

Universidad Americana

Facultad de Ingeniería y Arquitectura



Algoritmos y Estructuras de Datos, Grupo 4

Manejo de Funciones, Clases y Paquetes

Realizado por:

Joaquín Alberto Pérez Zúñiga

Nicolás Rafael Laguna Vallejos

Docente:

MSc. César Marín López

Ejercicio #1

Código:

```
> nu      ej1.py  U X
semana-3 > ejercicios_funciones-oop > ej1.py > ...
1  """
2  Desarrollar un programa que procese un conjunto de registros de ventas
3  (por ejemplo, listas de diccionarios) para extraer información relevante.
4  Los estudiantes deberán aplicar funciones internas como map, filter y reduce
5  para transformar y filtrar los datos, calculando totales y promedios.
6  """
7
8  from functools import reduce
9
10
11 def ingresar_ventas():
12     """
13     Permite al usuario ingresar registros de ventas.
14     """
15
16     ventas = []
17
18     print("\n=== INGRESO DE DATOS DE VENTAS ===")
19     print("(Dejar el producto en blanco para terminar)")
20
21     while True:
22         producto = input("\nProducto: ")
23         if not producto:
24             break
25
26         try:
27             precio = float(input("Precio: $"))
28             cantidad = int(input("Cantidad: "))
29             vendedor = input("Vendedor: ")
30
31             ventas.append(
32                 {
33                     "producto": producto,
34                     "precio": precio,
35                     "cantidad": cantidad,
36                     "vendedor": vendedor,
37                 }
38             )
39         except ValueError:
40             print("\nError: Ingrese valores numéricos válidos.\n")
41
42     return ventas
43
```

> nu

ej1.py U X

semana-3 > ejercicios_funciones-oop > ej1.py > ...

```
45 def analizar_ventas(ventas):
46     """
47     ... Analiza los datos de ventas usando map, filter y reduce.
48     ... """
49
50     ... # Calcular subtotales con map
51     ... subtotales = list(map(lambda v: v["precio"] * v["cantidad"], ventas))
52
53     ... # Calcular total con reduce
54     ... total = reduce(lambda x, y: x + y, subtotales, 0)
55
56     ... # Calcular ventas por vendedor con filter y map
57     ... vendedores = set(map(lambda v: v["vendedor"], ventas))
58     ... ventas_por_vendedor = {}
59
60     ... for vendedor in vendedores:
61     ...     ventas_vendedor = list(filter(lambda v: v["vendedor"] == vendedor, ventas))
62     ...     ventas_por_vendedor[vendedor] = sum(
63     ...         map(lambda v: v["precio"] * v["cantidad"], ventas_vendedor)
64     ...     )
65
66     ... return subtotales, total, ventas_por_vendedor
67
68
69 def main():
70     """
71     ... Ejecución del programa.
72     ... """
73
74     ... # Ingreso de datos
75     ... ventas = ingresar_ventas()
76
77     ... if not ventas:
78     ...     print("\nNo se ingresaron datos para analizar.")
79     ...     return
80
81     ... # Análisis de datos
82     ... subtotales, total, ventas_por_vendedor = analizar_ventas(ventas)
83
84     ... # Mostrar resultados
85     ... print("\n=== RESULTADOS DEL ANÁLISIS ===")
86
87     ... print("\n1. Registros ingresados:")
88     ... for i, v in enumerate(ventas, 1):
89     ...     print(
90     ...         f"{i}. {v['producto']}: ${v['precio']} x {v['cantidad']}" +
91     ...         f" - Vendedor: {v['vendedor']}"
92     ...     )
```

> nu

ej1.py U X

semana-3 > ejercicios_funciones-oop > ej1.py > ...

```
69     def main():
84         ...# Mostrar resultados
85         print("\n=== RESULTADOS DEL ANÁLISIS ===")
86
87         print("\n1. Registros ingresados:")
88         for i, v in enumerate(ventas, 1):
89             print(
90                 f"    {i}. {v['producto']}: ${v['precio']} x {v['cantidad']}" +
91                 f"    - Vendedor: {v['vendedor']}"
92             )
93
94         print("\n2. Subtotales por venta:")
95         for i, (v, subtotal) in enumerate(zip(ventas, subtotales), 1):
96             print(f"    {i}. {v['producto']}: ${subtotal:.2f}")
97
98         print(f"\n3. Total de ventas: ${total:.2f}")
99
100        print("\n4. Ventas por vendedor:")
101        for vendedor, monto in ventas_por_vendedor.items():
102            print(f"    - {vendedor}: ${monto:.2f}")
103
104
105    if __name__ == "__main__":
106        main()
107
```

Output:

```
> nu      X  ej1.py  U
● C:\dev\algo-structs> python semana-3\ejercicios_funciones-oop\ej1.py

=== INGRESO DE DATOS DE VENTAS ===
(Dejar el producto en blanco para terminar)

Producto: Calculadora
Precio: $123.4
Cantidad: 3
Vendedor: Juan

Producto: Lapicero
Precio: $34.5
Cantidad: 5
Vendedor: Juan

Producto: Laptop
Precio: $2345.6
Cantidad: 1
Vendedor: Juanita

Producto:

=== RESULTADOS DEL ANÁLISIS ===

1. Registros ingresados:
  1. Calculadora: $123.4 x 3 - Vendedor: Juan
  2. Lapicero: $34.5 x 5 - Vendedor: Juan
  3. Laptop: $2345.6 x 1 - Vendedor: Juanita

2. Subtotales por venta:
  1. Calculadora: $370.20
  2. Lapicero: $172.50
  3. Laptop: $2345.60

3. Total de ventas: $2888.30

4. Ventas por vendedor:
  - Juanita: $2345.60
  - Juan: $542.70
○ C:\dev\algo-structs> █
```

Ejercicio #2

1. main.py:

```
> nu      main.py U X  ordenamiento.py U
semana-3 > ejercicios_funciones-oop > ej2 > main.py > fx main
1  """
2  Se requiere que los estudiantes diseñen un módulo independiente que
3  contenga implementaciones de algoritmos de ordenamiento simples (bubble sort).
4  A partir de una función principal, se deben invocar los métodos del
5  módulo para ordenar una lista de números.
6  """
7
8  from os import system
9  from random import randint
10
11  from ordenamiento import (
12  |...bubble_sort,
13  |...print_paso_a_paso
14  |)
15
16
17  def ingresar_lista():
18  |..."""
19  |...Permite al usuario ingresar una lista de números.
20  |..."""
21
22  |...lista = []
23  |...print("\nIngrese los números de la lista (deje en blanco para terminar):")
24
25  |...while True:
26  |...|...entrada = input("Número (o 'Enter' para terminar): ")
27  |...|...if entrada == "":
28  |...|...|...break
29
30  |...|...try:
31  |...|...|...numero = float(entrada)
32  |...|...|...lista.append(numero)
33  |...|...except ValueError:
34  |...|...|...print("\nError: Ingrese un número válido.\n")
35
36  |...return lista
37
38
```


> nu

main.py U X

ordenamiento.py U

semana-3 > ejercicios_funciones-oop > ej2 > main.py > fx main

```
39 def generar_lista_aleatoria():
40     """
41     ... Genera una lista de números aleatorios.
42     ... """
43
44     try:
45         cantidad = int(input("\n¿Cuántos números desea generar? "))
46         minimo = int(input("Valor mínimo: "))
47         maximo = int(input("Valor máximo: "))
48
49         if cantidad <= 0:
50             input("\nError: La cantidad debe ser mayor a cero.")
51             system("cls || clear")
52             return generar_lista_aleatoria()
53
54         if minimo > maximo:
55             minimo, maximo = maximo, minimo
56
57         return [randint(minimo, maximo) for _ in range(cantidad)]
58     except ValueError:
59         input("Error: Ingrese valores numéricos válidos.")
60         return generar_lista_aleatoria()
61
62
63 def mostrar_menu():
64     """
65     ... Muestra el menú principal.
66     ... """
67
68     print("\n=== PROGRAMA DE ORDENAMIENTO ===")
69     print("1. Ingresar lista manualmente")
70     print("2. Generar lista aleatoria")
71     print("3. Salir")
72
73     return input("\nSeleccione una opción: ")
74
75
```

> nu

main.py U X

ordenamiento.py U

semana-3 > ejercicios_funciones-oop > ej2 > main.py > ...

```
76 def main():
77     """
78     Función principal del programa.
79     """
80
81     lista = []
82     opcion = mostrar_menu()
83
84     while True:
85         system("cls || clear")
86
87         if opcion == "1":
88             system("cls || clear")
89             print("=== INGRESO MANUAL DE DATOS ===")
90             lista = ingresar_lista()
91         elif opcion == "2":
92             system("cls || clear")
93             print("=== GENERACIÓN DE LISTA ALEATORIA ===")
94             lista = generar_lista_aleatoria()
95         elif opcion == "3":
96             input("\n¡Gracias por utilizar el programa de ordenamiento!")
97             break
98         else:
99             input("\nOpción inválida. Intente nuevamente.")
100             continue
101
102         if not lista:
103             input("\nNo hay datos para ordenar.")
104             continue
105
106         system("cls || clear")
107         print("=== RESULTADOS DE ORDENAMIENTO ===")
108         print("\nLista original:", lista)
109
110         # Usamos el módulo de ordenamiento
111         lista_ordenada, iteraciones = bubble_sort(lista)
112
113         print("\nLista ordenada: ", lista_ordenada)
114         print(f"Ordenamiento completado en {iteraciones} iteraciones.")
115
116         ver_pasos = input("\n¿Desea ver el ordenamiento paso a paso? (s/n): ").lower()
117         if ver_pasos == "s":
118             system("cls || clear")
119             print("=== ORDENAMIENTO PASO A PASO ===")
120             print_paso_a_paso(lista)
121
122             input("\nPresione 'Enter' para regresar al menú...")
123             system("cls || clear")
```

2. ordenamiento.py

```
> nu main.py U ordenamiento.py U X
semana-3 > ejercicios_funciones-oop > ej2 > ordenamiento.py > ...
1  """
2  Módulo que implementa algoritmos de ordenamiento simples.
3  """
4
5
6  def bubble_sort(lista):
7      """
8      ... Implementación del algoritmo Bubble Sort.
9      ... Ordena una lista comparando elementos adyacentes y
10     ... realizando intercambios hasta que la lista esté ordenada.
11     ... """
12
13     ... # Trabajamos con una copia para no modificar la original
14     ... resultado = lista.copy()
15     ... n = len(resultado)
16
17     ... intercambio_realizado = True
18     ... iteraciones = 0
19
20     ... # Realizamos n - 1 pasadas como máximo
21     ... for i in range(n - 1):
22     ...     ... # Si no hubo intercambios en la pasada anterior, la lista ya está ordenada
23     ...     ... if not intercambio_realizado:
24     ...     ...     break
25
26     ...     ... intercambio_realizado = False
27     ...     ... iteraciones += 1
28
29     ...     ... # En cada pasada, comparamos elementos adyacentes
30     ...     ... for j in range(n - i - 1):
31     ...     ...     ... # Si el elemento actual es mayor que el siguiente, los intercambiamos
32     ...     ...     ... if resultado[j] > resultado[j + 1]:
33     ...     ...     ...     resultado[j], resultado[j + 1] = resultado[j + 1], resultado[j]
34     ...     ...     ... intercambio_realizado = True
35
36     ... return resultado, iteraciones
37
```

> nu

main.py U

ordenamiento.py U X

semana-3 > ejercicios_funciones-oop > ej2 > ordenamiento.py > ...

```
39 def print_paso_a_paso(lista):
40     """
41     Muestra la ejecución paso a paso del algoritmo bubble sort.
42     """
43
44     resultado = lista.copy()
45     n = len(resultado)
46
47     print("\nEstado inicial: ", resultado)
48
49     # Seguimiento de pasos
50     paso = 1
51
52     for i in range(n - 1):
53         hubo_cambios = False
54
55         print(f"\nPasada #{i + 1}:")
56
57         for j in range(n - i - 1):
58             print(f" -> Paso #{paso}: Comparando {resultado[j]} y {resultado[j + 1]}")
59
60             if resultado[j] > resultado[j + 1]:
61                 resultado[j], resultado[j + 1] = resultado[j + 1], resultado[j]
62                 print(f"Intercambio: {resultado}")
63                 hubo_cambios = True
64             else:
65                 print(f"Sin cambios: {resultado}")
66
67             paso += 1
68
69         if not hubo_cambios:
70             print("No se realizaron intercambios. Lista ordenada.")
71             break
72
73     return resultado
74
```

Output:

```
> python × main.py U ordenamiento.py U
● C:\dev\algo-structs> python semana-3\ejercicios_funciones-oop\ej2\main.py

=== PROGRAMA DE ORDENAMIENTO ===
1. Ingresar lista manualmente
2. Generar lista aleatoria
3. Salir

Seleccione una opción: █
```

```
> python × main.py U ordenamiento.py U

=== GENERACIÓN DE LISTA ALEATORIA ===

¿Cuántos números desea generar? 12
Valor mínimo: 1
Valor máximo: 50

Lista generada exitosamente. Presione 'Enter' para ordenar la lista generada. █
```

```
=== RESULTADOS DE ORDENAMIENTO ===

Lista original: [42, 19, 18, 12, 21, 29, 28, 6, 27, 25, 31, 29]
Lista ordenada: [6, 12, 18, 19, 21, 25, 27, 28, 29, 29, 31, 42]

Ordenamiento completado en 8 iteraciones.

¿Desea ver el ordenamiento paso a paso? (s/n): █
```

> python



main.py U

ordenamiento.py U

○ === ORDENAMIENTO PASO A PASO ===

Estado inicial: [20, 12, 32, 30, 44, 42, 45, 30, 30, 23, 5, 27]

Pasada #1:

-> Paso #1: Comparando 20 y 12

Intercambio: [12, 20, 32, 30, 44, 42, 45, 30, 30, 23, 5, 27]

-> Paso #2: Comparando 20 y 32

Sin cambios: [12, 20, 32, 30, 44, 42, 45, 30, 30, 23, 5, 27]

-> Paso #3: Comparando 32 y 30

Intercambio: [12, 20, 30, 32, 44, 42, 45, 30, 30, 23, 5, 27]

-> Paso #4: Comparando 32 y 44

Sin cambios: [12, 20, 30, 32, 44, 42, 45, 30, 30, 23, 5, 27]

-> Paso #5: Comparando 44 y 42

Intercambio: [12, 20, 30, 32, 42, 44, 45, 30, 30, 23, 5, 27]

-> Paso #6: Comparando 44 y 45

Sin cambios: [12, 20, 30, 32, 42, 44, 45, 30, 30, 23, 5, 27]

-> Paso #7: Comparando 45 y 30

Intercambio: [12, 20, 30, 32, 42, 44, 30, 45, 30, 23, 5, 27]

-> Paso #8: Comparando 45 y 30

Intercambio: [12, 20, 30, 32, 42, 44, 30, 30, 45, 23, 5, 27]

-> Paso #9: Comparando 45 y 23

Intercambio: [12, 20, 30, 32, 42, 44, 30, 30, 23, 45, 5, 27]

-> Paso #10: Comparando 45 y 5

Intercambio: [12, 20, 30, 32, 42, 44, 30, 30, 23, 5, 45, 27]

-> Paso #11: Comparando 45 y 27

Intercambio: [12, 20, 30, 32, 42, 44, 30, 30, 23, 5, 27, 45]

Ejercicio #3

1. main.py:

```
> nu      main.py U x  busqueda_binaria.py U  busqueda_lineal.py U
semana-3 > ejercicios_funciones-oop > ej3 > main.py > ...
1  """
2  Desarrollar un programa para organizar diferentes algoritmos de búsqueda
3  en un paquete estructurado. Los estudiantes deberán crear al menos dos
4  módulos que contengan implementaciones de búsqueda lineal y búsqueda
5  binaria, configurando adecuadamente el archivo init.py. Posteriormente,
6  desde un script principal, se utilizarán estas funciones para
7  localizar elementos específicos en una colección de datos.
8  """
9
10 from algoritmos import (
11     busqueda_binaria,
12     busqueda_lineal
13 )
14
15
16 def demostrar_busquedas():
17     """
18     Demostración de algoritmos.
19     """
20
21     ...# Datos de ejemplo
22     numeros_desordenados = [45, 12, 85, 32, 89, 39, 69, 44, 42, 1, 6, 8]
23     numeros_ordenados = sorted(numeros_desordenados)
24
25     print("Lista desordenada:", numeros_desordenados)
26     print("Lista ordenada:", numeros_ordenados)
27     print("-" * 50)
28
29     ...# Elemento a buscar
30     elemento = 42
31
```



```
>- nu      main.py U X      busqueda_binaria.py U      busqueda_lineal.py U
semana-3 > ejercicios_funciones-oop > ej3 > main.py > ...
16  def demostrar_busquedas():
31
32      ...# Demostración de búsqueda lineal
33      ...print(f"Buscando el elemento {elemento} con búsqueda lineal:")
34      ...indice = busqueda_lineal(numeros_desordenados, elemento)
35
36      ...if indice != -1:
37      ...|... print(f"Elemento encontrado en la posición {indice}")
38      ...else:
39      ...|... print("Elemento no encontrado")
40
41      ...# Demostración de búsqueda binaria
42      ...print(f"\nBuscando el elemento {elemento} con búsqueda binaria:")
43
44      ...# Notar que usamos la lista ordenada para la búsqueda binaria
45      ...indice = busqueda_binaria(numeros_ordenados, elemento)
46
47      ...if indice != -1:
48      ...|... print(f"Elemento encontrado en la posición {indice}")
49      ...else:
50      ...|... print("Elemento no encontrado")
51
52      ...# Demostración con un elemento que no existe
53      ...elemento_ausente = 99
54      ...print(f"\nBuscando el elemento {elemento_ausente} que no existe:")
55
56      ...indice_lineal = busqueda_lineal(numeros_desordenados, elemento_ausente)
57      ...print(
58      ...|... "Búsqueda lineal: " +
59      ...|... f'{"No encontrado" if indice_lineal == -1 else f"Posición {indice_lineal}"}
60      ...|... )
61
```

```
>- nu      main.py U X      busqueda_binaria.py U      busqueda_lineal.py U
semana-3 > ejercicios_funciones-oop > ej3 > main.py > ...
16  def demostrar_busquedas():
62      ...indice_binario = busqueda_binaria(numeros_ordenados, elemento_ausente)
63      ...print(
64      ...|... "Búsqueda binaria: " +
65      ...|... f'{"No encontrado" if indice_binario == -1 else f"Posición {indice_binario}"}
66      ...|... )
67
68
69  if __name__ == "__main__":
70      ...demostrar_busquedas()
71
```

2. `__init__.py`:

```
>- nu      main.py U      __init__.py U X      busqueda_binaria.py U      busqueda_lineal.py U
semana-3 > ejercicios_funciones-oop > ej3 > algoritmos > __init__.py > ...
1  """
2  Paquete que implementa algoritmos de búsqueda (lineal y binaria).
3  """
4
5  from .busqueda_lineal import busqueda_lineal
6  from .busqueda_binaria import busqueda_binaria
7
8
9  # Facilita la importación directa de las funciones
10 __all__ = [
11     "busqueda_lineal",
12     "busqueda_binaria"
13 ]
14
```

3. `busqueda_binaria.py`:

```
>- nu      main.py U      busqueda_binaria.py U X      busqueda_lineal.py U
semana-3 > ejercicios_funciones-oop > ej3 > algoritmos > busqueda_binaria.py > ...
1  """
2  Algoritmo de búsqueda binaria.
3  """
4
5
6  def busqueda_binaria(coleccion, elemento):
7      """
8      Implementa el algoritmo de búsqueda binaria.
9      Requiere que la colección esté ordenada.
10     """
11
12     izquierda, derecha = 0, len(coleccion) - 1
13
14     while izquierda <= derecha:
15         medio = (izquierda + derecha) // 2
16
17         # Verificar si el elemento está en el medio
18         if coleccion[medio] == elemento:
19             return medio
20
21         # Si el elemento es mayor, ignorar la mitad izquierda
22         if coleccion[medio] < elemento:
23             izquierda = medio + 1
24
25         # Si el elemento es menor, ignorar la mitad derecha
26         else:
27             derecha = medio - 1
28
29     # Elemento no encontrado
30     return -1
31
```

4. busqueda_lineal.py:

```
> nu      main.py U      busqueda_binaria.py U      busqueda_lineal.py U X
semana-3 > ejercicios_funciones-oop > ej3 > algoritmos > busqueda_lineal.py > ...
1  """
2  Algoritmo de búsqueda lineal.
3  """
4
5
6  def busqueda_lineal(coleccion, elemento):
7      """
8      Implementa el algoritmo de búsqueda lineal.
9      """
10
11     for i, elem in enumerate(coleccion):
12         if elem == elemento:
13             return i
14
15     return -1
16
```

Output:

```
> nu      X      main.py U      busqueda_binaria.py U      busqueda_lineal.py U
● C:\dev\algo-structs> python semana-3\ejercicios_funciones-oop\ej3\main.py
Lista desordenada: [45, 12, 85, 32, 89, 39, 69, 44, 42, 1, 6, 8]
Lista ordenada: [1, 6, 8, 12, 32, 39, 42, 44, 45, 69, 85, 89]
-----
Buscando el elemento 42 con búsqueda lineal:
Elemento encontrado en la posición 8

Buscando el elemento 42 con búsqueda binaria:
Elemento encontrado en la posición 6

Buscando el elemento 99 que no existe:
Búsqueda lineal: No encontrado
Búsqueda binaria: No encontrado
○ C:\dev\algo-structs> |
```

Ejercicio #4

1. main.py:

```
>- nu  main.py  X  inventario.py  producto.py
semana-3 > ejercicios_funciones-oop > ej4 > main.py > ...
2  Diseñar una clase Producto que incluya atributos como código, nombre, precio y
3  cantidad en stock. Además, los estudiantes deberán implementar una clase
4  Inventario que administre una colección de objetos Producto, incorporando
5  métodos para agregar, buscar, actualizar y eliminar productos.
6  """
7
8  from os import system
9
10 from inventario import Inventario # pylint: disable=E0401
11
12
13 def menu_principal(inventario: Inventario) -> None:
14     """
15     Loop principal del programa que le muestra un menú de opciones al usuario.
16     """
17
18     opcion = ""
19     while opcion != "6":
20         system("cls || clear")
21         print("\nGestión de Inventario")
22         print("-----\n")
23         print("1. Agregar producto")
24         print("2. Consultar un producto específico")
25         print("3. Mostrar todos productos registrados")
26         print("4. Modificar producto")
27         print("5. Eliminar producto")
28         print("6. Salir")
29
30         opcion = input("\n-> ")
31
32         match opcion:
33             case "1":
34                 system("cls || clear")
35                 inventario.agregar_producto()
36             case "2":
37                 system("cls || clear")
38                 inventario.consultar_producto()
39                 input("\nPresione 'Enter' para regresar al menú principal...")
40             case "3":
41                 system("cls || clear")
42                 inventario.mostrar_productos()
43             case "4":
44                 system("cls || clear")
45                 inventario.modificar_producto()
46             case "5":
47                 system("cls || clear")
48                 inventario.eliminar_producto()
49             case "6":
50                 print("\nSaliendo del programa...\n")
51             case _:
52                 input("\n;Opción inválida, intente nuevamente!")
53     inventario.productos.sort(key=lambda p: p.codigo)
```

```
>- nu      main.py  X  inventario.py  producto.py
semana-3 > ejercicios_funciones-oop > ej4 > main.py > ...
55  def main() -> None:
56      """
57      Ejecución del programa.
58      """
59
60      menu_principal(Inventario())
61
62
63  if __name__ == "__main__":
64      main()
65
```

2. producto.py:

```
>- nu      main.py  inventario.py  producto.py X
semana-3 > ejercicios_funciones-oop > ej4 > producto.py > ...
1  """
2  Definición de la clase Producto.
3  """
4
5
6  class Producto:
7      """
8      Clase de datos para representar un producto con:
9      - un código,
10     - un nombre,
11     - una cantidad,
12     - un precio
13     """
14
15     def __init__(
16         self,
17         codigo: int = 0,
18         nombre: str = "",
19         cantidad: int = 0,
20         precio: float = 0.0
21     ):
22         self.codigo = codigo
23         self.nombre = nombre
24         self.cantidad = cantidad
25         self.precio = precio
26
```

3. inventario.py:

```
>- nu      main.py      inventario.py X      producto.py
semana-3 > ejercicios_funciones-oop > ej4 > inventario.py > ...
1  """
2  Definición de la clase Inventario.
3  """
4
5  from os import system
6  from typing import Optional
7
8  from producto import Producto  # pylint: disable=E0401
9
10
11 class Inventario:
12     """
13     Representa el inventario de productos de un negocio.
14     """
15
16     def __init__(self, lista_productos: Optional[list[Producto]] = None) -> None:
17         if lista_productos is None:
18             self.productos = []
19         else:
20             self.productos = lista_productos
21
22
23     def _pedir_codigo(self, accion: str) -> int:
24         """
25         Pide el código del producto a mostrar/modificar/eliminar.
26         """
27
28         try:
29             codigo = int(
30                 input(f"\nIngrese el código del producto que desea {accion}: ")
31             )
32
33             if codigo < 0:
34                 raise ValueError
35
36         except (TypeError, ValueError):
37             input("¡El código debe ser un número entero positivo!")
38             return self._pedir_codigo(accion)  # volver a pedir input
39
40         return codigo
41
42
```

```
> nu      main.py      inventario.py X      producto.py
semana-3 > ejercicios_funciones-oop > ej4 > inventario.py > ...
11  class Inventario:
43      def _buscar_producto(self, codigo: int) -> Optional[Producto]:
44          """
45          Busca el índice del producto con el código indicado
46          en self.productos. Retorna -1 si el producto no existe.
47          """
48
49          for producto in self.productos:
50              if producto.codigo == codigo:
51                  return producto
52
53          return None
54
55
56      def agregar_producto(self) -> None:
57          """
58          Pide los datos de un producto nuevo y lo agrega a self.productos.
59          """
60
61          print("\nAgregar Producto Nuevo:")
62          print("-----")
63
64          print("\nIngresa los datos del producto:")
65          try:
66              codigo = int(input("Código: "))
67              nombre = input("Nombre: ")
68              cantidad = int(input("Cantidad: "))
69              precio = float(input("Precio (en C$): "))
70
71              if codigo < 0 or cantidad < 0 or precio <= 0:
72                  raise ValueError
73              if self._buscar_producto(codigo) is not None:
74                  raise NameError
75
76          except (TypeError, ValueError):
77              input(
78                  "\nEl código y la cantidad del producto deben ser números enteros"
79                  + " positivos, y el precio debe ser un número real positivo!"
80              )
81
82          system("cls || clear")
83          return self.agregar_producto() # volver a pedir input
84
85          except NameError:
86              input("\nEl código ingresado ya existe en el registro de productos!")
87              system("cls || clear")
88              return self.agregar_producto() # volver a pedir input
89
90
91          self.productos.append(Producto(codigo, nombre, cantidad, precio))
92          input("\nProducto agregado exitosamente!")
93          return None
```


> nu

main.py

inventario.py X

producto.py

semana-3 > ejercicios_funciones-oop > ej4 > inventario.py > ...

```
11 class Inventario:
12
13
14
15
16     def consultar_producto(self, codigo: Optional[int] = None) -> None:
17         """
18         Busca un producto específico en self.productos y muestra sus datos.
19         """
20
21         cd = codigo if codigo is not None else self._pedir_codigo("consultar")
22         producto = self._buscar_producto(cd)
23
24         if producto is None:
25             input(f"\n¡No existe un producto con el código '{cd}'!")
26             return
27
28         print(f"\nProducto #{producto.codigo}:")
29         print("-----")
30         print(f"Nombre: {producto.nombre}")
31         print(f"Cantidad: {producto.cantidad}")
32         print(f"Precio: {producto.precio}")
33         print("-----")
34
35
36     def mostrar_productos(self) -> None:
37         """
38         Llama consultar_producto() para todos los productos registrados.
39         """
40
41         print("\nProductos Registrados:")
42         print("-----")
43
44         for producto in self.productos:
45             self.consultar_producto(producto.codigo)
46
47         input("\nPresione 'Enter' para regresar al menú principal...")
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
```

```
> nu      main.py      inventario.py X      producto.py
semana-3 > ejercicios_funciones-oop > ej4 > inventario.py > ...
11  class Inventario:
129
130  ... def modificar_producto(self) -> None:
131  ...     """
132  ...     Permite al usuario ingresar los datos de un producto de nuevo para modificarlos
133  ...     (excepto el código, ya que se utiliza para identificar los productos).
134  ...     """
135
136  ...     codigo = self._pedir_codigo("modificar")
137  ...     producto_modificado = self._buscar_producto(codigo)
138
139  ...     if producto_modificado is None:
140  ...         input(f"\n¡No existe un producto con el código '{codigo}'!")
141  ...         return None
142
143  ...     print("\nModificar Producto:")
144  ...     print("-----")
145
146  ...     self.consultar_producto(codigo)
147  ...     print()
148
149  ...     # eliminar el producto original para sobrescribirlo
150  ...     self.productos.remove(producto_modificado)
151
152  ...     # pedir nuevos datos
153  ...     print(
154  ...         "Ingrese los nuevos datos del producto (dejar en blanco para no modificar):"
155  ...     )
156
157  ...     nuevo_nombre = input("Nombre: ")
158  ...     nueva_cantidad = input("Cantidad: ")
159  ...     nuevo_precio = input("Precio: ")
160
```

>- nu

main.py

inventario.py X

producto.py

semana-3 > ejercicios_funciones-ooop > ej4 > inventario.py > ...

```
11  class Inventario:
130      def modificar_producto(self) -> None:
161          .....# Los strings vacios se evaluan como False,
162          .....# entonces si el usuario decide no modificar un atributo,
163          .....# el valor del objeto original permanecera igual
164
165          .....if nuevo_nombre:
166          .....|.....producto_modificado.nombre = nuevo_nombre
167
168          .....try:
169          .....|.....if nueva_cantidad:
170          .....|.....|.....producto_modificado.cantidad = int(nueva_cantidad)
171          .....|.....if nuevo_precio:
172          .....|.....|.....producto_modificado.precio = float(nuevo_precio)
173
174          .....|.....if producto_modificado.cantidad < 0 or producto_modificado.precio <= 0:
175          .....|.....|.....raise ValueError
176
177          .....except (TypeError, ValueError):
178          .....|.....input(
179          .....|.....|....."\n¡El código y la cantidad del producto deben ser números enteros"
180          .....|.....|.....+ " positivos, y el precio debe ser un número real positivo!"
181          .....|.....|.....)
182
183          .....|.....system("cls || clear")
184          .....|.....return self.modificar_producto()
185
186          .....self.productos.append(producto_modificado)
187          .....input(
188          .....|.....f"\n¡Datos del producto #{producto_modificado.codigo}"
189          .....|.....+ " fueron actualizados exitosamente!"
190          .....|.....)
191
192          .....return None
193
```

>- nu

main.py

inventario.py X

producto.py

semana-3 > ejercicios_funciones-oop > ej4 > inventario.py > ...

```
11  class Inventario:
195  .... def eliminar_producto(self) -> None:
196  ....     """
197  ....     Encuentra el índice del producto a eliminar, y lo quita de self.productos.
198  ....     """
199
200  ....     codigo = self._pedir_codigo("eliminar")
201  ....     producto = self._buscar_producto(codigo)
202
203  ....     if producto is None:
204  ....         input(f"\n¡No existe un producto con el código '{codigo}'!")
205  ....         return
206
207  ....     print("\nEliminar Producto:")
208  ....     print("-----")
209
210  ....     self.consultar_producto(codigo)
211  ....     print()
212
213  ....     confirmacion = input(
214  ....         f"\n¿Está seguro que desea eliminar el producto seleccionado? (s/n) "
215  ....     ).lower()
216
217  ....     if confirmacion == "s":
218  ....         self.productos.remove(producto)
219  ....         input(f"\nProducto #{producto.codigo} eliminado exitosamente!")
220  ....     elif confirmacion == "n":
221  ....         input("\nDe acuerdo! Regresando al menú principal...")
222  ....     else:
223  ....         input("\nRespuesta inválida! Regresando al menú principal...")
224
```

Ejercicio #5

1. main.py

```
>- nu  main.py  ×  pedido.py  clientes.py  producto.py
semana-3 > ejercicios_funciones-oop > ej5 > main.py > ...
1  """
2  Crear una clase Cliente con atributos básicos (por ejemplo, ID, nombre y contacto)
3  y una clase Pedido que contenga información sobre el cliente, la lista de productos
4  solicitados y el total de la venta. Se podrá incluir el uso de herencia para diferenciar
5  entre tipos de clientes (regular, VIP, etc.) y aplicar descuentos especiales.
6  """
7
8  from clientes import (
9      Cliente,
10     ClienteOro,
11     ClientePlatino
12 )
13
14 from pedido import Pedido
15 from producto import Producto  # pylint: disable=E0401
16
17
18 def main() -> None:
19     """
20     Ejecución del programa.
21     """
22
23     cliente1 = Cliente("0001", "Juancito Pérez", "7945-9316")
24     cliente2 = ClienteOro("0002", "Juanita X", "1243-6549")
25     cliente3 = ClientePlatino("0003", "Persona Random", "8764-9842")
26
27     productos1 = [
28         Producto(100, "Calculadora", 2, 123.4),
29         Producto(101, "Laptop", 1, 2345.6),
30         Producto(102, "Borrador", 3, 45.6),
31     ]
32
33     productos2 = [
34         Producto(103, "iPhone", 1, 3456.7),
35         Producto(104, "Galaxy S24", 1, 2345.6),
36         Producto(105, "Galaxy Buds", 2, 1234.5),
37     ]
38
39     productos3 = [
40         Producto(106, "Mochila", 1, 123.4),
41         Producto(107, "Cuaderno", 3, 78.9),
42         Producto(108, "Lapicero", 3, 45.6),
43     ]
44
45     pedido1 = Pedido(cliente1, productos1)
46     pedido2 = Pedido(cliente2, productos2)
47     pedido3 = Pedido(cliente3, productos3)
48
```

> nu

main.py



pedido.py

clientes.py

producto.py

semana-3 > ejercicios_funciones-oop > ej5 > main.py > ...

```
18 def main() -> None:
49     print("\nDatos de Pedido #1:")
50     print(f"Cliente: {pedido1.cliente.nombre} ({pedido1.cliente.telefono}")
51     print(f"Productos:\n{pedido1.get_producto_str()}")
52     print(f"Total del pedido #1: {pedido1.total}")
53
54     print("\nDatos de Pedido #2:")
55     print(f"Cliente: {pedido2.cliente.nombre} ({pedido2.cliente.telefono}")
56     print(f"Productos:\n{pedido2.get_producto_str()}")
57     print(f"Total del pedido #2: {pedido2.total}")
58
59     print("\nDatos de Pedido #3:")
60     print(f"Cliente: {pedido3.cliente.nombre} ({pedido3.cliente.telefono}")
61     print(f"Productos:\n{pedido3.get_producto_str()}")
62     print(f"Total del pedido #3: {pedido3.total}")
63
64     print()
65
66
67 if __name__ == "__main__":
68     main()
69
```

2. producto.py

```
semana-3 > ejercicios_funciones-oop > ej5 > producto.py > ...
1  """
2  Definición de la clase Producto.
3  """
4
5
6  class Producto:
7      """
8      Clase de datos para representar un producto con:
9      - un código,
10     - un nombre,
11     - una cantidad,
12     - un precio
13     """
14
15     def __init__(
16         self,
17         codigo: int = 0,
18         nombre: str = "",
19         cantidad: int = 0,
20         precio: float = 0.0
21     ):
22         self.codigo = codigo
23         self.nombre = nombre
24         self.cantidad = cantidad
25         self.precio = precio
26
27     def __str__(self) -> str:
28         return (
29             f"Código: {self.codigo}, Nombre: {self.nombre}, " +
30             f"Cantidad: {self.cantidad}, Precio: C${self.precio:.2f}"
31         )
32
```


3. cliente.py

```

> nu      main.py  pedido.py  clientes.py X  producto.py
semana-3 > ejercicios_funciones-oo > ej5 > clientes.py > ...
1  """
2  Definición de las clases Cliente, ClienteOro y ClientePlatino.
3  """
4
5  class Cliente:
6      """
7      Representa un cliente del negocio y sus detalles.
8      """
9
10     def __init__(
11         self,
12         codigo: str,
13         nombre: str,
14         telefono: str
15     ):
16         self.codigo = codigo
17         self.nombre = nombre
18         self.telefono = telefono
19         self.descuento = 1.0 # 0%
20
21
22     class ClienteOro(Cliente):
23         """
24         Representa un cliente 'Oro' del negocio,
25         ue tiene un descuento de 15% en sus compras.
26         """
27
28         def __init__(
29             self,
30             codigo: str,
31             nombre: str,
32             telefono: str
33         ):
34             Cliente.__init__(self, codigo, nombre, telefono)
35             self.descuento = 0.85 # 15%
36
37
38     class ClientePlatino(Cliente):
39         """
40         Representa un cliente 'Platino' del negocio,
41         ue tiene un descuento de 30% en sus compras.
42         """
43
44         def __init__(
45             self,
46             codigo: str,
47             nombre: str,
48             telefono: str
49         ):
50             Cliente.__init__(self, codigo, nombre, telefono)
51             self.descuento = 0.70 # 30%
52
```

4. pedido.py

```
> nu main.py pedido.py X clientes.py producto.py
semana-3 > ejercicios_funciones-oop > ej5 > pedido.py > ...
1  """
2  Definición de la clase Pedido.
3  """
4
5  from typing import Union
6  from clientes import (
7      Cliente,
8      ClienteOro,
9      ClientePlatino
10 )
11
12 from producto import Producto  # pylint: disable=E0401
13
14
15 class Pedido:
16     """
17     Representa una compra.
18     """
19
20     def __init__(
21         self,
22         cliente: Union[Cliente, ClienteOro, ClientePlatino],
23         productos: list[Producto]
24     ):
25         self.cliente = cliente
26         self.productos = productos
27         self.total = self._calcular_total()
28
29     def _calcular_total(self) -> float:
30         return sum(
31             p.precio * p.cantidad for p in self.productos
32         ) * self.cliente.descuento
33
34     def get_producto_str(self) -> str:
35         """
36         Convierte la lista de productos a un string para imprimir.
37         """
38
39         return "\n".join(str(p) for p in self.productos)
40
```

Output:

```
> nu      X      main.py      pedido.py      producto.py

C:\dev\algo-structs> python semana-3\ejercicios_funciones-oop\ej5\main.py

Datos de Pedido #1:
Cliente: Juancito Pérez (7945-9316)
Productos:
Código: 100, Nombre: Calculadora, Cantidad: 2, Precio: C$123.40
Código: 101, Nombre: Laptop, Cantidad: 1, Precio: C$2345.60
Código: 102, Nombre: Borrador, Cantidad: 3, Precio: C$45.60
Total del pedido #1: 2729.2

Datos de Pedido #2:
Cliente: Juanita X (1243-6549)
Productos:
Código: 103, Nombre: iPhone, Cantidad: 1, Precio: C$3456.70
Código: 104, Nombre: Galaxy S24, Cantidad: 1, Precio: C$2345.60
Código: 105, Nombre: Galaxy Buds, Cantidad: 2, Precio: C$1234.50
Total del pedido #2: 7030.605

Datos de Pedido #3:
Cliente: Persona Random (8764-9842)
Productos:
Código: 106, Nombre: Mochila, Cantidad: 1, Precio: C$123.40
Código: 107, Nombre: Cuaderno, Cantidad: 3, Precio: C$78.90
Código: 108, Nombre: Lapicero, Cantidad: 3, Precio: C$45.60
Total del pedido #3: 347.83

C:\dev\algo-structs> 
```

Ejercicio #6

1. main.py

```
> nu      main.py  ×  factura.py M  clientes.py  producto.py
semana-3 > ejercicios_funciones-oop > ej6 > main.py > ...
1  """
2  Crear una clase Factura que simule el proceso de facturación de una venta.
3  Los estudiantes deberán encapsular los datos internos de la factura
4  (como los detalles de los productos, cantidades, precios y descuentos)
5  y proveer métodos para calcular el total de la venta, generar
6  reportes simples y validar la integridad de la información.
7  """
8
9  from clientes import ( # type: ignore # pylint: disable=E0401
10      ... Cliente,
11      ... ClienteOro,
12      ... ClientePlatino
13  )
14
15  from factura import Factura # type: ignore # pylint: disable=E0401
16  from producto import Producto # type: ignore # pylint: disable=E0401
17
18
19  def main() -> None:
20      """
21      Ejecución del programa.
22      """
23
24      cliente1 = Cliente("0001", "Juancito Pérez", "7945-9316")
25      cliente2 = ClienteOro("0002", "Juanita X", "1243-6549")
26      cliente3 = ClientePlatino("0003", "Persona Random", "8764-9842")
27
28      productos1 = [
29          ... Producto(100, "Calculadora", 2, 123.4),
30          ... Producto(101, "Laptop", 1, 2345.6),
31          ... Producto(102, "Borrador", 3, 45.6),
32      ]
33
34      productos2 = [
35          ... Producto(103, "iPhone", 1, 3456.7),
36          ... Producto(104, "Galaxy S24", 1, 2345.6),
37          ... Producto(105, "Galaxy Buds", 2, 1234.5),
38      ]
39
```

> nu

main.py ×

factura.py M

clientes.py

producto.py

semana-3 > ejercicios_funciones-oop > ej6 > main.py > ...

```
19 def main() -> None:
40     ... productos3 = [
41     ...     ... Producto(106, "Mochila", 1, 123.4),
42     ...     ... Producto(107, "Cuaderno", 3, 78.9),
43     ...     ... Producto(108, "Lapicero", 3, 45.6),
44     ... ]
45
46     ... factura1 = Factura("001", cliente1, productos1)
47     ... factura2 = Factura("002", cliente2, productos2)
48     ... factura3 = Factura("003", cliente3, productos3)
49
50     ... print()
51     ... print(factura1.generar_reporte())
52     ... print()
53     ... print(factura2.generar_reporte())
54     ... print()
55     ... print(factura3.generar_reporte())
56
57
58 if __name__ == "__main__":
59     ... main()
60
```

2. producto.py

```
semana-3 > ejercicios_funciones-oop > ej5 > producto.py > ...
1  """
2  Definición de la clase Producto.
3  """
4
5
6  class Producto:
7      """
8      Clase de datos para representar un producto con:
9      - un código,
10     - un nombre,
11     - una cantidad,
12     - un precio
13     """
14
15     def __init__(
16         self,
17         codigo: int = 0,
18         nombre: str = "",
19         cantidad: int = 0,
20         precio: float = 0.0
21     ):
22         self.codigo = codigo
23         self.nombre = nombre
24         self.cantidad = cantidad
25         self.precio = precio
26
27     def __str__(self) -> str:
28         return (
29             f"Código: {self.codigo}, Nombre: {self.nombre}, " +
30             f"Cantidad: {self.cantidad}, Precio: C${self.precio:.2f}"
31         )
32
```

3. cliente.py

```

>- nu      main.py      pedido.py      clientes.py X      producto.py
semana-3 > ejercicios_funciones-oop > ej5 > clientes.py > ...
1  """
2  Definición de las clases Cliente, ClienteOro y ClientePlatino.
3  """
4
5  class Cliente:
6      """
7      Representa un cliente del negocio y sus detalles.
8      """
9
10     def __init__(
11         self,
12         codigo: str,
13         nombre: str,
14         telefono: str
15     ):
16         self.codigo = codigo
17         self.nombre = nombre
18         self.telefono = telefono
19         self.descuento = 1.0 # 0%
20
21
22     class ClienteOro(Cliente):
23         """
24         Representa un cliente 'Oro' del negocio,
25         ue tiene un descuento de 15% en sus compras.
26         """
27
28         def __init__(
29             self,
30             codigo: str,
31             nombre: str,
32             telefono: str
33         ):
34             Cliente.__init__(self, codigo, nombre, telefono)
35             self.descuento = 0.85 # 15%
36
37
38     class ClientePlatino(Cliente):
39         """
40         Representa un cliente 'Platino' del negocio,
41         ue tiene un descuento de 30% en sus compras.
42         """
43
44         def __init__(
45             self,
46             codigo: str,
47             nombre: str,
48             telefono: str
49         ):
50             Cliente.__init__(self, codigo, nombre, telefono)
51             self.descuento = 0.70 # 30%
52

```


4. factura.py

```
semana-3 > ejercicios_funciones-oop > ej6 > factura.py > ...
1  """
2  Definición de la clase Factura.
3  """
4
5  from typing import Union
6  from clientes import (  # type: ignore # pylint: disable=E0401
7      Cliente,
8      ClienteOro,
9      ClientePlatino
10 )
11
12 from producto import Producto  # type: ignore # pylint: disable=E0401
13
14
15 class Factura:
16     """
17     Representa una factura.
18     """
19
20     def __init__(
21         self,
22         codigo: str,
23         cliente: Union[Cliente, ClienteOro, ClientePlatino],
24         productos: list[Producto]
25     ):
26         self.__codigo = codigo
27         self.__cliente = cliente
28         self.__productos = productos
29
30     @property
31     def codigo(self) -> str:
32         """
33         Getter para el atributo protegido __codigo.
34         """
35
36         return self.__codigo
37
```

> nu

main.py

factura.py M X

clientes.py

producto.py

semana-3 > ejercicios_funciones-oop > ej6 > factura.py > ...

```
15 class Factura:
38     @property
39     def cliente(self) -> Union[Cliente, ClienteOro, ClientePlatino]:
40         """
41         Getter para el atributo protegido __cliente.
42         """
43
44         return self.__cliente
45
46     @property
47     def productos(self) -> list[Producto]:
48         """
49         Getter para el atributo protegido __productos.
50         """
51
52         return self.__productos
53
54     def get_producto_str(self) -> str:
55         """
56         Convierte la lista de productos a un string para imprimir.
57         """
58
59         return "\n".join(str(p) for p in self.productos)
60
61     def calcular_total(self) -> float:
62         """
63         Suma todos los montos de todos los productos en
64         self.productos y le aplica el descuento del cliente.
65         """
66
67         return sum(
68             p.precio * p.cantidad for p in self.productos
69         ) * self.cliente.descuento
70
71     def generar_reporte(self) -> str:
72         """
73         Organiza los datos de la factura para imprimirlos.
74         """
75
76         total = self.calcular_total()
77         subtotal = total * (1 - self.cliente.descuento + 1)
78         reporte = ""
79
```

> nu

main.py

factura.py M X

clientes.py

producto.py

semana-3 > ejercicios_funciones-oop > ej6 > factura.py > ...

```
15 class Factura:
71     def generar_reporte(self) -> str:
72         """
73         Organiza los datos de la factura para imprimirlos.
74         """
75
76         total = self.calcular_total()
77         subtotal = total * (1 - self.cliente.descuento + 1)
78         reporte = ""
79
80         reporte += "=" * 75
81         reporte += f"\nFACTURA #{self.codigo}\n"
82         reporte += "=" * 75
83
84         reporte += "\nDATOS DEL CLIENTE:\n"
85         reporte += "-" * 75
86         reporte += f"\nNombre: {self.cliente.nombre}"
87         reporte += f"\nNúmero de teléfono: {self.cliente.telefono}"
88         reporte += f"\nDescuento aplicable: {(1 - self.cliente.descuento) * 100:.0f}%\n"
89
90         reporte += "-" * 75
91         reporte += "\nPRODUCTOS:\n"
92         reporte += "-" * 75
93         reporte += "\n" + self.get_producto_str()
94
95         reporte += "\n" + "=" * 75
96         reporte += f"\nSUBTOTAL: C${subtotal:.2f}"
97         reporte += f"\nDESCUENTO: {(1 - self.cliente.descuento) * 100:.0f}%"
98         reporte += f"\nTOTAL A PAGAR: C${total:.2f}\n"
99         reporte += "=" * 75
100        reporte += "\n"
101
102        return reporte
103
```

Output:

```
> nu  X  main.py  factura.py M  clientes.py  producto.py

C:\dev\algo-structs> python semana-3\ejercicios_funciones-oop\ej6\main.py

=====
FACTURA #001
=====
DATOS DEL CLIENTE:
-----
Nombre: Juancito Pérez
Número de teléfono: 7945-9316
Descuento aplicable: 0%
-----
PRODUCTOS:
-----
Código: 100, Nombre: Calculadora, Cantidad: 2, Precio: C$123.40
Código: 101, Nombre: Laptop, Cantidad: 1, Precio: C$2345.60
Código: 102, Nombre: Borrador, Cantidad: 3, Precio: C$45.60
=====
SUBTOTAL: C$2729.20
DESCUENTO: 0%
TOTAL A PAGAR: C$2729.20
=====

=====
FACTURA #002
=====
DATOS DEL CLIENTE:
-----
Nombre: Juanita X
Número de teléfono: 1243-6549
Descuento aplicable: 15%
-----
PRODUCTOS:
-----
Código: 103, Nombre: iPhone, Cantidad: 1, Precio: C$3456.70
Código: 104, Nombre: Galaxy S24, Cantidad: 1, Precio: C$2345.60
Código: 105, Nombre: Galaxy Buds, Cantidad: 2, Precio: C$1234.50
=====
SUBTOTAL: C$8085.20
DESCUENTO: 15%
TOTAL A PAGAR: C$7030.60
=====
```