Práctica de Laboratorio 02

En la primera semana se desarrolló la primera clase de laboratorio en donde se explicó como mediante librerías potentes como NumPy y Pandas podemos emplear ciertos principios de algoritmos evolutivos para optimizar ciertos recursos aplicados a casos de la vida real de un estudiante (como ejemplo).

Actividad de esta semana: Empleando las librerías de NumPy o Pandas codificar en Python los siguientes enunciados de ejercicios propuestos teniendo como referencia lo trabajado en el Laboratorio 1.

Nota: Emplear Google Colab o Visual Studio Code. Puedes subir (compartir) su enlace de Google Colab o subir su Repositorio de GitHub de la práctica desarrollada. *No olvidar agregar a GxJohan como colaborador de sus proyectos GitHub relativos al curso.*

1. **FOTOCOPIAS PARA APUNTES**
   - **Contexto:** Martina tiene **S/ 8** para fotocopiar sus apuntes. Hay tres copisterías cerca con precios por página: S/ 0.10, S/ 0.12 y S/ 0.08.
   - **Objetivo:**
     1. Calcular cuántas páginas puede fotocopiar en cada copistería.
     2. Identificar en cuál obtiene más páginas con su presupuesto.
   - **Pista:** usa *np.floor* para dividir el presupuesto por el precio y redondear hacia abajo, y *argmax* para hallar el índice de la mejor opción.

2. **VIAJES AL CAMPUS**
   - **Contexto:** Carlos dispone de **S/ 15** para transporte la próxima semana. Un pasaje de bus cuesta S/ 2.50, de combi S/ 3.00 y de tren S/ 1.80.
   - **Objetivo:**
     1. Determinar cuántos viajes puede pagar con cada medio.
     2. Hallar el medio de transporte que le permite *más viajes*.
   - **Pista:** crea un *np.array* de precios y usa *array.max()* y *array.argmax()*.

3. **PRÉSTAMO DE LIBROS EN LA BIBLIOTECA**
   - **Contexto:** En la tabla tienes cinco compañeros y los días que retuvieron un libro:

Estudiante	Días_prestamo
Rosa	7
David	10
Elena	5
Mario	12
Paula	3

   - **Objetivo:**

     Crear un *DataFrame* con esos datos.

     Calcular el **promedio** y el **máximo** de días de préstamo.

     Filtrar quiénes retuvieron el libro **más de 8 días**.

   - **Pista:** emplea *df['Días_prestamo'].describe()* para resumen y *df[df['Días_prestamo'] > 8]* para el filtro.

4. **GASTOS DE ALMUERZO SEMANAL**
   - **Contexto:** Ana apunta lo que gasta cada día en el comedor universitario en una lista: [4.0, 3.5, 5.0, 4.2, 3.8] (de lunes a viernes).
   - **Objetivo:**
     1. Pasar esa lista a un *DataFrame* con columna Gasto.
     2. Calcular el **gasto total** y el **gasto medio** de la semana.
     3. Identificar los días en que gastó **más que el promedio**.
   - **Pista:** usa *df['Gasto'].sum(), df['Gasto'].mean() y luego df[df['Gasto'] > df['Gasto'].mean()].*

5. **RECARGA DE DATOS MÓVILES**

Contexto: Juan tiene un plan de datos con distintos paquetes:

Paquete (GB)	Precio (S/)
1	5
2	9
5	20
10	35

- **Objetivo:**
  1. Con un array de NumPy, calcular el **costo por GB** para cada paquete.
  2. Encontrar el **paquete más económico** en precio por GB.
- **Pista:** crea *np.array* para GB y para precios, luego divide y aplica *min()* y *argmin()*.

6. **OPTIMIZACIÓN DE RUTA DE ENTREGA (Hill Climbing)**
   - **Contexto:** Juan debe entregar volantes en 6 facultades distintas alrededor del campus. Cada facultad está numerada del 0 al 5, y la "distancia" entre facultades se representa con una matriz de costos (en minutos) predeterminada. Juan parte de la facultad 0.
   - **Objetivo:**
     1. Partiendo de una ruta inicial (por ejemplo, [0,1,2,3,4,5,0]), generar **vecinos** intercambiando dos facultades consecutivas.
     2. Calcular el **tiempo total** de la ruta (suma de los costos entre paradas).
     3. Aplicar el **criterio hill climbing**: si un vecino tiene menor tiempo total, reemplazar la ruta actual.
   - **Pista:** recorre los índices, intercambia pares adyacentes para crear vecinos, usa *sum()* sobre un array de distancias y un bucle *for* para comparar.

7. **AJUSTE DE HORARIO DE ESTUDIO (Simulated Annealing)**
   - **Contexto:** María quiere planificar 5 sesiones de estudio al día, en intervalos de 1 h, para cubrir 5 materias distintas. Hay un "puntaje de fatiga" asociado a cambiar de materia (un número entre 1 y 5).
   - **Objetivo:**
     1. Representa una solución como una lista de 5 materias en orden, calcula el **costo** total sumando los puntajes de fatiga de los cambios consecutivos.
     2. Implementa una iteración de **simulated annealing** sencilla: genera un vecino permutando dos materias al azar, y
        - Si el vecino tiene menor costo, acéptalo.

- Si es peor, acéptalo con probabilidad exp(−Δcoste / T), donde T desciende linealmente (p. ej. de 10 a 1).

3. Tras 100 iteraciones, devuelve la mejor secuencia encontrada y su costo.

- **Pista:** usa *random.shuffle* o *np.random.choice* para permutar, *math.exp* para la probabilidad de aceptación y una variable *T* que decremente en cada iteración.