

## Examen final

Titulo	Estructura de Datos y Algoritmos Caso Estudio		
Semana	8		
Resultado/s de aprendizaje/s	Todos los aprendizajes esperados de la asignatura.		
Criterios de evaluación	Todos los criterios de evaluación de cada semana.		
Producto por entregar	Informe		
Forma de trabajo	Individual	Nro. Estudiantes	1
Formato entrega actividad	Informe		

### Estimadas y estimados estudiantes:

Bienvenidos al examen final de la asignatura, correspondiente a la semana 8. En esta actividad deberán analizar un caso de estudio práctico en Python y resolver los requerimientos planteados.

Las tareas incluyen la implementación de clases y objetos, la gestión de clientes, productos y pedidos mediante listas y diccionarios, el uso de estructuras de decisión e iteración para validar stock y calcular totales, y la elaboración de un reporte de *testing* que compare resultados esperados con los obtenidos.

A partir del caso de estudio entregado y de las herramientas revisadas semana a semana, deberán seguir las siguientes instrucciones para desarrollar y documentar su solución.

## Caso de estudio:

La empresa ShopFast es una tienda en línea que ha tenido un rápido crecimiento en sus operaciones. Actualmente, la gestión de clientes, productos y pedidos se realiza de forma manual, lo que ha generado duplicación de datos, errores en el stock y problemas en el cálculo de ventas. El área de informática ha solicitado un sistema en Python que automatice este proceso y permita mayor control sobre la información.

El sistema debe permitir registrar clientes con sus datos básicos (RUT, nombre, correo y teléfono) y productos con su respectivo código, nombre, precio y *stock* disponible. Además, el sistema debe permitir la creación de pedidos, donde cada cliente puede seleccionar uno o más productos, indicando las cantidades solicitadas. Antes de confirmar un pedido, el programa debe validar el *stock* disponible, descontar automáticamente las unidades vendidas y calcular el total de la compra.

El sistema debe implementar clases y objetos para modelar clientes, productos y pedidos; usar listas y diccionarios para organizar los datos; y aplicar estructuras de decisión e iteración para verificar condiciones y recorrer elementos.

Asimismo, se debe implementar un conjunto de métodos que gestionen las operaciones principales: agregar y eliminar productos del pedido, calcular el total y actualizar el inventario. Finalmente, deberá elaborar un reporte de *testing* con al menos tres casos de prueba, comparando los resultados esperados con los obtenidos, para verificar el correcto funcionamiento del sistema.

## Objetivo

Desarrollar un sistema en Python aplicando los saberes de la asignatura (POO, estructuras de control, listas, diccionarios, métodos y *testing*), que permita simular la gestión de pedidos de una tienda de manera eficiente y documentada.

## Instrucciones

- Implementar en Python las clases Cliente, Producto y Pedido, con atributos y métodos adecuados.
- Usar listas y diccionarios para organizar clientes, productos y pedidos.
- Implementar métodos para agregar/eliminar productos, calcular totales y actualizar *stock*.
- Validar con estructuras condicionales e iterativas el *stock* y la creación de pedidos.
- Entregar código fuente comentado y un reporte en PDF o Word (2–4 páginas) que incluya:
  - Explicación del diseño del sistema.
  - Capturas o ejemplos de ejecución.
  - Reporte de testing con mínimo 3 casos de prueba (resultado esperado vs obtenido).
- Reflexión final sobre dificultades y aprendizajes.

## Rúbrica de evaluación

Indicadores disciplinares/ niveles de logro	Excelente	Muy bueno	Bueno	Aceptable	Insuficiente
POO (clases y métodos)	<b>20 puntos</b>	<b>16 puntos</b>	<b>12 puntos</b>	<b>8 puntos</b>	<b>1 punto</b>
	Implementa todas las clases y métodos con alta modularidad, reutilización y buenas prácticas de POO, sin errores.	Implementa todas las clases y métodos correctamente, con modularidad y claridad.	Implementa clases y métodos con errores menores pero funcionales.	Implementación incompleta o poco clara, con errores que afectan la comprensión.	No aplica correctamente POO o el código no funciona.
Estructuras de control	<b>20 puntos</b>	<b>16 puntos</b>	<b>12 puntos</b>	<b>8 puntos</b>	<b>1 punto</b>
	Aplica estructuras de control (if, for, while) de manera óptima, eficiente y legible, cubriendo todos los casos necesarios.	Aplica correctamente if, for y while en todas las validaciones.	Aplica estructuras, con errores menores o limitaciones en su uso.	Uso parcial, poco eficiente o con errores frecuentes en validaciones.	No aplica estructuras de control o las aplica de forma incorrecta.
Listas y diccionario	<b>20 puntos</b>	<b>16 puntos</b>	<b>12 puntos</b>	<b>8 puntos</b>	<b>1 punto</b>
	Usa listas y diccionarios de forma eficiente y creativa, optimizando la gestión de datos y evitando redundancias.	Uso adecuado en la gestión de clientes, productos y pedidos.	Uso con errores menores o incompleto en alguna parte del sistema.	Uso limitado, sin aprovechar las ventajas de las estructuras.	No se utilizan listas ni diccionarios o se usan de forma errónea.
Testing (casos de prueba)	<b>20 puntos</b>	<b>16 puntos</b>	<b>12 puntos</b>	<b>8 puntos</b>	<b>1 punto</b>
	Incluye ≥3 pruebas completas, bien documentadas, con análisis crítico de	Incluye ≥3 pruebas completas, claras y bien documentadas	Incluye pruebas correctas pero poco detalladas o solo 2 completas.	Incluye pruebas incompletas o con errores de interpretación.	No presenta pruebas o las presentadas no son válidas.

	diferencias entre esperado y obtenido.				
Claridad del código y reporte	20 puntos	16 puntos	12 puntos	8 puntos	1 punto
	Código impecable: legible, comentado con estándares, reporte completo con diseño explicado, ejemplos y reflexión profunda.	Código bien estructurado y comentado, con reporte completo.	Código claro con pequeños errores en documentación o reporte básico.	Código poco claro, con reporte incompleto o deficiente.	Código desordenado o sin reporte entregado.