Metodología:

Nos dividimos los distintos bloques de código de esta sección. En mi caso lo que hice fueron los siguientes puntos:

1. Función para la Determinación de Años Bisiestos

Desarrollé la función es_bisiesto(anio), la cual es esencial para identificar "años especiales". Esta función toma un año como entrada y aplica las reglas del calendario gregoriano: un año es bisiesto si es divisible por 4, excepto si también es divisible por 100, a menos que sea divisible por 400. La función devuelve True si el año cumple estas condiciones y es bisiesto, y False en caso contrario.

```
27  def es_bisiesto(anio):
28      if (anio % 4 == 0 and anio % 100 != 0) or (anio % 400 == 0):
29          return True
30      else:
31          return False
```

2. Conteo de Años Pares e Impares

Implementé el bloque de código encargado de clasificar los años de nacimiento en pares o impares. Si se han ingresado años, el programa recorre la lista completa de años de nacimiento. Por cada año, verifica si es divisible por 2 para determinar si es par o impar y actualiza contadores específicos. Una vez procesados todos los años, se imprime el total de años pares y el total de años impares, ofreciendo una perspectiva inicial sobre la distribución de los años en el grupo.

```
44      if anios_nacimiento:
45          contador_pares = 0
46          contador_impares = 0
47
48          for anio in anios_nacimiento:
49              if anio % 2 == 0:
50                  contador_pares += 1
51              else:
52                  contador_impares += 1
53
54          print(f"La cantidad de años pares es: {contador_pares}")
55          print(f"La cantidad de años impares: {contador_impares}")
56      else:
57          print ("No hay elementos en la lista")
```

3. Verificación de la "Generación Z"

Me encargué de la sección que comprueba si el grupo pertenece a la "Generación Z". Esta lógica se basa en la condición de que todos los integrantes deben haber nacido después del año 2000. Se utiliza una bandera gen_z, que se inicializa en True (asumiendo que es

Generación Z). El código luego itera por cada año de nacimiento; si encuentra un solo año anterior o igual a 2000, la bandera gen_z se cambia a False. Finalmente, se imprime un mensaje indicando si el grupo cumple o no con la condición de ser "Generación Z".

```
59    if anios_nacimiento:
60        gen_z = True
61
62        for anio in anios_nacimiento:
63            if anio < 2000:
64                gen_z = False
65
66        if gen_z:
67            print("Grupo Z")
68        else:
69            print("No es un 'Grupo Z'. Al menos un integrante antes del año 2000.")
70    else:
71        print ("No hay elementos en la lista")
```

4. Detección de Años de Nacimiento Bisiestos

Implementé el bloque de código que identifica si algún integrante del grupo nació en un año bisiesto. Este proceso recorre la lista de años de nacimiento y, para cada año, invoca la función es_bisiesto() que definí previamente. Si la función retorna True para alguno de los años, una bandera hay_anios_bisiestos se activa. A diferencia de otros enfoques, el bucle continúa hasta el final de la lista incluso después de encontrar un bisiesto. Esto garantiza que todos los años sean revisados si fuera necesario para otras lógicas (aunque para este punto, basta con encontrar uno). Finalmente, el programa informa si "Tenemos un año especial" o si ningún integrante nació en un año bisiesto.

```
73    if anios_nacimiento:
74        hay_anios_bisiestos = False
75
76        for anio in anios_nacimiento:
77            if es_bisiesto(anio):
78                hay_anios_bisiestos = True
79
80        if hay_anios_bisiestos:
81            print("Tenemos un año especial")
82        else:
83            print("No hay años bisiestos")
84    else:
85        print("No hay elementos en la lista")
```