**1. Explicación resumida línea por línea**

1. **import numpy as np**  
   Importa la librería NumPy y la asigna al alias np.
2. **import matplotlib.pyplot as plt**  
   Importa la parte de Pyplot de Matplotlib con el alias plt.
3. **from mpl\_toolkits.mplot3d import Axes3D**  
   Importa la clase Axes3D necesaria para crear gráficos en 3D.
4. **from matplotlib.animation import FuncAnimation**  
   Importa FuncAnimation para crear animaciones con Matplotlib.
5. ***(Línea en blanco)***  
   Espacio en blanco para separar secciones.
6. **# Crear una malla par alos ejes X e Y**  
   Comentario que indica la intención de crear una malla para los ejes X e Y.
7. **x = np.linspace(-5, 5, 50)**  
   Crea un arreglo de 50 números igualmente espaciados entre -5 y 5 para el eje X.
8. **y = np.linspace(-5, 5, 50)**  
   Crea un arreglo similar para el eje Y.
9. **X, Y = np.meshgrid(x, y)**  
   Genera dos matrices a partir de los arreglos x e y para representar la malla de coordenadas.
10. ***(Línea en blanco)***  
    Espacio en blanco para separar secciones.
11. **# Definir tres superficies planas (planos)**  
    Comentario que introduce la definición de tres superficies.
12. **# Plano 1: z = 0 (plano horizontal)**  
    Comentario que describe el primer plano: un plano horizontal en z=0.
13. **Z1 = np.zeros\_like(X)**  
    Crea una matriz del mismo tamaño que X, llena de ceros para definir el plano z=0.
14. ***(Línea en blanco)***  
    Espacio en blanco para separar secciones.
15. **# Plano 2: z = 0.5\*x + 0.2\*y + 1**  
    Comentario que describe el segundo plano con la ecuación dada.
16. **Z2 = 0.5 \* X + 0.2 \* Y + 1**  
    Calcula los valores de z para el segundo plano usando la ecuación lineal.
17. ***(Línea en blanco)***  
    Espacio en blanco para separar secciones.
18. **# Plano 3: z = -0.3\*x + 0.4\*y - 1**  
    Comentario que describe el tercer plano (aunque la ecuación en el comentario y en el código difieren levemente).
19. **Z3 = 0.3 \* X + 0.4 \* Y - 1**  
    Calcula los valores de z para el tercer plano (en este caso la ecuación usada es z = 0.3*x + 0.4*y - 1).
20. ***(Línea en blanco)***  
    Espacio en blanco para separar secciones.
21. **# Crear la figura y el eje 3D**  
    Comentario que indica la creación de la figura y del eje tridimensional.
22. **fig = plt.figure(figsize=(10, 8))**  
    Crea una figura con un tamaño de 10x8 pulgadas.
23. **ax = fig.add\_subplot(111, projection="3d")**  
    Agrega un subgráfico 3D a la figura.
24. ***(Línea en blanco)***  
    Espacio en blanco para separar secciones.
25. **# Graficar cada superficie con transparencia para ver las intersecciones**  
    Comentario que explica que se van a graficar las superficies con cierto nivel de transparencia.
26. **ax.plot\_surface(**  
    Inicia la llamada para graficar la superficie del primer plano.
27. **X, Y, Z1, alpha=0.5, color="blue", rstride=1, cstride=1, edgecolor="none"**  
    Parámetros para graficar la superficie: usa la malla X, Y y la superficie Z1, con transparencia, color azul y sin bordes.
28. **)**  
    Cierra la llamada a plot\_surface para el primer plano.
29. **ax.plot\_surface(**  
    Inicia la llamada para graficar la superficie del segundo plano.
30. **X, Y, Z2, alpha=0.5, color="red", rstride=1, cstride=1, edgecolor="none"**  
    Parámetros para graficar la superficie: usa la malla X, Y y la superficie Z2, con transparencia, color rojo y sin bordes.
31. **)**  
    Cierra la llamada a plot\_surface para el segundo plano.
32. **ax.plot\_surface(**  
    Inicia la llamada para graficar la superficie del tercer plano.
33. **X, Y, Z3, alpha=0.5, color="green", rstride=1, cstride=1, edgecolor="none"**  
    Parámetros para graficar la superficie: usa la malla X, Y y la superficie Z3, con transparencia, color verde y sin bordes.
34. **)**  
    Cierra la llamada a plot\_surface para el tercer plano.
35. ***(Línea en blanco)***  
    Espacio en blanco para separar secciones.
36. **# Configurar etiquetas y título**  
    Comentario que indica la configuración de las etiquetas de los ejes y el título del gráfico.
37. **ax.set\_xlabel("Eje X")**  
    Establece la etiqueta del eje X.
38. **ax.set\_ylabel("Eje Y")**  
    Establece la etiqueta del eje Y.
39. **ax.set\_zlabel("Eje Z")**  
    Establece la etiqueta del eje Z.
40. **ax.set\_title("Superficies Planas en 3D con Animación")**  
    Establece el título del gráfico.
41. ***(Línea en blanco)***  
    Espacio en blanco para separar secciones.
42. **def update(angle):**  
    Define una función llamada update que recibe un ángulo como parámetro.
43. **ax.view\_init(elev=30, azim=angle)**  
    Dentro de la función, actualiza la vista 3D con una elevación fija de 30° y un ángulo de azimut variable.
44. **return (ax,)**  
    La función retorna una tupla con el objeto ax.
45. ***(Línea en blanco)***  
    Espacio en blanco para separar secciones.
46. **# Crear la animación: se rotará la vista de 0 a 360 grados**  
    Comentario que explica la creación de la animación.
47. **anim = FuncAnimation(fig, update, frames=np.arange(0, 360, 2), interval=50)**  
    Crea una animación que rota la vista desde 0 hasta 360 grados en pasos de 2 y con un intervalo de 50 ms entre cuadros.
48. ***(Línea en blanco)***  
    Espacio en blanco para separar secciones.
49. **# Mostrar la gráfica animada**  
    Comentario que indica que se mostrará la gráfica.
50. **plt.show()**  
    Muestra la figura con la animación en una ventana.

**2. Explicación resumida del código**

Este código genera tres superficies planas en un espacio tridimensional utilizando Matplotlib y NumPy. Se crean mallas para los ejes X e Y, se definen tres superficies con distintas ecuaciones lineales y se grafican en una figura 3D. Además, se configura una animación que rota la vista de la gráfica para apreciar las intersecciones y relaciones espaciales entre los planos.

**3. Explicación detallada línea por línea**

1. **import numpy as np**  
   Importa la biblioteca NumPy, que permite trabajar con arreglos y realizar operaciones matemáticas de forma eficiente, y se asigna el alias np para facilitar su uso en el código.
2. **import matplotlib.pyplot as plt**  
   Importa la librería pyplot de Matplotlib, la cual se utiliza para crear gráficos y visualizaciones, y se asigna el alias plt para simplificar las llamadas a sus funciones.
3. **from mpl\_toolkits.mplot3d import Axes3D**  
   Importa la clase Axes3D del módulo mpl\_toolkits.mplot3d, la cual es necesaria para generar gráficos en tres dimensiones.
4. **from matplotlib.animation import FuncAnimation**  
   Importa FuncAnimation desde matplotlib.animation, una herramienta que facilita la creación de animaciones actualizando la visualización en cada cuadro.
5. ***(Línea en blanco)***  
   Línea en blanco para separar lógicamente la sección de importación de librerías del resto del código.
6. **# Crear una malla par alos ejes X e Y**  
   Comentario que indica el inicio del proceso para crear una malla de puntos que cubrirá los ejes X e Y.
7. **x = np.linspace(-5, 5, 50)**  
   Utiliza np.linspace para generar un arreglo de 50 números equidistantes que van desde -5 hasta 5. Estos valores representan las coordenadas en el eje X.
8. **y = np.linspace(-5, 5, 50)**  
   De forma similar, crea un arreglo de 50 números equidistantes para el eje Y, cubriendo el mismo rango.
9. **X, Y = np.meshgrid(x, y)**  
   Genera dos matrices (X e Y) a partir de los arreglos creados, donde cada par (x, y) corresponde a una coordenada de la malla. Esto es fundamental para graficar superficies en 3D.
10. ***(Línea en blanco)***  
    Se deja un espacio en blanco para separar la sección de creación de la malla de la definición de las superficies.
11. **# Definir tres superficies planas (planos)**  
    Comentario que introduce la definición de tres superficies planas que serán graficadas.
12. **# Plano 1: z = 0 (plano horizontal)**  
    Comentario que explica que el primer plano es horizontal, con z constante en 0.
13. **Z1 = np.zeros\_like(X)**  
    Crea una matriz Z1 con las mismas dimensiones que X, llenándola de ceros. Esto define la altura z=0 para todos los puntos de la malla.
14. ***(Línea en blanco)***  
    Espacio para separar la definición del primer plano del siguiente.
15. **# Plano 2: z = 0.5\*x + 0.2\*y + 1**  
    Comentario que describe la ecuación del segundo plano, una combinación lineal de x e y con una constante.
16. **Z2 = 0.5 \* X + 0.2 \* Y + 1**  
    Calcula los valores de z para cada punto de la malla según la ecuación del segundo plano, generando la matriz Z2.
17. ***(Línea en blanco)***  
    Espacio en blanco para separar la definición de los planos.
18. **# Plano 3: z = -0.3\*x + 0.4\*y - 1**  
    Comentario que indica la ecuación del tercer plano. (Nótese que en el código la ecuación resulta en z = 0.3*x + 0.4*y - 1, lo que implica una discrepancia leve con el comentario).
19. **Z3 = 0.3 \* X + 0.4 \* Y - 1**  
    Calcula los valores de z para el tercer plano utilizando la ecuación especificada en el código, generando la matriz Z3.
20. ***(Línea en blanco)***  
    Espacio para separar la sección de definición de superficies de la creación de la figura.
21. **# Crear la figura y el eje 3D**  
    Comentario que indica que a continuación se crea la figura donde se mostrarán los gráficos en 3D.
22. **fig = plt.figure(figsize=(10, 8))**  
    Crea una figura de Matplotlib con un tamaño de 10 pulgadas de ancho por 8 pulgadas de alto.
23. **ax = fig.add\_subplot(111, projection="3d")**  
    Agrega un subgráfico a la figura configurado para gráficos 3D. El argumento 111 indica que es el único subgráfico en la figura.
24. ***(Línea en blanco)***  
    Espacio en blanco para separar la creación de la figura de la sección de graficado.
25. **# Graficar cada superficie con transparencia para ver las intersecciones**  
    Comentario que explica que se graficarán las superficies con cierto nivel de transparencia para visualizar superposiciones e intersecciones.
26. **ax.plot\_surface(**  
    Inicia la función para graficar la primera superficie en el eje 3D.
27. **X, Y, Z1, alpha=0.5, color="blue", rstride=1, cstride=1, edgecolor="none"**  
    Define los parámetros de la gráfica: utiliza la malla (X, Y) y la superficie Z1, establece una transparencia del 50% (alpha=0.5), el color azul, y configura el espaciado de filas y columnas y quita el color de borde.
28. **)**  
    Cierra la función plot\_surface para el primer plano.
29. **ax.plot\_surface(**  
    Inicia la función para graficar la segunda superficie.
30. **X, Y, Z2, alpha=0.5, color="red", rstride=1, cstride=1, edgecolor="none"**  
    Configura los parámetros para la segunda superficie usando Z2, con transparencia, color rojo y sin bordes.
31. **)**  
    Cierra la función plot\_surface para el segundo plano.
32. **ax.plot\_surface(**  
    Inicia la función para graficar la tercera superficie.
33. **X, Y, Z3, alpha=0.5, color="green", rstride=1, cstride=1, edgecolor="none"**  
    Configura los parámetros para la tercera superficie usando Z3, con transparencia, color verde y sin bordes.
34. **)**  
    Cierra la función plot\_surface para el tercer plano.
35. ***(Línea en blanco)***  
    Espacio en blanco para separar la sección de graficado de la configuración de etiquetas.
36. **# Configurar etiquetas y título**  
    Comentario que indica que se configurarán las etiquetas de los ejes y el título de la gráfica.
37. **ax.set\_xlabel("Eje X")**  
    Establece la etiqueta del eje X con el texto "Eje X".
38. **ax.set\_ylabel("Eje Y")**  
    Establece la etiqueta del eje Y con el texto "Eje Y".
39. **ax.set\_zlabel("Eje Z")**  
    Establece la etiqueta del eje Z con el texto "Eje Z".
40. **ax.set\_title("Superficies Planas en 3D con Animación")**  
    Define el título del gráfico, indicando que se muestran superficies planas en 3D con animación.
41. ***(Línea en blanco)***  
    Espacio en blanco para separar la configuración de la figura de la definición de la función de actualización.
42. **def update(angle):**  
    Define una función llamada update que recibe un parámetro angle para actualizar la vista de la gráfica.
43. **ax.view\_init(elev=30, azim=angle)**  
    Dentro de la función, se actualiza la vista del gráfico en 3D fijando la elevación a 30° y modificando el ángulo de azimut según el parámetro recibido.
44. **return (ax,)**  
    Retorna una tupla con el objeto ax, requisito de la función de animación para saber qué se actualiza.
45. ***(Línea en blanco)***  
    Espacio en blanco para separar la función de actualización de la creación de la animación.
46. **# Crear la animación: se rotará la vista de 0 a 360 grados**  
    Comentario que explica que se va a crear una animación rotativa de la vista del gráfico.
47. **anim = FuncAnimation(fig, update, frames=np.arange(0, 360, 2), interval=50)**  
    Crea la animación aplicando la función update a la figura fig. La animación varía el ángulo de 0 a 360 grados en pasos de 2, con un intervalo de 50 milisegundos entre cada cuadro.
48. ***(Línea en blanco)***  
    Espacio en blanco para separar la animación de la instrucción final de mostrar la gráfica.
49. **# Mostrar la gráfica animada**  
    Comentario que indica que se mostrará la figura con la animación en una ventana.
50. **plt.show()**  
    Llama a la función show() de Matplotlib para renderizar y mostrar la ventana con la gráfica animada.

