entre 0. Se añade a la lista **lifestore_productores** los datos que se van a mostrar (Id de cada producto, Nombre de cada producto y su contador de ventas totales y promedio) por medio de la función *append.*

```
#Se realiza una validacion para que no se hagan divisiones entre 0
if n > 0:
  #para sacar el promedio, la sumatoria se divide entre el numero de ventas del
  producto
  promedio = suma/n
else:
  promedio = 0

if promedio > 0:
  #Se añade a la lista los datos que se van a mostrar (Id de cada producto, Nombre
  de cada producto, ventas totales y promedio de calificacion )
  lifestore_productores.append([lifestore_products[i][0],lifestore_products[i][1],n,
  round(promedio,2)])
```

50. En una nueva lista llamada **lifestore_masranked**, se realiza una funcion sorted, para ordenar los datos de la lista lifestore_masranked con filtro en el promedio de cada artículo, con valor reverse=True para visualizar los productos con mejores reseñas de mayor a menor
51. Se crea un nuevo ciclo for que permite recorrer la lista lifestore_masranked previamente ordenada en un rango de 0 a 20 para mostrar solo los 20 productos con mejores reseñas y por último se hace una espera de 3 segundos.

Opción 6. Top 20 Productos con peores reseñas

52. Se inicializa una variable n en 0 que servirá de acumulador en el conteo de ventas totales.
53. Se inicializa una variable suma en 0 que servirá de acumulador en el conteo de las calificaciones de cada producto que servirá de acumulador en el conteo de ventas totales.
54. Se inicializa una lista llamada **lifestore_ productores = [ ]** para posteriormente añadir los valores finales.
55. Se crea un ciclo for para itera la lista lifestore_products por medio de índices, Nuevamente se crea un ciclo for para itera la lista lifestore_sales por medio de índices. Se crea una sentencia if con la que se realiza una validación en donde la condición indica que si en cada elemento 1 de cada sublista de la lista lifestore_sales (id_producto) coincide con el elemento 0 de cada sublista

de la lista lifestore_products (id_producto) ejecutara las siguientes instrucciones. Esto para relacionar ambas listas con cada producto y cada venta.

56. Se suma 1 al acumulador n y se comienza a hacer la sumatoria de las calificaciones de cada producto.

```
n=n+1
suma=suma + lifestore_sales[j][2]
```

57. Para sacar el promedio, la sumatoria se divide entre el número de ventas del producto y se realiza una validación con un if para asegurarse que no se hagan divisiones entre 0. Se añade a la lista **lifestore_productores** los datos que se van a mostrar (Id de cada producto, Nombre de cada producto y su contador de ventas totales y promedio) por medio de la función *append.*

```
#Se realiza una validacion para que no se hagan divisiones entre 0
if n > 0:
  #para sacar el promedio, la sumatoria se divide entre el numero de ventas del
  producto
  promedio = suma/n
else:
  promedio = 0

if promedio > 0:
  #Se añade a la lista los datos que se van a mostrar (Id de cada producto, Nombre
  de cada producto, ventas totales y promedio de calificacion )
  lifestore_productores.append([lifestore_products[i][0],lifestore_products[i][1],n,
  round(promedio,2)])
```

58. En una nueva lista llamada **lifestore_menosranked**, se realiza una funcion sorted, para ordenar los datos de la lista lifestore_masranked con filtro en el promedio de cada artículo, con valor reverse=True para visualizar los productos con mejores reseñas de mayor a menor
59. Se crea un nuevo ciclo for que permite recorrer la lista lifestore_menosranked previamente ordenada en un rango de 0 a 20 para mostrar solo los 20 productos con peores reseñas y por último se hace una espera de 3 segundos.

Opción 7. Total de Ingresos Mensuales y Ventas promedio Mensuales

60. Se inicializa una variable nmes en 0 que servirá de acumulador en el conteo de ventas totales del mes.
61. Se inicializa una variable suma en 0 que servirá de acumulador en el conteo de ingresos.
62. Se inicializa una variable prom en 0 que servirá para guardar el promedio de ingresos de junio.
63. Se inicializa una variable "months" con los nombres de cada mes para después relacionarlos.

```
months = ("Enero","Febrero","Marzo","Abril","Mayo","Junio",
Julio","Agosto","Septiembre","Octubre","Noviembre","Diciembre")
```

64. Se valida si la selección del usuario es igual a 1 al 12 correspondiente a los meses del año y la variable "**seleccioncat**" toma el valor según sea el mes elegido. Esto para Homologar el input del usuario con la representación del mes en Letra

```
seleccioncat=input("\nElegir Mes (1-12): ")

if seleccioncat == "1":
  var="Enero"
elif seleccioncat == "2":
  var="Febrero"
elif seleccioncat == "3":
  var="Marzo"
elif seleccioncat == "4":
  var="Abril"
elif  seleccioncat == "5":
  var="Mayo"
elif  seleccioncat == "6":
  var="Junio"
elif seleccioncat == "7":
  var="Julio"
elif  seleccioncat == "8":
  var="Agosto"
elif seleccioncat == "9":
  var="Septiembre"
elif seleccioncat == "10":
  var="Octubre"
elif seleccioncat == "11":
  var="Noviembre"
elif seleccioncat == "12":
  var="Diciembre"
```

65. El primer ciclo for itera la lista **lifestore_sales** por medio de índices
66. Se formatea en una variable **fecha_dt** la fecha de cada elemento de cada sublista de lifestore_sale en su posición 3. De manera que pueda ser manejable.
67. En la variable mes se captura el mes de cada fecha para que se homologue con la representación del mes en letra

```
mes=months[fecha_dt.month - 1]
```

68. Se valida si la variable mes es igual a la elección del mes del usuario.
69. La variable nmes se incrementa + 1 para continuar con la sumatoria de los siguientes meses
70. El siguiente ciclo for itera la lista lifestore_products por medio de índices
71. Con la sentencia if se realiza una validación en donde la condición indica que si en cada elemento 1 de cada sublista de la lista lifestore_sales (id_producto) coincide con el elemento 0 de cada sublista de la lista lifestore_products (id_producto) ejecutara las siguientes instrucciones. Esto para relacionar ambas listas con cada producto y cada venta.
72. En la variable suma se comienza a hacer la sumatoria de los ingresos de todos los productos por mes
73. Con el print se muestra la variable var (mes en cuestión) y la suma de ingresos de ese mes en una función locale.currency para formatearlo en moneda.
74. Con el print se muestra la variable var además de las ventas totales de cada mes.
75. En la variable prom, se guarda el promedio de las ventas del mes en cuestión.
76. Y se muestra en un print con formateo de moneda.

```
    mes=months[fecha_dt.month - 1]
    #Se valida si la variable mes es igual a la eleccion del mes del usuario
    if mes == var :
     #la variable nmes se incrementa + 1 para continuar con la sumatoria de los
     siguientes meses
      nmes=nmes+1
      #El siguiente ciclo for itera la lista lifestore_products por medio de indices
      for y in range(len(lifestore_products)):
        #Con la sentencia if se realiza una validacion en donde la condicion indica que
        si en cada elemento 1 de cada sublista de la lista lifestore_sales (id_producto)
         coincide con el elemento 0 de cada sublista de la lista lifestore_products
        (id_producto) ejecutara las siguientes instrucciones. Esto para relacionar
        ambas listas con cada producto y cada venta
        if lifestore_sales[x][1] == lifestore_products[y][0]:
          #En la variable suma se comienza a hacer la sumatoria de los ingresos de
          todos los productos por mes
          suma=suma+lifestore_products[y][2]
#Con el print se muestra la variable var (mes en cuestion) y la suma de ingresos de
ese mes en una funcion locale.currency para formatearlo en moneda.
print("Total Ingresos de",var,":", locale.currency(suma,grouping=True))
#Con el print se muestra la variable var ademas de las ventas totales de cada mes
print("Ventas de",var,":",nmes - 1)
#En la variable prom, se guarda el promedio de las ventas del mes en cuestion
prom=suma/(nmes-1)
#Y se muestra en un print con formateo de moneda
print("Ventas promedio de",var,":", locale.currency(prom,grouping=True))
time.sleep(3)
```

Opción 8. Total Anual y Meses con más ventas.

77. Se inicializa una variable n en 0 que servirá de acumulador en el conteo de ventas totales.
78. Se inicializa una variable suma en 0 que servira de acumulador en el conteo de ingresos mensuales
79. Se inicializa una variable suma_anual en 0 que servira de acumulador en el conteo de ingresos anuales
80. Se inicializa una lista llamada **lifestore_mes = []** para posteriormente añadir los valores finales

81. Se inicializa una variable "months" con los nombres de cada mes para después relacionarlos.

```
months = ("Enero","Febrero","Marzo","Abril","Mayo","Junio",
Julio","Agosto","Septiembre","Octubre","Noviembre","Diciembre")
```

82. El primer ciclo for itera la lista months
83. El siguiente ciclo for itera la lista lifestore_sales por medio de índices
84. Se formatea en una variable fecha_dt la fecha de cada elemento de cada sublista de lifestore_sale en su posicion 3. De manera que pueda ser manejable
85. En la variable mes se captura el mes de cada fecha para que se homologue con la representación del mes en Letra
86. Se valida si la variable mes es igual a la iteración de cada mes
87. El siguiente ciclo for itera la lista lifestore_products por medio de índices
88. En la variable suma se comienza a hacer la sumatoria de los ingresos de todos los productos por mes

```
#El primer ciclo for itera la lista months
for var in months:
    #El siguiente ciclo for itera la lista lifestore_sales por medio de indices
    for x in range(len(lifestore_sales)):
        #Se formatea en una variable fecha_dt la fecha de cada elemento de cada
        sublista de lifestore_sale en su posicion 3. De manera que pueda ser manejable
        fecha_dt = datetime.strptime(lifestore_sales[x][3],"%d/%m/%Y")
        #en la variable mes se captura el mes de cada fecha para que se homologue con
        la representacion del mes en Letra
        mes=months[fecha_dt.month - 1]
        #Se valida si la variable mes es igual a la iteracion de cada mes
        if mes == var:
            #El siguiente ciclo for itera la lista lifestore_products por medio de indices

            for y in range(len(lifestore_products)):
                if lifestore_sales[x][1] == lifestore_products[y][0]:
                    #En la variable suma se comienza a hacer la sumatoria de los ingresos de
                    todos los productos por mes
                    suma=suma+lifestore_products[y][2]
                    n=n+1
```

89. Se añade a la lista los datos que se van a mostrar (mes, Ingresos del mes y ventas)
90. En la variable suma_anual se incrementa con la suma de cada mes

91. En una nueva lista, se realiza una funcion sorted, para ordenar los datos de la lista lifestore_mesrank con filtro en las ventas de cada mes para visualizar los meses con más ventas de mayor a menor.

```
lifestore_mesrank=sorted(lifestore_mes, key = lambda x: int(x[2]),reverse=True)
```

92. Se crea un contador p=0 para enumerar los meses con más ventas del mayor a menor
93. El siguiente ciclo for itera la lista lifestore_mesrank para visualizar todos los meses
94. Finalmente se hace la suma del ingreso de todos los meses, Además se vuelve a utilizar la funcion locale.currency para imprimir en formato moneda.

```
for m in lifestore_mesrank:
  p+=1
  print(p,"|  Mes:",m[0],"| Ingresos del mes:",locale.currency(m[1],grouping=True),"|
  Ventas: ",m[2],"|")
#Finalmente se hace la suma del ingreso de todos los meses, Ademas se vuelve a
utilizar la funcion locale.currency para imprimir en formato moneda
print("--------------------------")
print("Total anual:",locale.currency(suma_anual,grouping=True))
time.sleep(3)
```

Opción 9. Salir

95. Con un break se sale del ciclo while para regresal al ingreso de usuario y contraseña.

```
elif seleccion == "9":
  print("Saliendo...")
  time.sleep(3)
  #Con el break se sale del ciclo while para regresal al ingreso de usuario y contraseña
  break
#-----------------------------------------------------
```

Opción en caso de introducir cualquier otro dato no contemplado en el menú.

```
#-------------------------------------------------------
#Con la sentencia else realiza la validacion para contemplar el ingreso de digitos
distintos a las opciones mostradas en el menu
else:
  print("\nSelecciona una opcion valida\n")
  time.sleep(3)
```

Opción en caso de introducir un usuario o contraseña incorrectos

```
      else:
        print("\nUsuario o Contraseña Incorrecta. Reintente de nuevo\n")
        time.sleep(3)
```

Solución al problema

Gracias a la programación del programa, pude obtener datos y se realizó el análisis siguiente:

1. Productos más vendidos

Los 10 productos más vendidos que fueron arrojados por el programa fueron los siguientes:

- Id Producto: 54 | Nombre: SSD Kingston A400, 120GB, SATA III, 2.5", 7mm |No. Ventas: 50
- Id Producto: 3 | Nombre: Procesador AMD Ryzen 5 2600, S-AM4, 3.40GHz, Six-Core, 16MB L3 Cache, con Disipador Wraith Stealth |No. Ventas: 42
- Id Producto: 5 | Nombre: Procesador Intel Core i3-9100F, S-1151, 3.60GHz, Quad-Core, 6MB Cache (9na. Generación - Coffee Lake) |No. Ventas: 20
- Id Producto: 42 | Nombre: Tarjeta Madre ASRock Micro ATX B450M Steel Legend, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD |No. Ventas: 18
- Id Producto: 57 | Nombre: SSD Adata Ultimate SU800, 256GB, SATA III, 2.5", 7mm |No. Ventas: 15
- Id Producto: 29 | Nombre: Tarjeta Madre ASUS micro ATX TUF B450M-PLUS GAMING, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD |No. Ventas: 14
- Id Producto: 2 | Nombre: Procesador AMD Ryzen 5 3600, S-AM4, 3.60GHz, 32MB L3 Cache, con Disipador Wraith Stealth |No. Ventas: 13
- Id Producto: 4 | Nombre: Procesador AMD Ryzen 3 3200G con Gráficos Radeon Vega 8, S-AM4, 3.60GHz, Quad-Core, 4MB L3, con Disipador Wraith Spire |No. Ventas: 13
- Id Producto: 47 | Nombre: SSD XPG SX8200 Pro, 256GB, PCI Express, M.2 |No. Ventas: 11
- Id Producto: 12 | Nombre: Tarjeta de Video ASUS NVIDIA GeForce GTX 1660 SUPER EVO OC, 6GB 192-bit GDDR6, PCI Express x16 3.0 |No. Ventas: 9

2. Productos rezagados

Los productos con menores ventas por categoría son los siguientes:

Procesadores:

- Id Producto: 9 | Nombre: Procesador Intel Core i3-8100, S-1151, 3.60GHz, Quad-Core, 6MB Smart Cache (8va. Generación - Coffee Lake) |No. Ventas: 0 |Categoría: procesadores
- Id Producto: 1 | Nombre: Procesador AMD Ryzen 3 3300X S-AM4, 3.80GHz, Quad-Core, 16MB L2 Cache |No. Ventas: 2 |Categoría: procesadores

- Id Producto: 6 | Nombre: Procesador Intel Core i9-9900K, S-1151, 3.60GHz, 8-Core, 16MB Smart Cache (9na. Generación Coffee Lake) |No. Ventas: 3 |Categoría: procesadores

Tarjetas de video:

- Id Producto: 14 | Nombre: Tarjeta de Video EVGA NVIDIA GeForce GT 710, 2GB 64-bit GDDR3, PCI Express 2.0 |No. Ventas: 0 |Categoría: tarjetas de video
- Id Producto: 15 | Nombre: Tarjeta de Video EVGA NVIDIA GeForce GTX 1660 Ti SC Ultra Gaming, 6GB 192-bit GDDR6, PCI 3.0 |No. Ventas: 0 |Categoría: tarjetas de video
- Id Producto: 16 | Nombre: Tarjeta de Video EVGA NVIDIA GeForce RTX 2060 SC ULTRA Gaming, 6GB 192-bit GDDR6, PCI Express 3.0 |No. Ventas: 0 |Categoría: tarjetas de video

Tarjetas madre:

- Id Producto: 30 | Nombre: Tarjeta Madre AORUS ATX Z390 ELITE, S-1151, Intel Z390, HDMI, 64GB DDR4 para Intel |No. Ventas: 0 |Categoría: tarjetas madre
- Id Producto: 32 | Nombre: Tarjeta Madre ASRock Z390 Phantom Gaming 4, S-1151, Intel Z390, HDMI, 64GB DDR4 para Intel |No. Ventas: 0 |Categoría: tarjetas madre
- Id Producto: 34 | Nombre: Tarjeta Madre ASUS ATX ROG STRIX B550-F GAMING WI-FI, S-AM4, AMD B550, HDMI, max. 128GB DDR4 para AMD |No. Ventas: 0 |Categoría: tarjetas madre

Discos duros:

- Id Producto: 61 | Nombre: Kit Memoria RAM Corsair Vengeance LPX DDR4, 2400MHz, 32GB, Non-ECC, CL16 |No. Ventas: 0 |Categoría: memorias usb
- Id Producto: 60 | Nombre: Kit Memoria RAM Corsair Dominator Platinum DDR4, 3200MHz, 16GB (2x 8GB), Non-ECC, CL16, XMP |No. Ventas: 1 |Categoría: memorias usb

Pantallas:

- Id Producto: 62 | Nombre: Makena Smart TV LED 32S2 32", HD, Widescreen, Gris |No. Ventas: 0 |Categoría: pantallas
- Id Producto: 63 | Nombre: Seiki TV LED SC-39HS950N 38.5, HD, Widescreen, Negro |No. Ventas: 0 |Categoría: pantallas
- Id Producto: 64 | Nombre: Samsung TV LED LH43QMREBGCXGO 43, 4K Ultra HD, Widescreen, Negro |No. Ventas: 0 |Categoría: pantallas

Bocinas:

- Id Producto: 75 | Nombre: Lenovo Barra de Sonido, Alámbrico, 2.5W, USB, Negro |No. Ventas: 0 |Categoría: bocinas
- Id Producto: 76 | Nombre: Acteck Bocina con Subwoofer AXF-290, Bluetooth, Inalámbrico, 2.1, 18W RMS, 180W PMPO, USB, Negro |No. Ventas: 0 |Categoría: bocinas

- Id Producto: 77 | Nombre: Verbatim Bocina Portátil Mini, Bluetooth, Inalámbrico, 3W RMS, USB, Blanco |No. Ventas: 0 |Categoría: bocinas

Audifonos:

- Id Producto: 86 | Nombre: ASUS Audífonos Gamer ROG Theta 7.1, Alámbrico, USB C, Negro |No. Ventas: 0 |Categoría: audifonos
- Id Producto: 87 | Nombre: Acer Audífonos Gamer Galea 300, Alámbrico, 3.5mm, Negro |No. Ventas: 0 |Categoría: audifonos
- Id Producto: 88 | Nombre: Audífonos Gamer Balam Rush Orphix RGB 7.1, Alámbrico, USB, Negro |No. Ventas: 0 |Categoría: audífonos

Los anteriores artículos por categoría se han quedado rezagados y no tuvieron ni una sola venta en todo el año, por lo que se propone descontinuarlos.

3. Productos por reseña en el servicio

Los artículos con mejor reseña fueron los siguientes:

- Id Producto: 1 | Nombre: Procesador AMD Ryzen 3 3300X S-AM4, 3.80GHz, Quad-Core, 16MB L2 Cache |No. Ventas: 2 |Promedio: 5.0
- Id Producto: 6 | Nombre: Procesador Intel Core i9-9900K, S-1151, 3.60GHz, 8-Core, 16MB Smart Cache (9na. Generación Coffee Lake) |No. Ventas: 3 |Promedio: 5.0
- Id Producto: 7 | Nombre: Procesador Intel Core i7-9700K, S-1151, 3.60GHz, 8-Core, 12MB Smart Cache (9na. Generación Coffee Lake) |No. Ventas: 7 |Promedio: 5.0
- Id Producto: 8 | Nombre: Procesador Intel Core i5-9600K, S-1151, 3.70GHz, Six-Core, 9MB Smart Cache (9na. Generiación - Coffee Lake) |No. Ventas: 4 |Promedio: 5.0
- Id Producto: 11 | Nombre: Tarjeta de Video ASUS AMD Radeon RX 570, 4GB 256-bit GDDR5, PCI Express 3.0 |No. Ventas: 3 |Promedio: 5.0
- Id Producto: 21 | Nombre: Tarjeta de Video MSI AMD Mech Radeon RX 5500 XT MECH Gaming OC, 8GB 128-bit GDDR6, PCI Express 4.0 |No. Ventas: 2 |Promedio: 5.0
- Id Producto: 22 | Nombre: Tarjeta de Video MSI NVIDIA GeForce GTX 1050 Ti OC, 4GB 128-bit GDDR5, PCI Express x16 3.0 |No. Ventas: 1 |Promedio: 5.0
- Id Producto: 25 | Nombre: Tarjeta de Video Sapphire AMD Pulse Radeon RX 5500 XT Gaming, 8GB 128-bit GDDR6, PCI Express 4.0 |No. Ventas: 2 |Promedio: 5.0
- Id Producto: 28 | Nombre: Tarjeta de Video Zotac NVIDIA GeForce GTX 1660 Ti, 6GB 192-bit GDDR6, PCI Express x16 3.0 |No. Ventas: 1 |Promedio: 5.0
- Id Producto: 40 | Nombre: Tarjeta Madre Gigabyte XL-ATX TRX40 Designare, S-sTRX4, AMD TRX40, 256GB DDR4 para AMD |No. Ventas: 1 |Promedio: 5.0
- Id Producto: 49 | Nombre: Kit SSD Kingston KC600, 1TB, SATA III, 2.5, 7mm |No. Ventas: 3 |Promedio: 5.0
- Id Producto: 50 | Nombre: SSD Crucial MX500, 1TB, SATA III, M.2 |No. Ventas: 1 |Promedio: 5.0
- Id Producto: 52 | Nombre: SSD Western Digital WD Blue 3D NAND, 2TB, M.2 |No. Ventas: 2 |Promedio: 5.0
- Id Producto: 60 | Nombre: Kit Memoria RAM Corsair Dominator Platinum DDR4, 3200MHz, 16GB (2x 8GB), Non-ECC, CL16, XMP |No. Ventas: 1 |Promedio: 5.0
- Id Producto: 66 | Nombre: TCL Smart TV LED 55S425 54.6, 4K Ultra HD, Widescreen, Negro |No. Ventas: 1 |Promedio: 5.0
- Id Producto: 67 | Nombre: TV Monitor LED 24TL520S-PU 24, HD, Widescreen, HDMI, Negro |No. Ventas: 1 |Promedio: 5.0

- Id Producto: 84 | Nombre: Logitech Audífonos Gamer G332, Alámbrico, 2 Metros, 3.5mm, Negro/Rojo |No. Ventas: 1 |Promedio: 5.0
- Id Producto: 85 | Nombre: Logitech Audífonos Gamer G635 7.1, Alámbrico, 1.5 Metros, 3.5mm, Negro/Azul |No. Ventas: 2 |Promedio: 5.0
- Id Producto: 57 | Nombre: SSD Adata Ultimate SU800, 256GB, SATA III, 2.5", 7mm |No. Ventas: 15 |Promedio: 4.87
- Id Producto: 3 | Nombre: Procesador AMD Ryzen 5 2600, S-AM4, 3.40GHz, Six-Core, 16MB L3 Cache, con Disipador Wraith Stealth |No. Ventas: 42 |Promedio: 4.81

Los artículos con peor reseña fueron los siguientes:

- Id Producto: 17 | Nombre: Tarjeta de Video Gigabyte AMD Radeon R7 370 OC, 2GB 256-bit GDDR5, PCI Express 3.0 |No. Ventas: 1 |Promedio: 1.0
- Id Producto: 45 | Nombre: Tarjeta Madre ASRock ATX H110 Pro BTC+, S-1151, Intel H110, 32GB DDR4, para Intel |No. Ventas: 1 |Promedio: 1.0
- Id Producto: 31 | Nombre: Tarjeta Madre AORUS micro ATX B450 AORUS M (rev. 1.0), S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD |No. Ventas: 6 |Promedio: 1.83
- Id Producto: 46 | Nombre: Tarjeta Madre Gigabyte micro ATX GA-H110M-DS2, S-1151, Intel H110, 32GB DDR4 para Intel |No. Ventas: 1 |Promedio: 2.0
- Id Producto: 89 | Nombre: Cougar Audífonos Gamer Phontum Essential, Alámbrico, 1.9 Metros, 3.5mm, Negro. |No. Ventas: 1 |Promedio: 3.0

Estrategias

Los datos obtenidos por cada mes y el total anual fueron los siguientes:

```
1 |  Mes: Abril | Ingresos del mes: $193,295.00 | Ventas:  75 |
2 |  Mes: Enero | Ingresos del mes: $120,237.00 | Ventas:  53 |
3 |  Mes: Marzo | Ingresos del mes: $164,729.00 | Ventas:  51 |
4 |  Mes: Febrero | Ingresos del mes: $110,139.00 | Ventas:  41 |
5 |  Mes: Mayo | Ingresos del mes: $96,394.00 | Ventas:  36 |
6 |  Mes: Junio | Ingresos del mes: $36,949.00 | Ventas:  11 |
7 |  Mes: Julio | Ingresos del mes: $26,949.00 | Ventas:  11 |
8 |  Mes: Agosto | Ingresos del mes: $3,077.00 | Ventas:  3 |
9 |  Mes: Septiembre | Ingresos del mes: $4,199.00 | Ventas:  1 |
10 |  Mes: Noviembre | Ingresos del mes: $4,209.00 | Ventas:  1 |
11 |  Mes: Octubre | Ingresos del mes: $0.00 | Ventas:  0 |
12 |  Mes: Diciembre | Ingresos del mes: $0.00 | Ventas:  0 |
-------------------------
Total anual: $760,177.00
```

Como podemos observar a partir de el mes de agosto y hasta finalizar el año las ventas caen drásticamente inclusive en octubre y diciembre no se tuvo ninguna venta y por lo tanto ningún ingreso.

En primer lugar, en estos meses se pueden realizar campañas de publicidad y promociones exclusivas para obtener más ganancias de los productos que están

rezagados y posteriormente reinvertirlo en los productos que tienen más ventas como lo son SSD Kingston A400, 120GB, SATA III, 2.5", 7mm, Procesador AMD Ryzen 5 2600, Procesador Intel Core i3-9100F, etc.

Como observación adicional, las categorías que tienen más ventas son las que están relacionadas a componentes de computadora como lo son procesadores, tarjetas madre y de video.

Conclusión

Como conclusión se lograron los objetivos de las instrucciones dadas para el ejercicio práctico, se logró un mejor aprendizaje de los temas practicando cada uno de los temas y poniéndolos en ejecución, además de que se entendió mejor el comportamiento del lenguaje de programación Python, que a pesar de haber aprendido otros lenguajes de programación suele ser un poco distinto y la sintaxis que si bien es de alto nivel, hay que poner especial atención en la identación.

Al terminar todo el programa y al revisarlo, me pude percatar de que se pudo haber simplificado y optimizado mejor el código utilizando partes de código generales para las diversas opciones del programa además de que se pudieron ahorrar tiempo de programación, y tener en cuenta que esto puede afectar negativamente el tiempo de ejecución en manejo de información de mayores proporciones.

Anexos

Repositorio del programa cargado en GitHub:
https://github.com/daarleond/proyectofinal-python

Librerías y archivos adicionales utilizados:

- Lifestore_file.py (proporcionado por EMTECH): https://github.com/emtechinstitute/proyecto1
- Datetime : https://docs.python.org/3/library/datetime.html
- Time : https://docs.python.org/3/library/time.html?highlight=time#module-time
- Locale: https://docs.python.org/3/library/locale.html?highlight=locale#module-locale