

# UNIVERSIDAD GERARDO BARRIOS – SEDE CENTRAL SAN MIGUEL FACULTAD DE CIENCIA Y TECNOLOGÍA

Actividad Sumativa – Laboratorio 2 (30%) Primer Cómputo – Semana 4

Objetivo: Que el estudiante identifique y aplique los principios de la Programación Orientada a Objetos (POO) en sus programas, demostrando su capacidad para diseñar y desarrollar soluciones efectivas a problemas reales

Indicaciones: Realizar la presente guía de ejercicios de manera individual.

1. Realizar una investigación bibliográfica sobre el siguiente libro: Muñoz Guerrero, L. E. (Il.) & Trejos Buriticá, O. I. (2021). Introducción a la programación con Python (1ª ed.). RA-MA Editorial. El libro está disponible en la biblioteca de la universidad o en línea a través del siguiente enlace: <https://elibro.net/es/lc/bibliotecaugb/titulos/230298> .
2. Completar los ejercicios del taller de repaso ubicados en la lección 10 del libro.
3. Diseñar un ejercicio que aplique los principios de la Programación Orientada a Objetos (POO), con su respectiva implementación e instanciación. El ejercicio debe permitir la entrada de datos por teclado. Además, proporciona una explicación detallada sobre qué hace el programa y qué problema resuelve, basado en una situación problemática de tu entorno.

# \*Los ejercicios irán en archivos separados y deberán subirse a un repositorio en GitHub.

**RÚBRICA DE EVALUACIÓN**

* + Ejercicios del libro realizados y sin errores – 5 puntos
  + Ejercicio basado en una situación problemática con su respectiva explicación - 4 puntos
  + Código comentado - 1 punto

**1. Investigación Bibliográfica del Libro:**

**Referencia del Libro:**

* **Título:** Introducción a la programación con Python
* **Autores:** Muñoz Guerrero, L. E. (Il.) & Trejos Buriticá, O. I.
* **Edición:** 1ª edición, 2021
* **Editorial:** RA-MA Editorial

**Contenido del Libro:** Este libro es una introducción completa a la programación con Python, un lenguaje ampliamente utilizado tanto en la industria como en el ámbito académico. Cubre desde los conceptos básicos de programación hasta temas más avanzados como la Programación Orientada a Objetos (POO). El texto es adecuado para principiantes y proporciona numerosos ejemplos y ejercicios prácticos para reforzar el aprendizaje.

**Estructura del Libro:**

1. **Fundamentos de la Programación**: Explicación de los conceptos básicos de programación.
2. **Sintaxis y Estructuras de Control en Python**: Descripción de cómo se usan los bucles, condicionales, y funciones en Python.
3. **Estructuras de Datos**: Cobertura de listas, tuplas, diccionarios y conjuntos.
4. **Programación Orientada a Objetos (POO)**: Introducción a los principios de POO y su implementación en Python.
5. **Ejercicios y Talleres**: Cada capítulo termina con ejercicios para reforzar los conceptos aprendidos, incluyendo talleres de repaso, como el ubicado en la lección 10.

**2. Completar los Ejercicios del Taller de Repaso (Lección 10):**

1. **Veterinaria que Atiende Perros**

**class Perro:**

**def \_\_init\_\_(self, nombre, raza, edad, peso, color, sexo, vacunado, chip):**

**self.nombre = nombre**

**self.raza = raza**

**self.edad = edad**

**self.peso = peso**

**self.color = color**

**self.sexo = sexo**

**self.vacunado = vacunado**

**self.chip = chip**

**self.estado = "NO ATENDIDO"**

**self.tamano = "Perro Grande" if peso >= 10 else "Perro Pequeño"**

**def registrar(self):**

**self.estado = "ATENDIDO"**

**self.mostrar\_informacion()**

**def mostrar\_informacion(self):**

**print(f"Nombre: {self.nombre}")**

**print(f"Raza: {self.raza}")**

**print(f"Edad: {self.edad} años")**

**print(f"Peso: {self.peso} kg ({self.tamano})")**

**print(f"Color: {self.color}")**

**print(f"Sexo: {self.sexo}")**

**print(f"Vacunado: {'Sí' if self.vacunado else 'No'}")**

**print(f"Chip: {'Sí' if self.chip else 'No'}")**

**print(f"Estado: {self.estado}")**

**# Ejemplo de uso**

**perro1 = Perro("Max", "Labrador", 5, 12, "Negro", "Macho", True, True)**

**perro1.registrar()**

1. **Papelería que Vende Cuadernos y Lápices**

**class ArticuloPapeleria:**

**def \_\_init\_\_(self, tipo, caracteristicas, precio\_compra):**

**self.tipo = tipo**

**self.caracteristicas = caracteristicas**

**self.precio\_compra = precio\_compra**

**self.precio\_venta = precio\_compra \* 1.3**

**def mostrar\_informacion(self):**

**print(f"Tipo: {self.tipo}")**

**print(f"Características: {self.caracteristicas}")**

**print(f"Precio de compra: {self.precio\_compra}")**

**print(f"Precio de venta: {self.precio\_venta}")**

**# Ejemplo de uso**

**cuaderno = ArticuloPapeleria("Cuaderno HOJITAS", "100 hojas", 3.5)**

**lapiz = ArticuloPapeleria("Lápiz RAYAS", "Grafito", 0.5)**

**cuaderno.mostrar\_informacion()**

**lapiz.mostrar\_informacion()**

1. **Concesionario de Autos**

**class Auto:**

**def \_\_init\_\_(self, marca, modelo, anio, tipo, transmision, color, motor, combustible, nacionalidad, precio\_compra):**

**self.marca = marca**

**self.modelo = modelo**

**self.anio = anio**

**self.tipo = tipo**

**self.transmision = transmision**

**self.color = color**

**self.motor = motor**

**self.combustible = combustible**

**self.nacionalidad = nacionalidad**

**self.precio\_compra = precio\_compra**

**self.precio\_venta = precio\_compra \* 1.4**

**self.ruedas = 4**

**self.capacidad\_pasajeros = 5**

**def mostrar\_informacion(self):**

**print(f"Marca: {self.marca}")**

**print(f"Modelo: {self.modelo}")**

**print(f"Año: {self.anio}")**

**print(f"Tipo: {self.tipo}")**

**print(f"Transmisión: {self.transmision}")**

**print(f"Color: {self.color}")**

**print(f"Motor: {self.motor}")**

**print(f"Combustible: {self.combustible}")**

**print(f"Nacionalidad: {self.nacionalidad}")**

**print(f"Precio de compra: {self.precio\_compra}")**

**print(f"Precio de venta: {self.precio\_venta}")**

**print(f"Ruedas: {self.ruedas}")**

**print(f"Capacidad de pasajeros: {self.capacidad\_pasajeros}")**

**# Ejemplo de uso**

**auto1 = Auto("Toyota", "Corolla", 2021, "Sedán", "Automática", "Blanco", "1.8L", "Gasolina", "Importado", 15000)**

**auto1.mostrar\_informacion()**

1. **Almacén de Dispositivos Electrónicos**

**class DispositivoElectronico:**

**def \_\_init\_\_(self, tipo, modelo, marca, almacenamiento, ram, procesador, precio\_compra):**

**self.tipo = tipo**

**self.modelo = modelo**

**self.marca = marca**

**self.almacenamiento = almacenamiento**

**self.ram = ram**

**self.procesador = procesador**

**self.precio\_compra = precio\_compra**

**self.precio\_venta = precio\_compra \* 1.7**

**self.importado = True**

**def mostrar\_informacion(self):**

**print(f"Tipo: {self.tipo}")**

**print(f"Modelo: {self.modelo}")**

**print(f"Marca: {self.marca}")**

**print(f"Almacenamiento: {self.almacenamiento}")**

**print(f"RAM: {self.ram}")**

**print(f"Procesador: {self.procesador}")**

**print(f"Precio de compra: {self.precio\_compra}")**

**print(f"Precio de venta: {self.precio\_venta}")**

**print(f"Importado: {self.importado}")**

**# Ejemplo de uso**

**celular = DispositivoElectronico("Celular", "Galaxy S21", "PHR", "128GB", "8GB", "Exynos 2100", 600)**

**celular.mostrar\_informacion()**

**3. Diseño de un Ejercicio Basado en POO:**

**Ejercicio Propuesto: Sistema de Gestión de Tareas**

* **Descripción del Programa:** Este programa simula un sistema simple de gestión de tareas. El usuario puede crear tareas, marcar tareas como completadas y ver la lista de tareas pendientes y completadas.
* **Problema que Resuelve:** En un entorno donde se manejan múltiples tareas (por ejemplo, en un proyecto de desarrollo de software o en la administración de un hogar), este programa permite al usuario organizar y realizar un seguimiento de sus tareas de manera eficiente.

Implementación del Ejercicio en Python: class Tarea:

def \_\_init\_\_(self, titulo, descripcion):

self.titulo = titulo

self.descripcion = descripcion

self.completada = False

def marcar\_completada(self):

self.completada = True

class GestorDeTareas:

def \_\_init\_\_(self):

self.tareas = []

def agregar\_tarea(self, titulo, descripcion):

nueva\_tarea = Tarea(titulo, descripcion)

self.tareas.append(nueva\_tarea)

def mostrar\_tareas(self):

print("Tareas pendientes:")

for tarea in self.tareas:

if not tarea.completada:

print(f"- {tarea.titulo}: {tarea.descripcion}")

print("\nTareas completadas:")

for tarea in self.tareas:

if tarea.completada:

print(f"- {tarea.titulo}: {tarea.descripcion}")

def marcar\_completada(self, titulo):

for tarea in self.tareas:

if tarea.titulo == titulo:

tarea.marcar\_completada()

print(f"Tarea '{titulo}' marcada como completada.")

return

print(f"No se encontró una tarea con el título '{titulo}'.")

# Ejemplo de uso:

gestor = GestorDeTareas()

gestor.agregar\_tarea("Estudiar para el examen", "Repasar los capítulos 4 y 5")

gestor.agregar\_tarea("Comprar comida", "Comprar frutas y vegetales")

# Mostrar las tareas antes de marcar alguna como completada

gestor.mostrar\_tareas()

# Marcar una tarea como completada

gestor.marcar\_completada("Estudiar para el examen")

# Mostrar las tareas después de marcar una como completada

gestor.mostrar\_tareas()