Emerging Technologies Institute

Grupo 1

Fundamentos de Programación con Python

Reporte

Presenta:

Hugo Sevilla Gómez Llanos

Tijuana, B.C., 25 de septiembre de 2020

Índice

Introducción

Este primer proyecto busca aplicar todo lo visto en la primera parte del curso que son todos los fundamentos esenciales de cualquier lenguaje de programación en este caso específico en Python con los ciclos, condicionales y la manera de tener un flujo de programa. Se nos encargan ciertas tareas que nos harán investigar mucho más y pensar cómo se puede utilizar, más que un primer problema analítico a lo libre, es más guiado o lineal pues se nos dice qué instrucciones seguir y aunque sirven para analizar no viene de nuestras primeras soluciones aunque quizá esto sea diferente en el siguiente proyecto número 2.

Definición del código


```python
# Primero tenemos la importación de las librerías y las bases de datos

from lifestore_file import lifestore_products as products, lifestore_sales as sales, lifestore_searches as searches
from operator import itemgetter
from collections import Counter
import os
import numpy as np
from copy import deepcopy
from tabulate import tabulate


# Aquí tenemos a nuestros usuarios Admin y no Admin
usuarios = [['Hugo', '123', 'Admin'], ['Paco', '456', 'No Admin'], ['Luis','789', 'Admin'], ['Cesar', '321', 'No Admin']]

# opción default e inputs para login / iniciar sesión
opcion = ''
usuario = input('Ingresa nombre de usuario: ')
password = input('Ingresa contraseña de usuario: ')

# esto es para hacer clear en la consola
os.system('cls' if os.name == 'nt' else 'clear')

# generamos listas de usuarios únicamente administradores para comprar nombres de los que sí son usuarios y cuáles de esos son administradores
admins = list(filter(lambda x: str(x[2]) == 'Admin', usuarios))
nombres_todos = list(map(itemgetter(0), usuarios))
nombres_admins = list(map(itemgetter(0), admins))
```

Primero tenemos la importación de todas las librerías y bases de datos que vamos a utilizar, así como la generación de usuarios, el input para el login y otros listados tomando en cuenta los nombres de todos y únicamente los de los administradores para el inicio de sesión.

```python

# aquí tenemos el bucle para iniciar sesión
inicio_sesion = 'No'
if usuario in nombres_todos:
  if usuario in nombres_admins:
    usuario_encontrado = [usuario, password, 'Admin']
    if usuario_encontrado in usuarios:
      print('Inicio de sesión exitoso, BIENVENIDO.')
      inicio_sesion = 'Si'
    else:
      print('Contraseña incorrecta. Programa finalizado. Adiós.')
  else:
    print('Usuario indentificado, pero no es Administrador. Programa Finalizado. Adiós.')
else:
  print('No Existe Usuario. Programa Finalizado. Adiós.')
  opcion = 0

# Listado de Productos de más búsquedas, nos enfocamos en nombres de productos para hacer una lista y las búqeudas que hay, así como las categorías de los productos y por medio de un filtro para búsquedas únicas de los productos que fueron buscados de la misma manera el filtrado para las categorías de los productos que fueron buscados.
busquedas_producto = list(map(itemgetter(1), searches))
nombres_productos = list(map(itemgetter(1), products))

categorias_productos = list(map(itemgetter(3), products))
categorias_unicas = np.unique(categorias_productos)
categorias_unicas = list(map(lambda el:[el], categorias_unicas)) # separar en lista de listas

# Hacemos un diccionario para tener id de producto y la cantidad de búsquedas, generamos una copia para hacer modificaciones a ese diccionario cambiando id por nombre de producto y luego imprimir
busquedas = dict(Counter(busquedas_producto))
busquedas_copia = busquedas.copy()
busquedas_categoria = []

# Este es el ciclo mencionado, cambiamos las llaves de id por el nombre del producto
for busqueda in busquedas:
  busqueda_nombre = nombres_productos[int(busqueda)]
  llave_nueva = busqueda_nombre
  llave_vieja = busqueda
  busquedas_copia[llave_nueva] = busquedas_copia.pop(llave_vieja)
```

Después procedemos a los listados de los productos con más búsquedas y ventas (yo pensando del lado del cliente lo llamé compras aunque me refiero a las ventas de la empresa), también tomamos importancia en las categorías de los productos y por medio de un filtro generamos más listados búsquedas y ventas

de productos sin repeticiones (unique). Hacemos un diccionario para tener ID del producto y la cantidad de búsquedas. Generamos una copia para hacer modificaciones a ese diccionario cambiando ID por nombre de producto y luego imprimir más adelante en el ciclo final del código.

```
# hacemos una copia y vaciamos la otra lista únicamente para liberar un poco de espacio, es buena práctica.
busquedas = busquedas_copia.copy()
busquedas_copia.clear()

# Procedemos a ordenar las búsquedas de los más buscados a menos buscados
busquedas_ordenadas = sorted(busquedas.items(), key=lambda x: x[1], reverse=True)

# Listado de los Productos con más Ventas
# Todo lo anterior se repite con la diferencia de que ahora se aplica a las ventas
# Yo por equivocación puse compras pensando que los clientes compran los productos aunque del lado de la empresa en realidad son
las:
# VENTAS
compras_producto = list(map(itemgetter(1), sales))
compras = dict(Counter(compras_producto))

# igual hacemos la copia para cambiar id a nombre de producto
compras_copia = compras.copy()
for compra in compras:
  compra_nombre = nombres_productos[int(compra)]
  llave_nueva = compra_nombre
  llave_vieja = compra
  compras_copia[llave_nueva] = compras_copia.pop(llave_vieja)

# liberamos espacio con copias y ordenamos con sort, así cómo variables para ventas únicas
compras = compras_copia.copy()
compras_copia.clear()
compras_ordenadas = sorted(compras.items(), key=lambda x: x[1], reverse=True)
categorias_unicas_ventas = categorias_unicas.copy()
```

La siguiente parte es hacer copias y liberar las listas originales para liberar espacio de variables. Y aplicamos lambdas, maps, filtros para hacer todo mucho más eficaz que con ciclos for. Es importante hacer sort para ordenar los datos por cantidad de búsquedas o ventas de mayor a menor o viceversa dependiendo como lo queramos manejar.

```
# Lista en reversa, aquí hacemos un ciclo anidado para que dentro de una lista de listas, dependiendo de su categoría vaya imprimiendo y se recorre en inversa la lista porque nos piden los productos menos buscados y
menos vendidos.
for venta in compras_ordenadas[::-1]:
  for producto in products:
    if producto[1] == venta[0]:
      if producto[3] == 'audifonos':
        categorias_unicas_ventas[0].append(venta);
      elif producto[3] == 'bocinas':
        categorias_unicas_ventas[1].append(venta);
      elif producto[3] == 'discos duros':
        categorias_unicas_ventas[2].append(venta);
      elif producto[3] == 'memorias usb':
        categorias_unicas_ventas[3].append(venta);
      elif producto[3] == 'pantallas':
        categorias_unicas_ventas[4].append(venta)
      elif producto[3] == 'procesadores':
        categorias_unicas_ventas[5].append(venta);
      elif producto[3] == 'tarjetas de video':
        categorias_unicas_ventas[6].append(venta);
      elif producto[3] == 'tarjetas madre':
        categorias_unicas_ventas[7].append(venta);
      break;

# lo mismo de lo anterior que era para ventas, ahora para búsquedas, ciclo anidado, recorrer lista invertida para tener de menor a mayor en orden por categoría
categorias_unicas_busquedas = categorias_unicas.copy()
for busqueda in compras_ordenadas[::-1]:
  for producto in products:
    if producto[1] == busqueda[0]:
      if producto[3] == 'audifonos':
        categorias_unicas_busquedas[0].append(busqueda);
      elif producto[3] == 'bocinas':
        categorias_unicas_busquedas[1].append(busqueda);
      elif producto[3] == 'discos duros':
        categorias_unicas_busquedas[2].append(busqueda);
      elif producto[3] == 'memorias usb':
        categorias_unicas_busquedas[3].append(busqueda);
      elif producto[3] == 'pantallas':
        categorias_unicas_busquedas[4].append(busqueda)
      elif producto[3] == 'procesadores':
        categorias_unicas_busquedas[5].append(busqueda);
      elif producto[3] == 'tarjetas de video':
        categorias_unicas_busquedas[6].append(busqueda);
      elif producto[3] == 'tarjetas madre':
        categorias_unicas_busquedas[7].append(busqueda);
      break;
```

Procedemos a hacer iteraciones en listas de listas pues dependiendo de la categoría es donde posicionamos los valores y al final facilitar las impresiones con un mejor formato. Cabe destacar que aquí recorrimos la lista inversamente puesto que nos piden en la parte de categorías los productos con menos búsquedas y menos ventas.

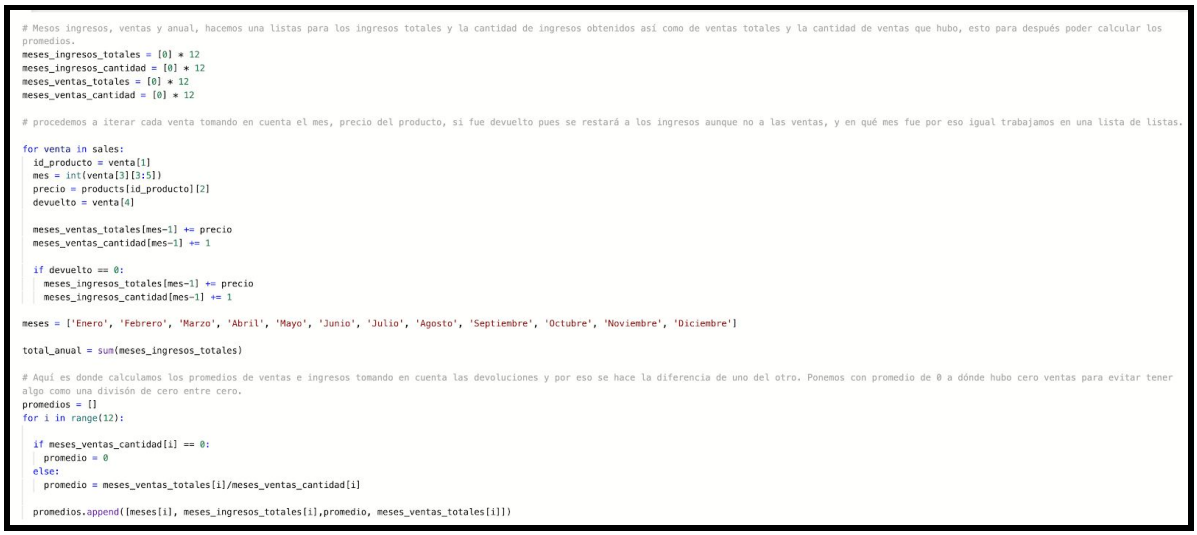
```
# Listas de 20 productos con mejores y peores reseñas

ventas_por_calificacion = deepcopy(sales) # para que la copia no afecte la original necesitamos un deepcopy

reviews_por_producto = []

# usamos contadores y ciclamos las ventas, guardamos los puntajes de las reseñas, si acaso fueron devueltos pues esto luego restará o hará alguna clase de diferencia y el producto
contador = 0
for review in ventas_por_calificacion:

  review_puntos = review[2]
  producto_review = nombres_productos[review[1]-1]

  #reviews_por_producto.append([[producto_review], [review_puntos]])
  reviews_por_producto.append([[producto_review], [review_puntos], [review[4]]])

  contador +=1

productos_reviewed = []

# ahora para productos reseñados los agregamos y procedemos a hacer una lista de los únicos productos reseñados eliminando las repeticiones, buen filtro y lo separamos en listas de listas para después agregar valores
con mayor facilidad y tanto para imprimir también con un mejor formato.
for producto_review in reviews_por_producto:
  productos_reviewed.append(producto_review[0])

productos_reviewed_unicos = list(np.unique(productos_reviewed))
productos_reviewed_unicos_puntuados = deepcopy(productos_reviewed_unicos)
productos_reviewed_unicos_puntuados = list(map(lambda el:[el], productos_reviewed_unicos_puntuados))

# agregamos 2 nuevas secciones/listas que serán puntajes para reseñas y devoluciones.
contador = 0
for producto in productos_reviewed_unicos_puntuados:
  productos_reviewed_unicos_puntuados[contador].append(0)
  productos_reviewed_unicos_puntuados[contador].append(0)
  contador += 1
```

Ahora vamos a la parte de las 20 listas con mejores y peores reseñas y para no afectar las listas originales necesitamos un deepcopy, hacemos igual listas para los únicos productos que fueron reseñados, es un buen filtrado que luego pasamos a listas de listas y agregamos 2 nuevas listas que es dónde se agregaran los valores para cada producto de la puntuación de reseñas y la cantidad de devoluciones.

```
# Aquí es dónde vamos agregando o sumando la calificación de reseña por reseña así como la cantidad de devoluciones.
contador = 0
for review in reviews_por_producto:
  reviewed_nombre = review[0][0]
  reviewed_calificacion = review[1][0]
  devuelto = review[2][0]

  reviewed_index = productos_reviewed_unicos.index(reviewed_nombre)

  # sumar calificacion
  productos_reviewed_unicos_puntuados[reviewed_index][1] += reviewed_calificacion
  # sumar devueltos
  productos_reviewed_unicos_puntuados[reviewed_index][2] += devuelto

  contador += 1

# invertir valores para itemgetter ya que en este caso entre más devueltos queremos ponerlo como peor reseñado mientras que itemgetter se basa en números más grandes, entonces los ponemos negativos y procedemos a
utilizar itemgitter para ordenar en un orden descendiente correcto tomando en cuenta tanto el puntaje de la reseña como el de la cantidad de devoluciones.
contador = 0
for producto in productos_reviewed_unicos_puntuados:
  productos_reviewed_unicos_puntuados[contador][2] = - (producto[2])
  contador += 1

# Ordenar lista de listas, aplicar por puntaje reseñas y suma devoluciones.
productos_reviewed_unicos_puntuados = sorted(productos_reviewed_unicos_puntuados, key=itemgetter(1, 2))

 # invertir valores después itemgetter para que todo vuelva a la normalidad no puede haber negativos en devoluciones.
contador = 0
for producto in productos_reviewed_unicos_puntuados:
  productos_reviewed_unicos_puntuados[contador][2] = - (producto[2])
  contador += 1

# Mesos ingresos, ventas y anual, hacemos una listas para los ingresos totales y la cantidad de ingresos obtenidos así como de ventas totales y la cantidad de ventas que hubo, esto para después poder calcular los
promedios.
meses_ingresos_totales = [0] * 12
meses_ingresos_cantidad = [0] * 12
meses_ventas_totales = [0] * 12
meses_ventas_cantidad = [0] * 12
```

Hacemos las sumas de reseñas, para los ciclos utilizamos contadores y sumamos el valor, aquí trabajamos únicamente con una lista. invertimos los valores para itemgetter ya que en este caso entre más devueltos queremos

ponerlo como peor reseñado mientras que itemgetter se basa en números más grandes, entonces los ponemos negativos y procedemos a utilizar itemgetter para ordenar en un orden descendiente correcto tomando en cuenta tanto el puntaje de la reseña como el de la cantidad de devoluciones y ya ordenada la lista hacemos esos valores positivos.



```
# Mesos ingresos, ventas y anual, hacemos una listas para los ingresos totales y la cantidad de ingresos obtenidos así como de ventas totales y la cantidad de ventas que hubo, esto para después poder calcular los
promedios.
meses_ingresos_totales = [0] * 12
meses_ingresos_cantidad = [0] * 12
meses_ventas_totales = [0] * 12
meses_ventas_cantidad = [0] * 12

# procedemos a iterar cada venta tomando en cuenta el mes, precio del producto, si fue devuelto pues se restará a los ingresos aunque no a las ventas, y en qué mes fue por eso igual trabajamos en una lista de listas.

for venta in sales:
  id_producto = venta[1]
  mes = int(venta[3][3:5])
  precio = products[id_producto][2]
  devuelto = venta[4]

  meses_ventas_totales[mes-1] += precio
  meses_ventas_cantidad[mes-1] += 1

  if devuelto == 0:
    meses_ingresos_totales[mes-1] += precio
    meses_ingresos_cantidad[mes-1] += 1

meses = ['Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo', 'Junio', 'Julio', 'Agosto', 'Septiembre', 'Octubre', 'Noviembre', 'Diciembre']

total_anual = sum(meses_ingresos_totales)

# Aquí es donde calculamos los promedios de ventas e ingresos tomando en cuenta las devoluciones y por eso se hace la diferencia de uno del otro. Ponemos con promedio de 0 a dónde hubo cero ventas para evitar tener
algo como una divisón de cero entre cero.
promedios = []
for i in range(12):

  if meses_ventas_cantidad[i] == 0:
    promedio = 0
  else:
    promedio = meses_ventas_totales[i]/meses_ventas_cantidad[i]

  promedios.append([meses[i], meses_ingresos_totales[i],promedio, meses_ventas_totales[i]])
```

Generemos listas tomando en cuenta 12 meses y una lista con los 1 meses, además del mes, nos importan las ventas e ingresos, en cantidad, totales (que tenemos la diferencia de cuándo hay devoluciones se tienen menores ingresos totales que ventas) y finalmente con esos últimos 2 se pueden calcular los promedios.

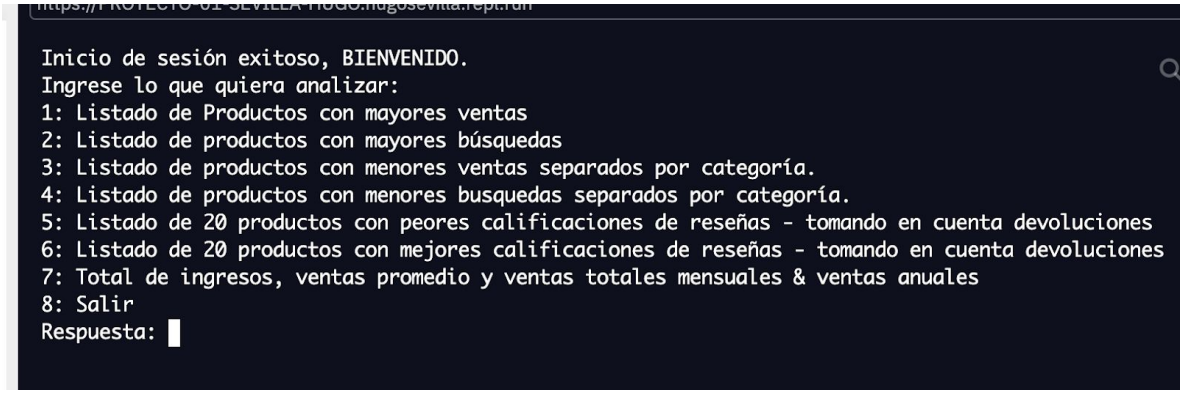
```
# Aquí tenemos el loop final con las opciones para imprimir todo lo desarrollado en el código, dividimos en 7 opciones y una extra para salir del programa y finalizarlo. Hacemos ciclos for para ir imprimiendo con un formato que señale el producto y la cantidad, o por categorías vaya haciendo las impresiones, o que enumere las listas de los 20 productos por ejemplo. Para la opción de las ventas e
ingresos tanto mensuales como anual utilizamos una tabulación que le da mejor formato a la tabla y un formato de dinero para imprimir el valor de las ganancias totales anuales.

while(opcion != 0):

  opcion = input('Ingrese la que quiere analizar:\n\t1: Listado de Productos con mayores ventas\n\t2: Listado de productos con mayores búsquedas\n\t3: Listado de productos con menores ventas separados por categoría.\n\t4: Listado de productos con menores busquedas separados por categoría.\n\t5: Listado de 20 productos con peores calificaciones de reseñas - tomando en cuenta devoluciones\n\t6: Listado de 20
  productos con mejores calificaciones de reseñas - tomando en cuenta devoluciones\n\t7: Total de ingresos, ventas promedio y ventas totales mensuales & ventas anuales\n\t0: Salir\nRespuesta: ')
  os.system('cls' if os.name == 'nt' else 'clear')

  if opcion == '1':
    print('PRODUCTOS MAS VENDIDOS:\n')
    for venta in compras_ordenadas:
      if venta[1] != 1:
        print('Venta de producto: "'+ venta[0]+'". Realizadas:',venta[1],'veces.\n')
      else:
        print('Venta de producto: "'+ venta[0]+'". Realizada:',venta[1],'vez.\n')

  elif opcion == '2':
    print('PRODUCTOS MAS BUSCADOS:\n')
    for busqueda in busquedas_ordenadas:
      if busqueda[1] != 1:
        print('Búsqueda de producto: "'+ busqueda[0]+'". Realizadas:',busqueda[1],'veces.\n')
      else:
        print('Búsqueda de producto: "'+ busqueda[0]+'". Realizada:',busqueda[1],'vez.\n')
      #print(products.index(busqueda[0]))

  elif opcion == '3':
    print('PRODUCTOS MENOS VENDIDOS - POR CATEGORIA:\n')
    for venta in categorias_unicas_ventas:
      contador = 0
      for venta_unica in ventas:
        if contador != 0:
          if venta_unica[2] != 1:
            print ('Producto: "'+venta_unica[0]+'." Vendidos:',venta_unica[2],'veces.')
          else:
            print ('Producto: "'+venta_unica[0]+'." Vendidos:',venta_unica[2],'vez.')
        else:
          print ('CATEGORIA:',venta_unica)
        contador += 1
      print()

  elif opcion == '4':
    print('PRODUCTOS MENOS BUSCADOS - POR CATEGORIA:\n')
    for busqueda in categorias_unicas_busquedas:
      contador = 0
      for busqueda_unica in busqueda:
        if contador != 0:
          if busqueda_unica[1] != 1:
            print ('Producto: "'+busqueda_unica[0]+'." Buscados',busqueda_unica[1],'veces.')
          else:
            print ('Producto: "'+busqueda_unica[0]+'." Buscados',busqueda_unica[1],'vez.')
        else:
          print ('CATEGORIA:',busqueda_unica,'\n')
        contador += 1
      print()

  elif opcion == '5':
    print('LISTA DE 20 PRODUCTOS CON PEOR CALIFICACION DE RESEÑAS - TOMANDO EN CUENTA DEVOLUCIONES:\n')
    contador = 1
    for producto in productos_reviewed_unicos_puntuados[:20]:
      print(str(contador)+'- Producto: "'+str(producto[0])+'". Con una puntuación de reseñas de: '+str(producto[1])+'. y una cantidad de devoluciones(es) de: '+ str(producto[2])+'\n' )
      contador += 1

  elif opcion == '6':
    print('LISTA DE 20 PRODUCTOS CON MEJOR CALIFICACION DE RESEÑAS - TOMANDO EN CUENTA DEVOLUCIONES:\n')
    contador = 1
    for producto in productos_reviewed_unicos_puntuados[:-21:-1]:
      print(str(contador)+'- Producto: "'+str(producto[0])+'". Con una puntuación de reseñas de: '+str(producto[1])+'. y una cantidad de devoluciones(es) de: '+ str(producto[2])+'\n' )
      contador += 1

  elif opcion == '7':
    print('Total de ingresos, ventas promedio y ventas totales mensuales:\n')

    print(tabulate(promedios, headers=['Mes', 'Total de Ingresos', 'Ventas Promedio', 'Ventas Totales'], tablefmt="fancy_grid"))
    print('Total anual = ${:,.2f}'.format(total_anual),'\n')
  elif opcion == '0':
    print('Programa finalizado. Adiós.')
    break
  else:
    print('Opción inexistente, Ingrese una opción correcta...\n')

  input ("\'Click \'ENTER\'  Key to Continue\'")
  os.system('cls' if os.name == 'nt' else 'clear')
```

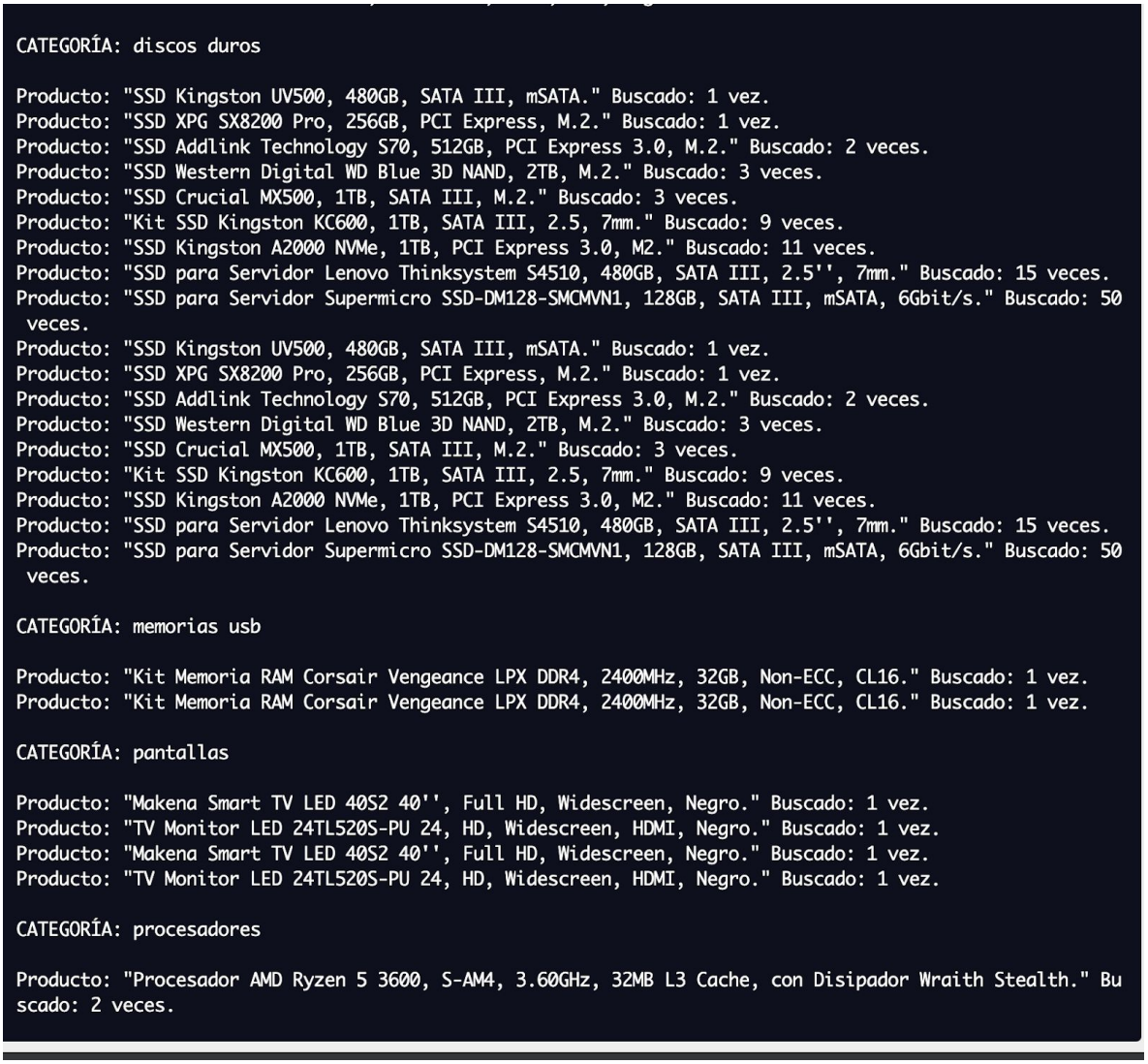
Finalmente tenemos el loop final con las opciones para imprimir todo lo desarrollado en el código, dividimos en 7 opciones y una extra para salir del

programa y finalizarlo. Hacemos ciclos para ir imprimiendo con un formato que señale el producto y la cantidad de manera entendible entre comillas -por eso cambiamos el ID por el nombre-, también iterando distintas categorías en sus ejercicios que lo pidieron, incluso enumerando las listas de los 20 productos de mejores y peores reseñas y para la opción de las ventas e ingresos tanto mensuales como anual utilizamos una tabulación que le da un formato muy elegante a la tabla y un formato de dinero para imprimir el valor de las ganancias totales anuales de manera más formal.

Cabe destacar que este ciclo de opciones/impresiones se repite infinitamente al menos que se utiliza la opción 8 para finalizar el programa. No hay bugs ni ciclos infinitos sin opción de salida ni fallas en las operaciones, todo fue bien cuidado.

Solución al problema





19 - Producto: "Procesador AMD Ryzen 3 3300X S-AM4, 3.80GHz, Quad-Core, 16MB L2 Cache". Con una puntuación de reseñas de: 10. y una cantidad de devolucion(es) de: 0

20 - Producto: "SSD Western Digital WD Blue 3D NAND, 2TB, M.2". Con una puntuación de reseñas de: 10. y una cantidad de devolucion(es) de: 0

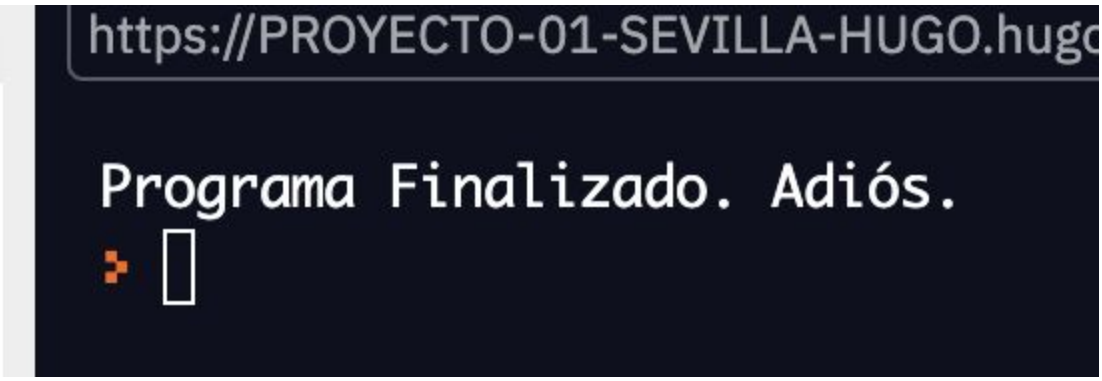
'Click "ENTER" Key to Continue'

Total de ingresos, ventas promedio y ventas totales mensuales:

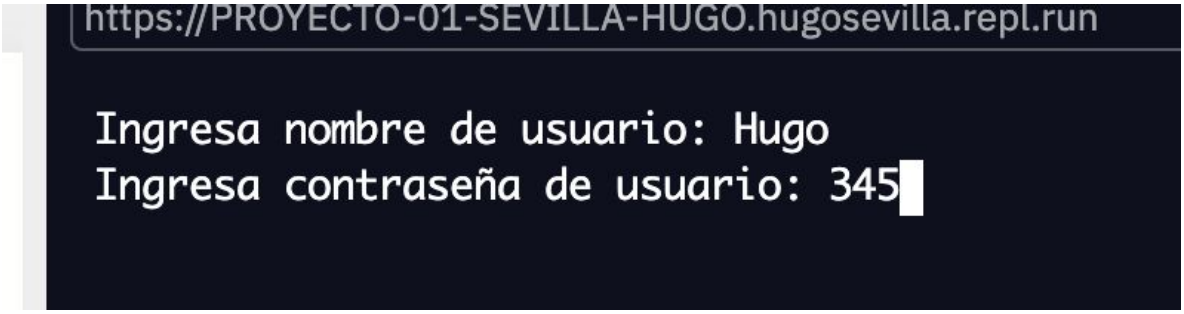
Mes	Total de Ingresos	Ventas Promedio	Ventas Totales
Enero	177138	3418.25	181167
Febrero	178464	4390.32	180003
Marzo	216261	4350.37	221869
Abril	313596	4238.73	317905
Mayo	187006	5434	195624
Junio	39759	3614.45	39759
Julio	51449	4677.18	51449
Agosto	11937	3979	11937
Septiembre	0	2199	2199
Octubre	0	0	0
Noviembre	0	3089	3089
Diciembre	0	0	0

Total anual = $1,175,610.00

'Click "ENTER" Key to Continue'


https://PROYECTO-01-SEVILLA-HUGO.hugo
Programa Finalizado. Adiós.


https://PROYECTO-01-SEVILLA-HUGO.hugosevilla.repl.run
Contraseña incorrecta. Programa Finalizado. Adiós.


https://PROYECTO-01-SEVILLA-HUGO.hugosevilla.repl.run
Ingresa nombre de usuario: Hugo
Ingresa contraseña de usuario: 345


https://PROYECTO-01-SEVILLA-HUGO.hugosevilla.repl.run
No Existe Usuario. Programa Finalizado. Adiós.


Usuario indentificado, pero no es Administrador. Programa Finalizado. Adiós.

Conclusión

Es un gran reto por todo lo que pide pues no se dan todas las bases y esto nos lleva a investigar por nuestra parte para poder realizar ciertas funcionalidades, el gran reto es hacer búsquedas, crear listas a partir de listas existentes con copias ya que trabajamos con listas de listas y hace que su manipulación sea más compleja al menos si queremos hacer un código eficiente, aunque para este primer proyecto no les importa que utilicemos los comandos más eficientes. Vemos la importancia de utilizar ciclos anidados y de cómo teniendo las bases se puede llegar a cualquier reto, ningún trabajo se debe subestimar pues estos proyectos son de alto calibre. Tener todas estas bases son suficientes para poder entender el siguiente tema de los CSV y DataFrame, pero debemos manejar de preferencia entendiendo con otro tipo de datos como funcionan, iterar, manipular y así podremos ir a algo más complejo. Vemos la importancia de tener bien comentado el código y explicado además de señalar de manera explícita las variables para que cualquier persona pueda entender nuestros código desde otros programadores hasta gente que no se dedica a la programación pueda entender su funcionamiento de una manera clara.