Trabajo Práctico: Evaluación Python Básico e Intermedio

Instrucciones Generales

1. Resuelve todos los ejercicios en archivos Python (.py) separados
2. Documenta tu código con comentarios explicativos
3. Prueba cada función con diferentes casos de entrada
4. Para los proyectos, incluye un archivo README.md explicando tu solución

PARTE I: EJERCICIOS BÁSICOS

Ejercicio 1: Tipos de Datos y Variables

Crea una función que reciba tres parámetros: nombre (string), edad (entero) y salario (float). La función debe retornar un diccionario con esta información y calcular el salario anual.

```
def crear_perfil_empleado(nombre, edad, salario_mensual):

    # Tu código aquí

    pass
```

Ejercicio 2: Manipulación de Strings

Escribe una función que reciba una frase y retorne:

- La frase en mayúsculas
- La frase en minúsculas
- El número de palabras
- La frase con las palabras en orden inverso

Ejercicio 3: Operaciones con Listas

Dada una lista de números, crea funciones para:

- Encontrar el mayor y menor número
- Calcular la suma y promedio
- Filtrar solo los números pares
- Ordenar la lista de mayor a menor

Ejercicio 4: Trabajo con Tuplas

Crea una función que reciba una tupla con coordenadas (x, y) de varios puntos y retorne:

- El punto más cercano al origen (0,0)
- La distancia promedio de todos los puntos al origen

Ejercicio 5: Operaciones con Conjuntos

Dados dos conjuntos de números, implementa funciones para:

- Encontrar elementos comunes (intersección)
- Encontrar elementos únicos de cada conjunto
- Crear un conjunto con todos los elementos (unión)
- Verificar si un conjunto es subconjunto del otro

Ejercicio 6: Manipulación de Diccionarios

Crea un sistema de inventario usando diccionarios. Implementa funciones para:

- Agregar un producto con su cantidad y precio
- Actualizar la cantidad de un producto existente
- Calcular el valor total del inventario
- Encontrar el producto más caro

Ejercicio 7: Funciones con Parámetros Variables

Escribe una función que acepte un número variable de argumentos numéricos y:

- Calcule estadísticas básicas (suma, promedio, máximo, mínimo)
- Permita especificar una operación mediante un parámetro keyword

Ejercicio 8: Comprensión de Listas

Usa list comprehension para:

- Crear una lista de cuadrados de números pares del 1 al 20
- Filtrar palabras que contengan más de 5 letras de una lista
- Crear una matriz 3x3 con números del 1 al 9

PARTE II: EJERCICIOS INTERMEDIOS

Ejercicio 9: Funciones Recursivas

Implementa las siguientes funciones recursivas:

- Factorial de un número
- Secuencia de Fibonacci

- Suma de dígitos de un número

Ejercicio 10: Clases y Objetos

Diseña una clase `CuentaBancaria` con:

- Atributos: titular, saldo, número de cuenta
- Métodos: depositar, retirar, consultar_saldo, transferir
- Validaciones apropiadas (no permitir saldos negativos)

Ejercicio 11: Herencia y Polimorfismo

Extiende la clase anterior creando `CuentaAhorro` y `CuentaCorriente`:

- CuentaAhorro: genera intereses mensuales
- CuentaCorriente: permite sobregiros hasta un límite

Ejercicio 12: Módulos y Paquetes

Crea un módulo `utilidades.py` con funciones matemáticas útiles:

- Verificar si un número es primo
- Calcular el máximo común divisor
- Convertir entre diferentes bases numéricas

Ejercicio 13: Trabajo con Archivos

Implementa funciones para:

- Leer un archivo CSV con datos de ventas
- Calcular totales por mes y por producto
- Generar un reporte en formato texto
- Manejar errores de archivo no encontrado

Ejercicio 14: JSON y Serialización

Crea un sistema que:

- Convierta objetos Python a JSON
- Guarde y cargue configuraciones desde archivos JSON
- Valide la estructura de datos JSON

Ejercicio 15: Expresiones Regulares

Usando la librería `re`, crea funciones para:

- Validar direcciones de email
- Extraer números de teléfono de un texto

- Limpiar y formatear texto (eliminar caracteres especiales)

PROYECTOS INTEGRADORES

Proyecto 1: Sistema de Gestión de Biblioteca (10 puntos)

Desarrolla un sistema completo de gestión de biblioteca que incluya:

Requisitos funcionales:

- Gestión de libros (agregar, buscar, eliminar, actualizar)
- Gestión de usuarios (registro, búsqueda)
- Sistema de préstamos y devoluciones
- Reportes de libros más prestados
- Multas por retraso en devoluciones

Requisitos técnicos:

- Usar clases para modelar Libro, Usuario, Prestamo
- Persistir datos en archivos JSON
- Implementar búsquedas eficientes usando diccionarios
- Manejo completo de excepciones
- Interfaz de línea de comandos intuitiva

Estructura sugerida:

```
biblioteca/
├── main.py
├── modelos/
│   ├── libro.py
│   ├── usuario.py
│   └── prestamo.py
├── servicios/
│   ├── biblioteca_service.py
│   └── reporte_service.py
├── datos/
│   ├── libros.json
│   ├── usuarios.json
```

```
|   └── prestamos.json
└── README.md
```

Proyecto 2: Analizador de Datos de Ventas

Crea un sistema de análisis de datos de ventas que:

Requisitos funcionales:

- Cargar datos de ventas desde archivos CSV
- Generar estadísticas por período, producto y vendedor
- Identificar tendencias y patrones
- Crear reportes automatizados
- Detectar anomalías en las ventas

Requisitos técnicos:

- Usar pandas para manipulación de datos (si está disponible) o implementar funcionalidad similar
- Generar gráficos simples usando matplotlib (opcional)
- Implementar funciones de agregación personalizadas
- Usar decoradores para logging de operaciones
- Sistema de configuración usando archivos JSON

Funcionalidades específicas:

- Calcular métricas: ventas totales, promedio, crecimiento mensual
- Top 10 productos más vendidos
- Performance por vendedor
- Predicciones simples basadas en tendencias históricas
- Exportar reportes en diferentes formatos (TXT, CSV, JSON)

Datos de ejemplo a procesar:

```
fecha,producto,vendedor,cantidad,precio_unitario,region

2024-01-15,Laptop,Juan,2,1200.50,Norte

2024-01-16,Mouse,Maria,10,25.90,Sur

...
```

Criterios de Evaluación

Funcionalidad (40%)

- El código ejecuta correctamente
- Cumple con todos los requisitos especificados
- Maneja casos edge apropiadamente

Calidad del Código (30%)

- Código limpio y bien estructurado
- Uso apropiado de convenciones de Python (PEP 8)
- Comentarios y documentación adecuados
- Nombres de variables y funciones descriptivos

Uso de Conceptos Python (20%)

- Aplicación correcta de estructuras de datos
- Uso eficiente de funciones y clases
- Implementación adecuada de manejo de archivos
- Aplicación de principios de programación orientada a objetos

Creatividad e Innovación (10%)

- Soluciones creativas a problemas complejos
- Implementación de funcionalidades adicionales
- Optimizaciones de rendimiento
- Interfaz de usuario intuitiva

Entrega

1. **Código fuente:** Todos los archivos Python necesarios
2. **Documentación:** README.md para cada proyecto explicando:
   - Cómo ejecutar el programa
   - Dependencias requeridas
   - Ejemplos de uso
   - Decisiones de diseño tomadas
1. **Datos de prueba:** Archivos de ejemplo para probar los proyectos
2. **Reflexión:** Documento breve (1-2 páginas) sobre:
   - Principales desafíos enfrentados
   - Conceptos de Python que fueron más útiles
   - Áreas donde necesitas mejorar

Recursos Permitidos y Recomendados

Documentación Oficial

- **Python.org:** https://docs.python.org/3/ - Documentación oficial completa
- **Python Tutorial:** https://docs.python.org/3/tutorial/ - Tutorial oficial para principiantes
- **Python Library Reference:** https://docs.python.org/3/library/ - Referencia de bibliotecas estándar

Libros Recomendados

Nivel Básico-Intermedio

1. **"Automate the Boring Stuff with Python" - Al Sweigart**
   - Disponible gratis: https://automatetheboringstuff.com/
   - Excelente para aplicaciones prácticas de Python
1. **"Python Crash Course" - Eric Matthes**
   - Cubre desde básicos hasta proyectos completos
   - Incluye ejercicios prácticos similares a este taller
1. **"Learning Python" - Mark Lutz**
   - Referencia completa de Python
   - Ideal para conceptos fundamentales

Nivel Intermedio-Avanzado

1. **"Effective Python" - Brett Slatkin**
   - Mejores prácticas y patrones de Python
   - Útil para los ejercicios avanzados
1. **"Python Tricks: The Book" - Dan Bader**
   - Técnicas avanzadas y trucos de Python
   - Disponible en: https://realpython.com/

Sitios Web y Recursos Online

Tutoriales y Guías

- **Real Python:** https://realpython.com/
- Tutoriales detallados sobre todos los temas del taller
- Artículos sobre clases, decoradores, manejo de archivos
- **GeeksforGeeks Python:** https://www.geeksforgeeks.org/python-programming-language/
- Ejemplos prácticos y explicaciones claras
- Sección específica de estructuras de datos
- **Python.org Beginner's Guide:** https://wiki.python.org/moin/BeginnersGuide
- Recursos organizados por nivel de experiencia

Datasets para Proyectos

- **Kaggle Datasets:** https://www.kaggle.com/datasets
- Para el proyecto de análisis de datos
- **JSON Generator:** https://www.json-generator.com/
- Para crear datos de prueba en formato JSON

Reglas de Uso de Recursos

Permitido:

- Consultar documentación y tutoriales
- Usar ejemplos como referencia para entender conceptos
- Adaptar código de ejemplos a tu solución específica
- Materiales del curso y notas de clase

No Permitido:

- Copiar código completo de internet sin entender
- Usar soluciones exactas encontradas online
- Plagiar código de otros estudiantes
- Usar herramientas de IA para generar código completo

Consejos para Usar los Recursos

1. **Lee primero, codifica después:** Entiende el concepto antes de implementar
2. **Experimenta:** Prueba los ejemplos en tu entorno local
3. **Documenta tu aprendizaje:** Toma notas sobre lo que aprendes
4. **Practica regularmente:** Usa los sitios de ejercicios para fortalecer conceptos
5. **Pregunta en foros:** Si tienes dudas específicas, las comunidades ayudan