

Salvador Garcilita Arguello

Proyecto EMTECH

índice

- Introducción: página 3
- Definición del código: páginas 4-13
- Solución al problema:
- Conclusión

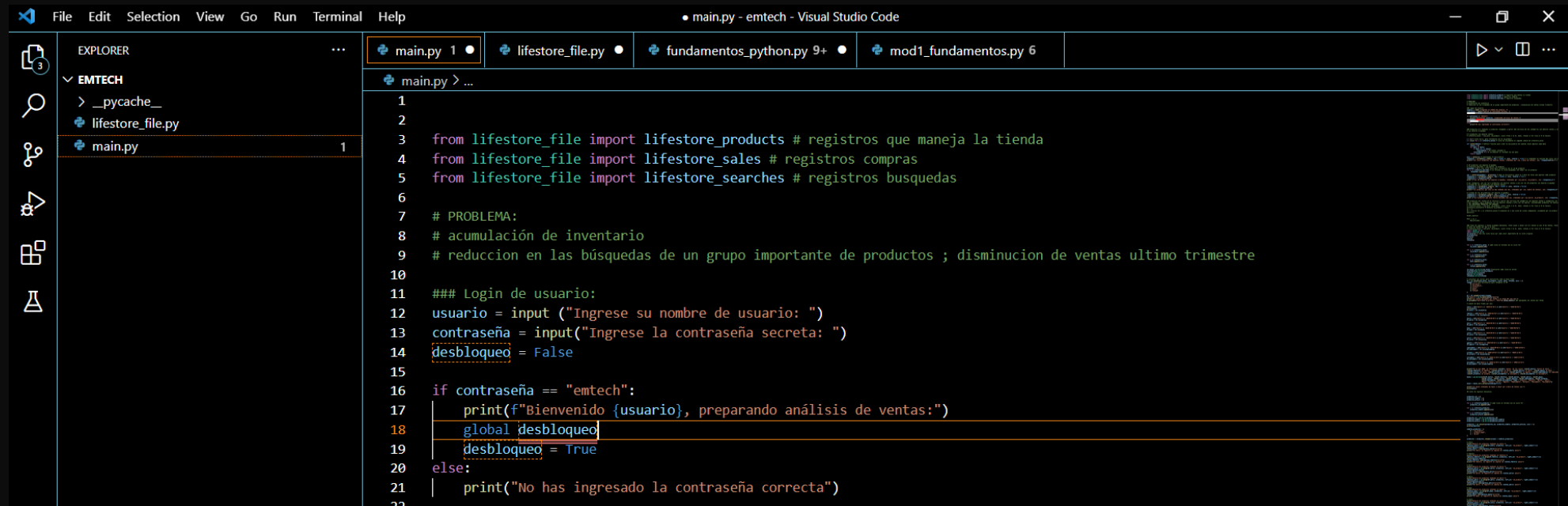
Link a repositorio github: <https://github.com/salva17ga?tab=repositories>

Introducción

- La programación es fundamental para analizar grandes cantidades de datos y automatizar procesos, y dominarla es clave para tener una exitosa inserción a la era digital.
- Para que cualquier organización tenga éxito, es necesario que base la toma de decisiones en el análisis de sus datos.
- Python es un lenguaje de programación de alto nivel, débilmente tipado, interpretado que tiene por característica ser muy flexible en sus áreas de aplicación, además de una lectura y comprensión sencilla.
- Una de sus principales aplicaciones es para la Ciencia de datos, para lo cual cuenta con numerosas librerías como pandas, numpy, etc.

Definición del código

- El programa cuenta con un log in para usuario; es necesario registrarse con su propio nombre y escribir la contraseña “emtech”



```
1
2
3 from lifestore_file import lifestore_products # registros que maneja la tienda
4 from lifestore_file import lifestore_sales # registros compras
5 from lifestore_file import lifestore_searches # registros busquedas
6
7 # PROBLEMA:
8 # acumulación de inventario
9 # reduccion en las búsquedas de un grupo importante de productos ; disminucion de ventas ultimo trimestre
10
11 ### Login de usuario:
12 usuario = input ("Ingrese su nombre de usuario: ")
13 contraseña = input("Ingrese la contraseña secreta: ")
14 desbloqueo = False
15
16 if contraseña == "emtech":
17     print(f"Bienvenido {usuario}, preparando análisis de ventas:")
18     global desbloqueo
19     desbloqueo = True
20 else:
21     print("No has ingresado la contraseña correcta")
22
```

Definición del código

- Para obtener los productos más vendidos, se extrajo de la lista el índice de interés y se creó un diccionario para contar la frecuencia de aparición de cada elemento.

```
31 x = [i[1] for i in lifestore_sales] # lista de unicamente el segundo indice de lifestore_sales
32
33 def contar(datos): # definir función para crear un diccionario de cuantas veces aparece cada dato
34     result = {}
35     for dato in datos:
36         if dato not in result:
37             result[dato] = 0 # meter contador 0
38             result[dato] += 1 # incrementar el contador de ese dato
39     return result
40
41 dicc = contar(x) # almacenamos el diccionario
42 respuesta = sorted(dicc.items(), key = lambda x: x[1], reverse = True) # lo ordenamos en función del valor con ayuda de una función
43 print(f"Los cinco productos con mayores ventas, ordenados por (id, número de ventas), son: {respuesta[0:4]}") # con un slicing sele
44
45
46 # 10 productos con mayores búsquedas
47 # lifestore_searches = [id_search, id_product]
```

```
>>> respuesta = sorted(dicc.items(), key = lambda x: x[1], reverse = True) # lo ordenamos en función del valor con ayuda de una función lambda
os el top 5
Los cinco productos con mayores ventas, ordenados por (id, número de ventas), son: [(54, 50), (3, 42), (5, 20), (42, 18)]
>>> def contar(datos): # definir función para crear un diccionario de cuantas veces aparece cada dato
...     result = {}
...     for dato in datos:
...         if dato not in result:
...             result[dato] = 0 # meter contador 0
...             result[dato] += 1 # incrementar el contador de ese dato
...     return result
...
>>> dicc = contar(x) # almacenamos el diccionario
>>> respuesta = sorted(dicc.items(), key = lambda x: x[1], reverse = True) # lo ordenamos en función del valor con ayuda de una función lambda
os el top 5
Los cinco productos con mayores ventas, ordenados por (id, número de ventas), son: [(54, 50), (3, 42), (5, 20), (42, 18)]
>>>
```

Terminal Help • main.py - emtech - Visual Studio Code

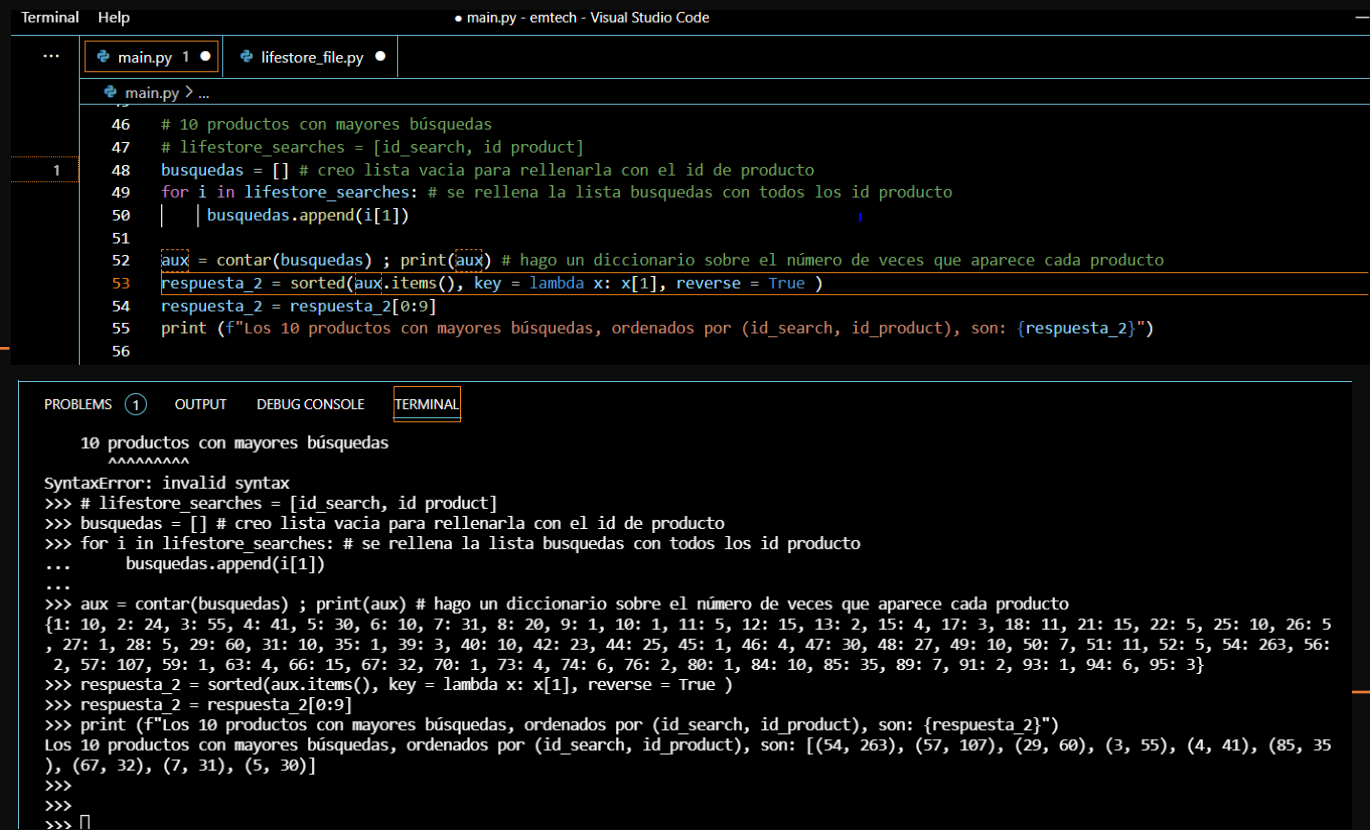
main.py 1 • lifestore_file.py

main.py > ...

```
21 | print("No has ingresado la contraseña correcta")
22 |
23 |
24 | ### Productos más vendidos y productos rezagados a partir del análisis de las categorías con menores
25 | # con menores búsquedas
26 |
27 | # 5 productos con mayores ventas:
28 | # lifestore_sales = [id_sale, id_product, score (from 1 to 5), date, refund (1 for true or 0 to fals
29 |
30 | x = [] # lista vacía, para rellenarse con el id_product
31 | x = [i[1] for i in lifestore_sales] # lista de unicamente el segundo indice de lifestore_sales
32 |
33 | def contar(datos): # definir función para crear un diccionario de cuantas veces aparece cada dato
34 |     result = {}
35 |     for dato in datos:
36 |         if dato not in result:
37 |             result[dato] = 0 # meter contador 0
38 |             result[dato] += 1 # incrementar el contador de ese dato
39 |     return result
40 |
41 | dicc = contar(x) # almacenamos el diccionario
42 | respuesta = sorted(dicc.items(), key = lambda x: x[1], reverse = True) # lo ordenamos en función del
43 | print(f"Los cinco productos con mayores ventas, ordenados por (id, número de ventas), son: {respuest
44 |
```

Definición del código

- Para obtener los 10 productos con mayores búsquedas, se realizó un proceso similar



The image shows a Visual Studio Code window with a Python file named `main.py` and a terminal window below it.

main.py

```
46 # 10 productos con mayores búsquedas
47 # lifestore_searches = [id_search, id product]
48 busquedas = [] # creo lista vacia para rellenarla con el id de producto
49 for i in lifestore_searches: # se rellena la lista busquedas con todos los id producto
50     | busquedas.append(i[1])
51
52 aux = contar(busquedas) ; print(aux) # hago un diccionario sobre el número de veces que aparece cada producto
53 respuesta_2 = sorted(aux.items(), key = lambda x: x[1], reverse = True )
54 respuesta_2 = respuesta_2[0:9]
55 print (f"Los 10 productos con mayores búsquedas, ordenados por (id_search, id_product), son: {respuesta_2}")
56
```

TERMINAL

```
10 productos con mayores búsquedas
^^^^^^^^
SyntaxError: invalid syntax
>>> # lifestore_searches = [id_search, id product]
>>> busquedas = [] # creo lista vacia para rellenarla con el id de producto
>>> for i in lifestore_searches: # se rellena la lista busquedas con todos los id producto
...     busquedas.append(i[1])
...
>>> aux = contar(busquedas) ; print(aux) # hago un diccionario sobre el número de veces que aparece cada producto
{1: 10, 2: 24, 3: 55, 4: 41, 5: 30, 6: 10, 7: 31, 8: 20, 9: 1, 10: 1, 11: 5, 12: 15, 13: 2, 15: 4, 17: 3, 18: 11, 21: 15, 22: 5, 25: 10, 26: 5, 27: 1, 28: 5, 29: 60, 31: 10, 35: 1, 39: 3, 40: 10, 42: 23, 44: 25, 45: 1, 46: 4, 47: 30, 48: 27, 49: 10, 50: 7, 51: 11, 52: 5, 54: 263, 56: 2, 57: 107, 59: 1, 63: 4, 66: 15, 67: 32, 70: 1, 73: 4, 74: 6, 76: 2, 80: 1, 84: 10, 85: 35, 89: 7, 91: 2, 93: 1, 94: 6, 95: 3}
>>> respuesta_2 = sorted(aux.items(), key = lambda x: x[1], reverse = True )
>>> respuesta_2 = respuesta_2[0:9]
>>> print (f"Los 10 productos con mayores búsquedas, ordenados por (id_search, id_product), son: {respuesta_2}")
Los 10 productos con mayores búsquedas, ordenados por (id_search, id_product), son: [(54, 263), (57, 107), (29, 60), (3, 55), (4, 41), (85, 35), (67, 32), (7, 31), (5, 30)]
>>>
>>>
>>> □
```

Definición del código

- Aprovechando los objetos creados (diccionario) se ordenaron con ayuda de `sorted` y una función `lambda` para obtener la información inversa
-

```
56
57 # por categoria, uno con los 5 productos con menores ventas y uno con los 10 productos con menores búsquedas
58 # listado de los 5 productos con menores ventas:
59 respuesta_3 = sorted(dicc.items(), key = lambda x: x[1], reverse = False)
60 respuesta_3 = respuesta_3[0:15]
61 print(f"los productos que sólo se han vendido una vez, ordenados por (id, numero de ventas), son: {respuesta_3}")
62
63 # listado de los 10 productos con menores búsquedas:
64 respuesta_4 = sorted(aux.items(), key = lambda x: x[1], reverse = False)
65 respuesta_4 = respuesta_4[0:9] ; print(respuesta_4)
66 print (f"Los productos que sólo fueron buscados una vez, ordenados por (id_search, id_product), son: {respuesta_4}")
67
68
```

```
>>> respuesta_3 = sorted(dicc.items(), key = lambda x: x[1], reverse = False)
>>> respuesta_3 = respuesta_3[0:15]
>>> print(f"los productos que sólo se han vendido una vez, ordenados por (id, numero de ventas), son: {respuesta_3}")
los productos que sólo se han vendido una vez, ordenados por (id, numero de ventas), son: [(10, 1), (13, 1), (17, 1), (22, 1), (28, 1), (40, 1), (45, 1), (46, 1), (50, 1), (60, 1), (66, 1), (67, 1), (84, 1), (89, 1), (94, 1)]
>>> # listado de los 10 productos con menores búsquedas:
>>>
>>> respuesta_4 = sorted(aux.items(), key = lambda x: x[1], reverse = False)
>>>
>>> respuesta_4 = respuesta_4[0:9] ; print(respuesta_4)
[(9, 1), (10, 1), (27, 1), (35, 1), (45, 1), (59, 1), (70, 1), (80, 1), (93, 1)]
>>> print (f"Los productos que sólo fueron buscados una vez, ordenados por (id_search, id_product), son: {respuesta_4}")
Los productos que sólo fueron buscados una vez, ordenados por (id_search, id_product), son: [(9, 1), (10, 1), (27, 1), (35, 1), (45, 1), (59, 1), (70, 1), (80, 1), (93, 1)]
>>>
```

Definición del código

- Para obtener los datos por mes y año, se utilizaron los paquetes pandas y datetime.
- Primero, se convirtió en serie a cada lista de datos de interés

```
69
70 ### total de ingresos y ventas promedio mensuales, total anual y meses con más ventas al año (# de ventas, total de ingresos)
71 # total de ventas por mes y al año:
72 # lifestore_sales = [id_sale, id_product, score (from 1 to 5), date, refund (1 for true or 0 to false)]
73 import pandas as pd
74 import datetime as dt
75 id_sales=[] # creo una lista vacia por cada valor importante de la lista original
76 id_product=[]
77 score=[]
78 date=[]
79 refund=[]
80
81 for i in lifestore_sales: # cada lista se rellena con un ciclo for
82     id_sales.append(i[0])
83
84 for i in lifestore_sales:
85     id_product.append(i[1])
86
87 for i in lifestore_sales:
88     score.append(i[2])
89
90 for i in lifestore_sales:
91     date.append(i[3])
92
93 for i in lifestore_sales:
94     refund.append(i[4])
95
96 id_sales= pd.Series(id_sales) #convierto cada lista en series
97 id_product=pd.Series(id_product)
98 score=pd.Series(score)
99 date=pd.Series(date)
100 refund=pd.Series(refund)
101
```


Definición del código

- Posteriormente, se hizo una agrupación con concat para las series y poderlas trabajar como data frame
 - Tras esto, se hicieron subtablas por cada mes
-

```
102 # concateno las series para analizarlas como un data frame:
103 df = pd.concat([id_sales, id_product, score, date, refund], axis = 1)
104 rename = { # creo diccionario para renombrar el DF
105     0:"id_sales",
106     1:"id_product",
107     2:"score",
108     3:"date",
109     4:"refund"
110 }
111
112 df = df.rename(columns=rename)
113 df["date"] = pd.to_datetime(df["date"])
114 print("Las ventas agrupadas por fecha a lo largo del año son:")
115 df.groupby(["date"])[["id_product", "date"]].value_counts() #df agrupando las ventas por fecha
116
117 # subset de data frames por mes:
118
119 enero = (df["date"] >= "2020-01-01") & (df["date"] < "2020-02-01")
120 print(enero)
121 df_enero = df.loc[enero]
122
123 febrero = (df["date"] >= "2020-02-01") & (df["date"] < "2020-03-01")
124 df_febrero = df.loc[febrero]
125
126 marzo = (df["date"] >= "2020-03-01") & (df["date"] < "2020-04-01")
127 df_marzo = df.loc[marzo]
128
129 abril = (df["date"] >= "2020-04-01") & (df["date"] < "2020-05-01")
130 df_abril = df.loc[abril]
131
132 mayo = (df["date"] >= "2020-05-01") & (df["date"] < "2020-06-01")
133 df_mayo = df.loc[mayo]
134
135 junio = (df["date"] >= "2020-06-01") & (df["date"] < "2020-07-01")
```

Definición del código

- Con estas subtablas, se realizó el análisis descriptivo, que se imprime en consola con ayuda de f strings.

```
print(f"En el año 2020, se realizaron {len(df)} ventas, de las cuales {len(df_enero)} fueron en enero, \
{len(df_febrero)} en febrero, {len(df_marzo)} en marzo, {len(df_abril)} en abril, {len(df_mayo)} en mayo, \
{len(df_junio)} en junio, {len(df_julio)} en julio, {len(df_agosto)} en agosto, {len(df_septiembre)} en septiembre, \
{len(df_octubre)} en octubre, {len(df_noviembre)} en noviembre y {len(df_diciembre)} en diciembre")
```

```
meses = pd.Series([len(df_enero), len(df_febrero), len(df_marzo), len(df_abril), len(df_mayo),
len(df_junio), len(df_julio), len(df_agosto), len(df_septiembre), len(df_octubre),
len(df_noviembre), len(df_diciembre)], index = ["Enero", "Febrero", "Marzo", "Abril",
"Mayo", "Junio", "Julio", "Agosto", "Septiembre", "Octubre", "Noviembre", "Diciembre"])
meses = meses.sort_values(ascending=False)
```

```
print("Los meses ordenados de mayor a menor por número de ventas son:")
print(meses)
```

```
En el año 2020, se realizaron 283 ventas, de las cuales 39 fueron en enero, 38 en febrero, 42 en marzo, 72 en abril, 20 en mayo, 16 en junio,
14 en julio, 5 en agosto, 14 en septiembre, 9 en octubre, 10 en noviembre y 3 en diciembre
```

```
>>>
>>> print("Los meses ordenados de mayor a menor por número de ventas son:")
Los meses ordenados de mayor a menor por número de ventas son:
>>> print(meses)
Abril      72
Marzo      42
Enero      39
Febrero    38
Mayo       20
Junio      16
Julio      14
Septiembre 14
Noviembre  10
Octubre     9
Agosto     5
Diciembre   3
dtype: int64
>>> []
```

Definición del código

- Finalmente, para obtener el total de ingresos mensuales, se realizó un data frame ahora con `lifestore_products` siguiendo un proceso similar al anterior
 - Finalmente, se hizo uso de la función `merge` para hacer joins del data frame de ventas de cada mes con el data frame de nombre y precios de cada producto
-

```
## total de ingresos mensuales:
productos_id = []
productos_nombre = []
productos_precio = []

for i in lifestore_products: # cada lista se rellena con un ciclo for
    productos_id.append(i[0])

for i in lifestore_products:
    productos_nombre.append(i[1])

for i in lifestore_products:
    productos_precio.append(i[2])

productos_id = pd.Series(productos_id)
productos_nombre = pd.Series(productos_nombre)
productos_precio = pd.Series(productos_precio)

productos = pd.concat([productos_id, productos_nombre, productos_precio], axis = 1)
print(productos)

nombres_productos = {
    0 : "id_product",
    1 : "product_name",
    2 : "price"
}

productos = productos.rename(columns = nombres_productos)

# genera
```

Definición del código

- Por ultimo, se aplicaron fstrings y funciones de agregación para imprimir la información deseada por cada mes

```
# enero
print("Detalle de productos vendidos en enero:")
ingresos_enero = pd.merge(df_enero, productos, left_on= "id_product", right_index=True)
print(ingresos_enero)
ventas_enero = sum(ingresos_enero["price"])
print(f"En enero, se registró un ingreso de {ventas_enero} pesos")

# febrero
print("Detalle de productos vendidos en febrero:")
ingresos_febrero = pd.merge(df_febrero, productos, left_on= "id_product", right_index=True)
print(ingresos_febrero)
ventas_febrero= sum(ingresos_febrero["price"])
print(f"En febrero, se registró un ingreso de {ventas_febrero} pesos")

# marzo
print("Detalle de productos vendidos en marzo:")
ingresos_marzo = pd.merge(df_marzo, productos, left_on= "id_product", right_index=True)
print(ingresos_marzo)
ventas_marzo = sum(ingresos_marzo["price"])
print(f"En marzo, se registró un ingreso de {ventas_marzo} pesos")

# abril
print("Detalle de productos vendidos en abril:")
ingresos_abril = pd.merge(df_abril, productos, left_on= "id_product", right_index=True)
print(ingresos_abril)
ventas_abril= sum(ingresos_abril["price"])
print(f"En abril, se registró un ingreso de {ventas_abril} pesos")

# mayo
print("Detalle de productos vendidos en mayo:")
ingresos_mayo = pd.merge(df_mayo, productos, left_on= "id_product", right_index=True)
print(ingresos_mayo)
```

Definición del código

- Vista de output del programa:

```
>>> print("Detalle de productos vendidos en noviembre:")
Detalle de productos vendidos en noviembre:
>>> ingresos_noviembre = pd.merge(df_noviembre, productos, left_on= "id_product", right_index=True)
>>> print(ingresos_noviembre)
```

	id_product	id_sales	id_product_x	score	date	refund	id_product_y	product_name	price
17	3	18	3	5	2020-11-06	0	4	Procesador AMD Ryzen 3 3200G con Gráficos Rade...	2209
18	3	19	3	5	2020-11-06	0	4	Procesador AMD Ryzen 3 3200G con Gráficos Rade...	2209
36	3	37	3	5	2020-11-03	0	4	Procesador AMD Ryzen 3 3200G con Gráficos Rade...	2209
48	3	49	3	5	2020-11-01	0	4	Procesador AMD Ryzen 3 3200G con Gráficos Rade...	2209
68	4	69	4	4	2020-11-01	0	5	Procesador Intel Core i3-9100F, S-1151, 3.60GH...	1779
149	31	150	31	1	2020-11-01	0	32	Tarjeta Madre ASRock Z390 Phantom Gaming 4, S-...	4309
172	44	173	44	5	2020-11-04	0	45	Tarjeta Madre ASRock ATX H110 Pro BTC+, S-1151...	2869
177	45	178	45	1	2020-11-02	1	46	Tarjeta Madre Gigabyte micro ATX GA-H110M-DS2,...	1539
187	47	188	47	3	2020-11-02	0	48	SSD Kingston A2000 NVMe, 1TB, PCI Express 3.0, M2	2559
215	54	216	54	5	2020-11-05	0	55	SSD para Servidor Supermicro SSD-DM128-SMCMVN1...	4399

```
>>> ventas_noviembre= sum(ingresos_noviembre["price"])
>>> print(f"En noviembre, se registró un ingreso de {ventas_noviembre} pesos")
En noviembre, se registró un ingreso de 26290 pesos
... □
```

Solución al problema

- Lifestore Company debe adaptar su oferta de productos; posiblemente sea útil un estudio de segmentación de mercado, para entender por qué ciertos productos se venden mucho y otros casi no se venden.
- Por el tipo de artículos más vendidos, deberían aumentar en oferta procesadores y discos duros.
- Los meses de junio a diciembre tienen ventas demasiado bajas en comparación a los demás; una estrategia podría ser aumentar la publicidad y promociones.
- Agrupar los productos por grupos funcionales (categorías de mercado) para poder identificar que segmentos de productos venden más y cuales menos, y de qué forma podrían añadirse nuevas de forma estratégica.

Conclusión

- La versatilidad de objetos con los que Python ofrece trabajar aumenta las posibilidades de crear soluciones con nuestro código.
- Las listas son muy adaptables y útiles para trabajar con ellas, pues son mutables, iterables e indexables. No obstante, para procesar datos y analizarlos, se recomiendan otras opciones.
- Cada empresa debería tener un equipo de científicos de datos que trabajen en convertir datos en información para tomar las mejores decisiones posibles.