PRÁCTICAS DE LABORATORIO

No. De Control	Nombre	Fecha de realización
21690270	Jorge Emmanuel Pérez Martínez	25/02/2025

CARRERA	PLAN DE ESTUDIO	CLAVE DE ASIGNATURA
Ingeniería en Sistemas Computacionales		SCC - 1012
NOMBRE DE LA ASIGNATURA	**PRACTICA No.**	**NOMBRE DE LA PRÁCTICA**
Inteligencia Artificial	1	Python

2 OBJETIVO (COMPETENCIA)

Proporcionar una introducción práctica a los conceptos básicos de Python y su entorno de desarrollo, capacitándolos para implementar soluciones iniciales en el ámbito de la Inteligencia Artificial.

3 INTRODUCCIÓN:

Python se ha consolidado como uno de los lenguajes de programación más utilizados en el campo de la Inteligencia Artificial. Su sintaxis clara y la amplia disponibilidad de bibliotecas especializadas, como TensorFlow, Keras y PyTorch, facilitan el desarrollo eficiente de modelos de aprendizaje automático y redes neuronales. Además, herramientas como Jupyter Notebook ofrecen entornos interactivos que enriquecen el proceso de experimentación y visualización de datos, aspectos fundamentales en proyectos de IA.

Python se ha consolidado como uno de los lenguajes de programación más utilizados en el campo de la Inteligencia Artificial. Su **sintaxis clara y legible** facilita el aprendizaje, haciéndolo ideal tanto para principiantes como para expertos. Además, ofrece las siguientes características clave que lo convierten en una herramienta poderosa para el desarrollo de IA:

- **Multiparadigma:** Soporta programación **orientada a objetos, funcional e imperativa**, lo que permite flexibilidad en el diseño de soluciones.
- **Bibliotecas y Frameworks especializados:** Dispone de un **ecosistema robusto** para IA y aprendizaje automático, incluyendo **TensorFlow, Keras, PyTorch, Scikit-learn, NumPy y Pandas**, que agilizan el desarrollo y la implementación de modelos complejos.
- **Entorno interactivo:** Herramientas como **Jupyter Notebook** permiten escribir código, visualizar datos y documentar experimentos en un solo lugar, favoreciendo el **análisis exploratorio de datos**.
- **Portabilidad y compatibilidad:** Funciona en **Windows, macOS y Linux** sin cambios en el código, lo que asegura que los proyectos sean fácilmente compartibles y reproducibles.
- **Comunidad activa y soporte extenso:** Cuenta con una **amplia comunidad de desarrolladores**, lo que se traduce en recursos de aprendizaje, documentación actualizada y soluciones rápidas a problemas comunes.

4 MATERIALES Y EQUIPO

Requerimientos de hardware:

1. **Procesador**: Intel Core i3 o equivalente.
2. **Memoria RAM**: 4 GB mínimo (recomendado 8 GB para un rendimiento más fluido).
3. **Almacenamiento**: Al menos 20 GB de espacio libre en disco.

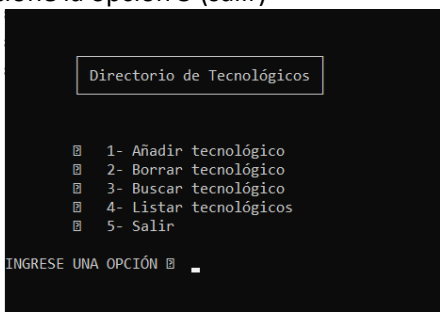
4. **Resolución de pantalla**: 1366x768 píxeles (recomendado 1920x1080 para una mejor experiencia de desarrollo).
5. **Conexión a internet**: Para instalar dependencias y paquetes desde pip.

Requerimientos de software:

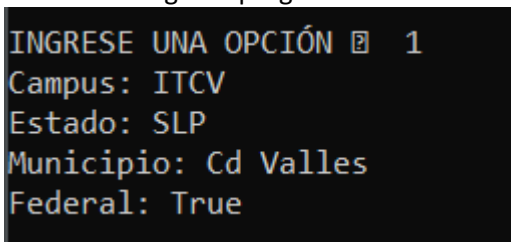
1. **Sistema Operativo:** Compatible con Windows, macOS o distribuciones de Linux.
2. **Software adicional:** Navegador web actualizado para acceder a Jupyter Notebook.

5 PROCEDIMIENTOS (DESCRIPCIÓN)

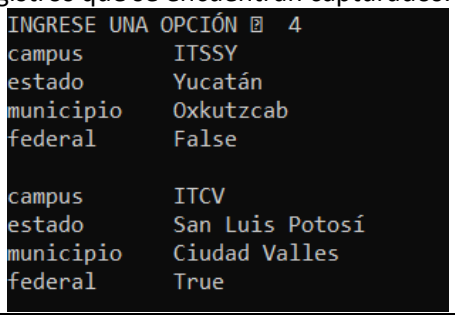
1. Crea un cuaderno Jupyter Notebook
2. Documenta en todo momento utilizando celdas Markdown y comentarios en código.
3. Desarrolla un programa en Python para gestionar registros; darlos de alta, baja y consultarlos, de acuerdo a los siguientes requerimientos:
    a. Estructura de datos.
        - "campus": str,
        - "estado": str,
        - "municipio": str,
        - "federal": boolean
    b. Menú de opciones. El programa debe continuar en ejecución mientras no se seleccione la opción 5 (salir)



    c. Inserta un nuevo registro preguntando la entrada de cada campo.



    d. Elimina y Consulta un registra a través del nombre del Tecnológico.
    e. Lista los registros que se encuentran capturados.



f. El código debe estar estructurado en funciones para cada operación. Cada función debe contener comentarios al interior del código, independientemente de la documentación de este reporte.

6 SUGERENCIAS DIDÁCTICAS

1. https://docs.python.org/es/3/

7 DOCUMENTACIÓN DEL PROCEDIMIENTOS

Código

Registro de Campus

```
[38]: import json # Se usa para formatear la impresión de los registros
```

```
[39]: registros = [] # lista vacía para almacenar los registros
```

Se importa el módulo json para formatear la salida de los registros en un formato más legible. Luego, se define una lista vacía llamada registros, que almacenará la información ingresada por el usuario, funcionando como una base de datos en memoria.

Funcion Mostrar menu

```
[40]: def mostrar_menu():
          print("\nMenú de opciones:")
          print("1. Insertar un nuevo registro")
          print("2. Eliminar un registro")
          print("3. Consultar un registro")
          print("4. Listar todos los registros")
          print("5. Salir")
```

Se define la función mostrar_menu(), que imprime en la consola un listado con las opciones disponibles para gestionar los registros. Este menú permite al usuario seleccionar la operación que desea realizar, como insertar, eliminar, consultar, listar registros o salir del programa.

Funcion insertar

```
[12]: def insertar_registro():
          campus = input("Ingrese el nombre del campus: ")  # Solicita el nombre del campus
          estado = input("Ingrese el estado: ")  # Solicita el estado donde se encuentra
          municipio = input("Ingrese el municipio: ")  # Solicita el municipio
          federal = input("¿Es federal? (si/no): ").strip().lower() == "si"  # Convierte la respuesta en booleano

          registros.append({  # Agrega el nuevo registro a la lista de registros
              "campus": campus,
              "estado": estado,
              "municipio": municipio,
              "federal": federal
          })
          print("Registro agregado exitosamente.")
```

La función insertar_registro() solicita al usuario los datos de un campus, incluyendo su nombre, estado, municipio y si es federal. La respuesta sobre si es federal se convierte en un valor booleano (True o False) verificando si el usuario ingresó "si". Una vez obtenidos los datos, se almacenan en la lista registros como un diccionario y se muestra un mensaje de confirmación al usuario.

Funcion eliminar

```
[13]: def eliminar_registro():
          """Elimina un registro de la base de datos por nombre de campus."""
          campus = input("Ingrese el nombre del campus a eliminar: ")  # Solicita el nombre del campus
          for registro in registros:
              if registro["campus"].lower() == campus.lower():  # Busca el registro en la lista
                  registros.remove(registro)  # Usa remove para eliminar el registro encontrado
                  print("Registro eliminado exitosamente.")
                  return
          print("Registro no encontrado.")
```

La función eliminar_registro() permite eliminar un campus de la lista de registros. Primero, solicita al usuario el nombre del campus que desea eliminar. Luego, recorre la lista en busca de un registro con un nombre coincidente (ignorando mayúsculas y minúsculas) y, si lo encuentra, lo elimina usando remove(). Si el registro no se encuentra, se notifica al usuario.

Funcion de Consulta

```
[14]: def consultar_registro():
          campus = input("Ingrese el nombre del campus a consultar: ")  # Solicita el nombre del campus
          for registro in registros:
              if registro["campus"].lower() == campus.lower():  # Compara sin distinguir mayúsculas
                  print(json.dumps(registro, indent=4, ensure_ascii=False))  # Imprime el registro en formato JSON
                  return
          print("Registro no encontrado.")
```

La función consultar_registro() permite buscar un campus en la lista ingresando su nombre. Recorre la lista para encontrar un registro cuyo nombre coincida con el ingresado por el usuario y, si lo encuentra, lo imprime en formato JSON para mejorar su legibilidad. Si el registro no existe, se muestra un mensaje informando que no se encontró.

Funcion para imprimir registros

```
[15]: def listar_registros():
          """Lista todos los registros almacenados."""
          if registros:  # Verifica si hay registros en la lista
              for registro in registros:
                  print(json.dumps(registro, indent=4, ensure_ascii=False))  # Imprime cada registro en formato JSON
          else:
              print("No hay registros almacenados.")
```

La función listar_registros() muestra todos los registros almacenados en la lista. Si la lista contiene registros, los imprime uno por uno en formato JSON; de lo contrario, indica que no hay registros disponibles. Esto permite al usuario ver toda la información ingresada de manera organizada.

Funcion principal

```
def main():
    """Función principal que ejecuta el menú en bucle utilizando match-case."""
    while True:
        mostrar_menu()
        opcion = input("Seleccione una opción: ")  # Solicita al usuario una opción
        match opcion:
            case "1":
                insertar_registro()
            case "2":
                eliminar_registro()
            case "3":
                consultar_registro()
            case "4":
                listar_registros()
            case "5":
                print("Saliendo del programa...")
                break  # Termina la ejecución del programa
            case _:
                print("Opción no válida. Intente nuevamente.")
```

```
main()# Inicia el programa ejecutando la función principal
```

La función main() ejecuta un bucle que permite al usuario interactuar con el programa hasta que decida salir. Primero, muestra el menú y espera la selección del usuario. Luego, usa match-case para llamar a la función correspondiente según la opción ingresada, permitiendo insertar,

eliminar, consultar o listar registros. Si el usuario elige la opción "5", se muestra un mensaje de salida y se rompe el bucle con break. Si la opción ingresada no es válida, se informa al usuario.

Para iniciar el programa, se llama a la función `main()`, lo que permite que el usuario pueda interactuar con el sistema de gestión de registros desde el menú de opciones.

Ejecución

```
Menú de opciones:
1. Insertar un nuevo registro
2. Eliminar un registro
3. Consultar un registro
4. Listar todos los registros
5. Salir
Seleccione una opción:
```

```
Seleccione una opción:  1
Ingrese el nombre del campus:  ITCV
Ingrese el estado:  SLP
Ingrese el municipio:  Cd Valles
¿Es federal? (si/no):  si
Registro agregado exitosamente.
```

```
Menú de opciones:
1. Insertar un nuevo registro
2. Eliminar un registro
3. Consultar un registro
4. Listar todos los registros
5. Salir
Seleccione una opción:  3
Ingrese el nombre del campus a consultar:  ITCV
{
    "campus": "ITCV",
    "estado": "SLP",
    "municipio": "Cd Valles",
    "federal": true
}
```

```
Menú de opciones:
1. Insertar un nuevo registro
2. Eliminar un registro
3. Consultar un registro
4. Listar todos los registros
5. Salir
Seleccione una opción:  4
{
    "campus": "ITCV",
    "estado": "SLP",
    "municipio": "Cd Valles",
    "federal": true
}
{
    "campus": "ITYP",
    "estado": "Yugopotamia",
    "municipio": "Tlaxcala",
    "federal": false
}
```

```
Menú de opciones:
1. Insertar un nuevo registro
2. Eliminar un registro
3. Consultar un registro
4. Listar todos los registros
5. Salir
Seleccione una opción:  2
Ingrese el nombre del campus a eliminar:  ITCV
Registro eliminado exitosamente.

Menú de opciones:
1. Insertar un nuevo registro
2. Eliminar un registro
3. Consultar un registro
4. Listar todos los registros
5. Salir
Seleccione una opción:  4
{
    "campus": "ITYP",
    "estado": "Yugopotamia",
    "municipio": "Tlaxcala",
    "federal": false
}
```