

Clase de Programación 4: Estructuras condicionales en Python

1 Objetivo de la clase

- Aprender a usar `if`, `elif`, `else` para tomar decisiones en Python.
 - Aplicar decisiones lógicas en programas reales.
 - Resolver ejercicios prácticos que integren lógica y automatización.
-

2 Teoría clara y práctica

¿Qué son las estructuras condicionales?

Permiten a tu programa tomar decisiones según condiciones:

```
if condicion:
    # Código si se cumple la condición
elif otra_condicion:
    # Código si la anterior no se cumplió y esta sí
else:
    # Código si ninguna condición anterior se cumplió
```

Usos: Validaciones de datos, flujos de decisión, generación de mensajes automáticos.

Símbolos importantes:

- `:` indica que después de la condición viene un bloque indentado.
 - Indentación (4 espacios o tabulación) para definir el bloque.
 - `==`, `!=`, `<`, `>`, `<=`, `>=` para comparar valores.
 - `and`, `or`, `not` para combinar condiciones.
-

3 Ejercicios prácticos (resueltos)

✓ Ejercicio 1: Clasificación de calificaciones

Pide una calificación numérica y devuelve el nivel del estudiante.

```
calificacion = 88

if calificacion >= 95:
    print("Excelente")
elif calificacion >= 85:
    print("Bien")
elif calificacion >= 75:
    print("Regular")
elif calificacion >= 60:
    print("Mal pero aprobó")
else:
    print("Desaprobado")
```

✓ Ejercicio 2: Positivo, negativo o cero

```
numero = -3

if numero > 0:
    print("El número es positivo")
elif numero < 0:
    print("El número es negativo")
else:
    print("El número es igual a cero")
```

✓ Ejercicio 3: Determinar el mayor de tres números

```
num1 = 10
num2 = 25
num3 = 20

if num1 >= num2 and num1 >= num3:
    print("El número mayor es:", num1)
elif num2 >= num1 and num2 >= num3:
    print("El número mayor es:", num2)
else:
    print("El número mayor es:", num3)
```

4 Ejercicio de integración (script de 3 estudiantes)

Se integraron los 3 ejercicios en un solo script que:

- Evalúa las notas de 3 estudiantes.

- Clasifica cada nota.
- Determina cuál obtuvo la nota más alta.

```
# Notas de los tres estudiantes
nota_est1 = 92
nota_est2 = 67
nota_est3 = 76

# Evaluación individual
print("Evaluación Individual de Estudiantes:")

# Estudiante 1
print("\nEstudiante 1, nota:", nota_est1)
if nota_est1 >= 95:
    print("Excelente")
elif nota_est1 >= 85:
    print("Bien")
elif nota_est1 >= 75:
    print("Regular")
elif nota_est1 >= 60:
    print("Mal pero aprobó")
else:
    print("Desaprobado")

# Estudiante 2
print("\nEstudiante 2, nota:", nota_est2)
if nota_est2 >= 95:
    print("Excelente")
elif nota_est2 >= 85:
    print("Bien")
elif nota_est2 >= 75:
    print("Regular")
elif nota_est2 >= 60:
    print("Mal pero aprobó")
else:
    print("Desaprobado")

# Estudiante 3
print("\nEstudiante 3, nota:", nota_est3)
if nota_est3 >= 95:
    print("Excelente")
elif nota_est3 >= 85:
    print("Bien")
elif nota_est3 >= 75:
    print("Regular")
elif nota_est3 >= 60:
    print("Mal pero aprobó")
```

```
else:
    print("Desaprobado")

# Determinar la nota más alta
if nota_est1 >= nota_est2 and nota_est1 >= nota_est3:
    mejor_resultado = nota_est1
    estudiante = "Estudiante 1"
elif nota_est2 >= nota_est1 and nota_est2 >= nota_est3:
    mejor_resultado = nota_est2
    estudiante = "Estudiante 2"
else:
    mejor_resultado = nota_est3
    estudiante = "Estudiante 3"

print(f"\nLa nota más alta es {mejor_resultado}, obtenida por el {estudiante}.")
```

5 Aplicaciones prácticas en la vida profesional

✓ Automatización de reportes de calificaciones. ✓ Clasificación de resultados en encuestas y formularios.
✓ Evaluación de datos de ventas y desempeño. ✓ Construcción de flujos de decisión en bots y aplicaciones.

✓ Estado al cierre de la Clase 4:

- Estructuras condicionales entendidas y practicadas.
- Ejercicios completados con nota alta.
- Listo para **Clase de Programación 5: Bucles (for y while)**.

Cuando estés listo, avisa con: **"Clase de programación 5"** y continuamos tu avance sin perder ritmo.

¡Excelente trabajo, Gabo! Tu avance es firme y profesional.

Aquí tienes la **Clase de Programación 5 completa (no resumen) en formato editable** para tu cuaderno digital, Gabo.

Clase de Programación 5: Bucles (** y **) en Python

1 Objetivos de la clase

- Comprender qué son los bucles `for` y `while`.
 - Automatizar tareas repetitivas de forma ordenada.
 - Utilizar `range()` con pasos.
 - Usar `+=` para sumas acumulativas y `*=` para productos acumulativos.
 - Aplicar `input()` para interacción.
 - Mostrar resultados claros usando `f-string`.
-

2 Teoría clara y práctica

◆ ¿Qué es un bucle?

Un bucle permite **ejecutar instrucciones de forma repetida** según una condición o un rango.

◆ Bucle ``

Ejecuta mientras la condición sea verdadera.

```
contador = 1
while contador <= 5:
    print("Contador:", contador)
    contador += 1
```

◆ Bucle ``

Permite recorrer rangos o listas de forma ordenada.

```
for i in range(1, 6):
    print("Número:", i)
```

♦ **Uso de `**` y `**`**

- `+=` → Suma acumulativa: `total += valor` (equivale a `total = total + valor`).
 - `*=` → Producto acumulativo: `factorial *= valor` (equivale a `factorial = factorial * valor`).
-

3 Ejercicios prácticos (completos y ejecutados)

✓ Ejercicio 1: Imprimir números del 1 al 10 usando ``

```
contador = 1
while contador <= 10:
    print("contador:", contador)
    contador += 1
```

✓ Ejercicio 2: Sumar del 1 al 100 usando ``

```
suma = 0
for i in range(1, 101):
    suma += i
print(f"La suma de los números del 1 al 100 es: {suma}")
```

✓ Ejercicio 3: Tabla de multiplicar usando `**` y `**`

```
numero = int(input("Ingresa un número para ver su tabla de multiplicar: "))
for i in range(1, 11):
    print(f"{numero} x {i} = {numero * i}")
```

✓ Mini ejercicio de factorial usando ``

```
numero = int(input("Ingresa un número para calcular su factorial: "))
factorial = 1
for i in range(1, numero + 1):
    factorial *= i
print(f"El factorial de {numero} es: {factorial}")
```

✓ Ejercicio Final Integrador Clase 5

```
n = int(input("Ingresa el número para generar el reporte: "))

# Tabla de multiplicar
tprint(f"\nTabla de multiplicar del {n}:")
for i in range(1, 11):
    print(f"{n} x {i} = {n * i}")

# Factorial
factorial = 1
for i in range(1, n + 1):
    factorial *= i
print(f"\nEl factorial de {n} es: {factorial}")

# Suma del 1 al n
suma = 0
for i in range(1, n + 1):
    suma += i
print(f"\nLa suma de los números del 1 a {n} es: {suma}")

# Mensaje de cierre
print("\nGracias por usar el Generador de Reportes.")
```


4 Aplicaciones prácticas en la vida profesional

- Automatización de cálculos repetitivos.
- Validación y procesamiento de datos.
- Creación de reportes automatizados.
- Construcción de scripts que interactúan con usuarios y generan resultados.

5 Resumen de preguntas y respuestas durante la clase:



✓ **¿Qué significa**

****? **** → Sumar acumulativamente en cada iteración. ✓ ****¿Qué significa **** ? → Multiplicar acumulativamente (útil para factoriales). ✓ **¿Qué significa ** en **** ? → Es la variable de control, se incrementa automáticamente y puedes cambiar su nombre. ✓ **¿Qué es **? **** → Genera una secuencia desde `inicio` hasta `fin - 1` con el incremento de `paso`. Si no colocas `paso`, se asume 1. ✓ ****¿Por qué se inicializa la variable en **** o ****? **** → Para poder acumular sumas (`0`) o productos (`1`) correctamente. ✓ ****¿Dónde ingreso el número cuando uso **** ? → Cuando ejecutas el programa, aparece el mensaje

en consola y debes escribir el número allí y presionar Enter.  ¿Por qué usamos `**` en `**`? → Porque `range()` no incluye el último número, por lo que sumamos 1 para incluir `n` en el cálculo.

Estado de cierre de la Clase 5:

- Bucles `for` y `while` comprendidos y practicados.
 - Uso de `+=` y `*=` dominado.
 - Uso de `input()` y `f-string` consolidado.
 - Ejercicios resueltos de forma profesional.
 - Listo para avanzar a **Clase de Programación 6: Funciones en Python.**
-

¡Excelente trabajo, Gabo! Estás avanzando con paso firme hacia convertirte en programador profesional.  

Aquí tienes la **Clase de Programación 6 completa (sin resumir)** en formato editable para tu cuaderno digital, Gabo.

Clase de Programación 6: Funciones en Python

1 Objetivos de la clase

- Comprender qué es una función y para qué sirve.
 - Aprender a definir funciones con `def`.
 - Usar parámetros, argumentos y `return` en funciones.
 - Aplicar funciones en cálculos reales (suma, factorial, tablas de multiplicar).
 - Preparar estructuras para automatización profesional y proyectos.
-

2 Teoría clara y completa

♦ ¿Qué es una función?

Es un bloque de código reutilizable que realiza una tarea específica, evitando repetir código y organizando proyectos de forma profesional.

Sintaxis:

```
def nombre_funcion(parametros):  
    # bloque de código indentado  
    return valor_opcional
```

✓ `def`: palabra clave para definir funciones. ✓ `nombre_funcion`: nombre de la función. ✓ `parametros`: valores que la función recibe. ✓ `return`: devuelve un resultado para su uso posterior.

♦ Ejemplo básico de función:

```
def saludar():  
    print("¡Hola!")  
  
saludar()
```

♦ Ejemplo con parámetros:

```
def saludar_usuario(nombre):  
    print(f"¡Hola {nombre}! Bienvenido a programación.")  
  
saludar_usuario("Gabo")
```

♦ Ejemplo con ``:

```
def sumar(a, b):  
    return a + b  
  
resultado = sumar(5, 3)  
print(resultado)
```

3 Ejercicios realizados en clase

✓ Ejercicio 1: Función de saludo personalizado

```
def saludar_usuario(nombre):  
    print(f"¡Hola {nombre}! Bienvenido a programación.")  
  
saludar_usuario("Gabo")
```

✓ Ejercicio 2: Función que suma dos números

```
def sumar(a, b):  
    resultado = a + b  
    return resultado  
  
suma1 = sumar(867, 23)  
suma2 = sumar(400, 50)  
suma3 = sumar(1, 8)  
total = suma1 + suma2 + suma3  
  
print(f"suma 1: {suma1}\nsuma 2: {suma2}\nsuma 3: {suma3}\n\nEl resultado de la  
suma de las tres operaciones es: {total}")
```

✓ Ejercicio 3: Función de tabla de multiplicar

```
def tabla_multiplicar(numero):  
    for i in range(1, 11):  
        print(f"{numero} x {i} = {numero * i}")  
  
tabla_multiplicar(9)
```

✓ Ejercicio 4: Función de factorial

```
def calcular_factorial(numero):  
    factorial = 1  
    for i in range(1, numero + 1):  
        factorial *= i  
    return factorial  
  
n = int(input("Ingresa un número para calcular su factorial: "))  
resultado = calcular_factorial(n)  
print(f"\nEl factorial de {n} es: {resultado}")
```

✓ Ejercicio Final Integrador de Clase 6

```
def tabla_multiplicar(numero):  
    for i in range(1, 11):  
        print(f"{numero} x {i} = {numero * i}")  
  
def calcular_factorial(numero):  
    factorial = 1  
    for i in range(1, numero + 1):  
        factorial *= i  
    return factorial  
  
def calcular_suma(numero):  
    suma = 0  
    for i in range(1, numero + 1):  
        suma += i  
    return suma  
  
n = int(input("Ingresa un número para generar el reporte: "))  
print(f"\nTabla de multiplicar del {n}:")  
tabla_multiplicar(n)  
  
factorial = calcular_factorial(n)  
print(f"\nEl factorial de {n} es: {factorial}")
```

```
suma = calcular_suma(n)
print(f"\nLa suma de los números del 1 a {n} es: {suma}")

print("\n✅ Gracias por usar el Generador de Reportes con Funciones.")
```

4 Aplicaciones profesionales de lo aprendido

- Automatizar cálculos y reportes de negocio.
- Estructurar scripts de automatización de ventas o inventario.
- Preparación para usar funciones en proyectos con APIs y bases de datos.
- Uso de funciones para limpieza de datos y procesos de OCR.

5 Resumen de preguntas y respuestas de la clase

✅ **¿Por qué usar funciones?** Para reutilizar código, mantener orden y facilitar automatización. ✅ **¿Qué es `**?`** Devuelve un resultado para su uso fuera de la función. ✅ ****?** Por qué indentamos después de `**`? Python lo requiere para entender el bloque de la función. ✅ **¿Puedo llamar una función varias veces?** Sí, las funciones pueden reutilizarse con diferentes valores. ✅ **¿Qué sucede si no coloco ```** La función no devuelve un valor, solo ejecuta las instrucciones internas.

✅ Cierre de la Clase 6

- Comprendes la creación y uso de funciones.
- Realizaste ejercicios aplicando funciones en casos prácticos.
- Estás preparado para avanzar a **Clase 7: Estructuras de Datos (listas, tuplas, diccionarios, sets)**.

¡Excelente trabajo, Gabo! Vas en camino firme a ser un programador profesional, estructurando código real para tus proyectos. 🚫 NEW

Clase 7.5 – Proyecto Integrador con Menú y Persistencia (COMPLETA)

Incluye:

- Teoría completa de listas, tuplas, diccionarios y sets con explicaciones claras.
- Scripts de todos los mini ejercicios.
- Ejercicio Integrador completo con menú interactivo y funciones.
- Preguntas y respuestas de Gabo con explicaciones.
- Preparado sin resumir para estudio directo en lienzo.

Teoría:

- Listas: colecciones ordenadas, mutables, uso de `.append()` y `.remove()`.
- Tuplas: colecciones ordenadas, inmutables, acceso por índices.
- Diccionarios: pares clave\valor, uso de `.items()` y actualización de valores.
- Sets: colecciones desordenadas sin duplicados, uso de `.add()` y `.remove()`.

Scripts:

Mini Ejercicio Listas:

```
tareas = []
for i in range(3):
    tarea = input(f"Tarea {i+1}: ")
    tareas.append(tarea)
tarea_eliminar = input("¿Desea eliminar alguna tarea?: ")
if tarea_eliminar in tareas:
    tareas.remove(tarea_eliminar)
print(tareas)
```

Mini Ejercicio Tuplas:

```
dias = ("lunes", "martes", "miércoles", "jueves", "viernes", "sábado",
"domingo")
print(dias[0])
print(dias[-1])
```

Mini Ejercicio Diccionarios:

```
usuario = {"nombre": "Gabo", "edad": 26, "pais": "Cuba"}
for k, v in usuario.items():
    print(f"{k.capitalize()}: {v}")
nuevo_pais = input("Nuevo país: ")
usuario["pais"] = nuevo_pais
print(usuario)
```

Mini Ejercicio Sets:

```
prioridades = {"alta", "media", "baja"}
nueva_etiqueta = input("Nueva etiqueta: ")
prioridades.add(nueva_etiqueta)
eliminar = input("¿Desea eliminar una etiqueta?: ")
if eliminar in prioridades:
    prioridades.remove(eliminar)
print(prioridades)
```

Ejercicio Final Integrador Clase 7.5 (COMPLETO)

```
# Proyecto Integrador Clase 7.5 con Menú Interactivo

# Datos iniciales
tareas = []
dias_semana = ("lunes", "martes", "miércoles", "jueves", "viernes", "sábado",
"domingo")
usuario = {"nombre": "Gabriel", "edad": 27, "pais": "Rusia", "ciudad": "San
Petersburgo"}
prioridades = {"alta", "media", "baja"}

# Funciones
def gestionar_tareas():
    print("\n📌 Gestión de Tareas")
    for i in range(3):
        tarea = input(f"Ingrese la tarea {i+1}: ")
        tareas.append(tarea)
    eliminar = input("¿Desea eliminar una tarea? (si/no): ").lower()
    if eliminar in ["si", "sí"]:
        tarea_eliminar = input("Ingrese la tarea a eliminar: ")
        if tarea_eliminar in tareas:
            tareas.remove(tarea_eliminar)
            print(f"🗑️ '{tarea_eliminar}' eliminada correctamente.")
```

```

        else:
            print("🔍 Tarea no encontrada.")
print("\n🔗 Lista final de tareas:")
for t in tareas:
    print(f"- {t}")

def seleccionar_dia():
    print("\n📌 Selección del Día")
    for idx, dia in enumerate(dias_semana):
        print(f"{idx+1}. {dia.capitalize()}")
    seleccion = int(input("Seleccione el número del día: "))
    if 1 <= seleccion <= 7:
        print(f"🔗 Día seleccionado: {dias_semana[seleccion - 1].capitalize()}")
    else:
        print("🔍 Selección inválida.")

def actualizar_usuario():
    print("\n📌 Datos del Usuario")
    for k, v in usuario.items():
        print(f"- {k.capitalize(): {v}")
    modificacion_pais = input("\n¿Has cambiado de país? (si/no): ").lower()
    if modificacion_pais in ["si", "sí"]:
        nuevo_pais = input("Ingrese el nuevo país: ")
        usuario["pais"] = nuevo_pais
        print(f"🔗 País actualizado a {nuevo_pais}.")
    modificacion_ciudad = input("¿Has cambiado de ciudad? (si/no): ").lower()
    if modificacion_ciudad in ["si", "sí"]:
        nueva_ciudad = input("Ingrese la nueva ciudad: ")
        usuario["ciudad"] = nueva_ciudad
        print(f"🔗 Ciudad actualizada a {nueva_ciudad}.")
    print("\n🔗 Datos actuales:")
    for k, v in usuario.items():
        print(f"- {k.capitalize(): {v}")

def gestionar_prioridades():
    print("\n📌 Gestión de Prioridades")
    print("Prioridades actuales:")
    for p in prioridades:
        print(f"- {p.capitalize()}")
    nueva = input("Ingrese una nueva prioridad: ").lower()
    prioridades.add(nueva)
    eliminar = input("¿Desea eliminar una prioridad? (si/no): ").lower()
    if eliminar in ["si", "sí"]:
        eliminar_p = input("Ingrese la prioridad a eliminar: ").lower()
        if eliminar_p in prioridades:
            prioridades.remove(eliminar_p)
            print(f"🔗 Prioridad '{eliminar_p}' eliminada.")
    else:

```

```


        print("🔍 Prioridad no encontrada.")
    print("\n🔗 Prioridades actuales:")
    for p in prioridades:
        print(f"- {p.capitalize()}")

def ver_resumen():
    print("\n📌 Resumen General")
    print("Tareas:")
    for t in tareas:
        print(f"- {t}")
    print("\nUsuario:")
    for k, v in usuario.items():
        print(f"- {k.capitalize(): {v}")
    print("\nPrioridades:")
    for p in prioridades:
        print(f"- {p.capitalize()}")

# Menú Principal
while True:
    print("""
    === Menú Principal ===
    [1] Gestionar lista de tareas
    [2] Seleccionar día de la semana
    [3] Ver o actualizar datos de usuario
    [4] Gestionar etiquetas de prioridad
    [5] Ver resumen de todo
    [0] Salir
    """)
    opcion = input("Seleccione una opción: ")
    if opcion == "1":
        gestionar_tareas()
    elif opcion == "2":
        seleccionar_dia()
    elif opcion == "3":
        actualizar_usuario()
    elif opcion == "4":
        gestionar_prioridades()
    elif opcion == "5":
        ver_resumen()
    elif opcion == "0":

print("🔗 Gracias por usar el proyecto integrador, Gabo. ¡Has completado la Clase 7.5!")
    break
else:
    print("🔍 Opción inválida. Intente de nuevo.")

```


 Entrega solicitada por Gabo: Scripts completos y explicaciones de la Clase 8 y Clase 9 en formato organizado y listo para estudio.

Clase 8 – Manejo de Archivos y Persistencia de Datos

Descripción general:

En esta clase aprendiste:

- Qué es persistencia de datos.
- Lectura y escritura de archivos `.txt` en Python.
- Uso de `open()`, `.read()`, `.write()`, `.writelines()`, `.readlines()`.
- Uso de `"r"`, `"w"`, `"a"` en modo de apertura de archivos.
- Uso de `utf-8` para evitar errores con caracteres especiales.
- Estructuración de un mini proyecto con persistencia real.

Script completo de Clase 8: Mini Proyecto Gestor de Tareas con Persistencia

```
archivo_tareas = "tareas_gabo.txt"

def agregar_tarea():
    tarea = input("Ingrese la tarea: ")
    with open(archivo_tareas, "a", encoding="utf-8") as archivo:
        archivo.write(f"P|{tarea}\n")
    print(f"    Tarea '{tarea}' agregada correctamente.")

def ver_tareas():
    try:
        with open(archivo_tareas, "r", encoding="utf-8") as archivo:
            tareas = archivo.readlines()
        if tareas:
            for idx, linea in enumerate(tareas, 1):
                estado, tarea = linea.strip().split("|")
                estado_legible = "Pendiente" if estado == "P" else "Completada"
                print(f"{idx}. [{estado_legible}] {tarea}")
            else:
                print("    No hay tareas registradas.")
    except FileNotFoundError:
        print("    No existe el archivo de tareas aún. Agregue una tarea primero.")
```

```

def eliminar_tarea():
    try:
        with open(archivo_tareas, "r", encoding="utf-8") as archivo:
            tareas = archivo.readlines()
        if tareas:
            for idx, linea in enumerate(tareas, 1):
                estado, tarea = linea.strip().split("|")
                estado_legible = "Pendiente" if estado == "P" else "Completada"
                print(f"{idx}. [{estado_legible}] {tarea}")
            numero = int(input("Ingrese el número de la tarea a eliminar: "))
            if 1 <= numero <= len(tareas):
                tarea_eliminada = tareas.pop(numero - 1)
                with open(archivo_tareas, "w", encoding="utf-8") as archivo:
                    archivo.writelines(tareas)
                print(f"    Tarea eliminada: {tarea_eliminada.strip()}")
            else:
                print("    Número inválido.")
        else:
            print("    No hay tareas para eliminar.")
    except FileNotFoundError:
        print("    No existe el archivo de tareas aún.")

def marcar_completada():
    try:
        with open(archivo_tareas, "r", encoding="utf-8") as archivo:
            tareas = archivo.readlines()
        if tareas:
            for idx, linea in enumerate(tareas, 1):
                estado, tarea = linea.strip().split("|")
                estado_legible = "Pendiente" if estado == "P" else "Completada"
                print(f"{idx}. [{estado_legible}] {tarea}")
            numero = int(input("Ingrese el número de la tarea a marcar como
completada: "))
            if 1 <= numero <= len(tareas):
                estado, tarea = tareas[numero - 1].strip().split("|")
                if estado == "P":
                    tareas[numero - 1] = f"C|{tarea}\n"
                    with open(archivo_tareas, "w", encoding="utf-8") as archivo:
                        archivo.writelines(tareas)
                    print(f"    Tarea marcada como completada: {tarea}")
                else:
                    print(f"    La tarea '{tarea}' ya estaba completada.")
            else:
                print("    Número inválido.")
        else:
            print("    No hay tareas registradas.")
    except FileNotFoundError:

```

```

        print("    No existe el archivo de tareas aún.")

def ver_estadisticas():
    try:
        with open(archivo_tareas, "r", encoding="utf-8") as archivo:
            tareas = archivo.readlines()
            total = len(tareas)
            completadas = sum(1 for linea in tareas if linea.startswith("C|"))
            pendientes = sum(1 for linea in tareas if linea.startswith("P|"))
            print("\n=== Estadísticas de Tareas ===")
            print(f"Total de tareas: {total}")
            print(f"Tareas completadas: {completadas}")
            print(f"Tareas pendientes: {pendientes}")
    except FileNotFoundError:
        print("    No existe el archivo de tareas aún.")

while True:
    print("""
=== Gestor de Tareas de Gabo ===
[1] Agregar tarea
[2] Ver tareas
[3] Eliminar tarea
[4] Marcar tarea como completada
[5] Ver estadísticas
[0] Salir
""")
    opcion = input("Seleccione una opción: ")

    if opcion == "1":
        agregar_tarea()
    elif opcion == "2":
        ver_tareas()
    elif opcion == "3":
        eliminar_tarea()
    elif opcion == "4":
        marcar_completada()
    elif opcion == "5":
        ver_estadisticas()
    elif opcion == "0":
        print("    Saliendo del gestor de tareas. ¡Buen trabajo, Gabo!")
        break
    else:
        print("    Opción inválida. Intente de nuevo.")

```

Clase 9 – Proyecto Integrador Módulo 1

Descripción general:

En esta clase aprendiste:

- Cómo estructurar proyectos grandes con Python.
- Aplicar estructuras de control, funciones, persistencia y modularidad.
- Uso de listas y manejo de archivos.
- Creación de menús interactivos.
- Cómo simular aplicaciones de escritorio en consola con persistencia de datos.

Proyecto realizado: Gestor de Tareas Mejorado con Persistencia y Estadísticas.

Script completo de Clase 9

(Este es el que me enviaste y corregimos juntos, preparado para que lo estudies de forma limpia)

```
# === Proyecto Integrador Clase 9 ===
# Gestor de Tareas Mejorado con Persistencia

archivo_tareas = "tareas_gabo.txt"

def agregar_tareas():
    tarea = input("Ingrese la tarea: ")
    with open(archivo_tareas, "a", encoding="utf-8") as archivo:
        archivo.write(f"P|{tarea}\n")
    print(f"    Tarea '{tarea}' agregada correctamente.")

def ver_tarea():
    try:
        with open(archivo_tareas, "r", encoding="utf-8") as archivo:
            tareas = archivo.readlines()
        if tareas:
            for idx, linea in enumerate(tareas, 1):
                estado, tarea = linea.strip().split("|")
                estado_legible = "Pendiente" if estado == "P" else "Completada"
                print(f"{idx}. [{estado_legible}] {tarea}")
        else:
            print("    No hay tareas registradas.")
    except FileNotFoundError:
        print("    No existe el archivo aún. Agregue una tarea primero.")

def eliminar_tarea():
```

```

try:
    with open(archivo_tareas, "r", encoding="utf-8") as archivo:
        tareas = archivo.readlines()
    if tareas:
        for idx, linea in enumerate(tareas, 1):
            estado, tarea = linea.strip().split("|")
            estado_legible = "Pendiente" if estado == "P" else "Completada"
            print(f"{idx}. [{estado_legible}] {tarea}")
            numero = int(input("Ingrese el número de la tarea a eliminar: "))
            if 1 <= numero <= len(tareas):
                tarea_eliminada = tareas.pop(numero - 1)
                with open(archivo_tareas, "w", encoding="utf-8") as archivo:
                    archivo.writelines(tareas)
                print(f"    Tarea eliminada: {tarea_eliminada.strip()}")
            else:
                print("    Número inválido.")
        else:
            print("    No hay tareas para eliminar.")
except FileNotFoundError:
    print("    No existe el archivo de tareas aún.")

def marcar_completadas():
    try:
        with open(archivo_tareas, "r", encoding="utf-8") as archivo:
            tareas_completadas = archivo.readlines()
        if tareas_completadas:
            for idx, linea in enumerate(tareas_completadas, 1):
                estado, tarea = linea.strip().split("|")
                estado_legible = "Pendiente" if estado == "P" else "Completada"
                print(f"{idx}. [{estado_legible}] {tarea}")
                numero = int(input("Ingrese el número de la tarea a marcar como
completada: "))
                if 1 <= numero <= len(tareas_completadas):
                    estado, tarea = tareas_completadas[numero -
1].strip().split("|")
                    if estado == "P":
                        tareas_completadas[numero - 1] = f"C|{tarea}\n"
                        with open(archivo_tareas, "w", encoding="utf-8") as archivo:
                            archivo.writelines(tareas_completadas)
                        print(f"    Tarea marcada como completada: {tarea}")
                    else:
                        print(f"    La tarea '{tarea}' ya estaba completada.")
                else:
                    print("    Número inválido.")
            else:
                print("    No hay tareas registradas.")
        except FileNotFoundError:
            print("    No existe el archivo de tareas aún.")

```

```

def ver_estadisticas():
    try:
        with open(archivo_tareas, "r", encoding="utf-8") as archivo:
            tareas = archivo.readlines()
            total = len(tareas)
            completadas = sum(1 for linea in tareas if linea.startswith("C|"))
            pendientes = sum(1 for linea in tareas if linea.startswith("P|"))
            print("\n=== Estadísticas de Tareas ===")
            print(f"Total de tareas: {total}")
            print(f"Tareas completadas: {completadas}")
            print(f"Tareas pendientes: {pendientes}")
    except FileNotFoundError:
        print("    No existe el archivo aún.")

while True:
    print("""
=== Gestor de Tareas de Gabo ===
[1] Agregar tarea
[2] Ver tareas
[3] Eliminar tarea
[4] Marcar tarea como completada
[5] Ver estadísticas
[0] Salir
""")
    opcion = input("Seleccione una opción: ")

    if opcion == "1":
        agregar_tareas()
    elif opcion == "2":
        ver_tarea()
    elif opcion == "3":
        eliminar_tarea()
    elif opcion == "4":
        marcar_completadas()
    elif opcion == "5":
        ver_estadisticas()
    elif opcion == "0":
        print("    Saliendo del gestor de tareas. ¡Buen trabajo, Gabo!")
        break
    else:
        print("    Opción inválida. Intente de nuevo.")

```