# Lista 12: Gráficos

Demuestras un buen dominio de las herramientas de visualización matplotlib, seaborn y networkx. Los gráficos están bien diseñados y cumplen con los requisitos de los ejercicios, mostrando una buena creatividad y personalización. Sin embargo, en algunos casos, se podrían simplificar las configuraciones y mejorar la legibilidad del código.

* **Cosas positivas:**
  + Uso creativo de matplotlib para gráficos interactivos y personalización avanzada.
  + Implementación de conceptos más avanzados como grafos y diagramas con colores y etiquetas.
  + Configuración de tamaños, colores, y transparencia de manera efectiva.
* **Posibles mejoras:**
  + Simplificar la lógica de algunos gráficos complejos.
  + Añadir más comentarios en las configuraciones avanzadas para facilitar la comprensión.
  + Explorar el uso de estilos predefinidos de matplotlib para ahorrar tiempo en personalización.

# Ejercicios

### **Ejercicio 1: Gráfico de dispersión**

**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** No es necesario; el uso de plt.scatter con atributos como alpha y colorbar es adecuado.  
**Comentario:** Excelente manejo de gráficos de dispersión. La personalización con colores y tamaños mejora la visualización.

### **Ejercicio 2: Conversión de colores RGB a hexadecimal**

**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** Simplifica la lógica de la conversión.

**Comentario:** Muy buena implementación; la optimización sugerida mejora la legibilidad.

### **Ejercicio 3: Gráfico de funciones sinusoidales**

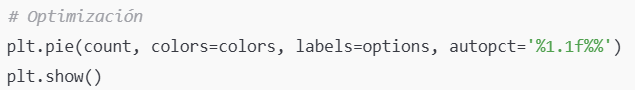
**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** No es necesario.  
**Comentario:** Los gráficos son claros y los colores están bien seleccionados para diferenciar las funciones. Uso adecuado de legend y linspace.

### **Ejercicio 4: Subgráficos de funciones sinusoidales**

**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** No es necesario.  
**Comentario:** Excelente uso de subgráficos para organizar múltiples visualizaciones. Los títulos y configuraciones son apropiados.

### **Ejercicio 5: Diagrama de sectores (pie chart)**

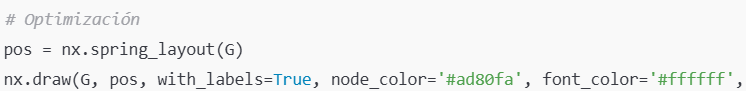
**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** Usa autopct para mostrar porcentajes directamente en el gráfico.



**Comentario:** El gráfico de sectores está bien implementado, pero mostrar porcentajes mejora la comprensión.

### **Ejercicio 6: Grafo básico con** networkx

**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** Usa spring\_layout para organizar mejor los nodos.

****

**Comentario:** Excelente manejo de grafos. La optimización mejora la distribución visual de los nodos.

### **Ejercicio 7: Grafo con tamaños de nodos personalizados**

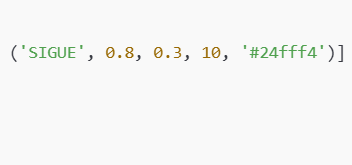
**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** No es necesario.  
**Comentario:** Buena personalización del tamaño de nodos y configuración del grafo.

### **Ejercicio 8: Grafo con pesos en las aristas**

**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** No es necesario.  
**Comentario:** Muy buena implementación de grosores proporcionales a los pesos de las aristas.

### **Ejercicio 9: Gráfico con texto personalizado**

**Estado del ejercicio:** Correcto.  
**Errores encontrados:** Ninguno.  
**Código optimizado:** Simplifica las configuraciones repetitivas.

****

**Comentario:** Buen uso de plt.text para crear un gráfico creativo. La optimización simplifica el código y mejora la reutilización.