REPORTE 01

INTRODUCCIÓN A PYTHON

Descripción breve

Poner en práctica las bases de Python para análisis y clasificación de datos mediante datos de entrada de usuario y validaciones, uso y definición de variables y listas, operadores lógicos y condicionales.

Raúl Torres Duque Vázquez Aldana
marinervii@gmail.com

Reporte 01 – Introducción a Python

Introducción

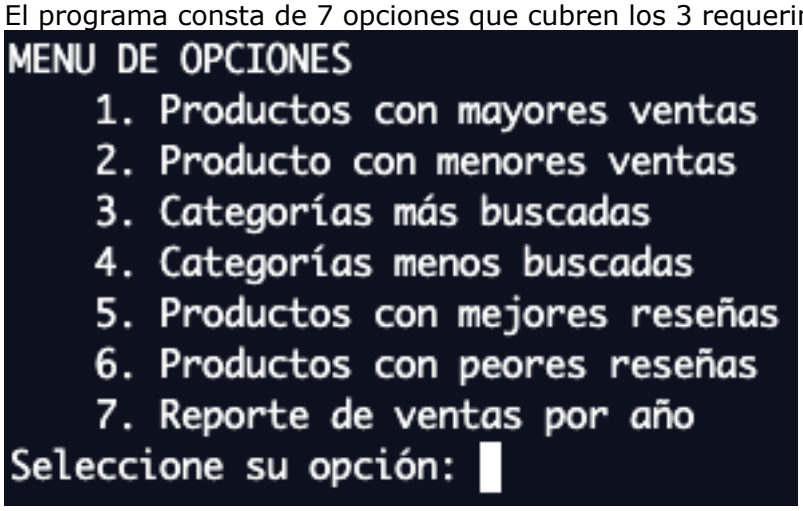
Mediante el uso de los fundamentos del lenguaje, se integraran las consignas solicitadas para el análisis de datos del caso propuesto de Lifestore. El programa deberá resolver los objetivos propuestos, sin el uso de librerias o programas de terceros. Adicionalmente, se agregó una rutina para manternese en el programa mientras no se opte por salir del mismo, asi como un contador de intentos de logueo.

Objetivo

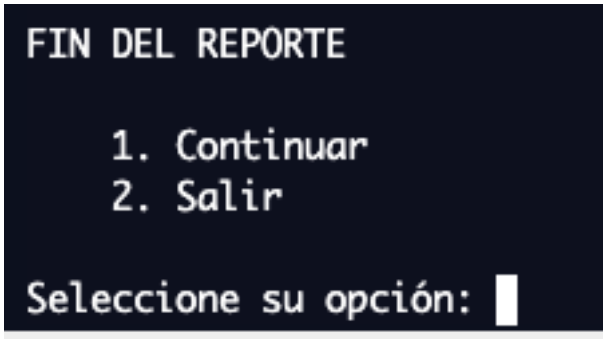
Hacer uso de los elementos básicos del lenguaje: tipos de variable, validaciones, listas, índices, slicing, control de flujo para resolver un problema real que se puede implementar para un volumen de datos considerable

Procedimiento

El programa consta de 3 archivos: main.py, el archivo de datos provisto lifestore_file.py y un archivo con una lista utilizada para las categorías, lifestore_categories.
Se crearon tres funciones para para la creación de listas reutilizables a lo largo del programa.
Se creó una rutina de logueo con contador de errores
El programa consta de 7 opciones que cubren los 3 requerimientos:



Se creó también una rutina para mantenerse en el programa hasta que se opte por salir



Definición del código

Archivo lifestore_categories.py. Es simplemente un listado de categorías que se usa en un archivo por separado para claridad en el código

```
"""
Este es un archivo adicional para las categorías de productos

Sacado del listado de productos
Ordenado alfabéticamente
"""

product_categories = ['audifonos', 'bocinas', 'discos duros',
'memorias usb', 'pantallas', 'procesadores','tarjetas de video',
'tarjetas madre']
```

Archivo main.py.

```
from lifestore_file import lifestore_searches
from lifestore_file import lifestore_sales
from lifestore_file import lifestore_products
from lifestore_categories import product_categories

"""Funciones para listas reutilizables.
Esta contabiliza las veces que una categoría fue buscada
Se usa en las opciones 1 a 4 del menú principal"""
# Función contadora de búsquedas
def search_counter():
  def_counter = []
  for product_searches in lifestore_products:
    counter = 0
    for search in lifestore_searches:
      if search[1] == product_searches[0]:
        counter += 1

    def_counter.append([product_searches[3], counter])
```

```
  return def_counter
```

“””Funciones para listas reutilizables.
Esta crea una lista que contabiliza las veces que se vendió un producto
Se usa para las opciones 1,2, y 5,6 y 7”””

```
# Función contadora de ventas
def sales_counter(year, month):
  search_period = month + '/' + year
  #print('periodo de búsqueda: ' + month + '-' + year)
  def_counter = []
  for product in lifestore_products:
    counter = 0
    for sales in lifestore_sales:
      dt = sales[3]
      sales_date = dt[3:10]
      #print('sales_date' + sales_date)
      if sales[1] == product[0] and sales_date == search_period and
sales[4] != 1:
        def_counter.append([product[1], product[2], product[3],
sales[3]])

  return def_counter
```

“””Funciones para listas reutilizables.
Esta contabiliza las veces que una categoría tuvo una venta real (sin devolución)”””

```
def category_sales():
  # Menú de opciones
  years = ['2019', '2020']
  months =
['01','02','03','04','05','06','07','08','09','10','11','12']
  year = 0
  month = 0

  while year not in years:
    print("""SELECCIONE AÑO
    2020
    2019""")
    year = input("Seleccione el año: ")
  while month not in months:
    print("""\nMes en formato de dos dígitos (01, 09, 11, etc)
    """)
```

```
    month = input("Mes: ")
    sales = sales_counter(year,month)
    print('\n')

  unordered_category_sales = []
  for category in product_categories:
    counter = 0
    for product in lifestore_products:
      if product[3] == category:
        for prod_sales in sales:
          if prod_sales[0] == product[1]:
            counter += 1
        unordered_category_sales.append([category, product[1],
counter])
  print('PERIODO: ' + str(month) + '-' + str(year))
  return unordered_category_sales


def category_sum():
  #Sumarizar por categoría
  counter = 0
  category_searches_list = []
  category = ''

  for cat_counter in searches_counter:
    if not category:
      category = cat_counter[0]
      counter = cat_counter[1]
    elif cat_counter[0] == category:
      counter += cat_counter[1]
    else:
      category_searches_list.append([category, counter])
      category = cat_counter[0]
      counter = cat_counter[1]

  """
  Para la última categoría de la lista, porque ya no entra a la
comparación del loop pero si tiene la cuenta de las búsquedas
  """
  category_searches_list.append([category, counter])

  return category_searches_list
```

“””Inicio del programa”””

```
# Inicio main

“””Estas tuplas de usuarios permiten enviar mensajes del resultado
de logueo, así como permite mantener el progrma activo si se comete
un error en el logueo”””

# Lista de usuarios autorizados
admin_list = [["javier", "123"], ["pedro", "456"], ["daniel",
"789"]]

“””El programa permite 3 errores de logueo antes de terminarlo”””
# Variables de logueo
login_attempts = 0;
login_credentials = []


“””Con el while podemos mantener la pantalla de logueo en caso de
error”””

# Proceso de logueo
while login_credentials not in admin_list:
  if login_attempts >= 3:
    print("Ha intentado loguearse demasiadas veces. El acceso ha
sido bloquedo")
    break
  elif login_attempts >= 1:
    print("Su nombre de usuario o contraseña no son correctos. Por
favor, intente de nuevo")

  username = input("Ingrese su nombre de usuario: ")
  userpwd = input("Ingrese su contraseña: ")
  login_credentials = [username, userpwd]
  login_attempts += 1

if login_credentials in admin_list:
  print("Bienvenido!\n")
else:
  print("Si tiene problemas para acceder, contacte al administrador
del sistema.")
```

```
“””Lo que se logra con este while es mantener el programa activo
(una vez logueado), mientras el usuario no seleccione Salir”””

continue_program = True
while continue_program == True:

“””Se usa el mismo proceso que el logueo para mantener la pantalla
de menú si no se selecciona una opción válida”””

  # Menú de opciones
  options = [1, 2, 3, 4, 5, 6, 7] #Se usa en el while del menú. Si
el dato ingresado no existe en la lista, sigue mostrando el menú
hasta que se seleccione una opción válida
  option = 0 # Se requiere un valor inicial para el while

  while option not in options:
    print("""MENU DE OPCIONES
    1. Productos con mayores ventas
    2. Producto con menores ventas
    3. Categorías más buscadas
    4. Categorías menos buscadas
    5. Productos con mejores reseñas
    6. Productos con peores reseñas
    7. Reporte de ventas por año""")
    option = int(input("Seleccione su opción: "))

    #Verificar si la respuesta se puede convertir a entero o
repetir las opciones
    try:
      option = int(option)
    except ValueError:
      option = 0

“””La opciones 1, 2, 3 y 4 cubren el primer objetivo solicitado de
ventas”””

  # OPCION 1 mayores ventas
  if option == 1:
    unordered_category_sales = category_sales()
    #Ordenamiento de mayor a menor de la lista
    ordered_category_sales = []
    while unordered_category_sales:
      for category in product_categories:
        max_searched = unordered_category_sales[0][2]
```

```
        current_max = [unordered_category_sales[0][0],
unordered_category_sales[0][1], unordered_category_sales[0][2]]

        for product in unordered_category_sales:
          if product[2] > max_searched:
            max_searched = product[2]
            current_max = [product[0], product[1], product[2]]

        ordered_category_sales.append(current_max)
        unordered_category_sales.remove(current_max)

    #Salida de datos solicitado
    print('PRODUCTOS CON MAYORES VENTAS')
    print('No. VENTAS\t|| PRODUCTO')
    for index in range(50):
      print(str(ordered_category_sales[index][2]) + '        \t|| '
+ ordered_category_sales[index][1])

  # OPCION 2 menores ventas
  if option == 2:
    unordered_category_sales = category_sales()
    #Ordenamiento de mayor a menor de la lista
    ordered_category_sales = []
    while unordered_category_sales:
      for category in product_categories:
        min_searched = unordered_category_sales[0][2]
        current_min = [unordered_category_sales[0][0],
unordered_category_sales[0][1], unordered_category_sales[0][2]]

        for product in unordered_category_sales:
          if product[2] < min_searched:
            min_searched = product[2]
            current_min = [product[0], product[1], product[2]]

        ordered_category_sales.append(current_min)
        unordered_category_sales.remove(current_min)

    #Salida de datos solicitado
    print('PRODUCTOS CON MENORES VENTAS')
    print('No. VENTAS\t|| PRODUCTO')
    for index in range(50):
      print(str(ordered_category_sales[index][2]) + '        \t|| '
+ ordered_category_sales[index][1])
```

```
  # OPCION 3 categorías más buscadas
  elif option == 3:
    """Contar el número de búsquedas para cada categoría"""
    #Sacar los totales por producto
    searches_counter = search_counter()

    #Sumarizar las búsquedas por categoría
    category_searches_list = category_sum()

    #Ordenar las búsquedas de mayor a menor
    ordered_list = []
    while category_searches_list:
      max_searched = category_searches_list[0][1]
      current_max = [category_searches_list[0][0],
category_searches_list[0][1]]

      for category_listing in category_searches_list:
        if category_listing[1] > max_searched:
          max_searched = category_listing[1]
          current_max = [category_listing[0], category_listing[1]]

      ordered_list.append(current_max)
      category_searches_list.remove(current_max)

    #Imprimir el listado ordenado al usuario
    print("\nCategorías más buscadas")
    print('No. BUSQUEDAS\t|| CATEGORIA')
    for index in range(5):
      print(str(ordered_list[index][1]) + '           \t|| ' +
ordered_list[index][0])


  # OPCION 4 categorías menos buscadas
  elif option == 4:
    """Contar el número de búsquedas para cada categoría"""
    #Sacar los totales por producto
    searches_counter = search_counter()

    #Sumarizar las búsquedas por categoría
    category_searches_list = category_sum()

    #Ordenar las búsquedas de mayor a menor
    ordered_list = []
    while category_searches_list:
```

```
      min_searched = category_searches_list[0][1]
      current_min = [category_searches_list[0][0],
category_searches_list[0][1]]

      for category_listing in category_searches_list:
        if category_listing[1] < min_searched:
          min_searched = category_listing[1]
          current_min = [category_listing[0], category_listing[1]]

      ordered_list.append(current_min)
      category_searches_list.remove(current_min)

    #Imprimir el listado ordenado al usuario
    print("\nCategorías menos buscadas")
    print('No. BUSQUEDAS\t|| CATEGORIA')
    for index in range(5):
      print(str(ordered_list[index][1]) + '           \t|| ' +
ordered_list[index][0])

“””las opciones 5 y 6 cubren el objetivo 2 del ejercicio”””

  # OPCION 5 Mejores reseñas
  elif option == 5:
    #reviews = lifestore_sales
    reviews = []
    #Ordenar las reseñas de mayor a menor
    ordered_list = []
    remove_item = []
    for product in lifestore_products:
      # Usamos estas dos variables para sacar promedio de
valoraciones por producto
      prod_review_sum = 0
      prod_review_count = 0
      for sales in lifestore_sales:
        if product[0] == sales[1] and sales[4] != 1:
          prod_review_sum += sales[2]
          prod_review_count += 1
      # Promedio de valoraciones
      if prod_review_count == 0:
        prod_review_avg = 0
      else:
        prod_review_avg = int(prod_review_sum/prod_review_count)

      reviews.append([product[1], prod_review_avg])
```

```
    while reviews:
      max_review = reviews[0][1]
      current_best = [reviews[0][0], reviews[0][1]]
      remove_item = reviews[0]

      for review_listing in reviews:
        if review_listing[1] > max_review:
          max_review = review_listing[1]
          current_best = [review_listing[0], review_listing[1]]
          remove_item = review_listing

      ordered_list.append(current_best)
      reviews.remove(remove_item)

    #Imprimir el listado ordenado al usuario
    print('PROM RESEÑAS\t|| PRODUCTO')
    for index in range(20):
      print('          ' + str(ordered_list[index][1]) +' \t|| ' +
ordered_list[index][0])

  elif option == 6:
    #reviews = lifestore_sales
    reviews = []
    #Ordenar las reseñas de mayor a menor
    ordered_list = []
    remove_item = []
    for product in lifestore_products:
      # Usamos estas dos variables para sacar promedio de
valoraciones por producto
      prod_review_sum = 0
      prod_review_count = 0
      for sales in lifestore_sales:
        if product[0] == sales[1] and sales[4] != 1:
          prod_review_sum += sales[2]
          prod_review_count += 1
      # Promedio de valoraciones
      if prod_review_count == 0:
        prod_review_avg = 0
      else:
        prod_review_avg = int(prod_review_sum/prod_review_count)

      reviews.append([product[1], prod_review_avg])
```

```
    while reviews:
      min_review = reviews[0][1]
      current_min = [reviews[0][0], reviews[0][1]]
      remove_item = reviews[0]

      for review_listing in reviews:
        if review_listing[1] < min_review:
          min_review = review_listing[1]
          current_min = [review_listing[0], review_listing[1]]
          remove_item = review_listing

      ordered_list.append(current_min)
      reviews.remove(remove_item)

    #Imprimir el listado ordenado al usuario
    print('PROM RESEÑAS\t|| PRODUCTO')
    for index in range(20):
      print('          ' + str(ordered_list[index][1]) +' \t|| ' +
ordered_list[index][0])

“””La opción 7 se utiliza para el objetivo 3 de reporte de ventas
por año”””

  elif option == 7:
    years = ['2019', '2020']
    months = [['01', 'Ene'], ['02', 'Feb'], ['03', 'Mar'], ['04',
'Abr'], ['05', 'May'], ['06', 'Jun'], ['07', 'Jul'], ['08', 'Ago'],
['09', 'Sep'], ['10', 'Oct'], ['11', 'Nov'], ['12', 'Dic']]

    for year in years:
      month_sales = [] # Lista sin ordenar de ventas por mes
      ordered_mo_sales = [] # Lista ordenada de mayor a menor de
ventas por mes
      yr_sales = 0
      print('VENTAS AÑO ' + year)
      for month in months:
        mo_sales = 0 # Ventas del mes
        sales = sales_counter(year, month[0])
        for period_sales in sales:
          product = period_sales[0]
          for price in lifestore_products:
            if price[1] == product:
              ppt = price[2]
              break
```

```
          mo_sales += ppt
        print('\t\t' + month[1] + '     \t| ' + str(mo_sales) +
'.00')

        month_sales.append([month[1], mo_sales])
        yr_sales += mo_sales
      print('\t\t-----------------------------------')
      print('\t\tTOTAL     \t| ' + str(yr_sales) + '.00')
      """ PROMEDIO MENSUAL POR AÑO """
      mo_avg = yr_sales/12
      print('\t\t-----------------------------------')
      print('\t\tPROM MENSUAL| ' + str(mo_avg) + '\n')
      print('-------------------------------')

      # Meses con más ventas
      while month_sales:
        max_sales = month_sales[0][1]
        current_max = [month_sales[0][0], max_sales]

        for current_sale in month_sales:
          if current_sale[1] > max_sales:
            max_sales = current_sale[1]
            current_max = [current_sale[0], max_sales]

        ordered_mo_sales.append(current_max)
        month_sales.remove(current_max)

      print('\tMESES CON MEJORES VENTAS')
      for order in ordered_mo_sales:
        if order[1] > 0:
          print('\t' + order[0] + '     \t| ' + str(order[1]) +
'.00')
      print('_____________________________________\n')

“””Al final, el programa da la opción de seguir o salir del
programa”””

  """
  Fin del reporte. Menú de Salida
  """
  valid_resp = [1, 2]
  resp = 0
  print('\nFIN DEL REPORTE')
  while resp not in valid_resp:
```

```
    print("""
    1. Continuar
    2. Salir
    """)
    resp = input("Seleccione su opción: ")
    #Verificar si la respuesta se puede convertir a entero o
repetir las opciones
    try:
      resp = int(resp)
    except ValueError:
      resp = 0

    if int(resp) == 2:
      continue_program = False
      print('\n¡Hasta luego!')
```

Resultados

SALIDA DE LA OPCIÓN 1. EL NÚMERO DE VENTAS POR PRODUCTO, AGRUPADO POR CATEGORÍAS, ORDENADO DE MAYOR A MENOR. (LA OPCIÓN 2 ES LO MISMO PERO ORDENADO DE MENOR A MAYOR)

Seleccione su opción: 1

SELECCIONE AÑO

2020

2019

Seleccione el año: 2020

Mes en formato de dos dígitos (01, 09, 11, etc)

Mes: 07

PERIODO: 07-2020

PRODUCTOS CON MAYORES VENTAS

No. VENTAS || PRODUCTO

6 || Procesador Intel Core i7-9700K, S-1151, 3.60GHz, 8-Core, 12MB Smart Cache (9na. Generación Coffee Lake)

6 || Procesador Intel Core i5-9600K, S-1151, 3.70GHz, Six-Core, 9MB Smart Cache (9na. Generiación - Coffee Lake)

6 || Procesador Intel Core i3-8100, S-1151, 3.60GHz, Quad-Core, 6MB Smart Cache (8va. Generación - Coffee Lake)

5 || Procesador Intel Core i3-9100F, S-1151, 3.60GHz, Quad-Core, 6MB Cache (9na. Generación - Coffee Lake)

5 || Procesador Intel Core i9-9900K, S-1151, 3.60GHz, 8-Core, 16MB Smart Cache (9na. Generación Coffee Lake)

4 || SSD Adata Ultimate SU800, 256GB, SATA III, 2.5'', 7mm

4 || SSD para Servidor Lenovo Thinksystem S4510, 480GB, SATA III, 2.5'', 7mm

4 || SSD Samsung 860 EVO, 1TB, SATA III, M.2

4 || Procesador AMD Ryzen 5 2600, S-AM4, 3.40GHz, Six-Core, 16MB L3 Cache, con Disipador Wraith Stealth

4 || Procesador AMD Ryzen 3 3200G con Gráficos Radeon Vega 8, S-AM4, 3.60GHz, Quad-Core, 4MB L3, con Disipador Wraith Spire

3 || SSD Kingston A400, 120GB, SATA III, 2.5'', 7mm

3 || SSD para Servidor Supermicro SSD-DM128-SMCMVN1, 128GB, SATA III, mSATA, 6Gbit/s

3 || SSD para Servidor Lenovo Thinksystem S4500, 480GB, SATA III, 3.5'', 7mm

2 || Procesador AMD Ryzen 3 3300X S-AM4, 3.80GHz, Quad-Core, 16MB L2 Cache

2 || Procesador AMD Ryzen 5 3600, S-AM4, 3.60GHz, 32MB L3 Cache, con Disipador Wraith Stealth

1 || SSD XPG SX8200 Pro, 256GB, PCI Express, M.2

1 || SSD Kingston A2000 NVMe, 1TB, PCI Express 3.0, M2

1 || Kit SSD Kingston KC600, 1TB, SATA III, 2.5, 7mm

1 || SSD Crucial MX500, 1TB, SATA III, M.2

1 || SSD Kingston UV500, 480GB, SATA III, mSATA

1 || SSD Western Digital WD Blue 3D NAND, 2TB, M.2

1 || SSD Addlink Technology S70, 512GB, PCI Express 3.0, M.2

1 || Tarjeta Madre ASRock Micro ATX B450M Steel Legend, S-AM4, AMD B450, HDMI, 64GB DDR4 para AMD

1 || Tarjeta Madre ASUS ATX ROG STRIX Z390-E GAMING, S-1151, Intel Z390, HDMI, 64GB DDR4 para Intel

1 || Tarjeta Madre MSI ATX B450 TOMAHAWK MAX, S-AM4, AMD B450, 64GB DDR4 para AMD

1 || Tarjeta Madre ASRock ATX H110 Pro BTC+, S-1151, Intel H110, 32GB DDR4, para Intel

1 || Tarjeta Madre Gigabyte micro ATX GA-H110M-DS2, S-1151, Intel H110, 32GB DDR4 para Intel

0 || Logitech Audífonos Gamer G332, Alámbrico, 2 Metros, 3.5mm, Negro/Rojo

0 || Logitech Audífonos Gamer G635 7.1, Alámbrico, 1.5 Metros, 3.5mm, Negro/Azul

0 || ASUS Audífonos Gamer ROG Theta 7.1, Alámbrico, USB C, Negro

0 || Acer Audífonos Gamer Galea 300, Alámbrico, 3.5mm, Negro

0 || Audífonos Gamer Balam Rush Orphix RGB 7.1, Alámbrico, USB, Negro

0 || Cougar Audífonos Gamer Phontum Essential, Alámbrico, 1.9 Metros, 3.5mm, Negro.

0 || Energy Sistem Audífonos con Micrófono Headphones 1, Bluetooh, Inalámbrico, Negro/Grafito

0 || Genius GHP-400S Audífonos, Alámbrico, 1.5 Metros, Rosa

0 || Getttech Audífonos con Micrófono Sonority, Alámbrico, 1.2 Metros, 3.5mm, Negro/Rosa

0 || Ginga Audífonos con Micrófono GI18ADJ01BT-RO, Bluetooth, Alámbrico/Inalámbrico, 3.5mm, Rojo

0 || HyperX Audífonos Gamer Cloud Flight para PC/PS4/PS4 Pro, Inalámbrico, USB, 3.5mm, Negro

0 || Iogear Audífonos Gamer GHG601, Alámbrico, 1.2 Metros, 3.5mm, Negro

0 || Klip Xtreme Audífonos Blast, Bluetooth, Inalámbrico, Negro/Verde

0 || Logitech Bocinas para Computadora con Subwoofer G560, Bluetooth, Inalámbrico, 2.1, 120W RMS, USB, negro

0 || Lenovo Barra de Sonido, Alámbrico, 2.5W, USB, Negro

0 || Acteck Bocina con Subwoofer AXF-290, Bluetooth, Inalámbrico, 2.1, 18W RMS, 180W PMPO, USB, Negro

0 || Verbatim Bocina Portátil Mini, Bluetooth, Inalámbrico, 3W RMS, USB, Blanco

0 || Ghia Bocina Portátil BX300, Bluetooth, Inalámbrico, 40W RMS, USB, Rojo - Resistente al Agua

0 || Naceb Bocina Portátil NA-0301, Bluetooth, Inalámbrico, USB 2.0, Rojo

0 || Ghia Bocina Portátil BX800, Bluetooth, Inalámbrico, 2.1 Canales, 31W, USB, Negro

0 || Ghia Bocina Portátil BX900, Bluetooth, Inalámbrico, 2.1 Canales, 34W, USB, Negro - Resistente al Agua

0 || Ghia Bocina Portátil BX400, Bluetooth, Inalámbrico, 8W RMS, USB, Negro

0 || Ghia Bocina Portátil BX500, Bluetooth, Inalámbrico, 10W RMS, USB, Gris

FIN DEL REPORTE

SALIDA DE LA OPCIÓN 3. LAS 5 MAYORES POR EL NÚMERO DE BÚSQUEDAS ORDENADO DE MAYOR A MENOR. (LA OPCIÓN 4 ES LO MISMO PERO ORDENADO DE MENOR A MAYOR)

Seleccione su opción: 3

Categorías más buscadas
No. BUSQUEDAS || CATEGORIA
463 || discos duros
222 || procesadores
137 || tarjetas madre
82 || tarjetas de video
64 || audifonos

FIN DEL REPORTE

SALIDA DE LA OPCIÓN 5. LOS 20 PRODUCTOS CON EL MEJOR PROMEDIO DE RESEÑAS. (OPCIÓN 6 SON LOS 20 PRODUCTOS CON EL PEOR PROMEDIO)

Seleccione su opción: 5

PROM RESEÑAS || PRODUCTO

5 || Procesador AMD Ryzen 3 3300X S-AM4, 3.80GHz, Quad-Core, 16MB L2 Cache

5 || Procesador Intel Core i9-9900K, S-1151, 3.60GHz, 8-Core, 16MB Smart Cache (9na. Generación Coffee Lake)

5 || Procesador Intel Core i7-9700K, S-1151, 3.60GHz, 8-Core, 12MB Smart Cache (9na. Generación Coffee Lake)

5 || Procesador Intel Core i5-9600K, S-1151, 3.70GHz, Six-Core, 9MB Smart Cache (9na. Generiación - Coffee Lake)

5 || Tarjeta de Video ASUS AMD Radeon RX 570, 4GB 256-bit GDDR5, PCI Express 3.0

5 || Tarjeta de Video MSI AMD Mech Radeon RX 5500 XT MECH Gaming OC, 8GB 128-bit GDDR6, PCI Express 4.0

5 || Tarjeta de Video MSI NVIDIA GeForce GTX 1050 Ti OC, 4GB 128-bit GDDR5, PCI Express x16 3.0

5 || Tarjeta de Video Sapphire AMD Pulse Radeon RX 5500 XT Gaming, 8GB 128-bit GDDR6, PCI Express 4.0

5 || Tarjeta de Video Zotac NVIDIA GeForce GTX 1660 Ti, 6GB 192-bit GDDR6, PCI Express x16 3.0

5 || Tarjeta Madre Gigabyte XL-ATX TRX40 Designare, S-sTRX4, AMD TRX40, 256GB DDR4 para AMD

5 || Kit SSD Kingston KC600, 1TB, SATA III, 2.5, 7mm

5 || SSD Crucial MX500, 1TB, SATA III, M.2

5 || SSD Western Digital WD Blue 3D NAND, 2TB, M.2

5 || Kit Memoria RAM Corsair Dominator Platinum DDR4, 3200MHz, 16GB (2x 8GB), Non-ECC, CL16, XMP

5 || TCL Smart TV LED 55S425 54.6, 4K Ultra HD, Widescreen, Negro

5 || TV Monitor LED 24TL520S-PU 24, HD, Widescreen, HDMI, Negro

5 || Logitech Audífonos Gamer G332, Alámbrico, 2 Metros, 3.5mm, Negro/Rojo

5 || Logitech Audífonos Gamer G635 7.1, Alámbrico, 1.5 Metros, 3.5mm, Negro/Azul

4 || Procesador AMD Ryzen 5 3600, S-AM4, 3.60GHz, 32MB L3 Cache, con Disipador Wraith Stealth

4 || Procesador AMD Ryzen 5 2600, S-AM4, 3.40GHz, Six-Core, 16MB L3 Cache, con Disipador Wraith Stealth

FIN DEL REPORTE

OPCIÓN 7 ES UN REPORTE DE VENTAS QUE CALCULA LAS VENTAS POR MES EN EL AÑO (PARA CADA AÑO EN LA LISTA DE VENTAS), EL GRAN TOTAL POR AÑO, EL PROMEDIO DE VENTAS POR MES EN EL AÑO Y LOS MESES CON MAYORES VENTAS. PARA PROBAR EL CONCEPTO SE MODIFICÓ SOLO UN REGISTRO DEL LISTADO QUE DE TODAS MANERAS ESTABA MARCADO COMO 2002 CON EL OBJETIVO DE OBTENER DATOS EN MÁS DE UN AÑO. LAS LISTAS DE AÑOS ES HARD-CODED, NO CALCULADA

Seleccione su opción: 7

VENTAS AÑO 2019

Ene | 0.00

Feb | 0.00

Mar | 0.00

Abr | 0.00

May | 259.00

Jun | 0.00

Jul | 0.00

Ago | 0.00

Sep | 0.00

Oct | 0.00

Nov | 0.00

Dic | 0.00

TOTAL | 259.00

PROM MENSUAL| 21.583333333333332

MESES CON MEJORES VENTAS

May | 259.00

VENTAS AÑO 2020

Ene | 117738.00

Feb | 107270.00

Mar | 162931.00

Abr | 191066.00

May | 91677.00

Jun | 36949.00

Jul | 26949.00

Ago | 3077.00

Sep | 0.00

Oct | 0.00

Nov | 0.00

Dic | 0.00

TOTAL | 737657.00

PROM MENSUAL| 61471.416666666664

MESES CON MEJORES VENTAS

Abr | 191066.00

Mar | 162931.00

Ene | 117738.00

Feb | 107270.00

May | 91677.00

Jun | 36949.00

Jul | 26949.00

Ago | 3077.00

FIN DEL REPORTE

Conclusiones

El ejercicio se cumplió de manera razonable, de acuerdo a la interpretación de una definición vaga de la consigna.

Si bien fue un ejercicio muy interesante, de acuerdo al objetivo de resolver un problema complejo con el uso de los aspectos básicos de programación, considero que fue una tarea desproporcionada entre lo solicitado y lo expuesto y enseñado en el curso, considerando que hay una gran gama de perfiles en el grupo, incluyendo gente con conocimientos básicos.

También considero, que el tiempo para completar todo el ejercicio, incluyendo este reportes, no fue para nada suficiente.