PROYECTO 1

Fundamentos de Programación con Python

Elaborado por:

Juan Carlos Olivera Jiménez

Colima, Colima, México

Septiembre 06 2020

Contenido

Introducción

El presente trabajo tiene como objetivo poner en práctica las bases de programación en Python para el análisis y clasificación de datos mediante la creación de programas de entrada de usuario y validaciones, uso y definición de variables y listas, operadores lógicos y condicionales para la clasificación de información.

El contenido del trabajo es la definición de un código elaborado en el entorno de desarrollo Repl.it (liga de acceso al código en Repl.it https://repl.it/@JuanCarlosOJ/PROYECTO-01-OLIVERAJIMENEZ-JUANCARLOS#main.py) el cual contiene los elementos necesarios para la resolución de las instrucciones planteadas por la empresa LifeStore mismas que se muestran a continuación.

1. Productos más vendidos y productos rezagados.
   1.1 Generar un listado de los 50 productos con mayores ventas y uno con los 50 productos con mayores búsquedas.
   1.2 Por categoría, generar un listado con los 50 productos con menores ventas y uno con los 50 productos con menores búsquedas.
2. Productos por reseña en el servicio.
   2.1 Mostrar un listado de los 20 productos con las mejores reseñas y otro con las peores reseñas, considerando los productos con devolución.
3. Total de ingresos y ventas promedio mensuales, total anual y meses con más ventas al año.

El código elaborado cuenta con la definicion de variables y listas para el análisis de ventas, búsquedas, reseñas e ingresos manejando condicionales, operadores lógicos y bucles “while” y “for” para una correcta ejecución y conseguir un buen análisis, para posteriormente obtener conclusiones adecuadas y una solución a los diferentes problemas de manera eficaz y confiable.

Definición del código

El código comienza extrayendo información de las listas "lifestore_products", "lifestore_sales" y "lifestore_searches" localizadas en el documento de archivo "lifestore_file" con extensión para el Software de lenguaje de programación Python.

La lista lifestore_products contiene las siguientes variables para la elaboración de una lista con 96 productos incluyendo sus características generales de registro y venta al público.

La lista lifestore_sales contiene las variables para el registro de ventas de cada producto, la fecha de venta, una reseña de este y si el producto fue devuelto o no.

La lista lifestore_searches es un registro de las veces que fue buscado cada uno de los productos

```
Lifestore_products = [id_product, name, price, category, stock]

Lifestore_sales = [id_sale, id_product, score (from 1 to 5), date, refund (1 for true or 0 for false)]

Lifestore_searches = [id_search, id_product]
```

Dónde:

`Lifestore_products` corresponde al nombre de la lista que contiene las listas de los productos y sus características.

`Lifestore_sales` es el nombre de la lista que contiene los registros de venta de cada producto.

`Lifestore_searches` significa la lista que contiene las búsquedas de cada producto.

`= y []` hace referencia al contenido de la lista y al conjunto de valores contenidos en cada apartado de la lista respectivamente.

`id_product` es el número de identificación de cada uno de los productos, va de 1 hasta 96 para cada producto diferente.

`id_sale` se refiere al número de compra realizada tiene un total de 283 ventas.

`id_search` es el número de búsqueda realizada, tiene el registro de 1033 búsquedas.

`name` corresponde al nombre de cada producto y algunas de sus especificaciones.

`price` es el precio unitario del producto.

`category` es la categoría a la que pertenece cada producto, se encuentran 8 en total (procesadores, tarjetas de vídeo, tarjetas madre, discos duros, memorias USB, pantallas, bocinas y audífonos).

`stock` son las unidades disponibles en almacén para la venta de cada producto.

`score` corresponde al puntaje que obtuvo el producto al comprarse, el valor 1 corresponde a un valor muy bajo que podemos traducir en que el producto no fue de agrado para el consumidor y el valor 5 corresponde a un valor muy alto que hace referencia a que el producto fue de mucho agrado para el consumidor.

`date` es la fecha en la que fue comprado cada producto.

`refund` significa si el producto fue devuelto o no, si fue así, se muestra el valor 1, de lo contrario, el valor de devolución es 0.

La forma en la que se extraen las listas del documento es por medio de las funciones `from` e `import` por medio de las siguientes estructuras de código.

```
from lifestore_file import lifestore_products, lifestore_sales, lifestore_searches
```
(forma 1)

```
from lifestore_file import lifestore_products
```
(forma 2)

```
from lifestore_file import lifestore_sales
```
(forma 2)

```
from lifestore_file import lifestore_searches
```
(forma 2)

La función `from` funciona como una referencia a un archivo o documento, pues su traducción puede definirse como “a partir de” y la función `import` realiza la función de extracción de la lista a partir del documento referenciado por `from`

Una vez obtenidas las listas de productos, ventas y búsquedas, puede realizarse el análisis para la resolución de las instrucciones y obtener las listas correspondientes.

Análisis de ventas, las mayores y menores ventas.

```
                    #Código para análisis de ventas
counter = 0
total_de_ventas = []
for producto in lifestore_products:
  for venta in lifestore_sales:
    if producto [0] == venta [1]:
      counter +=1

  plantilla = [producto [1], counter]
  total_de_ventas.append(plantilla)
  counter = 0
  total_de_mas_ventas = total_de_ventas.copy()

                     #Los productos más vendidos
total_de_ventas_más = []
while total_de_mas_ventas:
  máximo = total_de_mas_ventas [0][1]
  lista_más_vendidas = total_de_mas_ventas [0]
  for Venta_mas in total_de_mas_ventas:
    if Venta_mas[1] > máximo:
      máximo = Venta_mas [1]
      lista_más_vendidas = Venta_mas
  total_de_ventas_más.append(lista_más_vendidas)
  total_de_mas_ventas.remove(lista_más_vendidas)

                    #Los productos menos vendidos
total_de_ventas_menos = []
while total_de_ventas:
  mínimo = total_de_ventas [0][1]
  lista_menos_vendidas = total_de_ventas[0]
  for Venta_menos in total_de_ventas:
    if Venta_menos[1] < mínimo:
      mínimo = Venta_menos[1]
      lista_menos_vendidas = Venta_menos
  total_de_ventas_menos.append(lista_menos_vendidas)
  total_de_ventas.remove(lista_menos_vendidas)
```

El código comienza con un contador (`counter`) equivalente a 0 para indicar el comienzo de una cuenta, posteriormente una lista vacía (`total_de_ventas`) que podrá capturar más adelante el conjunto de valores [nombre del producto, ventas] al realizar el análisis. Luego, se inicia un bucle `for` que servirá para extraer los conjuntos de lista unitarios incluidos en la

lista, a cada lista unitaria se le dio el nombre de `producto` por enfoque al análisis del producto y a la lista que contiene todas las listas unitarias se le otorgó el nombre de `lifestore_products`, un segundo bucle `for` servirá para extraer los conjuntos de lista unitarios de la lista `lifestore_sales` al cual se le da el nombre de `venta`, un condicional `if` compara que el índice [0] de producto, es decir, el id_product sea igual al índice [1] de venta, es decir, el id_product del mismo producto, si esta condición se cumple, el contador aumenta 1 valor, que quiere decir que se ha vendido el producto x una vez. Para facilitar la captura de datos en la lista `total_de_ventas` se elaboró una plantilla que contiene la estructura de `lista producto[1]` que equivale al nombre del producto y `counter` que corresponde a las veces en las que ese producto se ha vendido, con la función "`append`" se añade los valores del par [producto[1], counter] en la lista `total_de_ventas`, para definir que el contador a terminado de contar las ventas de un solo producto, debe reiniciarse, colocando `counter = 0` otra vez para comenzar a contar las veces que se vendió el siguiente producto. Por último, para ordenar los datos de ventas, se guarda una copia de la lista `total_de_ventas` por medio de la funcion `copy()`, creando la lista `total_de_mas_ventas` que será usada para obtener los productos con las mayores ventas.

Para los productos más vendidos se abre una nueva lista que lleva por nombre `total_de_ventas_mas` que concentrará los datos de las ventas de los productos de mayor a menor orden de venta, un bucle `while` para la lista `total_de_mas_ventas` ayudará a iniciar un ciclo de conteos y comparaciones que ayudarán a acomodar la lista en el orden deseado, para el producto con mayor cantidad de ventas y posteriormente los que siguen en orden descendente, se le asigna la variable `máximo` que equivale a la lista `total_de_mas_ventas` con los índices [0] (nombre del producto) y [1] (contador `counter`) luego una nueva variable denominada `lista_más_vendidas` equivale al nombre del producto de la lista `total_de_mas_ventas` en el índice [0]. Un bucle `for` que extrae las ventas unitarias de producto en la lista neta `total_de_mas_ventas` para después ejecutar la condicional if (si) `Venta_mas [1]` (esta indica el valor de counter de cada producto) es mayor que el valor de `máximo`, de ser cierto, `máximo` adquiere el valor del contador del producto y la `lista_mas_vendidas` se vuelve equivalente a la lista unitaria `Venta_mas`, como paso final se crea la lista `total_de_ventas_más` con la función

`append` que agrega el par de lista [nombre del producto, ventas del producto] y se remueve la lista `total_de_mas_ventas` para evitar interferencias o problemas al momento de imprimir los datos con la función `print`.

La obtención de los productos menos vendidos se obtiene de manera inversa de la que se obtuvo el listado de las mayores ventas. Con los productos menos vendidos se abre una nueva lista que lleva por nombre `total_de_ventas_menos` que concentrará los datos de las ventas de los productos de menor a mayor orden de venta, un bucle while para la lista `total_ventas` ayudará a iniciar un ciclo de conteos y comparaciones que ayudarán a acomodar la lista en el orden deseado, para el producto con menor cantidad de ventas y posteriormente los que siguen en orden ascendente, se le asigna la variable `mínimo` que equivale a la `lista total_de_ventas` con los índices [0] (nombre del producto) y [1] (contador `counter`) luego una nueva variable denominada `lista_menos_vendidas` equivale al nombre del producto de la lista `total_de_ventas` en el índice [0]. Un bucle `for` que extrae las ventas unitarias de producto en la lista neta `total_de_ventas` para después ejecutar la condicional `if` (si) `Venta_menos` [1] (esta indica el valor de `counter` de cada producto) es menor que el valor de `mínimo`, de ser cierto, mínimo adquiere el valor del contador del producto y la `lista_menos_vendidas` se vuelve equivalente a la lista unitaria `Venta_menos`, como paso final se crea la lista `total_de_ventas_menos` con la función `append` que agrega el par de lista [nombre del producto, ventas del producto] en `lista_menos_vendidas` y se remueve la lista `total_de_ventas` para evitar interferencias o problemas al momento de imprimir los datos con la función `print`.

Análisis de búsquedas, las mayores y menores búsquedas.

```
#Código para análisis de búsquedas
counter_b = 0
total_de_búsquedas = []

for producto in lifestore_products:
  for búsqueda in lifestore_searches:
    if producto [0] == búsqueda [1]:
      counter_b += 1

  plantilla_búsquedas = [producto [1], counter_b]
  total_de_búsquedas.append(plantilla_búsquedas)
```

```
  counter_b = 0
  total_de_busquedas_menos = total_de_búsquedas.copy()

                    #Los productos más buscados
Los_mas_buscados = []
while total_de_búsquedas:
  maximo = total_de_búsquedas [0][1]
  lista_mas_buscadas = total_de_búsquedas [0]
  for Busqueda_mas in total_de_búsquedas:
    if Busqueda_mas [1] > maximo:
      maximo = Busqueda_mas [1]
      lista_mas_buscadas = Busqueda_mas
  Los_mas_buscados.append(lista_mas_buscadas)
  total_de_búsquedas.remove(lista_mas_buscadas)

                   #Los productos menos buscados
Los_menos_buscados = []
while total_de_busquedas_menos:
  minimo = total_de_busquedas_menos [0][1]
  lista_menos_buscadas = total_de_busquedas_menos [0]
  for Busqueda_menos in total_de_busquedas_menos:
    if Busqueda_menos [1] < minimo:
     minimo = Busqueda_menos [1]
     lista_menos_buscadas = Busqueda_menos
  Los_menos_buscados.append(lista_menos_buscadas)
  total_de_busquedas_menos.remove(lista_menos_buscadas)
```

El contador `counter_b` funcionará para realizar los conteos del numero de búsquedas de cada producto, como es el inicio de las búsquedas, este contador debe adquirir el valor de 0, para capturar los datos de producto y sus búsquedas, se abre una nueva lista llamada `total_de_búsquedas`, un primer bucle `for` extrae el valor de una lista denominado `producto` en las listas de `lifestore_products` y un segundo bucle `for` extrae el valor de una lista que contiene las búsquedas, por ello se le asigna el nombre `búsqueda` y estas se obtienen de las listas de `lifestore_searches` obtenidos los valores unitarios de las listas, se establece el condicional si producto [0] (id_product) es igual a búsqueda[1] (que debe contener el id_product en la lista), si esto se cumple, `counter_b` aumenta un valor por cada vez que se cumpla esta condición. La captura de los datos para su posterior captura en `total_de_búsquedas` se realiza mediante plantilla `plantilla_búsquedas` que

adquiere los valores de `lista [producto[1]` (nombre del producto y `counter_b` (total de búsquedas para un producto)], estos valores de la plantilla, se añaden a `total_de_búsquedas` con la función `append` y para finalizar, se reinicia el `counter_b` para el siguiente producto y se copia la lista `total_de_búsquedas` con la función `copy()` para crear la lista `total_de_busquedas_menos`.

Para los productos más buscados se abre una nueva lista que lleva por nombre `Los_mas_buscados` que concentrará los datos de las ventas de los productos de mayor a menor orden de búsqueda, un bucle `while` para la lista `total_de_búsquedas` ayudará a iniciar un ciclo de conteos y comparaciones que ayudarán a acomodar la lista en el orden deseado, para el producto con mayor cantidad de búsquedas y posteriormente los que siguen en orden descendente, se le asigna la variable `máximo` que equivale a la lista `total_de_búsquedas` con los índices `[0]` (nombre del producto) y `[1]` (`counter_b`) luego una nueva variable denominada `lista_mas_buscadas` equivale al nombre del producto de la lista `total_de_búsquedas` en el índice [0]. Un bucle `for` que extrae las búsquedas unitarias de producto en la lista neta `total_de_búsquedas` en la lista `Búsqueda_mas` para después ejecutar la condicional `if` (si) `Búsqueda_mas [1]` (esta indica el valor de `counter_b` de cada producto) es mayor que el valor de `máximo`, de ser cierto, `máximo` adquiere el valor del contador del producto y la `lista_mas_buscadas` se vuelve equivalente a la lista unitaria `Busqueda_mas`, como paso final se llena la lista `Los_mas_buscados` con la función `append` que agrega el par de lista `[nombre del producto, búsquedas del producto]` y se remueve la lista `total_de_búsquedas` para evitar interferencias o problemas al momento de imprimir los datos con la función `print`.

La obtención de los productos menos buscados se obtiene de manera inversa de la que se obtuvo el listado de las mayores búsquedas. Con los productos menos buscados se abre una nueva lista que lleva por nombre `Los_menos_buscados` que concentrará los datos de las búsquedas de los productos de menor a mayor orden de venta, un bucle `while` para la lista `total_de_búsquedas_menos` ayudará a iniciar un ciclo de conteos y comparaciones que ayudarán a acomodar la lista en el orden deseado, para el producto con menor cantidad de búsquedas y posteriormente los que siguen en orden ascendente, se le asigna la variable

mínimo que equivale a la lista `total_de_búsquedas_menos` con los índices [0] (nombre del producto) y [1] (`counter_b`) luego una nueva variable denominada `lista_menos_buscadas` equivale al nombre del producto de la lista `total_de_busquedas_menos` en el índice [0]. Un bucle `for` que extrae las ventas unitarias de producto en la lista neta `total_de_búsquedas_menos` en `Búsqueda_menos` para después ejecutar la condicional `if` (si) `Busqueda_menos [1]` (esta indica el valor de `counter_b` de cada producto) es menor que el valor de `mínimo`, de ser cierto, `mínimo` adquiere el valor de `counter_b` y la `lista_menos_buscadas` se vuelve equivalente a la lista unitaria `Busqueda_menos`, como paso final se crea la lista `Los_menos_buscados` con la función `append` que agrega el par de lista `[nombre del producto, ventas del producto]` en `lista_menos_buscadas` y se remueve la lista `total_de_busquedas` para evitar interferencias o problemas al momento de imprimir los datos con la función `print`.

Análisis de reseñas, las mejores y las peores reseñas

```
#Código para análisis de reseñas
contador_puntos = 0
sumador_puntos = 0
puntaje_final = []

for producto in lifestore_products:
  for venta in lifestore_sales:
    if producto [0] == venta [1]:
      contador_puntos += 1
      sumador_puntos += venta [2]
  promedio_puntos = float(sumador_puntos/contador_puntos)
  lista_puntos = [producto [1], promedio_puntos]
  puntaje_final.append(lista_puntos)
  copia_puntaje_final = puntaje_final.copy()

              #Los productos con las mejores reseñas
total_reseñas = []
while puntaje_final:
  máximo = puntaje_final [0][1]
  mejores_reseñas_list = puntaje_final [0]
  for Reseña_mayor in puntaje_final:
    if Reseña_mayor [1] > máximo:
      máximo = Reseña_mayor [1]
      mejores_reseñas_list = Reseña_mayor
```

```
  total_reseñas.append(mejores_reseñas_list)
  puntaje_final.remove(mejores_reseñas_list)

              #Los productos con las peores reseñas
total_reseñas_peores = []
while copia_puntaje_final:
  mínimo = copia_puntaje_final [0][1]
  peores_reseñas_list = copia_puntaje_final [0]
  for Reseña_peor in copia_puntaje_final:
    if Reseña_peor[1] < mínimo:
      mínimo = Reseña_peor[1]
      peores_reseñas_list = Reseña_peor
  total_reseñas_peores.append(peores_reseñas_list)
  copia_puntaje_final.remove(peores_reseñas_list)
```

Al inicio del código se encuentra el contador `contador_puntos` que funcionará para realizar los conteos del número de reseñas que tiene cada producto, como es el inicio de las reseñas, este contador debe adquirir el valor de 0, después se encuentra el `sumador_puntos` que contará los valores de cada reseña de cada producto, para capturar los datos de producto y sus reseñas, se abre una nueva lista llamada `puntaje_final`, un primer bucle `for` extrae el valor de una lista denominado `producto` en las `listas de lifestore_products` y un segundo bucle `for` extrae el valor de una lista que contiene las reseñas por venta, para ello se le asigna el nombre `venta`, por referencia a las reseñas de cada vez que se vende un producto y estas se obtienen de las listas de `lifestore_sales` obtenidos los valores unitarios de las listas, se establece el condicional si `producto [0]` (id_product) es igual a `venta[1]` (que debe contener el id_product en la lista), si esto se cumple, `contador_puntos` aumenta un valor por cada vez que se cumpla esta condición, si esta misma condición se cumple `sumador_puntos` aumenta el valor de cada reseña según el producto vendido. Cuando se concentran los valores de la cantidad de reseñas (cuantas existen) y el valor neto de estas mismas (total), se establece un promedio que se captura en la variable `promedio_puntos` dividiendo el valor obtenido en `sumador_puntos` entre el valor de `contador_puntos` y convirtiendo este como un número flotante, de ahí el motivo de la función `float` dentro del código en la igualdad de la variable `promedio_puntos`. La captura de los datos para su posterior captura en `puntaje_final` se realiza mediante plantilla `lista_puntos` que adquiere los valores de lista [`producto`

[1] (nombre del producto y promedio_puntos (reseña promedio del producto)], estos valores de la plantilla, se añaden a puntaje_final con la función append y para finalizar, se copia la lista puntaje_final con la función copy() para crear la lista copia_puntaje_final.

Para los productos con las mejores reseñas se abre una nueva lista que lleva por nombre total_reseñas que concentrará los datos de las reseñas de los productos de mayor a menor orden, un bucle while para la lista puntaje_final ayudará a iniciar un ciclo de conteos y comparaciones que ayudarán a acomodar la lista en el orden deseado, para el producto con mayor promedio de reseñas y posteriormente los que siguen en orden descendente, se le asigna la variable máximo que equivale a la lista puntaje_final con los índices [0] (nombre del producto) y [1] (promedio_puntos) luego una nueva variable denominada mejores_reseñas_list equivale al nombre del producto de la lista puntaje_final en el índice [0]. Un bucle for que extrae las reseñas unitarias de producto en la lista neta puntaje_final en la lista Reseña_mayor para después ejecutar la condicional if (si) Reseña_mayor [1] (esta indica el valor de la reseña promedio de cada producto) es mayor que el valor de máximo, de ser cierto, máximo adquiere el valor de la reseña del producto y la lista mejores_reseñas_list se vuelve equivalente a la lista unitaria Reseña_mayor, como paso final se llena la lista total_reseñas con la función append que agrega el par de lista [nombre del producto, reseña promedio del producto] y se remueve la lista puntaje_final para evitar interferencias o problemas al momento de imprimir los datos con la función print.

Para los productos con las peores reseñas se abre una nueva lista que lleva por nombre total_reseñas_peores que concentrará los datos de las reseñas de los productos de menor a mayor orden, un bucle while para la lista copia_puntaje_final ayudará a iniciar un ciclo de conteos y comparaciones que ayudarán a acomodar la lista en el orden deseado, para el producto con menor promedio de reseñas y posteriormente los que siguen en orden ascendente, se le asigna la variable mínimo que equivale a la lista copia_puntaje_final con los índices [0] (nombre del producto) y [1] (promedio_puntos) luego una nueva variable denominada peores_reseñas_list equivale al nombre del producto de la lista Reseña_peor en el índice [0]. Un bucle for

que extrae las reseñas unitarias de producto en la lista neta `copia_puntaje_final` en la lista `Reseña_peor` para después ejecutar la condicional `if` (si) `Reseña_peor [1]` (esta indica el valor de la reseña promedio de cada producto) es mayor que el valor de `mínimo`, de ser cierto, `mínimo` adquiere el valor de la reseña del producto y la lista `peores_reseñas_list` se vuelve equivalente a la lista unitaria `Reseña_peor`, como paso final se llena la lista `total_reseñas_peores` con la función `append` que agrega el par de lista `[nombre del producto, reseña promedio del producto]` y se remueve la lista `copia_puntaje_final` para evitar interferencias o problemas al momento de imprimir los datos con la función `print`.

Análisis de ingresos, estadísticas por año y mes

```
#Código para estadísticas de ingresos
counter = 0
total_de_ventas = []

for producto in lifestore_products:
  for venta in lifestore_sales:
    if producto[0] == venta[1]:
      counter +=1
  plantilla = [producto[0], counter]
  total_de_ventas.append(plantilla)
  counter = 0
  total_de_mas_ventas = total_de_ventas.copy()

cuenta_total = 0
lista_ingresos_por_producto = []
for venta in total_de_ventas:
  for producto in lifestore_products:
    if venta[0] == producto[0]:
      producto_de_ventas = int(venta[1]*producto[2])
      cuenta_total += producto_de_ventas

                        # Ingresos por años
                              #2020
contador_productos_vendidos = 0
ganancias_2020 = 0
Ingresos_2020 = []

for producto in lifestore_products:
  for fecha in lifestore_sales:
    if producto[0] == fecha[1] and int(fecha[3][6:10]) == 2020:
```

```
      contador_productos_vendidos += 1
  venta_por_producto_año =
int(contador_productos_vendidos*producto[2])
  ganancias_2020 += venta_por_producto_año
  contador_productos_vendidos = 0
                              #2019
contador_productos_vendidos = 0
ganancias_2019 = 0
Ingresos_2019 = []

for producto in lifestore_products:
  for fecha in lifestore_sales:
    if producto[0] == fecha[1] and int(fecha[3][6:10]) == 2019:
      contador_productos_vendidos += 1
  venta_por_producto_año =
int(contador_productos_vendidos*producto[2])
  ganancias_2019 += venta_por_producto_año
  contador_productos_vendidos = 0

                              #2002
contador_productos_vendidos = 0
ganancias_2002 = 0
Ingresos_2002 = []

for producto in lifestore_products:
  for fecha in lifestore_sales:
    if producto[0] == fecha[1] and int(fecha[3][6:10]) == 2002:
      contador_productos_vendidos += 1
  venta_por_producto_año =
int(contador_productos_vendidos*producto[2])
  ganancias_2002 += venta_por_producto_año
  contador_productos_vendidos = 0

                          #Ingresos enero
contador_productos_vendidos = 0
ganancias_enero = 0
Ingresos_enero = []

for producto in lifestore_products:
  for fecha in lifestore_sales:
    if producto[0] == fecha[1] and int(fecha[3][3:5]) == 1:
      contador_productos_vendidos += 1
  venta_por_producto_mes =
int(contador_productos_vendidos*producto[2])
  ganancias_enero += venta_por_producto_mes
```

```
  contador_productos_vendidos = 0

                          #Ingresos febrero
contador_productos_vendidos = 0
ganancias_febrero = 0
Ingresos_febrero = []

for producto in lifestore_products:
  for fecha in lifestore_sales:
    if producto[0] == fecha[1] and int(fecha[3][3:5]) == 2:
      contador_productos_vendidos += 1
  venta_por_producto_mes =
int(contador_productos_vendidos*producto[2])
  ganancias_febrero += venta_por_producto_mes
  contador_productos_vendidos = 0

                           #Ingresos marzo
contador_productos_vendidos = 0
ganancias_marzo = 0
Ingresos_marzo = []

for producto in lifestore_products:
  for fecha in lifestore_sales:
    if producto[0] == fecha[1] and int(fecha[3][3:5]) == 3:
      contador_productos_vendidos += 1
  venta_por_producto_mes =
int(contador_productos_vendidos*producto[2])
  ganancias_marzo += venta_por_producto_mes
  contador_productos_vendidos = 0

                           #Ingresos abril
contador_productos_vendidos = 0
ganancias_abril = 0
Ingresos_abril = []

for producto in lifestore_products:
  for fecha in lifestore_sales:
    if producto[0] == fecha[1] and int(fecha[3][3:5]) == 4:
      contador_productos_vendidos += 1
  venta_por_producto_mes =
int(contador_productos_vendidos*producto[2])
  ganancias_abril += venta_por_producto_mes
  contador_productos_vendidos = 0

                           #Ingresos mayo
```

```
contador_productos_vendidos = 0
ganancias_mayo = 0
Ingresos_mayo = []

for producto in lifestore_products:
  for fecha in lifestore_sales:
    if producto[0] == fecha[1] and int(fecha[3][3:5]) == 5:
      contador_productos_vendidos += 1
  venta_por_producto_mes =
int(contador_productos_vendidos*producto[2])
  ganancias_mayo += venta_por_producto_mes
  contador_productos_vendidos = 0

                          #Ingresos junio
contador_productos_vendidos = 0
ganancias_junio = 0
Ingresos_junio = []

for producto in lifestore_products:
  for fecha in lifestore_sales:
    if producto[0] == fecha[1] and int(fecha[3][3:5]) == 6:
      contador_productos_vendidos += 1
  venta_por_producto_mes =
int(contador_productos_vendidos*producto[2])
  ganancias_junio += venta_por_producto_mes
  contador_productos_vendidos = 0

                          #Ingresos julio
contador_productos_vendidos = 0
ganancias_julio = 0
Ingresos_julio = []

for producto in lifestore_products:
  for fecha in lifestore_sales:
    if producto[0] == fecha[1] and int(fecha[3][3:5]) == 7:
      contador_productos_vendidos += 1
  venta_por_producto_mes =
int(contador_productos_vendidos*producto[2])
  ganancias_julio += venta_por_producto_mes
  contador_productos_vendidos = 0

                         #Ingresos agosto
contador_productos_vendidos = 0
ganancias_agosto = 0
Ingresos_agosto = []
```

```
for producto in lifestore_products:
  for fecha in lifestore_sales:
    if producto[0] == fecha[1] and int(fecha[3][3:5]) == 8:
      contador_productos_vendidos += 1
  venta_por_producto_mes =
int(contador_productos_vendidos*producto[2])
  ganancias_agosto += venta_por_producto_mes
  contador_productos_vendidos = 0

                        #Ingresos septiembre
contador_productos_vendidos = 0
ganancias_septiembre = 0
Ingresos_septiembre = []

for producto in lifestore_products:
  for fecha in lifestore_sales:
    if producto[0] == fecha[1] and int(fecha[3][3:5]) == 9:
      contador_productos_vendidos += 1
  venta_por_producto_mes =
int(contador_productos_vendidos*producto[2])
  ganancias_septiembre += venta_por_producto_mes
  contador_productos_vendidos = 0

                         #Ingresos octubre
contador_productos_vendidos = 0
ganancias_octubre = 0
Ingresos_octubre = []

for producto in lifestore_products:
  for fecha in lifestore_sales:
    if producto[0] == fecha[1] and int(fecha[3][3:5]) == 10:
      contador_productos_vendidos += 1
  venta_por_producto_mes =
int(contador_productos_vendidos*producto[2])
  ganancias_octubre += venta_por_producto_mes
  contador_productos_vendidos = 0

                        #Ingresos noviembre
contador_productos_vendidos = 0
ganancias_noviembre = 0
Ingresos_noviembre = []

for producto in lifestore_products:
  for fecha in lifestore_sales:
```

```
    if producto[0] == fecha[1] and int(fecha[3][3:5]) == 11:
      contador_productos_vendidos += 1
  venta_por_producto_mes =
int(contador_productos_vendidos*producto[2])
  ganancias_noviembre += venta_por_producto_mes
  contador_productos_vendidos = 0

                        #Ingresos diciembre
contador_productos_vendidos = 0
ganancias_diciembre = 0
Ingresos_diciembre = []

for producto in lifestore_products:
  for fecha in lifestore_sales:
    if producto[0] == fecha[1] and int(fecha[3][3:5]) == 12:
      contador_productos_vendidos += 1
  venta_por_producto_mes =
int(contador_productos_vendidos*producto[2])
  ganancias_diciembre += venta_por_producto_mes
  contador_productos_vendidos = 0

                        #Ventas por cada mes
   #Obtenidos a partir de un análisis previo con el código para
                      estadísticas de ingresos
Ingresos_por_cada_mes = [['Enero', 120237], ['Febrero', 110139],
['Marzo', 164729], ['Abril', 193295], ['Mayo', 96394], ['Junio',
36949], ['Julio', 26949], ['Agosto', 3077], ['Septiembre', 4199],
['Octubre', 0], ['Noviembre', 4209], ['Diciembre', 0]]

Ingresos_ordenados = []
while Ingresos_por_cada_mes:
  máximo = Ingresos_por_cada_mes[0][1]
  lista_orden_mes = Ingresos_por_cada_mes[0]
  for mes in Ingresos_por_cada_mes:
    if mes [1] > máximo:
      máximo = mes [1]
      lista_orden_mes = mes
  Ingresos_ordenados.append(lista_orden_mes)
  Ingresos_por_cada_mes.remove(lista_orden_mes)
```

Para el cálculo de los ingresos en general sin importar el año o mes, se requirió del código para el análisis de ventas, después, se definió la variable `cuenta_total` que cuenta y suma el valor de los ingresos de cada producto vendido por su precio luego se definió un primer bucle `for` que extrae los datos de venta de cada `producto` en el `total_de_ventas`, un

segundo bucle `for` extrae los datos de cada producto en `lifestore_products`, extraídas los datos de venta y producto, se establece un condicional que dice si `venta [0]` (id del producto) es igual a `producto [0]` (id del producto), se genera la variable `producto_de_ventas` que es igual al producto entero (int) de `venta[1]` (cantidad de productos vendidos) por `producto[2]` (precio unitario del producto) y este producto se agrega en `cuenta_total`.

Para calcular los ingresos por año (se muestran según la `lista lifestore_sales` fechas de los años 2002, 2019 y 2020) y por mes (desde enero hasta diciembre sin importar el año al que corresponda) se requirió de las variables contadoras `contador_productos_vendidos` y `ganancias_(año/mes)`. Definidas las variables se establece un primer bucle `for "for producto in lifestore_products"` para extraer los datos del producto en la lista y un segundo bucle `for "for fecha in lifestore_sales"` para extraer los años y meses de las fechas de venta de los productos, una vez hecho esto, se establece el condicional `if` para comparar `producto[0]` (id del producto) con `fecha[1]` (id del producto) y declarar que los índices de la lista fecha `posición 3 ([3]), caracteres de 6 a 10 ([6:10])` sean igual a `2020`, para asegurar esto, los índices de la lista fecha, se convirtieron a numero entero con la función `int`, de ser ciertas estas condiciones `contador_productos_vendidos` aumenta 1 valor y se crea una nueva variable `(venta_por_producto_[año/mes])` que equivale al producto del `contador_productos_vendidos` y `producto[2]` (precio unitario del producto), por último, se reinicia `contator_productos_vendidos` para continuar con la cuenta de los siguientes productos.

Despues de haber obtenido los análisis de los años y los meses, los ingresos mensuales fueron recabados manualmente en una lista `(Ingresos_por_cada_mes)` para despues ordenar estos ingresos desde el mes que tuvo más ingresos al mes con menos ingresos.

Para conseguir este orden de ingresos mensuales se comenzó por escribir una lista vacía (`Ingresos_ordenados`) y después se definió un bucle while en `Ingresos_por_cada_mes` para comenzar con los conteos y comparaciones respectivas, una variable máximo se igualó a `Ingresos_por_cada_mes [0][1]` y se definió una

nueva variable (`lista_orden_mes`) equivalente a `Ingresos_por_cada_mes [0]`, un bucle for extrajo los valores de los ingresos por cada mes en la lista `Ingresos_por_cada_mes` en la variable mes y despues se comparó con un condicional `if` que `mes[1]` (ingreso del mes) es mayor que `máximo`, al cumplirse la condición, `máximo` adquiere el valor de `mes[1]` y se captura el valor de mes en `lista_orden_mes`. Por último las capturas en `lista_orden_mes` se añadieron a `Ingresos_ordenados` con la función `append` y se removió la lista `Ingresos_por_cada_mes` con la función `remove`.

Mensaje de bienvenida en inicio de sesión

```
Mensaje_de_bienvenida = 'Bienvenido, por favor registrese o inicie
sesión para ingresar'
print (Mensaje_de_bienvenida)

es_admin = 0
Pregunta_de_Usuario = input('¿Usted cuenta con una cuenta de
usuario? (S/N) ')

if Pregunta_de_Usuario == 'S':
  print ('Escriba su usuario aquí')

  Usuario = 'Juan Carlos OJ'
  Contraseña = 'Olivera123'
  usuario = input('Ingrese su nombre de usuario: ')
  contraseña = input('Ingrese una contraseña: ')

  if Usuario == usuario and Contraseña == contraseña:
    es_admin = 1


  intentos = 0
  while Usuario != usuario and Contraseña != contraseña and
intentos <= 3:
    print('Usuario y/o contraseña incorrectos, por favor intente
otra vez')
    usuario = input('Ingrese su nombre de usuario: ')
    contraseña = input('Ingrese su contraseña: ')
    intentos += 1
  if intentos <= 3:
    es_admin = 1
    print('Bienvenido '+ usuario)
  if intentos > 3:
```

```
    print('Demasiados intentos, por favor intente más tarde o
registre un nuevo usuario y contraseña')

  breakpoint
if Pregunta_de_Usuario == 'N':
     es_admin == 1
  print('Por favor, introduzca un nombre de usuario y contraseña')
  usuario = input('Escriba su nombre de usuario: ')
  contraseña = input('Escriba una contraseña: ')
  print('Bienvenido '+ usuario)
```

Al iniciar a correr el código, se comienza por mostrar una variable denominada Mesaje_de_bienvenida a manera de saludo para el usuario, esto ocurre gracias a la función print ya que muestra las variables o elementos en el cuadro de prueba de Repl.it.
Despues de mostrar el saludo se muestra una pregunta de usuario interactiva preguntando al usuario si tiene una cuenta previa o no en el portal, se ser así, puede contestar S si es que tiene una cuenta previa o N si es que no la tiene.

Al escribir S en el cuadro de prueba una opción interactiva pide al usuario escribir su nombre de usuario y contraseña (variables nombradas de la misma manera respectivamente), una vez puestas, se ejecuta una comparación por medio de la condicional if para comprobar si el usuario escrito (usuario) es igual al usuario predeterminado (variable Usuario) y lo mismo ocurre para la contraseña (variable contraseña es igual a variable Contraseña) de ser ciertas ambas condiciones, se muestra un mensaje de bienvenida enfocado al usuario (print(‘Bienvenido ‘ + usuario). Si una (o ambas) condiciones no se cumplen, se imprime un mensaje indicando que el usuario o la contraseña no corresponden a los predeterminados y se pide una serie de intentos más, el total de intentos máximo contando el primer ingreso es de 5, por lo que cuenta con 5 oportunidades para poder escribir su nombre de usuario y contraseña correctamente, si en ninguno de los intentos tiene éxito para entrar, se imprime un mensaje indicando que fueron demasiados intentos y aconseja al usuario intentar más tarde o registrarse desde 0.

Al escribir N en el cuadro de prueba se da la indicación de escribir un nombre de usuario y una contraseña, una vez escritos se muestra el mismo mensaje de bienvenida que el caso de que se reconozcan el usuario y la contraseña predeterminados.

Menú de opciones

```
Opción = 0
while Opción != "8" and es_admin == 1:
  Pregunta_a_usuario = '\n Por favor seleccione una opción. \n 1.
Ver el top 50 de los productos más vendidos. \n 2. Ver el top 50
de los productos más buscados. \n 3. Ver el top 50 de los
productos con menos ventas. \n 4. Ver el top 50 de los productos
menos buscados. \n 5. Ver la lista de los 20 productos con las
mejores reseñas. \n 6. Ver la lista de los 20 productos con las
peores reseñas. \n 7. Ver un análisis de los ingresos en general y
un análisis de los ingresos mensuales comenzando por el mes con
mayores ingresos.\n 8. Salir\n'
  print(Pregunta_a_usuario)
  Opción = input('Opción: ')
  if Opción == '1':
    print('\n Aquí los 50 productos más vendidos')
    for venta_mas in total_de_ventas_más[0:50]:
      print('\n', venta_mas)
  if Opción == '2':
    print('\n Aquí los 50 productos más buscados')
    for mas_buscado in Los_mas_buscados[0:50]:
      print('\n', mas_buscado)
  if Opción == '3':
    print('\n Aquí los 50 productos menos vendidos')
    for venta_menos in total_de_ventas_menos[0:50]:
      print('\n', venta_menos)
  if Opción == '4':
    print('\n Aquí los 50 productos menos buscados')
    for menos_buscado in Los_menos_buscados[0:50]:
      print('\n', menos_buscado)
  if Opción == '5':
    print('\n Aquí los 20 productos con las mejores reseñas')
    for mejor_reseña in total_reseñas[0:20]:
      print('\n', mejor_reseña)
  if Opción == '6':
    print('\n Aquí los 20 productos con las peores reseñas')
    for peor_reseña in total_reseñas_peores[0:20]:
      print('\n', peor_reseña)
  if Opción == '7':
    print('\n Aquí las estadísticas de ingresos')
    print('\n El total de ingresos es ', cuenta_total, 'unidades')
    print('\n El total de ingresos en 2020 fue de ',
ganancias_2020, 'unidades')
```

```
    print('\n El total de ingresos en 2019 fue de ',
ganancias_2019, 'unidades')
    print('\n El total de ingresos en 2002 fue de ',
ganancias_2002, 'unidades')
    for ingreso_mes in Ingresos_ordenados:
      print('\n', ingreso_mes)
print('¡Gracias por visitarnos! Esperamos verlo por aquí muy
pronto.')
```

Una vez se ha conseguido ingresar con éxito al portal (registrar un usuario y contraseña nuevos o coincidir el usuario y la contraseña escritos con los predeterminados), se abre un menú interactivo de 8 opciones mostrando un menú interactivo a partir de un bucle while que muestra las opciones siguientes y sus ejecuciones.

1. Ver el top 50 de los productos más vendidos.

Esto ocurre si se escribe 1 como respuesta interactiva, una vez escrito, se imprime el resultado del análisis de `#Los productos más vendidos` a partir de un bucle for que extrae los datos de la lista total_de_ventas_más desde el primer valor hasta el valor 50 [0:50] en una variable denominada venta_mas, posteriormente, imprime cada valor de venta_mas.

2. Ver el top 50 de los productos más buscados.

Esto ocurre si se escribe 2 como respuesta interactiva, una vez escrito, se imprime el resultado del análisis de `#Los productos más buscados` a partir de un bucle for que extrae los datos de la lista Los_mas_buscados desde el primer valor hasta el valor 50 [0:50] en una variable denominada mas_buscado, posteriormente, imprime cada valor de mas_buscado.

3. Ver el top 50 de los productos menos venidos.

Esto ocurre si se escribe 3 como respuesta interactiva, una vez escrito, se imprime el resultado del análisis de `#Los productos menos vendidos` a partir de un bucle for que extrae los datos de la lista total_de_ventas_menos desde el primer valor hasta el valor 50 [0:50] en una variable denominada venta_menos, posteriormente, imprime cada valor de venta_menos.

4. Ver el top 50 de los productos menos buscados.

Esto ocurre si se escribe 4 como respuesta interactiva, una vez escrito, se imprime el resultado del análisis de `#Los productos menos buscados` a partir de un bucle for que extrae

los datos de la lista Los_menos_buscados desde el primer valor hasta el valor 50 [0:50] en una variable denominada menos_buscado, posteriormente, imprime cada valor de menos_buscado.

5. Ver el top 20 de los productos con las mejores reseñas.

Esto ocurre si se escribe 5 como respuesta interactiva, una vez escrito, se imprime el resultado del análisis de `#Los productos con las mejores reseñas` a partir de un bucle for que extrae los datos de la lista total_reseñas desde el primer valor hasta el valor 20 [0:20] en una variable denominada mejor_reseña, posteriormente, imprime cada valor de menos_buscado.

6. Ver el top 20 de los productos con las peores reseñas.

Esto ocurre si se escribe 6 como respuesta interactiva, una vez escrito, se imprime el resultado del análisis de `#Los productos con las peores reseñas` a partir de un bucle for que extrae los datos de la lista total_reseñas_peores desde el primer valor hasta el valor 20 [0:20] en una variable denominada peor_reseña, posteriormente, imprime cada valor de peor_buscado.

7. Ver un análisis de los ingresos en general y de los ingresos mensuales desde el mes con más ingresos al mes con menos ingresos.

Ocurre cuando se escribe 7 como respuesta interactiva, una vez escrito, se imprimen los ingresos netos, por año y por mes en orden descendente (de mayor ingreso a menor ingreso) obtenidos estos a partir de los códigos de Análisis de ingresos, estadísticas por año y mes.

8. Salir del menú de opciones y del programa.

Al escribir 8 como respuesta interactiva, el usuario sale del menú de opciones y del programa recibiendo antes un mensaje de agradecimiento y despedida (‘¡Gracias por visitarnos! Esperamos verlo por aquí muy pronto)

Solución a los problemas

Como soluciones a los problemas de inicio de sesión se tienen las siguientes propuestas.

- Tener presente que el usuario y contraseña deben memorizarse o bien, guardarse en un papel para un posterior uso. (En el código redactado el usuario puede ingresar sin importar que los caracteres estén en minúsculas o mayúsculas)
- Revisar que las mayúsculas y las minúsculas estén activas o desactivadas, según sea el caso del carácter a escribir.
- Minimizar el uso de caracteres extraños como algún símbolo o conjunto de número aleatorio.
- Escribir un usuario y contraseña entre 6 y 12 caracteres, relacionando este con un objeto o elemento con el que el usuario esté familiarizado.
- Revisar el usuario y la contraseña antes de teclear enter.

Como soluciones a los problemas de ejecución de los códigos se tienen las siguientes opciones.

- Revisar el indentado de la estructura del código, así como también la gramática y sintaxis de este.
- Evitar utilizar variables con nombres confusos para el programador.
- Revisar y asegurar el correcto uso de caracteres, operadores, bucles, condicionales y demás elementos en el código.
- Consultar a un experto ante el surgimiento de dudas o problemas

Conclusiones

- ✓ El producto con más ventas dentro del top 50 fue SSD Kingston A400, 120 GB, SATA III, 2.5”, 7mm con un total de 50 ventas.
- ✓ El producto con menos ventas dentro del top 50 fue Procesador Intel Core i3-8100, S-1151, 3.60 GHz, Quad-Core, 6MB Smart Cache (8va. Generación – Coffee Lake) con un total de 0 ventas
- ✓ El producto más buscado dentro del top 50 fue SSD Kingston A400, 120 GB, SATA III, 2.5”, 7mm con un total de 263 búsquedas.
- ✓ El producto menos buscado dentro del top 50 fue Tarjeta de Video EVGA NVIDIA GeForce GT 710, 2GB 64-bit GDDR3, PCI Express 2.0 con ninguna búsqueda.
- ✓ El procesador AMD Ryzen 3 3300X S-AM4, 3.80 GHz, Quad-Core 16 MB L2 Cache fue el producto con las mejores reseñas dentro del top, teniendo una calificación perfecta (5).
- ✓ El procesador AMD Ryzen 5 3600 S-AM4 3.60 GHz 32 MB L3 Cache con Disipador Wraith Stealth fue el producto con la peor referencia de reseñas teniendo 4.3 como puntaje de ellas.
- ✓ El total de ingresos neto hasta el momento fue de 760177 unidades, el año 2020 hasta ahora fue el año con el mayor número de ingresos teniendo un total de 755709 unidades, en el año 2019 solo se obtuvo un ingreso de 4209 unidades y el año 2002 (el año con el menor número de ingresos) obtuvo un total de 259 unidades (demostración en el gráfico 1).
- ✓ Sin importar el año al que pertenezca, abril es el mes con el mayor registro de ingresos teniendo un total de 193295 unidades y los meses de octubre y diciembre son los meses con los menores ingresos con un total de 0 unidades (demostración en el gráfico 2)

Gráfica 1. Gráfico de ingresos anuales.

Gráfica 2. Gráfico de ingresos mensuales