

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	3
Definición del código	4
Solución al problema	9
Conclusión	12

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

```
main.py
1 # Primero tenemos la importación de las librerías y las bases de datos
2
3 from lifestore_file import lifestore_products as products, lifestore_sales as sales, lifestore_searches as searches
4 from operator import itemgetter
5 from collections import Counter
6 import os
7 import numpy as np
8 from copy import deepcopy
9 from tabulate import tabulate
10
11
12 # Aquí tenemos a nuestros usuarios Admin y no Admin
13 usuarios = [['Hugo', '123', 'Admin'], ['Paco', '456', 'No Admin'], ['Luis', '789', 'Admin'], ['Cesar', '321', 'No Admin']]
14
15 # opción default e inputs para login / iniciar sesión
16 opcion = ''
17 usuario = input('Ingresa nombre de usuario: ')
18 password = input('Ingresa contraseña de usuario: ')
19
20 # esto es para hacer clear en la consola
21 os.system('cls' if os.name == 'nt' else 'clear')
22
23 # generamos listas de usuarios únicamente administradores para comprar nombres de los que sí son usuarios y cuáles de esos son
administradores
24 admins = list(filter(lambda x: str(x[2]) == 'Admin', usuarios))
25 nombres_todos = list(map(itemgetter(0), usuarios))
26 nombres_admins = list(map(itemgetter(0), admins))
27
```

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.

```
27
28 # aquí tenemos el bucle para iniciar sesión
29 inicio_sesion = 'No'
30 if usuario in nombres_todos:
31     if usuario in nombres_admins:
32         usuario_encontrado = [usuario, password, 'Admin']
33         if usuario_encontrado in usuarios:
34             print('Inicio de sesión exitoso, BIENVENIDO.')
35             inicio_sesion = 'Si'
36         else:
37             print('Contraseña incorrecta. Programa finalizado. Adiós.')
38     else:
39         print('Usuario indentificado, pero no es Administrador. Programa Finalizado. Adiós.')
40 else:
41     print('No Existe Usuario. Programa Finalizado. Adiós.')
42     opcion = 0
43
44 # Listado de Productos de más búsquedas, nos enfocamos en nombres de productos para hacer una lista y las búsquedas 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.
45 busquedas_producto = list(map(itemgetter(1), searches))
46 nombres_productos = list(map(itemgetter(1), products))
47
48 categorias_productos = list(map(itemgetter(3), products))
49 categorias_unicas = np.unique(categorias_productos)
50 categorias_unicas = list(map(lambda el: [el], categorias_unicas)) # separar en lista de listas
51
52 # 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
53 busquedas = dict(Counter(busquedas_producto))
54 busquedas_copia = busquedas.copy()
55 busquedas_categoria = []
56
57 # Este es el ciclo mencionado, cambiamos las llaves de id por el nombre del producto
58 for busqueda in busquedas:
59     busqueda_nombre = nombres_productos[int(busqueda)]
60     llave_nueva = busqueda_nombre
61     llave_vieja = busqueda
62     busquedas_copia[llave_nueva] = busquedas_copia.pop(llave_vieja)
63
```

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.

```
92 # 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
93 # menos vendidos.
94 for venta in compras_ordenadas[::-1]:
95     for producto in productos:
96         if producto[3] == 'audifonos':
97             categorias_unicas_ventas[0].append(venta)
98         elif producto[3] == 'laptops':
99             categorias_unicas_ventas[1].append(venta)
100         elif producto[3] == 'discos duros':
101             categorias_unicas_ventas[2].append(venta)
102         elif producto[3] == 'memorias usb':
103             categorias_unicas_ventas[3].append(venta)
104         elif producto[3] == 'pantallas':
105             categorias_unicas_ventas[4].append(venta)
106         elif producto[3] == 'procesadores':
107             categorias_unicas_ventas[5].append(venta)
108         elif producto[3] == 'tarjetas de video':
109             categorias_unicas_ventas[6].append(venta)
110         elif producto[3] == 'tarjetas madre':
111             categorias_unicas_ventas[7].append(venta)
112     break;
113
114 # 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
115 categorias_unicas_busquedas = categorias_unicas.copy()
116 for busqueda in busquedas_ordenadas[::-1]:
117     for producto in productos:
118         if producto[3] == 'audifonos':
119             categorias_unicas_busquedas[0].append(busqueda)
120         elif producto[3] == 'laptops':
121             categorias_unicas_busquedas[1].append(busqueda)
122         elif producto[3] == 'discos duros':
123             categorias_unicas_busquedas[2].append(busqueda)
124         elif producto[3] == 'memorias usb':
125             categorias_unicas_busquedas[3].append(busqueda)
126         elif producto[3] == 'pantallas':
127             categorias_unicas_busquedas[4].append(busqueda)
128         elif producto[3] == 'procesadores':
129             categorias_unicas_busquedas[5].append(busqueda)
130         elif producto[3] == 'tarjetas de video':
131             categorias_unicas_busquedas[6].append(busqueda)
132         elif producto[3] == 'tarjetas madre':
133             categorias_unicas_busquedas[7].append(busqueda)
134     break;
135
```

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 itemgetter 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

# Meses 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

```
# Meses ingresos, ventas y anual, hacemos una lista 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 división 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]])
```

```

4 Algui terminó el curso pero con algunas pocas faltas por cancelar todas las observaciones en el código, eliminando en su momento a una extra para salir del programa y finalizando. Hemos creado files para el desarrollo con los temas que cubren el producto y la cantidad, a por categorías para facilitar las impresiones, a que muestra las listas de los 28 productos por ejemplo. Para la opción de los ventas a
5 requiere tener en cuenta algunas cosas como el total de ventas a por categorías que lo de mejor forma a la tabla y a un formato de impresión para facilitar el valor de los precios totales muestra.
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
10
```

[illegible]

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

<https://projecto-01-sevilla-nodo.nagosevilla.tepti.ru/>

Inicio de sesión exitoso, BIENVENIDO.

Ingrese lo que quiera analizar:

- 1: Listado de Productos con mayores ventas
 - 2: Listado de productos con mayores búsquedas
 - 3: Listado de productos con menores ventas separados por categoría.
 - 4: Listado de productos con menores búsquedas separados por categoría.
 - 5: Listado de 20 productos con peores calificaciones de reseñas - tomando en cuenta devoluciones
 - 6: Listado de 20 productos con mejores calificaciones de reseñas - tomando en cuenta devoluciones
 - 7: Total de ingresos, ventas promedio y ventas totales mensuales & ventas anuales
 - 8: Salir
- Respuesta: █

CATEGORÍA: discos duros

Producto: "SSD Kingston UV500, 480GB, SATA III, mSATA." Buscado: 1 vez.
Producto: "SSD XPG SX8200 Pro, 256GB, PCI Express, M.2." Buscado: 1 vez.
Producto: "SSD Addlink Technology S70, 512GB, PCI Express 3.0, M.2." Buscado: 2 veces.
Producto: "SSD Western Digital WD Blue 3D NAND, 2TB, M.2." Buscado: 3 veces.
Producto: "SSD Crucial MX500, 1TB, SATA III, M.2." Buscado: 3 veces.
Producto: "Kit SSD Kingston KC600, 1TB, SATA III, 2.5, 7mm." Buscado: 9 veces.
Producto: "SSD Kingston A2000 NVMe, 1TB, PCI Express 3.0, M2." Buscado: 11 veces.
Producto: "SSD para Servidor Lenovo Thinksystem S4510, 480GB, SATA III, 2.5'', 7mm." Buscado: 15 veces.
Producto: "SSD para Servidor Supermicro SSD-DM128-SMCMVN1, 128GB, SATA III, mSATA, 6Gbit/s." Buscado: 50 veces.
Producto: "SSD Kingston UV500, 480GB, SATA III, mSATA." Buscado: 1 vez.
Producto: "SSD XPG SX8200 Pro, 256GB, PCI Express, M.2." Buscado: 1 vez.
Producto: "SSD Addlink Technology S70, 512GB, PCI Express 3.0, M.2." Buscado: 2 veces.
Producto: "SSD Western Digital WD Blue 3D NAND, 2TB, M.2." Buscado: 3 veces.
Producto: "SSD Crucial MX500, 1TB, SATA III, M.2." Buscado: 3 veces.
Producto: "Kit SSD Kingston KC600, 1TB, SATA III, 2.5, 7mm." Buscado: 9 veces.
Producto: "SSD Kingston A2000 NVMe, 1TB, PCI Express 3.0, M2." Buscado: 11 veces.
Producto: "SSD para Servidor Lenovo Thinksystem S4510, 480GB, SATA III, 2.5'', 7mm." Buscado: 15 veces.
Producto: "SSD para Servidor Supermicro SSD-DM128-SMCMVN1, 128GB, SATA III, mSATA, 6Gbit/s." Buscado: 50 veces.

CATEGORÍA: memorias usb

Producto: "Kit Memoria RAM Corsair Vengeance LPX DDR4, 2400MHz, 32GB, Non-ECC, CL16." Buscado: 1 vez.
Producto: "Kit Memoria RAM Corsair Vengeance LPX DDR4, 2400MHz, 32GB, Non-ECC, CL16." Buscado: 1 vez.

CATEGORÍA: pantallas

Producto: "Makena Smart TV LED 40S2 40'', Full HD, Widescreen, Negro." Buscado: 1 vez.
Producto: "TV Monitor LED 24T520S-PU 24, HD, Widescreen, HDMI, Negro." Buscado: 1 vez.
Producto: "Makena Smart TV LED 40S2 40'', Full HD, Widescreen, Negro." Buscado: 1 vez.
Producto: "TV Monitor LED 24T520S-PU 24, HD, Widescreen, HDMI, Negro." Buscado: 1 vez.

CATEGORÍA: procesadores

Producto: "Procesador AMD Ryzen 5 3600, S-AM4, 3.60GHz, 32MB L3 Cache, con Disipador Wraith Stealth." Buscado: 2 veces.

Producto de reseñas de: 10. y una cantidad de devolucion(es) de: 0

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 nuestro código desde otros programadores hasta gente que no se dedica a la programación pueda entender su funcionamiento de una manera clara.