Enunciado:

La siguiente tabla muestra la función de cuantía de una variable aleatoria, que representa el número de seguros de vida que vende diariamnente un ejecutivo.

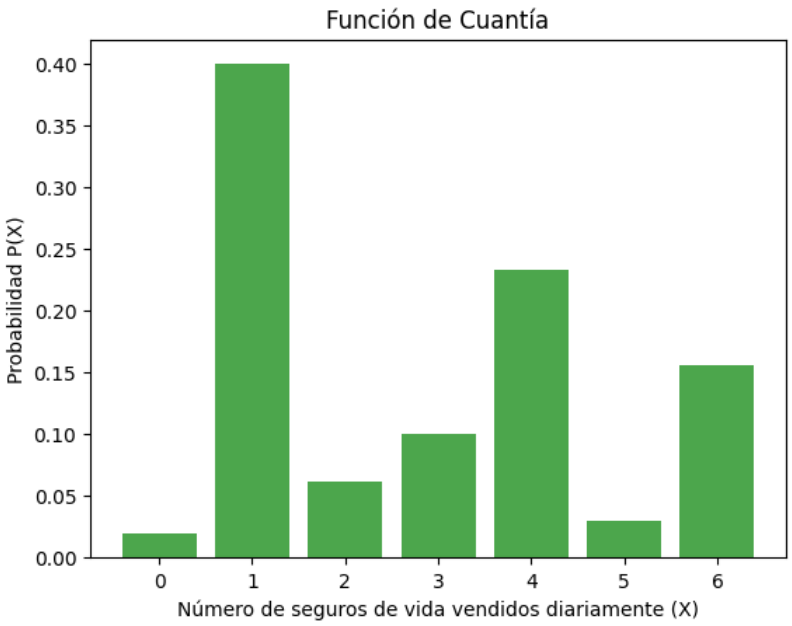
a. Determine el valor de k de modo que la distribución sea una función de cuantía.

In [ ]:
```
import matplotlib.pyplot as plt
from sympy import symbols, Eq, solve

# Definir la variable k
k = symbols('k')

# Definir la ecuación
ecuacion = Eq(0.25 * k + 0.4 + 0.8 * k + 0.1 + 3 * k + 0.03 + 2 * k, 1)

# Resolver la ecuación para k
valor_k = solve(ecuacion, k)

print(f"El valor de k es: {valor_k[0]:.2f}")
```

```
El valor de k es: 0.08
```

b. Grafique la distribución asociada al problema.

In [ ]:
```
# Valores de x y sus probabilidades asociadas
valores_x = [0, 1, 2, 3, 4, 5, 6]
probabilidades = [0.25 * valor_k[0], 0.4, 0.8 * valor_k[0], 0.1, 3 * valor_k[0], 0.03, 2 * valor_k[0]]

# Graficar la función de cuantía
plt.bar(valores_x, probabilidades, color='green', alpha=0.7)
plt.title('Función de Cuantía')
plt.xlabel('Número de seguros de vida vendidos diariamente (X)')
plt.ylabel('Probabilidad P(X)')
plt.show()
```



c. Si se eligen 15 ejecutivos, determine la probabilidad que 5 de ellos no vendan seguros.

In [ ]:
```
from scipy.stats import binom

# Definir la variable k
k = symbols('k')

# Resolver la ecuación para k
ecuacion = Eq(0.25 * k + 0.4 + 0.8 * k + 0.1 + 3 * k + 0.03 + 2 * k, 1)
valor_k = solve(ecuacion, k)

# Definir la probabilidad de "fracaso" en un solo ensayo
probabilidad_fracaso = 0.25 * valor_k[0]
```

```
# Número total de ejecutivos
n_ejecutivos = 15

# Número de "fracasos" deseados
k_fracasos = 5

# Calcular la probabilidad utilizando la distribución binomial
probabilidad_binomial = binom.pmf(k_fracasos, n_ejecutivos, probabilidad_fracaso)

# Imprimir el resultado
print(f"El valor de k es: {valor_k[0]}")
print(f"La probabilidad de que exactamente 5 ejecutivos no vendan seguros es: {probabilidad_binomial:.4f}")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
d:\UNAB\3_Trimestre\WS_Taller_metodos_cuantitativos\Clases\sumativa4_parte2\sumativa_4_parte2_preg2.ipynb Cell 7 line 2
     <a href='vscode-notebook-cell:/d%3A/UNAB/3_Trimestre/WS_Taller_metodos_cuantitativos/Clases/sumativa4_parte2/sumativa_4_parte2_preg2.ipynb#X30sZmlsZQ%3D%3D?line=16'>17</a> k_fracasos = 5
     <a href='vscode-notebook-cell:/d%3A/UNAB/3_Trimestre/WS_Taller_metodos_cuantitativos/Clases/sumativa4_parte2/sumativa_4_parte2_preg2.ipynb#X30sZmlsZQ%3D%3D?line=18'>19</a> # Calcular la probabilidad utilizando la distribución binomial
---> <a href='vscode-notebook-cell:/d%3A/UNAB/3_Trimestre/WS_Taller_metodos_cuantitativos/Clases/sumativa4_parte2/sumativa_4_parte2_preg2.ipynb#X30sZmlsZQ%3D%3D?line=19'>20</a> probabilidad_binomial = binom.pmf(k_fracasos, n_ejecutivos, probabilidad_fracaso)
     <a href='vscode-notebook-cell:/d%3A/UNAB/3_Trimestre/WS_Taller_metodos_cuantitativos/Clases/sumativa4_parte2/sumativa_4_parte2_preg2.ipynb#X30sZmlsZQ%3D%3D?line=21'>22</a> # Imprimir el resultado
     <a href='vscode-notebook-cell:/d%3A/UNAB/3_Trimestre/WS_Taller_metodos_cuantitativos/Clases/sumativa4_parte2/sumativa_4_parte2_preg2.ipynb#X30sZmlsZQ%3D%3D?line=22'>23</a> print(f"El valor de k es: {valor_k[0]}")

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\scipy\stats\_distn_infrastructure.py:3378, in rv_discrete.pmf(self, k, *args, **kwds)
   3376 if np.any(cond):
   3377     goodargs = argsreduce(cond, *((k,)+args))
-> 3378     place(output, cond, np.clip(self._pmf(*goodargs), 0, 1))
   3379 if output.ndim == 0:
   3380     return output[()]

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\scipy\stats\_discrete_distns.py:76, in binom_gen._pmf(self, x, n, p)
     74 def _pmf(self, x, n, p):
     75     # binom.pmf(k) = choose(n, k) * p**k * (1-p)**(n-k)
---> 76     return _boost._binom_pdf(x, n, p)

TypeError: ufunc '_binom_pdf' not supported for the input types, and the inputs could not be safely coerced to any supported types according to the casting rule ''safe''
```

d. Si el número de seguros que vende un ejecutivo por día es igual al valor esperado del modelo ¿cuál es la probabilidad de que en un día el ejecutivo venda mas de 6 seguros?

In [ ]:
```
from scipy.stats import rv_discrete

# Definir la distribución de cuantía
valores_x = [0, 1, 2, 3, 4, 5, 6]
probabilidades = [0.25 * valor_k[0], 0.4, 0.8 * valor_k[0], 0.1, 3 * valor_k[0], 0.03, 2 * valor_k[0]]
distribucion_cuantia = rv_discrete(name='distribucion_cuantia', values=(valores_x, probabilidades))

# Calcular la probabilidad de que el ejecutivo venda más de 6 seguros en un día
probabilidad_mas_6_seguros = 1 - distribucion_cuantia.cdf(6)

# Imprimir el resultado
print(f"La probabilidad de que el ejecutivo venda más de 6 seguros en un día es: {probabilidad_mas_6_seguros:.4f}")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
d:\UNAB\3_Trimestre\WS_Taller_metodos_cuantitativos\Clases\sumativa4_parte2\sumativa_4_parte2_preg2.ipynb Cell 9 line 6
      <a href='vscode-notebook-cell:/d%3A/UNAB/3_Trimestre/WS_Taller_metodos_cuantitativos/Clases/sumativa4_parte2/sumativa_4_parte2_preg2.ipynb#X32sZmlsZQ%3D%3D?line=3'>4</a> valores_x = [0, 1, 2, 3, 4, 5, 6]
      <a href='vscode-notebook-cell:/d%3A/UNAB/3_Trimestre/WS_Taller_metodos_cuantitativos/Clases/sumativa4_parte2/sumativa_4_parte2_preg2.ipynb#X32sZmlsZQ%3D%3D?line=4'>5</a> probabilidades = [0.25 * valor_k[0], 0.4, 0.8 * valor_k[0], 0.1, 3 * valor_k[0], 0.03, 2 * valor_k[0]]
----> <a href='vscode-notebook-cell:/d%3A/UNAB/3_Trimestre/WS_Taller_metodos_cuantitativos/Clases/sumativa4_parte2/sumativa_4_parte2_preg2.ipynb#X32sZmlsZQ%3D%3D?line=5'>6</a> distribucion_cuantia = rv_discrete(name='distribucion_cuantia', values=(valores_x, probabilidades))
      <a href='vscode-notebook-cell:/d%3A/UNAB/3_Trimestre/WS_Taller_metodos_cuantitativos/Clases/sumativa4_parte2/sumativa_4_parte2_preg2.ipynb#X32sZmlsZQ%3D%3D?line=7'>8</a> # Calcular la probabilidad de que el ejecutivo venda más de 6 seguros en un día
      <a href='vscode-notebook-cell:/d%3A/UNAB/3_Trimestre/WS_Taller_metodos_cuantitativos/Clases/sumativa4_parte2/sumativa_4_parte2_preg2.ipynb#X32sZmlsZQ%3D%3D?line=8'>9</a> probabilidad_mas_6_seguros = 1 - distribucion_cuantia.cdf(6)

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\scipy\stats\_distn_infrastructure.py:3887, in rv_sample.__init__(self, a, b, name, badvalue, moment_tol, values, inc, longname, shapes, seed)
   3885 if np.less(pk, 0.0).any():
   3886     raise ValueError("All elements of pk must be non-negative.")
-> 3887 if not np.allclose(np.sum(pk), 1):
   3888     raise ValueError("The sum of provided pk is not 1.")
   3890 indx = np.argsort(np.ravel(xk))

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\numpy\core\numeric.py:2241, in allclose(a, b, rtol, atol, equal_nan)
   2170 @array_function_dispatch(_allclose_dispatcher)
   2171 def allclose(a, b, rtol=1.e-5, atol=1.e-8, equal_nan=False):
   2172     """
   2173     Returns True if two arrays are element-wise equal within a tolerance.
   2174
   (...)
   2239
   2240     """
-> 2241     res = all(isclose(a, b, rtol=rtol, atol=atol, equal_nan=equal_nan))
   2242     return bool(res)

File ~\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-packages\numpy\core\numeric.py:2348, in isclose(a, b, rtol, atol, equal_nan)
   2345     dt = multiarray.result_type(y, 1.)
   2346     y = asanyarray(y, dtype=dt)
-> 2348 xfin = isfinite(x)
   2349 yfin = isfinite(y)
   2350 if all(xfin) and all(yfin):

TypeError: ufunc 'isfinite' not supported for the input types, and the inputs could not be safely coerced to any supported types according to the casting rule ''safe''
```