UES

Universidad Estatal de Sonora

La Fuerza del Saber Estimulará mi Espíritu

MANUAL DE PRÁCTICAS DE LABORATORIO

Seminario de Programación

Laboratorio de Programación

Programa Académico	**Ing. en Software**
Plan de Estudios	
Fecha de elaboración	**30/06/2025**
Versión del Documento	**1.0**

UES
Universidad Estatal de Sonora
La Fuerza del Saber Estimulará mi Espíritu

Dra. Martha Patricia Patiño Fierro
Rectora

Mtra. Ana Lisette Valenzuela Molina
Encargada del Despacho de la Secretaría General Académica

Mtro. José Antonio Romero Montaño
Secretario General Administrativo

Lic. Jorge Omar Herrera Gutiérrez
Encargado de Despacho de Secretario General de Planeación

Tabla de contenido

INTRODUCCIÓN

Como parte de las herramientas esenciales para la formación académica de los estudiantes de la Universidad Estatal de Sonora, se definen manuales de práctica de laboratorio como elemento en el cual se define la estructura normativa de cada práctica y/o laboratorio, además de representar una guía para la aplicación práctica del conocimiento y el desarrollo de las competencias clave en su área de estudio. Su diseño se encuentra alineado con el modelo educativo institucional, el cual privilegia el aprendizaje basado en competencias, el aprendizaje activo y la conexión con escenarios reales.

Con el propósito de fortalecer la autonomía de los estudiantes, su pensamiento crítico y sus habilidades para la resolución de problemas, las prácticas de laboratorio integran estrategias didácticas como el aprendizaje basado en proyectos, el trabajo colaborativo, la experimentación guiada y el uso de tecnologías educativas. De esta manera, se promueve un proceso de enseñanza-aprendizaje dinámico, en el que los estudiantes no solo adquieren conocimientos teóricos, sino que también desarrollan habilidades prácticas y reflexivas para su desempeño profesional.

- Propósito del manual
  El propósito del manual de laboratorio es proporcionar una guía estructurada para el desarrollo de prácticas relacionadas con la programación en Python, el manejo de bases de datos (SQLite y MySQL), el desarrollo web con Flask y la creación de interfaces gráficas. Estas prácticas buscan fortalecer la comprensión teórica y técnica de los estudiantes, fomentar el pensamiento lógico y computacional, y aplicar conocimientos en escenarios simulados y reales, alineados al perfil de egreso del programa académico.

- Justificación de su uso en el programa académico
  El manual se justifica como herramienta didáctica esencial que facilita el desarrollo de competencias profesionales, disciplinares y blandas requeridas por los estudiantes del programa de Ingeniería en Software. A través de experiencias prácticas, se promueve el aprendizaje activo, significativo y por competencias, en concordancia con el enfoque educativo de la UES. Permite además evaluar el desempeño de manera objetiva mediante rúbricas institucionales, garantizando la calidad y pertinencia del proceso de enseñanza-aprendizaje.

- Competencias a desarrollar
  - **Competencias blandas:**
    Ética profesional en el desarrollo de software.
    Responsabilidad en la ejecución y entrega de prácticas.

Trabajo colaborativo en el desarrollo de soluciones tecnológicas.
Uso eficiente de tecnologías de información y herramientas de programación.
Habilidades de comunicación escrita (reportes técnicos).

- **Competencias disciplinares:**

  Conocimiento y aplicación de la sintaxis y semántica del lenguaje Python.
  Manejo de estructuras de control, funciones, módulos y programación orientada a objetos.
  Diseño e implementación de bases de datos relacionales con SQLite y MySQL.
  Desarrollo de aplicaciones web usando el framework Flask.
  Creación de interfaces gráficas para interacción con el usuario.

- **Competencias profesionales:**

  Diseño e implementación de soluciones móviles y web alineadas a las necesidades organizacionales.
  Aplicación de técnicas de programación para el procesamiento de datos y automatización de procesos.
  Desarrollo de proyectos tecnológicos integradores con enfoque de innovación.
  Resolución de problemas reales mediante el uso de tecnologías emergentes.

IDENTIFICACIÓN

Nombre de la Asignatura		Seminario de Programación	
Clave	061CE047	Créditos	6
Asignaturas Antecedentes		Plan de Estudios	2021

Área de Competencia	Competencia del curso
Implementar soluciones e innovaciones tecnológicas, con el fin de contribuir a una planeación responsable de los recursos humanos, tecnológicos y financieros, con base en las necesidades, la problemática analizada y los estándares de calidad establecidos por la organización	Crear las nuevas tecnologías emergentes de computación móvil para evaluar, planificar e implementar soluciones móviles en las organizaciones bajo los estándares de calidad establecidos para el desarrollo de aplicaciones móviles con un enfoque de innovación y liderazgo

Carga Horaria de la asignatura

Horas Supervisadas			Horas Independientes	Total de Horas
Aula	Laboratorio	Plataforma		
3	1	1	2	7

Consignación del Documento

Unidad Académica	Unidad Académica Navojoa
Fecha de elaboración	30/06/2025
Responsables del diseño	Francisco Alan Espinoza Zallas
Validación	
Recepción	Coordinación de Procesos Educativos

NORMAS DE SEGURIDAD Y BUENAS PRÁCTICAS

Reglamento general del laboratorio

Acceso a Laboratorio de Programación

- El acceso a las salas de trabajo será sólo a las horas que no haya clases.
- Deberá entrar sólo con sus documentos de trabajo, el resto de sus cosas deberá dejarlo a la entrada del laboratorio de cómputo.
- No deberá entrar a las salas de trabajo con alimentos, bebidas o fumando.
- Queda estrictamente prohibido introducir animales.

Permanencia

- Antes de trabajar se debe revisar el equipo de cómputo donde fue designado y reportar alguna anomalía al encargado de la sala; si posteriormente se detecta alguna falla no reportada, se le responsabilizará sobre ésta.
- No deberá mover el equipo de cómputo ni mobiliario de su lugar bajo ningún motivo; verifique en su apartado el equipo que va a necesitar para su trabajo.
- El usuario que dañe el equipo de cómputo, sus instalaciones o cambie su configuración del software, incluso protectores de pantalla o colores de ventanas quedará responsabilizado de pagar su costo de reparación o adquisición.
- El comportamiento de todo usuario no debe ir en contra de la moral y las buenas costumbres dentro de las diferentes salas de cómputo.
- El usuario deberá mantener limpia su área de trabajo.
- El uso del equipo de cómputo es exclusivamente para fines didácticos y de investigación, por lo que se prohíbe el uso de juegos, trabajos personales ó de terceros con fines de lucro.
- Solo se permitirá el trabajo individual por computadora, a excepciones de aquellos cursos ó clases impartidas por instructores ó profesores.
- Los usuarios no podrán ingresar ningún tipo de radio, grabadora ó elementos que produzcan ruido y/o elementos magnéticos.
- Los estudiantes deben guardar respeto mantener los canales de comunicación y demás normas establecidas con los monitores y responsables del centro de cómputo y estos para con ellos.
- El acceso a los laboratorios es para el usuario que haya solicitado el servicio, bajo ninguna circunstancia podrán estar dos personas trabajando conjuntamente en una computadora.

Uso adecuado del equipo y materiales

- El equipo periférico que se presta, es solo para uso durante la sesión de trabajo por lo que deberá de devolverlo al término de esta.
- Para asesorías ó dudas con respecto al uso del equipo, debe dirigirse al responsable en turno dentro de la sala donde se encuentre ó bien dirigirse al personal encargado del laboratorio
- El usuario no debe desconectar ni destapar los ratones de las computadoras, las impresoras ó cualquier equipo periférico que se tenga instalado.
- Es obligación del usuario saber operar el equipo de cómputo y periféricos; debe consultar con el personal del servicio social o leer los folletos sobre el uso de periféricos distribuidos para tal efecto; ya que el daño por el mal uso será responsabilidad del usuario.
- En caso de necesitar tóner la impresora notifique al responsable en turno ó bien al encargado del laboratorio, para que el la verifique y éste sea remplazado.

Procedimientos en caso de emergencia

- Evacuación ordenada por rutas señalizadas.
- Uso de extintores solo por personal capacitado.
- Reporte inmediato de accidentes al personal responsable.

MATRIZ DE CORRESPONDENCIA

Señalar la relación de cada práctica con las competencias del perfil de egreso

PRÁCTICA	PERFIL DE EGRESO
Práctica No. 1 Instalación de Python	Desarrollar software con la finalidad de agilizar los procesos y la toma de decisiones en empresas públicas y privadas, bajo estándares de calidad nacional e internacional con enfoque de liderazgo.
Práctica No. 2 Programación de estructuras repetitivas en Python	Desarrollar software con la finalidad de agilizar los procesos y la toma de decisiones en empresas públicas y privadas, bajo estándares de calidad nacional e internacional con enfoque de liderazgo.
Práctica No. 3 Programación de módulos en Python	Aplicar soluciones e innovaciones tecnológicas con la finalidad de automatizar los procesos, atendiendo los principios de la organización y gestión efectiva de la información.
Práctica No. 4 Conexión, creación y desconexión a base de datos con SQLite en Python	Crear bases de datos para una gestión eficiente de la información, garantizando la integridad y seguridad de los datos, atendiendo los requerimientos de la organización con un sentido de liderazgo e innovación.
Práctica No. 5 Consultas SQL básicas con SQLite en Python	Crear bases de datos para una gestión eficiente de la información, garantizando la integridad y seguridad de los datos, atendiendo los requerimientos de la organización con un sentido de liderazgo e innovación.
Práctica No. 6 Consultas SQL intermedias con SQLite en Python	Crear bases de datos para una gestión eficiente de la información, garantizando la integridad y seguridad de los datos, atendiendo los requerimientos de la organización con un sentido de liderazgo e innovación.
Práctica No. 7 Operaciones CRUD con MySQL desde Python	Crear bases de datos para una gestión eficiente de la información, garantizando la integridad y seguridad de los datos, atendiendo los requerimientos de la organización con un sentido de liderazgo e innovación.
Práctica No. 8 Conexión, creación y desconexión a base de datos con SQLite en Python	Crear bases de datos para una gestión eficiente de la información, garantizando la integridad y seguridad de los datos, atendiendo los requerimientos de la organización con un sentido de liderazgo e innovación.
Práctica No. 9 Instalación del framework Flask y primera app	Aplicar soluciones e innovaciones tecnológicas con la finalidad de automatizar los procesos, atendiendo los principios de la organización y

	gestión efectiva de la información.
Práctica No. 10 Desarrollo de sitio web con Flask	Desarrollar software con la finalidad de agilizar los procesos y la toma de decisiones en empresas públicas y privadas, bajo estándares de calidad nacional e internacional con enfoque de liderazgo.
Práctica No. 11 Aplicaciones CRUD con Flask	Desarrollar software con la finalidad de agilizar los procesos y la toma de decisiones en empresas públicas y privadas, bajo estándares de calidad nacional e internacional con enfoque de liderazgo.

RELACIÓN DE PRÁCTICAS DE LABORATORIO POR ELEMENTO DE COMPETENCIA

Elemento de Competencia al que pertenece la práctica	EC I
	Identificar la sintaxis y semántica del lenguaje Python para desarrollar programas de cómputo que den solución a problemas de procesamiento de información, con una actitud ética y profesional

PRÁCTICA	NOMBRE	COMPETENCIA
Práctica No. 1	Instalación de Python	Instalar el entorno de desarrollo Python con la finalidad de conocer la interfaz IDLE, siguiendo los pasos indicados por el facilitador, en el contexto del aula, con ética.
Práctica No. 2	Programación de estructuras repetitivas en Python	Desarrollar programas que implementen estructuras repetitivas con la finalidad de resolver problemas simples, utilizando las estructuras while y for, en el laboratorio, con responsabilidad.
Práctica No. 3	Programación de módulos en Python	Construir módulos en Python con la finalidad de estructurar el código en componentes reutilizables, siguiendo el planteamiento del facilitador, en el laboratorio, con organización y responsabilidad.

Elemento de Competencia al que pertenece la práctica	EC II
	Diseñar bases de Datos relacionales para el desarrollo de sistemas de información que permitan dar solución a necesidades dentro de las organizaciones, utilizando el lenguaje Python y MySQL con una actitud analítica y responsable

PRÁCTICA	NOMBRE	COMPETENCIA
Práctica No. 4	Conexión, creación y desconexión a base de datos con SQLite en Python	Implementar la conexión y manipulación básica de bases de datos SQLite con el objetivo de gestionar información persistente, siguiendo las indicaciones del facilitador, en el laboratorio, con actitud analítica.
Práctica No. 5	Consultas SQL básicas con SQLite en Python	Ejecutar consultas SQL básicas con la finalidad de obtener datos relevantes de una base SQLite, aplicando instrucciones del facilitador, en el laboratorio, con

		responsabilidad.
Práctica No. 6	Consultas SQL intermedias con SQLite en Python	Aplicar consultas SQL intermedias para gestionar información almacenada, utilizando SQLite con Python, en condiciones guiadas por el facilitador, en el laboratorio, con actitud analítica.
Práctica No. 7	Operaciones CRUD con MySQL desde Python	Implementar operaciones CRUD desde Python con la finalidad de manipular datos en bases MySQL, con apoyo de los recursos del facilitador, en el laboratorio, trabajando en equipo y con responsabilidad.
Práctica No. 8	Conexión, creación y desconexión a base de datos con SQLite en Python	Implementar la conexión y manipulación básica de bases de datos SQLite con el objetivo de gestionar información persistente, siguiendo las indicaciones del facilitador, en el laboratorio, con actitud analítica.

Elemento de Competencia al que pertenece la práctica	**EC III**
	Crear interfaces gráficas con acceso a datos web para facilitar a las organizaciones acceder, procesar y mostrar datos provenientes de fuentes en línea utilizando Python de manera efectiva y eficiente.

PRÁCTICA	NOMBRE	COMPETENCIA
Práctica No. 9	Instalación del framework Flask y primera app	Instalar y ejecutar el framework Flask con la finalidad de comprender su estructura básica, siguiendo el ejemplo del facilitador, en el laboratorio, con actitud autodidacta.
Práctica No. 10	Desarrollo de sitio web con Flask	Desarrollar un sitio web básico con Flask para aplicar conocimientos de desarrollo web, bajo condiciones del facilitador, en el laboratorio, con autonomía y compromiso.
Práctica No. 11	Aplicaciones CRUD con Flask	Crear aplicaciones CRUD usando Flask con la finalidad de resolver necesidades locales, en condiciones de trabajo colaborativo, en el laboratorio, con actitud de liderazgo y eficiencia.

UES
Universidad Estatal de Sonora
La Fuerza del Saber Estimulará mi Espíritu

PRÁCTICAS

NOMBRE DE LA PRÁCTICA	**Práctica No. 1: Instalación de Python**
COMPETENCIA DE LA PRÁCTICA	Instalar el entorno de desarrollo Python con la finalidad de conocer la interfaz IDLE, siguiendo los pasos indicados por el facilitador, en el contexto del aula, con ética.

FUNDAMENTO TÉORICO

Python es un lenguaje de programación de alto nivel, interpretado y multipropósito. Se caracteriza por tener una sintaxis muy limpia y legible, lo que facilita su aprendizaje. Es uno de los lenguajes más populares del mundo debido a su gran versatilidad, que le permite ser utilizado en desarrollo web, análisis de datos, inteligencia artificial, automatización de tareas y mucho más. Una de sus grandes ventajas es su extensa biblioteca estándar y la enorme cantidad de librerías de terceros disponibles.

IDLE (Integrated Development and Learning Environment) es el entorno de desarrollo integrado que se incluye por defecto con la instalación de Python. Es una aplicación gráfica sencilla que proporciona las herramientas básicas para programar en Python, incluyendo:

Shell Interactivo: Una consola donde se pueden escribir y ejecutar comandos de Python uno por uno, obteniendo una respuesta inmediata. Es ideal para probar fragmentos de código y explorar el lenguaje.

Editor de Texto: Permite escribir, guardar y ejecutar scripts completos de Python (archivos con extensión .py). Ofrece funcionalidades como resaltado de sintaxis y autocompletado básico.

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

- 1 Computadora personal (de escritorio o laptop) con sistema operativo Windows, macOS o Linux.
- Navegador de internet para acceder al sitio de descarga.
- Instalador de Python descargado desde el sitio web oficial: https://www.python.org

PROCEDIMIENTO O METODOLOGÍA

A continuación, se detalla el proceso para instalar Python y verificar su funcionamiento.

1. **Descargar el Instalador:**
   - Abre tu navegador web y dirígete a la página oficial de Python: python.org.
   - Ve a la sección "Downloads" (Descargas). La página detectará automáticamente tu sistema operativo y te sugerirá la versión estable más reciente.
   - Haz clic en el botón de descarga para obtener el archivo instalador.
2. **Ejecutar la Instalación (Ejemplo para Windows):**
   - Localiza y ejecuta el archivo .exe que descargaste.
   - **¡Paso Crítico!** En la primera ventana del instalador, asegúrate de marcar la casilla que dice **"Add Python to PATH"** en la parte inferior. Esto permitirá ejecutar Python desde cualquier terminal de comandos.
   - Selecciona "Install Now" para una instalación estándar con las características recomendadas.

- El instalador comenzará a copiar los archivos necesarios. Espera a que el proceso finalice.
- Al terminar, verás un mensaje de "Setup was successful". Puedes cerrar el instalador.

3. **Verificar la Instalación:**
   - Abre el menú de inicio de tu sistema operativo.
   - Busca y abre la aplicación llamada **"IDLE (Python...)"**.
   - Se abrirá una ventana titulada "Python Shell". Verás un cursor parpadeando después del símbolo >>>.
4. **Primera Interacción con IDLE:**
   - En el Shell de IDLE, escribe el siguiente comando y presiona Enter:

     Python

     print("Hola, Mundo!")

   - El sistema deberá responder inmediatamente mostrando el texto Hola, Mundo! en la siguiente línea.

Precauciones:

- Descarga siempre el software desde el sitio web oficial para evitar versiones alteradas o malware.
- No omitas el paso de **"Add Python to PATH"** en Windows, ya que es fundamental para el uso correcto del lenguaje desde la línea de comandos.

RESULTADOS ESPERADOS

Tener el entorno de desarrollo de Python correctamente instalado y funcionando en su computadora.

Ubicar y ejecutar el entorno de desarrollo IDLE.

Ejecutar un comando básico de Python (print()) en el shell interactivo de IDLE y observar el resultado.

Comprender la función del shell interactivo para pruebas rápidas de código.

ANÁLISIS DE RESULTADOS

Para interpretar los resultados de la práctica, responde las siguientes preguntas:

- ¿Qué ocurrió exactamente al ejecutar el comando print("Hola, Mundo!")? Describe la relación entre el comando que escribiste y la salida que obtuviste.
- ¿Cuál es el propósito del prompt >>> que se muestra en el shell de IDLE?
- Si durante la instalación no hubieras marcado la casilla "Add Python to PATH", ¿qué problemas podrías esperar al intentar usar Python más adelante?

- Explora los menús de IDLE (File, Edit, Shell). ¿Cómo crearías un nuevo archivo de script para escribir un programa más largo?

CONCLUSIONES Y REFLEXIONES

En esta práctica se realizó la instalación del entorno de desarrollo de Python y su IDLE por defecto. Se siguieron los pasos de descarga y configuración, culminando con la verificación del entorno a través de la ejecución de un comando simple. Esta instalación es el primer paso indispensable para el desarrollo de cualquier aplicación o script en Python. Contar con un entorno bien configurado nos asegura que el código que escribamos podrá ser interpretado y ejecutado sin problemas, sentando las bases para futuros proyectos de programación en mi campo profesional, donde Python puede ser aplicado para análisis de datos, automatización o desarrollo de software.

ACTIVIDADES COMPLEMENTARIAS

Calculadora Instantánea: Utiliza el shell interactivo de IDLE como una calculadora. Realiza las siguientes operaciones y anota los resultados:

- 250 + 750
- 2025 - 50
- 100 * 5
- 400 / 20

Mi Primer Script:

- En IDLE, ve a File > New File para abrir el editor.
- Escribe un pequeño programa que imprima tu nombre completo y tu carrera.
- Guarda el archivo con el nombre mi_presentacion.py en un lugar que recuerdes.
- Ejecuta tu script presionando la tecla F5 o desde el menú Run > Run Module. Observa el resultado en la ventana del shell.

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE

Criterios de evaluación	Instalación correcta y funcional de Python. Capacidad para abrir y utilizar la interfaz de IDLE. Ejecución exitosa de comandos en el shell y de un script guardado.
Rúbricas o listas de cotejo para valorar desempeño	Rúbrica de práctica de laboratorio

Formatos de reporte de prácticas	Captura de pantalla del shell de IDLE mostrando la ejecución del comando print("Hola, Mundo!"). Captura de pantalla del shell de IDLE mostrando los resultados de las operaciones aritméticas de las actividades complementarias. El archivo mi_presentacion.py creado en la segunda actividad complementaria.

NOMBRE DE LA PRÁCTICA	**Práctica No. 2: Programación de estructuras repetitivas en Python**
COMPETENCIA DE LA PRÁCTICA	Desarrollar programas que implementen estructuras repetitivas con la finalidad de resolver problemas simples, utilizando las estructuras while y for, en el laboratorio, con responsabilidad.

FUNDAMENTO TÉORICO

Las estructuras repetitivas, también conocidas como bucles o ciclos, son estructuras de control que permiten ejecutar un mismo bloque de código múltiples veces. Son fundamentales en programación porque automatizan tareas repetitivas, evitando la necesidad de escribir la misma línea de código una y otra vez. Python ofrece principalmente dos tipos de bucles: while y for.

- **Bucle while (mientras):** Ejecuta un bloque de código mientras una condición específica se evalúe como verdadera (True). La condición se comprueba al inicio de cada repetición. Es crucial que dentro del bucle exista alguna instrucción que eventualmente haga que la condición se vuelva falsa (False), de lo contrario, se creará un **bucle infinito**. Se utiliza cuando no sabemos de antemano cuántas veces se debe repetir el ciclo.
  - **Sintaxis:**

    Python

    ```
    while condicion:
        # bloque de código a repetir
    ```

- **Bucle for (para):** Ejecuta un bloque de código para cada elemento de una secuencia o de un objeto iterable (como una lista, una cadena de texto o un rango de números). Es ideal para cuando se conoce la cantidad de veces que se debe ejecutar el ciclo o se quiere recorrer una colección de elementos.
  - **Sintaxis:**

    Python

    ```
    for variable in iterable:
        # bloque de código a repetir
    ```

  - La función range() se usa comúnmente con for para generar secuencias numéricas. Por ejemplo, range(5) genera los números 0, 1, 2, 3, 4.

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

Equipamiento:

- 1 Computadora personal (de escritorio o laptop).

Software:

- Python instalado en el equipo.
- IDLE (Entorno de Desarrollo Integrado y Aprendizaje de Python) o cualquier otro editor de código.

PROCEDIMIENTO O METODOLOGÍA

Se realizarán dos pequeños programas para ilustrar el uso de cada bucle.

Parte 1: Implementación del bucle while

1. Abre IDLE y crea un nuevo archivo (File > New File).
2. Escribe el siguiente código, el cual imprime los números del 1 al 5 usando un bucle while.

   Python

   ```
   # Programa que cuenta del 1 al 5 con un bucle while
   contador = 1
   print("Iniciando conteo con 'while':")
   while contador <= 5:
       print(contador)
       contador = contador + 1 # Incremento para evitar un bucle infinito
   print("Conteo finalizado.")
   ```

3. Guarda el archivo con el nombre practica_while.py.
4. Ejecuta el programa (presionando F5 o desde Run > Run Module).
5. Observa la salida en la ventana del Shell de Python.

Parte 2: Implementación del bucle for

1. Crea un nuevo archivo en IDLE.
2. Escribe el siguiente código, que realiza la misma tarea que el anterior pero utilizando un bucle for y la función range().

   Python

   ```
   # Programa que cuenta del 1 al 5 con un bucle for
   print("Iniciando conteo con 'for':")
   for numero in range(1, 6): # range(1, 6) genera números del 1 al 5
       print(numero)
   print("Conteo finalizado.")
   ```

3. Guarda el archivo con el nombre practica_for.py.
4. Ejecuta el programa y observa la salida. Compara el resultado y el código con el de la Parte 1.

Precauciones:

- Al usar while, siempre verifica que la variable de la condición se modifique dentro del bucle para garantizar que en algún momento termine.
- Presta atención a la **indentación** (el espacio al inicio de las líneas). Python la usa para saber qué código está dentro de un bucle.

RESULTADOS ESPERADOS

Al finalizar la práctica, el estudiante deberá:

- Comprender la sintaxis y el funcionamiento lógico de los bucles while y for.
- Ser capaz de escribir, guardar y ejecutar programas simples que utilicen ambas estructuras repetitivas.
- Diferenciar las situaciones en las que es más conveniente usar un bucle for en lugar de un while y viceversa.
- Identificar y evitar la creación de bucles infinitos.

ANÁLISIS DE RESULTADOS

Para reflexionar sobre los resultados de la práctica, responde las siguientes preguntas:

- ¿Cuál es la diferencia principal entre la estructura del programa practica_while.py y practica_for.py, si ambos producen el mismo resultado?
- En el código con while, ¿qué pasaría si eliminas la línea contador = contador + 1? Justifica tu respuesta.
- ¿Cómo modificarías el código del bucle for para que imprima los números pares del 2 al 10? (Pista: range() puede aceptar un tercer argumento para el "paso" o incremento).
- Describe una situación en la que sería obligatorio usar un bucle while en lugar de un for

CONCLUSIONES Y REFLEXIONES

En esta práctica se han implementado las estructuras repetitivas while y for, dos herramientas esenciales en la programación. Se observó que ambos bucles permiten automatizar tareas,

pero su enfoque es diferente: while se basa en una condición y for en recorrer una secuencia. Comprender su funcionamiento es clave para escribir código eficiente y resolver problemas complejos. Estas estructuras me permitirán, en mi desarrollo profesional, procesar grandes cantidades de datos, crear simulaciones o automatizar procesos que de otra forma serían tediosos e imprácticos.

ACTIVIDADES COMPLEMENTARIAS

Tabla de Multiplicar: Escribe un programa que solicite al usuario un número entero. A continuación, el programa deberá mostrar la tabla de multiplicar de ese número del 1 al 10. Utiliza un bucle for.

Suma Acumulada: Desarrolla un programa que pida al usuario números positivos uno por uno. El programa debe ir sumándolos. El ciclo debe terminar cuando el usuario ingrese un número negativo. Al final, muestra la suma total de los números positivos ingresados. Utiliza un bucle while.

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE	
Criterios de evaluación	• Uso correcto de la sintaxis de los bucles for y while. • Lógica de programación funcional para resolver los problemas planteados. • Código claro, comentado y correctamente indentado. • Los programas se ejecutan sin errores y producen el resultado esperado.
Rúbricas o listas de cotejo para valorar desempeño	Rúbrica de práctica de laboratorio
Formatos de reporte de prácticas	• El archivo .py del programa de la "Tabla de Multiplicar". • El archivo .py del programa de "Suma Acumulada". • Capturas de pantalla que muestren la ejecución y el resultado de ambos programas de las actividades complementarias.

NOMBRE DE LA PRÁCTICA	**Práctica No. 3: Programación de módulos en Python**
COMPETENCIA DE LA PRÁCTICA	Construir módulos en Python con la finalidad de estructurar el código en componentes reutilizables, siguiendo el planteamiento del facilitador, en el laboratorio, con organización y responsabilidad.

FUNDAMENTO TÉORICO

En Python, un **módulo** es simplemente un archivo con extensión .py que contiene código, como definiciones de funciones, clases y variables. La idea principal detrás de los módulos es agrupar código relacionado en un solo lugar para mantener los proyectos organizados y promover la reutilización de código.

Ventajas de usar módulos:

- **Organización:** Permiten dividir un programa grande en archivos más pequeños y manejables, cada uno con una responsabilidad específica.
- **Reutilización:** Se puede escribir una función o una clase una vez en un módulo y luego importarla en cualquier otro script que la necesite, sin tener que copiar y pegar código.
- **Espacio de nombres (Namespace):** Cada módulo tiene su propio espacio de nombres, lo que ayuda a evitar conflictos con nombres de variables o funciones entre diferentes partes de un programa.

Para utilizar el código de un módulo en otro archivo, se usa la instrucción **import**. Hay dos formas comunes de hacerlo:

1. **import nombre_modulo**: Importa el módulo completo. Para usar una función de ese módulo, se debe usar la sintaxis nombre_modulo.nombre_funcion().
2. **from nombre_modulo import nombre_funcion**: Importa un objeto específico (una función, clase, etc.) del módulo. Esto permite llamar a la función directamente por su nombre, sin necesidad del prefijo del módulo.

Cualquier archivo de Python puede actuar como un módulo para ser importado por otro.

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

Equipamiento:

- 1 Computadora personal (de escritorio o laptop).

Software:

- Python instalado en el equipo.
- IDLE (Entorno de Desarrollo Integrado y Aprendizaje de Python).

PROCEDIMIENTO O METODOLOGÍA

Se creará un módulo con operaciones matemáticas básicas y luego un programa principal que lo utilice.

Parte 1: Creación del Módulo de Operaciones

1. Abre IDLE y crea un nuevo archivo (File > New File).
2. En este archivo, define dos funciones simples: una para sumar y otra para restar.

   Python

   ```
   # Este es el módulo de operaciones matematicas
   # Archivo: operaciones.py

   def sumar(a, b):
       """Esta funcion devuelve la suma de dos numeros."""
       return a + b

   def restar(a, b):
       """Esta funcion devuelve la resta de dos numeros."""
       return a - b

   PI = 3.14159 # Ejemplo de una variable en un modulo
   ```

3. Guarda este archivo con el nombre **operaciones.py**. Este nombre de archivo es el nombre del módulo.

Parte 2: Uso del Módulo

1. Abre una segunda ventana de IDLE para crear otro archivo (File > New File). Este será tu programa principal.
2. **¡Paso Crítico!** Guarda este nuevo archivo con el nombre programa_principal.py **en la misma carpeta** donde guardaste operaciones.py. Python busca los módulos en el mismo directorio del script que se está ejecutando.
3. Escribe el siguiente código en programa_principal.py para importar y usar el módulo operaciones.

   ```
   # Este es el programa principal que usa el modulo 'operaciones'
   # Archivo: programa_principal.py
   ```

```
import operaciones

# Usamos las funciones del modulo
resultado_suma = operaciones.sumar(15, 7)
resultado_resta = operaciones.restar(100, 45)

print(f"El resultado de la suma es: {resultado_suma}")
print(f"El resultado de la resta es: {resultado_resta}")

# Tambien podemos usar las variables del modulo
print(f"El valor de PI segun nuestro modulo es: {operaciones.PI}")
```

4. Ejecuta programa_principal.py (presionando F5).
5. Observa la salida en la consola, la cual debe mostrar los resultados de las operaciones definidas en el otro archivo.

Precauciones:

- Asegúrate de que el archivo del módulo y el archivo del programa principal estén guardados en el mismo directorio. De lo contrario, Python no encontrará el módulo y lanzará un error (ModuleNotFoundError).
- No nombres tu script con el mismo nombre de un módulo estándar de Python (ej. math.py, random.py), ya que puede causar conflictos.

RESULTADOS ESPERADOS

Al finalizar la práctica, el estudiante podrá:

- Crear un archivo .py que funcione como un módulo con funciones y variables.
- Importar un módulo personalizado en otro script de Python.
- Utilizar las funciones y variables de un módulo importado.
- Entender la importancia de la organización de archivos para que la importación de módulos funcione correctamente.
- Diferenciar el uso de import modulo y from modulo import funcion.

ANÁLISIS DE RESULTADOS

Para analizar lo aprendido, contesta las siguientes preguntas:

- ¿Cuál es la principal ventaja de haber puesto las funciones sumar y restar en el archivo operaciones.py en lugar de en programa_principal.py?

- ¿Qué error esperas si intentas ejecutar programa_principal.py después de mover operaciones.py a otra carpeta? ¿Por qué ocurre?
- Modifica programa_principal.py. En lugar de import operaciones, usa from operaciones import sumar. ¿Cómo cambia la línea de código donde llamas a la función de suma?
- Si solo importas sumar con from operaciones import sumar, ¿podrías seguir accediendo a la variable PI del módulo? ¿Por qué?

CONCLUSIONES Y REFLEXIONES

En esta práctica se aprendió el concepto de modularidad en Python, creando un archivo con código reutilizable y luego importándolo desde un programa principal. Este enfoque es fundamental en el desarrollo de software, ya que promueve un código más limpio, organizado y fácil de mantener. Dividir un proyecto complejo en módulos lógicos no solo simplifica el trabajo individual, sino que también facilita la colaboración en equipo. Esta habilidad es esencial para construir aplicaciones robustas y escalables en mi futuro profesional.

ACTIVIDADES COMPLEMENTARIAS

Módulo de Geometría:

- Crea un nuevo módulo llamado **geometria.py**.
- Dentro de este módulo, define tres funciones:
  - area_circulo(radio): Devuelve el área de un círculo.
  - area_rectangulo(base, altura): Devuelve el área de un rectángulo.
  - perimetro_cuadrado(lado): Devuelve el perímetro de un cuadrado.

Programa de Cálculo Geométrico:

- Crea un programa principal que importe el módulo geometria.
- Usa las funciones del módulo para calcular el área de un círculo con radio 10 y el perímetro de un cuadrado de lado 5.
- Imprime los resultados de forma clara.

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE	
Criterios de evaluación	• Creación correcta del módulo geometria.py con las funciones solicitadas. • Uso adecuado de la instrucción import en el programa principal.

	• Lógica correcta en las llamadas a las funciones y presentación de resultados. • El programa se ejecuta sin errores y muestra los cálculos correctos.
Rúbricas o listas de cotejo para valorar desempeño	Rúbrica de práctica de laboratorio
Formatos de reporte de prácticas	• El archivo **geometria.py**. • El archivo del programa principal que utiliza el módulo de geometría. • Capturas de pantalla que muestren el código de ambos archivos y la salida final del programa principal al ser ejecutado.

NOMBRE DE LA PRÁCTICA	**Práctica No. 4: Conexión, creación y desconexión a base de datos con SQLite en Python**
COMPETENCIA DE LA PRÁCTICA	Implementar la conexión y manipulación básica de bases de datos SQLite con el objetivo de gestionar información persistente, siguiendo las indicaciones del facilitador, en el laboratorio, con actitud analítica.

FUNDAMENTO TÉORICO

SQLite es un sistema de gestión de bases de datos relacional auto-contenido, sin servidor y de configuración cero. A diferencia de sistemas como MySQL, no requiere un proceso de servidor separado. Los datos se guardan en un único archivo `.db`. El módulo `sqlite3` viene incluido en Python y proporciona una interfaz SQL para trabajar con estas bases de datos. Los objetos clave son la **conexión** (representa la base de datos) y el **cursor** (permite ejecutar comandos SQL).

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

- **Equipamiento:** Computadora personal.
- **Software:** Python con IDLE.

PROCEDIMIENTO O METODOLOGÍA

- **Importar:** Inicia tu script con `import sqlite3`.
- **Conectar:** Crea una conexión (y la base de datos si no existe): `conn = sqlite3.connect('mi_empresa.db')`.
- **Crear Cursor:** Obtén un objeto cursor: `cursor = conn.cursor()`.
- **Crear Tabla:** Ejecuta un comando `CREATE TABLE` usando el cursor para crear una tabla de `empleados` con `id`, `nombre` y `puesto`.

```
CREATE TABLE IF NOT EXISTS empleados (
    id INTEGER PRIMARY KEY,
    nombre TEXT NOT NULL,
    puesto TEXT NOT NULL
);
```

- **Confirmar Cambios:** Guarda los cambios en la base de datos: `conn.commit()`.

- **Cerrar Conexión:** Libera los recursos: `conn.close()`.

RESULTADOS ESPERADOS

- Creación de un archivo `.db` en la carpeta del proyecto.
- Comprensión del flujo: conectar, crear cursor, ejecutar, confirmar y cerrar.
- La tabla `empleados` debe existir dentro del archivo de la base de datos.

ANÁLISIS DE RESULTADOS

- ¿Qué sucede si ejecutas el script una segunda vez? ¿Por qué se usa `IF NOT EXISTS`?
- ¿Cuál es el propósito de `conn.commit()`? ¿Qué pasaría si lo omites después de crear una tabla?
- ¿Por qué es importante cerrar la conexión con `conn.close()`?

CONCLUSIONES Y REFLEXIONES

La capacidad de interactuar con bases de datos es fundamental para crear aplicaciones que manejen datos de forma persistente. SQLite y el módulo `sqlite3` ofrecen una manera sencilla y potente de integrar almacenamiento de datos en aplicaciones Python sin la complejidad de un servidor de base de datos.

ACTIVIDADES COMPLEMENTARIAS

1. **Base de Datos de Biblioteca:** Crea un nuevo script que genere una base de datos llamada `biblioteca.db`.
2. **Tabla de Libros:** Dentro de `biblioteca.db`, crea una tabla `libros` con las columnas: `isbn` (TEXTO, CLAVE PRIMARIA), `titulo` (TEXTO) y `autor` (TEXTO).

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE

Criterios de evaluación	• **Evidencias:** El script de Python para crear la base de datos `biblioteca.db` y su tabla. Una captura de pantalla de la carpeta del proyecto mostrando el archivo `biblioteca.db` creado.

Rúbricas o listas de cotejo para valorar desempeño	Rúbrica de práctica de laboratorio
Formatos de reporte de prácticas	

NOMBRE DE LA PRÁCTICA	**Práctica No. 5Consultas SQL básicas con SQLite en Python**
COMPETENCIA DE LA PRÁCTICA	Ejecutar consultas SQL básicas con la finalidad de obtener datos relevantes de una base SQLite, aplicando instrucciones del facilitador, en el laboratorio, con responsabilidad.

FUNDAMENTO TÉORICO

Una vez que una tabla tiene datos, podemos recuperarlos usando la instrucción SELECT.

- Insertar datos: Se usa la instrucción INSERT INTO. Para evitar ataques de inyección SQL, se deben usar placeholders (?) en lugar de formatear cadenas de texto.
- Seleccionar datos: SELECT se usa para leer la información.
- Recuperar resultados: El objeto cursor tiene métodos para obtener los resultados de un SELECT: fetchone() (obtiene una fila), fetchall() (obtiene todas las filas como una lista) y fetchmany(n) (obtiene n filas).

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

- Equipamiento: Computadora personal.
- Software: Python con IDLE y el archivo mi_empresa.db de la práctica anterior.

PROCEDIMIENTO O METODOLOGÍA

1. **Conectar:** Conéctate a la base de datos `mi_empresa.db`.
2. **Insertar Datos:** Inserta al menos tres empleados en la tabla `empleados` usando `cursor.execute()` y placeholders.

```
empleados_a_insertar = [     ('Ana Torres', 'Desarrolladora'),
('Carlos Ruiz', 'Diseñador'),     ('Sofia Lopez', 'Gerente de Proyecto') ]
cursor.executemany("INSERT INTO empleados (nombre, puesto) VALUES (?, ?)",
empleados_a_insertar) conn.commit()
```

3. **Consultar Datos:** Ejecuta una consulta para seleccionar todos los empleados: `cursor.execute("SELECT * FROM empleados")`.
4. **Mostrar Resultados:** Usa `fetchall()` y un bucle `for` para iterar sobre los resultados e imprimirlos en la consola.
5. **Cerrar Conexión.**

RESULTADOS ESPERADOS

- Insertar datos de forma segura en una tabla SQLite.
- Realizar una consulta SELECT para recuperar todos los registros de una tabla.
- Procesar y mostrar los resultados obtenidos en la consola de Python.

ANÁLISIS DE RESULTADOS

- ¿Cuál es la ventaja de usar executemany en lugar de múltiples execute?
- ¿Qué tipo de dato devuelve cursor.fetchall()? ¿Cómo se ve cada fila dentro de esa estructura?
- ¿Cómo modificarías la consulta SELECT para obtener solo los nombres de los empleados y no toda la información?

CONCLUSIONES Y REFLEXIONES

Insertar y recuperar datos son las operaciones más comunes en cualquier aplicación. Aprender a realizar estas tareas de forma segura y eficiente es crucial. Los placeholders son una práctica de seguridad no negociable para proteger la base de datos.

ACTIVIDADES COMPLEMENTARIAS

- Poblar Biblioteca: Escribe un script que inserte al menos 3 libros en la tabla libros de tu biblioteca.db.
- Consultar Libros: Escribe una consulta SELECT para obtener todos los libros de un autor específico (usa la cláusula WHERE). Imprime los resultados.

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE	
Criterios de evaluación	Evidencias: El script de Python que puebla y consulta la biblioteca.db. Captura de pantalla mostrando la salida de la consulta.
Rúbricas o listas de cotejo para valorar desempeño	Rúbrica de práctica de laboratorio
Formatos de reporte de prácticas	

NOMBRE DE LA PRÁCTICA	**Práctica No. 6 Consultas SQL intermedias con SQLite en Python**
COMPETENCIA DE LA PRÁCTICA	Aplicar consultas SQL intermedias para gestionar información almacenada, utilizando SQLite con Python, en condiciones guiadas por el facilitador, en el laboratorio, con actitud analítica.

FUNDAMENTO TÉORICO

Además de insertar y leer, es fundamental poder modificar y eliminar datos.

- UPDATE: Modifica registros existentes en una tabla. Es crucial usar una cláusula WHERE para especificar qué filas actualizar.
- DELETE: Elimina registros de una tabla. Al igual que con UPDATE, la cláusula WHERE es indispensable para no borrar datos accidentalmente.
- ORDER BY: Se usa junto con SELECT para ordenar los resultados según una o más columnas, de forma ascendente (ASC, por defecto) o descendente (DESC).

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

- Equipamiento: Computadora personal.
- Software: Python con IDLE y el archivo mi_empresa.db con datos.

PROCEDIMIENTO O METODOLOGÍA

1. **Conectar:** Conéctate a `mi_empresa.db`.
2. **Ordenar:** Realiza una consulta `SELECT * FROM empleados ORDER BY nombre ASC`. Imprime los resultados para ver el orden alfabético.
3. **Actualizar:** Usa una instrucción `UPDATE` para cambiar el puesto de un empleado específico. Usa un `WHERE` para identificar al empleado por su `id` o `nombre`. No olvides `conn.commit()`.

   ```
   cursor.execute("UPDATE empleados SET puesto = ? WHERE nombre = ?",
   ('Desarrolladora Senior', 'Ana Torres')) conn.commit()
   ```
4. **Eliminar:** Usa una instrucción `DELETE` para eliminar a un empleado. Usa `WHERE`. Commit.
5. **Verificar:** Realiza una última consulta `SELECT * FROM empleados` para confirmar que los cambios se aplicaron correctamente.
6. **Cerrar Conexión.**

RESULTADOS ESPERADOS

- Ordenar los resultados de una consulta.
- Actualizar de forma segura un registro existente en la base de datos.
- Eliminar de forma segura un registro específico.

ANÁLISIS DE RESULTADOS

- ¿Qué pasaría si ejecutas una instrucción `UPDATE` o `DELETE` sin la cláusula `WHERE`?
- ¿Cómo ordenarías a los empleados por nombre en orden descendente (de la Z a la A)?
- ¿Se pueden usar múltiples condiciones en una cláusula `WHERE`? ¿Cómo? (Investiga `AND`/`OR`).

CONCLUSIONES Y REFLEXIONES

La gestión completa del ciclo de vida de los datos (crear, leer, actualizar, eliminar) es el núcleo de las aplicaciones dinámicas. Las instrucciones UPDATE y DELETE son poderosas pero peligrosas si no se usan con una cláusula WHERE precisa.

ACTIVIDADES COMPLEMENTARIAS

1. Actualizar Libro: En tu biblioteca.db, escribe un script para actualizar el autor de un libro específico, identificado por su isbn.
2. Eliminar Libro: Escribe un script para eliminar un libro de la base de datos.
3. Verificar: Realiza una consulta final que seleccione todos los libros restantes y los ordene por título.

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE	
Criterios de evaluación	Evidencias: El script de Python que actualiza, elimina y consulta la biblioteca.db. Captura de pantalla de la salida final.
Rúbricas o listas de cotejo para valorar desempeño	Rúbrica de práctica de laboratorio
Formatos de reporte de prácticas	

NOMBRE DE LA PRÁCTICA	**Práctica No. 7 Operaciones CRUD con MySQL desde Python**
COMPETENCIA DE LA PRÁCTICA	Implementar operaciones CRUD desde Python con la finalidad de manipular datos en bases MySQL, con apoyo de los recursos del facilitador, en el laboratorio, trabajando en equipo y con responsabilidad.

FUNDAMENTO TÉORICO
MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

MySQL es un sistema de gestión de bases de datos cliente-servidor, a diferencia de SQLite. Para conectarse desde Python, se necesita una librería de terceros como mysql-connector-python. Las operaciones básicas se agrupan en el acrónimo CRUD:

- Create (Crear): INSERT
- Read (Leer): SELECT
- Update (Actualizar): UPDATE
- Delete (Eliminar): DELETE

La conexión requiere más parámetros: host, user, password y database.

PROCEDIMIENTO O METODOLOGÍA

- Equipamiento: Computadora personal.
- Software: Python, IDLE, acceso a un servidor de base de datos MySQL (local o remoto).
- Librería: Instalar el conector con pip install mysql-connector-python.

RESULTADOS ESPERADOS

1. **Instalar Conector:** Abre una terminal o CMD y ejecuta `pip install mysql-connector-python`.
2. **Importar:** `import mysql.connector`.
3. **Conectar:** Establece la conexión con tus credenciales de MySQL.
   ```
   conn = mysql.connector.connect( host="localhost", user="tu_usuario", password="tu_contraseña", database="tu_base_de_datos" )
   ```
4. **Realizar CRUD:**
   a. **Crear:** Inserta un nuevo registro en una tabla usando `INSERT`. El placeholder es `%s`.
   b. **Leer:** Lee los registros usando `SELECT`.
   c. **Actualizar:** Modifica un registro usando `UPDATE` y `WHERE`.
   d. **Eliminar:** Borra un registro usando `DELETE` y `WHERE`.
5. Recuerda usar `conn.commit()` después de cada operación que modifique datos.
6. **Cerrar Conexión.**

ANÁLISIS DE RESULTADOS

- ¿Cuáles son las principales diferencias entre conectarse a SQLite y a MySQL?
- ¿Por qué el placeholder cambia de ? a %s?

- ¿Qué errores de conexión podrías encontrar y cuáles serían sus posibles causas (ej. host incorrecto, contraseña inválida)?

CONCLUSIONES Y REFLEXIONES

Aunque los comandos SQL son estándar, la forma de conectar y manejar la base de datos varía según el sistema gestor. Aprender a trabajar con sistemas cliente-servidor como MySQL es un paso crucial para desarrollar aplicaciones web y empresariales más robustas.

ACTIVIDADES COMPLEMENTARIAS

- Tabla de Productos: En tu base de datos MySQL, crea una tabla productos (id, nombre_producto, precio, stock).
- Script CRUD Completo: Escribe un único script en Python que realice las 4 operaciones CRUD sobre la tabla productos: inserte un producto, lo lea, actualice su precio y finalmente lo elimine.

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE	
Criterios de evaluación	**Evidencias:** El script de Python que realiza el CRUD en MySQL. Capturas de pantalla mostrando el estado de la tabla en la base de datos después de cada operación.
Rúbricas o listas de cotejo para valorar desempeño	Rúbrica de práctica de laboratorio
Formatos de reporte de prácticas	

NOMBRE DE LA PRÁCTICA	**Práctica No. 8 Conexión, creación y desconexión a base de datos con SQLite en Python**
COMPETENCIA DE LA PRÁCTICA	Implementar la conexión y manipulación básica de bases de datos SQLite con el objetivo de gestionar información persistente, siguiendo las indicaciones del facilitador, en el laboratorio, con actitud analítica.

FUNDAMENTO TÉORICO

Esta práctica integra los conceptos anteriores para diseñar y crear una base de datos relacional simple. Una base de datos relacional utiliza múltiples tablas vinculadas entre sí mediante claves foráneas (Foreign Keys). Una clave foránea en una tabla es una columna que apunta a la clave primaria (PRIMARY KEY) de otra tabla, estableciendo una relación entre ellas. Por ejemplo, se puede tener una tabla de autores y una de libros, donde cada libro tiene una clave foránea que lo vincula a su autor.

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

- Equipamiento: Computadora personal.
- Software: Python con IDLE.

PROCEDIMIENTO O METODOLOGÍA

1. Diseño: Diseña un esquema de dos tablas:
   categorias (id_categoria INTEGER PRIMARY KEY, nombre_categoria TEXT)
   productos (id_producto INTEGER PRIMARY KEY, nombre_producto TEXT, id_categoria_fk INTEGER, FOREIGN KEY(id_categoria_fk) REFERENCES categorias(id_categoria))
2. Crear BD: Conéctate a una nueva base de datos llamada tienda.db.
3. Crear Tablas: Ejecuta los dos comandos CREATE TABLE para implementar el diseño.
4. Poblar Tablas: Inserta al menos dos categorías y tres productos, asegurándote de que los id_categoria_fk de los productos correspondan a id_categoria existentes.
5. Confirmar y Cerrar.
6. Verificación (Opcional Avanzado): Escribe una consulta con JOIN para ver los productos junto al nombre de su categoría.

RESULTADOS ESPERADOS

- Diseñar un esquema relacional simple con dos tablas.
- Implementar claves primarias y foráneas en SQLite.
- Crear y poblar una base de datos relacional desde Python.

ANÁLISIS DE RESULTADOS

- ¿Qué ventaja ofrece separar los productos y las categorías en dos tablas en lugar de tener una sola tabla con toda la información?
- ¿Qué pasaría si intentas insertar un producto con un id_categoria_fk que no existe en la tabla categorias?
- ¿Qué es la "integridad referencial" y cómo ayudan las claves foráneas a mantenerla

CONCLUSIONES Y REFLEXIONES

El verdadero poder de las bases de datos relacionales reside en su capacidad para modelar relaciones complejas del mundo real. El uso de claves foráneas para vincular tablas reduce la redundancia de datos y asegura su consistencia, lo cual es fundamental para construir sistemas de información fiables y eficientes.

ACTIVIDADES COMPLEMENTARIAS

- Diseño Estudiante-Curso: Diseña un esquema para una base de datos de una escuela con dos tablas: estudiantes (id_estudiante, nombre) y inscripciones (id_inscripcion, id_estudiante_fk, nombre_curso).
- Implementación: Escribe el script de Python para crear y poblar esta base de datos.

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE	
Criterios de evaluación	Evidencias: El script de Python que crea la base de datos tienda.db y sus tablas relacionales. Captura de pantalla de la carpeta del proyecto mostrando el archivo tienda.db.
Rúbricas o listas de cotejo para valorar desempeño	Rúbrica de práctica de laboratorio
Formatos de reporte de prácticas	

NOMBRE DE LA PRÁCTICA	**Práctica No. 9Instalación del framework Flask y primera app**
COMPETENCIA DE LA PRÁCTICA	Instalar y ejecutar el framework Flask con la finalidad de comprender su estructura básica, siguiendo el ejemplo del facilitador, en el laboratorio, con actitud autodidacta.

FUNDAMENTO TÉORICO

Flask es un **microframework** de desarrollo web para Python. Se le llama "micro" porque no incluye componentes que se pueden encontrar en frameworks más grandes (como un ORM para bases de datos o un sistema de autenticación), permitiendo al desarrollador una total libertad para elegir las herramientas que mejor se adapten a su proyecto. Sus componentes clave son:

- **Enrutamiento (Routing):** Asocia URLs (rutas) a funciones específicas de Python. Esto se logra con el decorador @app.route().
- **Funciones de Vista (View Functions):** Son las funciones de Python que se ejecutan cuando un usuario visita una ruta. Deben devolver una respuesta que se mostrará en el navegador.
- **Servidor de Desarrollo:** Flask incluye un servidor web integrado que es ideal para las fases de desarrollo y prueba.

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

- Equipamiento: Computadora personal.
- Software: Python con pip (gestor de paquetes), terminal o línea de comandos (CMD, PowerShell, etc.).

PROCEDIMIENTO O METODOLOGÍA

- **Crear Carpeta del Proyecto:** Crea una nueva carpeta para tu proyecto, por ejemplo, `mi_primera_app`.
- **Activar Entorno Virtual (Recomendado):** Abre una terminal en la carpeta del proyecto y ejecuta `python -m venv venv`. Luego, actívalo:
- En Windows: `venv\Scripts\activate`
- En macOS/Linux: `source venv/bin/activate`
- **Instalar Flask:** En la misma terminal con el entorno activado, ejecuta `pip install Flask`.
- **Crear la Aplicación:** Dentro de la carpeta, crea un archivo llamado `app.py` y escribe el siguiente código:

```
from flask import Flask  # Crea una instancia de la aplicacion Flask app = Flask(__name__)  # Define la ruta para la pagina principal@app.route('/')def hola_mundo():     return '¡Hola, Mundo desde Flask!'# Permite ejecutar la app directamente desde el scriptif __name__ == '__main__':     app.run(debug=True)
```

- **Ejecutar la App:** En la terminal, ejecuta `python app.py`. Verás un mensaje indicando que el servidor está corriendo en `http://127.0.0.1:5000/`.
- **Verificar:** Abre un navegador web y visita `http://127.0.0.1:5000`. Deberías ver el mensaje "¡Hola, Mundo desde Flask!".

RESULTADOS ESPERADOS

- Flask instalado correctamente en un entorno virtual.
- Creación y ejecución de una aplicación Flask mínima.
- Visualización del resultado de una ruta en un navegador web.
- Comprensión del rol del decorador @app.route().

ANÁLISIS DE RESULTADOS

- ¿Cuál es el propósito del argumento debug=True en app.run()?
- ¿Qué sucede si cambias la ruta de '/' a '/inicio'? ¿A qué URL tendrías que navegar para ver el mensaje?
- ¿Qué es __name__ y por qué se usa en la condición if __name__ == '__main__':?

CONCLUSIONES Y REFLEXIONES

Flask simplifica enormemente el proceso de crear un servidor web en Python. Con solo unas pocas líneas de código, es posible tener una aplicación funcional, lo que demuestra el poder de los frameworks para abstraer la complejidad y permitir a los desarrolladores enfocarse en la lógica de la aplicación.

ACTIVIDADES COMPLEMENTARIAS

- Nueva Ruta: Agrega una nueva ruta /despedida a tu archivo app.py que devuelva el texto "¡Hasta la próxima!".
- Ruta de Información: Crea una ruta /info que devuelva tu nombre y la carrera que estudias.

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE

Criterios de evaluación	Evidencias: El archivo app.py con las tres rutas (/, /despedida, /info). Capturas de pantalla del navegador mostrando el resultado de cada una de las tres rutas.
Rúbricas o listas de cotejo para valorar desempeño	Rúbrica de práctica de laboratorio
Formatos de reporte de prácticas	

NOMBRE DE LA PRÁCTICA	**Práctica No. 10 Desarrollo de sitio web con Flask**
COMPETENCIA DE LA PRÁCTICA	Desarrollar un sitio web básico con Flask para aplicar conocimientos de desarrollo web, bajo condiciones del facilitador, en el laboratorio, con autonomía y compromiso.

FUNDAMENTO TÉORICO

Para crear sitios web complejos, no es práctico devolver código HTML como cadenas de texto desde Python. Flask se integra con el motor de plantillas Jinja2 para solucionar esto.

- Templates (Plantillas): Son archivos HTML que contienen marcadores de posición para datos dinámicos. Por convención, se guardan en una carpeta llamada templates.
- render_template(): Es una función de Flask que toma el nombre de un archivo de plantilla y lo procesa, permitiendo pasar variables desde Python a la plantilla.
- Archivos Estáticos: Los archivos como CSS, JavaScript e imágenes se consideran "estáticos". Por convención, se guardan en una carpeta llamada static.
- Sintaxis de Jinja2:
- {{ variable }}: Imprime el valor de una variable.
- {% for item in lista %} ... {% endfor %}: Estructura de control para bucles.

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

- Equipamiento: Computadora personal.
- Software: Python con Flask, terminal, editor de código.

PROCEDIMIENTO O METODOLOGÍA

1. **Organizar Carpetas:** En tu carpeta de proyecto, crea dos subcarpetas: `templates` y `static`.
2. **Crear Plantilla HTML:** Dentro de `templates`, crea un archivo `index.html`.
3. HTML

```
<!DOCTYPE html><html><head>     <title>Mi Sitio Web</title>     <link
rel="stylesheet" href="{{ url_for('static', filename='style.css')
}}"></head><body>     <h1>{{ titulo_de_la_pagina }}</h1>
<p>¡Bienvenido a mi primera página web con Flask y
templates!</p></body></html>
```

4. **Crear Archivo CSS:** Dentro de `static`, crea un archivo `style.css` con alguna regla simple.
   CSS

```
body {     background-color: #f0f0f0;     font-family: sans-serif; }
```

5. **Actualizar `app.py`:** Modifica tu archivo principal para usar la plantilla.
   Python

```
from flask import Flask, render_template, url_for  app = Flask(__name__)

@app.route('/')def index():     # Pasa una variable a la plantilla
titulo = "Página de Inicio Dinámica"     return
render_template('index.html', titulo_de_la_pagina=titulo)

if __name__ == '__main__':     app.run(debug=True)
```

6. **Ejecutar y Verificar:** Corre `python app.py` y visita `http://127.0.0.1:5000`. Deberías ver tu página HTML con el título dinámico y el estilo CSS aplicado.

RESULTADOS ESPERADOS

- Implementar la estructura de carpetas estándar de Flask (templates, static).
- Renderizar una plantilla HTML desde una función de vista usando render_template.
- Pasar datos (variables) desde el script de Python a la plantilla HTML.
- Servir y enlazar correctamente archivos estáticos como CSS.

ANÁLISIS DE RESULTADOS

- ¿Por qué es fundamental separar la lógica (Python) de la presentación (HTML/CSS)?
- ¿Qué ventaja ofrece usar url_for('static', ...) en lugar de escribir la ruta /static/style.css directamente en el HTML?
- ¿Cómo pasarías una lista de elementos desde Python para mostrarla en la plantilla?

CONCLUSIONES Y REFLEXIONES

El uso de plantillas es un concepto clave en el desarrollo web moderno. Permite crear sitios dinámicos y mantenibles, donde el contenido y el diseño pueden evolucionar de forma independiente. Flask, a través de Jinja2, ofrece una solución elegante y potente para esta necesidad.

ACTIVIDADES COMPLEMENTARIAS

- Página "Acerca de": Crea una nueva plantilla about.html y una ruta /about que la renderice.
- Lista de Tareas: En tu ruta principal, crea una lista de tareas (strings) en Python. Pásala a la plantilla index.html y usa un bucle {% for %} de Jinja2 para mostrar cada tarea dentro de una lista <ul>.

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE

Criterios de evaluación	• **Evidencias:** El proyecto completo con su estructura de carpetas y archivos. Capturas de pantalla mostrando la página de inicio con la lista de tareas y la página "Acerca de".
Rúbricas o listas de cotejo para valorar desempeño	Rúbrica de práctica de laboratorio
Formatos de reporte de prácticas	

NOMBRE DE LA PRÁCTICA	**Práctica No. 11 Aplicaciones CRUD con Flask**
COMPETENCIA DE LA PRÁCTICA	Crear aplicaciones CRUD usando Flask con la finalidad de resolver necesidades locales, en condiciones de trabajo colaborativo, en el laboratorio, con actitud de liderazgo y eficiencia.

FUNDAMENTO TÉORICO

CRUD son las siglas de las cuatro operaciones básicas en la gestión de datos: Crear (Create), Leer (Read), Actualizar (Update) y Eliminar (Delete). Una aplicación CRUD permite al usuario interactuar con los datos de una aplicación.

- Formularios HTML: Se usan para enviar datos del cliente al servidor. El atributo method especifica cómo se envían los datos (GET para solicitar, POST para enviar/crear).
- Objeto request: En Flask, el objeto request (importado de flask) contiene la información de la petición del cliente. request.form se usa para acceder a los datos enviados mediante un formulario POST.
- Redirección: Después de una operación exitosa (como crear o eliminar un dato), es una buena práctica redirigir al usuario a otra página para evitar reenvíos accidentales del formulario. Esto se hace con las funciones redirect y url_for.

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

- Equipamiento: Computadora personal.
- Software: Python con Flask, terminal, editor de código.

PROCEDIMIENTO O METODOLOGÍA

Se creará una aplicación simple para gestionar una lista de tareas. Para simplificar, se usará una lista de Python como "base de datos" en memoria.

1. **Configuración Inicial (`app.py`):**

```
from flask import Flask, render_template, request, redirect, url_for  app
= Flask(__name__)  # Base de datos en memoria (una lista de diccionarios)
tareas = [{'id': 1, 'descripcion': 'Aprender Flask'}] next_id = 2
```

2. **Leer (Read):** Muestra todas las tareas en la página principal.
3. **Ruta en `app.py`:**

```
@app.route('/')
def listar_tareas():
      return render_template('tareas.html', tareas=tareas)
```

4. **Plantilla `tareas.html`:** Usa un bucle `{% for %}` para mostrar cada tarea.
5. **Crear (Create):** Agrega un formulario en `tareas.html` para añadir nuevas tareas.
6. **Formulario en `tareas.html`:**

```
<form action="{{ url_for('agregar_tarea') }}" method="post">     <input
type="text" name="descripcion" required>     <button type="submit">Agregar
Tarea</button></form>
```

7. **Ruta en `app.py`:**

```
@app.route('/add', methods=['POST'])
def agregar_tarea():
      global next_id
      descripcion = request.form['descripcion']
      tareas.append({'id': next_id, 'descripcion': descripcion})
            next_id += 1     return redirect(url_for('listar_tareas'))
```

8. **Eliminar (Delete):** Agrega un botón de eliminar junto a cada tarea.
9. **Formulario en el bucle de `tareas.html`:**

```
<form action="{{ url_for('eliminar_tarea', tarea_id=tarea.id) }}"
method="post" style="display:inline;">      <button
type="submit">Eliminar</button></form>
```

10. **Ruta en `app.py`:**

```
@app.route('/delete/<int:tarea_id>', methods=['POST'])
def eliminar_tarea(tarea_id):
     global tareas
     tareas = [t for t in tareas if t['id'] != tarea_id]
     return redirect(url_for('listar_tareas'))
```

11. **Ejecutar y Probar:** Corre la aplicación y prueba agregar y eliminar tareas.

RESULTADOS ESPERADOS

- Construir una aplicación web interactiva que gestione datos.
- Manejar datos de formularios HTML enviados con el método POST.
- Implementar un flujo de trabajo CRUD completo.
- Utilizar el patrón Post/Redirect/Get para mejorar la experiencia del usuario.

ANÁLISIS DE RESULTADOS

- ¿Por qué la ruta para agregar tareas especifica `methods=['POST']`?
- Explica el propósito de la función `redirect(url_for(...))` después de agregar o eliminar una tarea.
- Si quisieras que los datos fueran permanentes, ¿cómo adaptarías este código para usar la base de datos SQLite de las prácticas anteriores en lugar de una lista en memoria?

CONCLUSIONES Y REFLEXIONES

La combinación de manejo de rutas, plantillas y procesamiento de formularios convierte a Flask en una herramienta poderosa para crear aplicaciones web completas y dinámicas. El patrón CRUD es la base de la mayoría de las aplicaciones que vemos en internet, desde redes sociales hasta tiendas en línea.

ACTIVIDADES COMPLEMENTARIAS

- Integrar con SQLite: Reemplaza la lista en memoria tareas con una base de datos SQLite. Modifica las funciones agregar_tarea y eliminar_tarea para que ejecuten comandos INSERT y DELETE en la base de datos.
- Implementar Actualizar (Update): Añade la funcionalidad para editar la descripción de una tarea existente. Esto requerirá una nueva ruta y plantilla para mostrar el formulario de edición, y otra

ruta para procesar la actualización.

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE	
Criterios de evaluación	Evidencias: El proyecto completo de la aplicación de tareas. Capturas de pantalla mostrando: la lista de tareas, el formulario para agregar una nueva, y la lista después de haber agregado y eliminado tareas.
Rúbricas o listas de cotejo para valorar desempeño	Rúbrica de práctica de laboratorio
Formatos de reporte de prácticas	

FUENTES DE INFORMACIÓN

Flask. (n.d.). *Handling Form Data*. Pallets Projects. Recuperado el 30 de junio de 2025, de https://flask.palletsprojects.com/en/latest/patterns/wtforms/

Python Software Foundation. (n.d.). *Documentation*. Recuperado el 30 de junio de 2025, de https://www.python.org/doc/

Python Software Foundation. (n.d.). *IDLE*. En The Python Standard Library. Recuperado el 30 de junio de 2025, de https://docs.python.org/3/library/idle.html

Python Software Foundation. (n.d.). *Modules*. En The Python Tutorial. Recuperado el 30 de junio de 2025, de https://docs.python.org/3/tutorial/modules.html

Python Software Foundation. (n.d.). *Python for Beginners*. Recuperado el 30 de junio de 2025, de https://www.python.org/about/gettingstarted/

Python Software Foundation. (n.d.). *More control flow tools*. En The Python Tutorial. Recuperado el 30 de junio de 2025, de https://docs.python.org/3/tutorial/controlflow.html

W3Schools. (n.d.). *Python For Loops*. Recuperado el 30 de junio de 2025, de https://www.w3schools.com/python/python_for_loops.asp

W3Schools. (n.d.). *Python While Loops*. Recuperado el 30 de junio de 2025, de https://www.w3schools.com/python/python_while_loops.asp


UES
Universidad Estatal de Sonora
La Fuerza del Saber Estimulará mi Espíritu