Setup

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[06 Points] Task 01 - Gaussian Random Variable

Gaussian/Normal random variables are the most common type of random variable. They are characterized by a mean μ and a standard deviation σ. The probability density function of a Gaussian/Normal random variable is given by

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

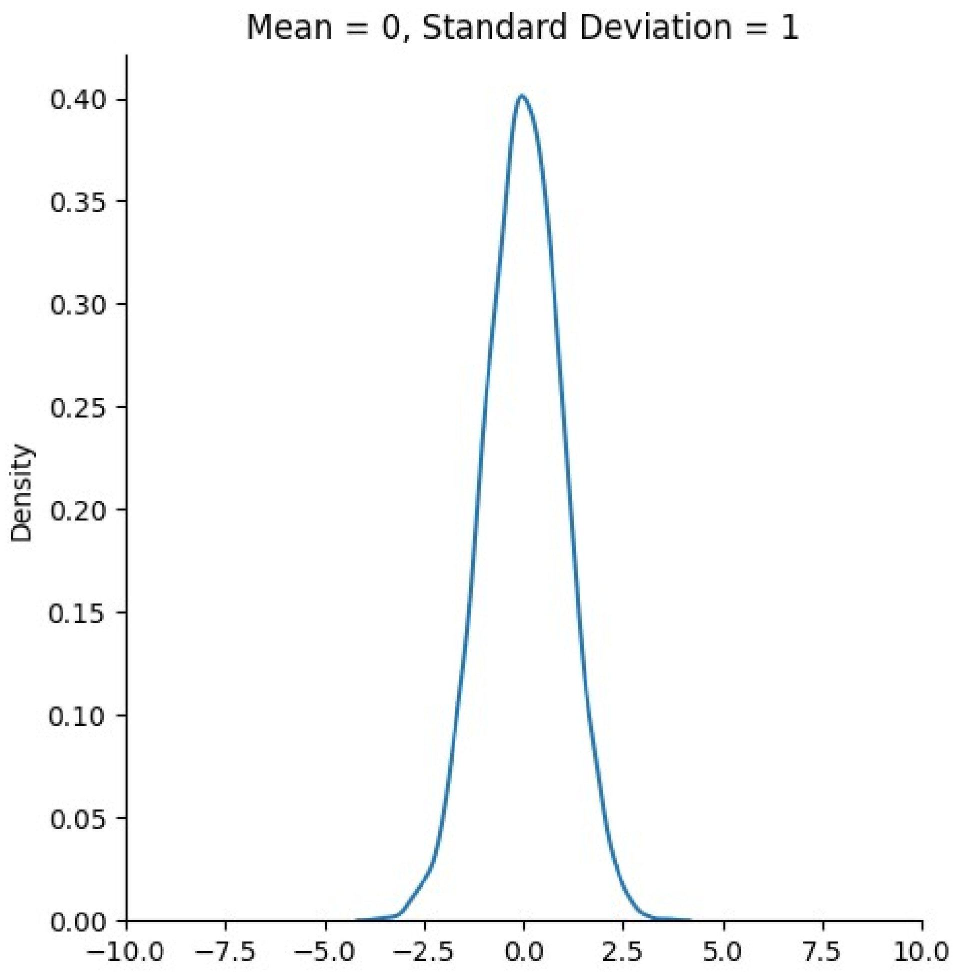
The standard normal distribution is the distribution of the Normal random variable with mean 0 and standard deviation 1 which is denoted by $ X \sim \mathcal{N}(0,1)$.

The following code generates 10,000 samples of standard Gaussian random variable X and plots an approximate PDF of X (based on its 10,000 samples) using the displot() function of Seaborn library.

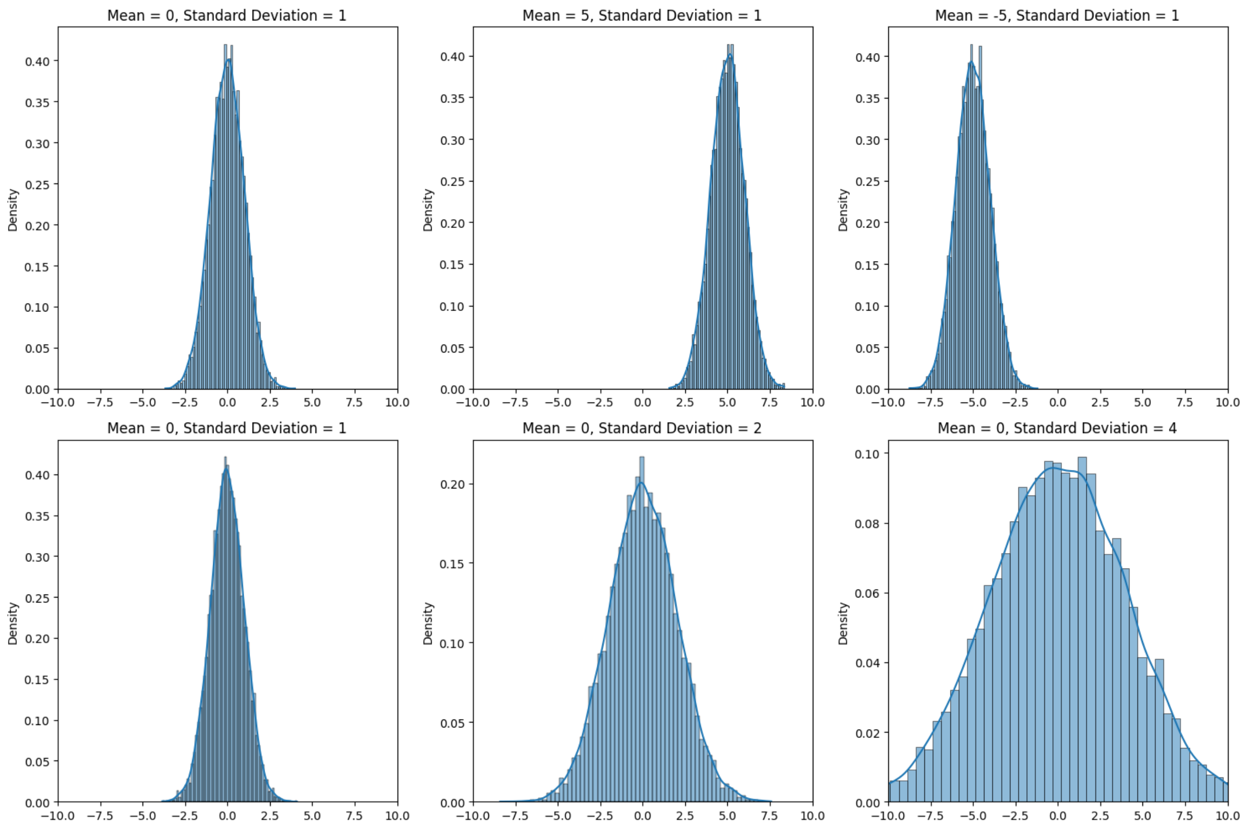
Modify the code to plot the approximate PDF of Gaussian random variable X for the following values of mean: 0, 5, -5 (keeping the standard deviation fixed at 1). Then, plot the approximate PDF of Gaussian random variable X for the following values of standard deviation: 1, 2, 4 (keeping the mean fixed at 0). Comment on the changes in the PDF shape for different values of mean and standard devivation.

```python
sample_size = 10000

x_mean0_sd1_sample = np.random.normal(0, 1, sample_size)


sns.displot(data=x_mean0_sd1_sample,kind="kde").set(title='Mean = 0, 
Standard Deviation = 1')
plt.xlim(-10, 10)
```

```
(-10.0, 10.0)
```



```
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
sample_size = 10000

# Generating samples for different means
x_means = [0, 5, -5]
x_mean_samples = [np.random.normal(mean, 1, sample_size) for mean in
x_means]

# Generating samples for different standard deviations
x_stdevs = [1, 2, 4]
x_stdev_samples = [np.random.normal(0, stdev, sample_size) for stdev
in x_stdevs]

fig, axes = plt.subplots(2, 3, figsize=(15, 10))

# Ploting PDFs for different means
for i, ax in enumerate(axes[0]):
    sns.histplot(data=x_mean_samples[i], kde=True, stat="density",
ax=ax)
    ax.set_title(f'Mean = {x_means[i]}, Standard Deviation = 1')
    ax.set_xlim(-10, 10)

# Ploting PDFs for different standard deviations
for i, ax in enumerate(axes[1]):
    sns.histplot(data=x_stdev_samples[i], kde=True, stat="density",
ax=ax)
    ax.set_title(f'Mean = 0, Standard Deviation = {x_stdevs[i]}')
    ax.set_xlim(-10, 10)

plt.tight_layout()
plt.show()
```



Comments

1.**Modifying the Standard Deviation while Maintaining the Mean Constant:**

- The data spreads wider when the standard deviation rises and the mean remains constant. This can be seen in the PDF's shape, which gets flatter and wider. A wider distribution is produced when the standard deviation increases, indicating that the data points are more dispersed from the mean.
- In contrast, a smaller and taller distribution results from a closer clustering of the data points around the mean when the standard deviation falls.

2.**Modifying the Mean while Maintaining a Constant Standard Deviation:**

- The entire distribution moves to the right when the mean rises and the standard deviation stays constant. This indicates that more data points have higher values than when the mean was lower, increasing the data's central tendency.
- On the other hand, when the mean falls, the distribution moves to the left, signifying a decline in the data's central tendency. Compared to times when the mean was higher, more data points have lower values.

In conclusion, altering the mean shifts the entire distribution along the x-axis without altering its spread, whereas altering the standard deviation modifies the data's distribution around the mean.

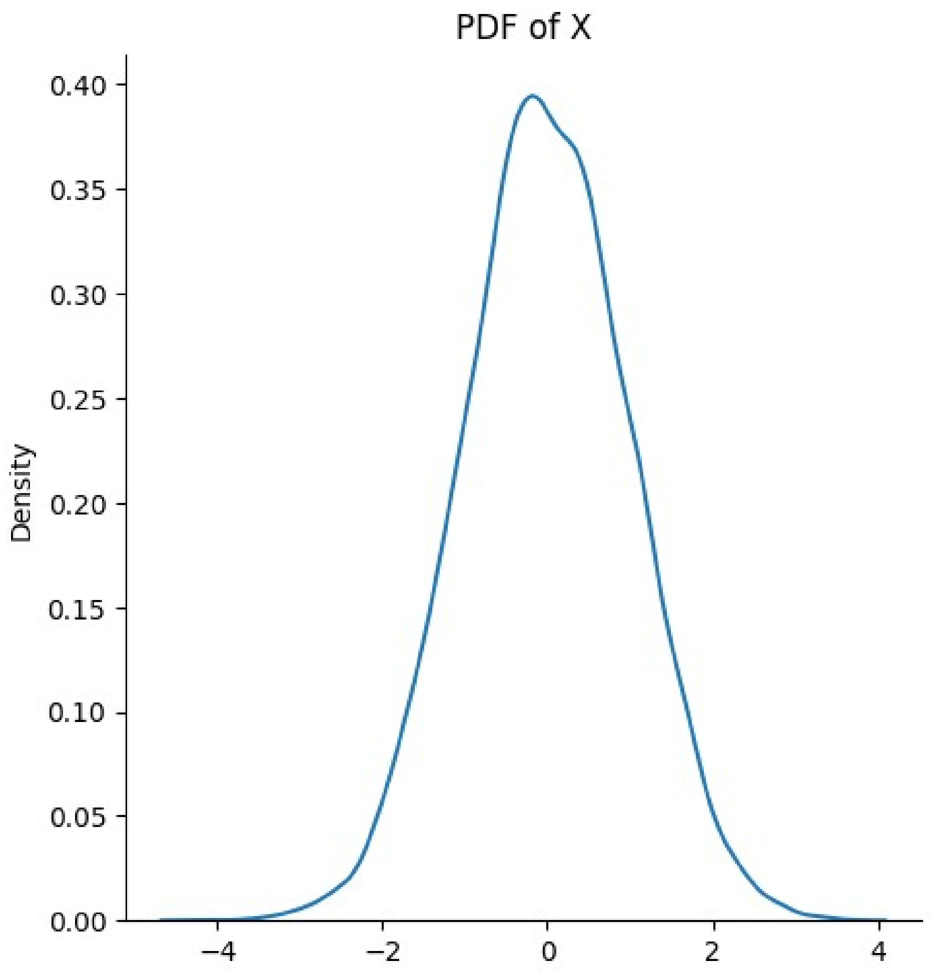
[20 Points] Task 02 - Functions of Gaussian Random Variable

In this task, you are required to comment on the PDF shape of different functions of a Gaussian random variable.

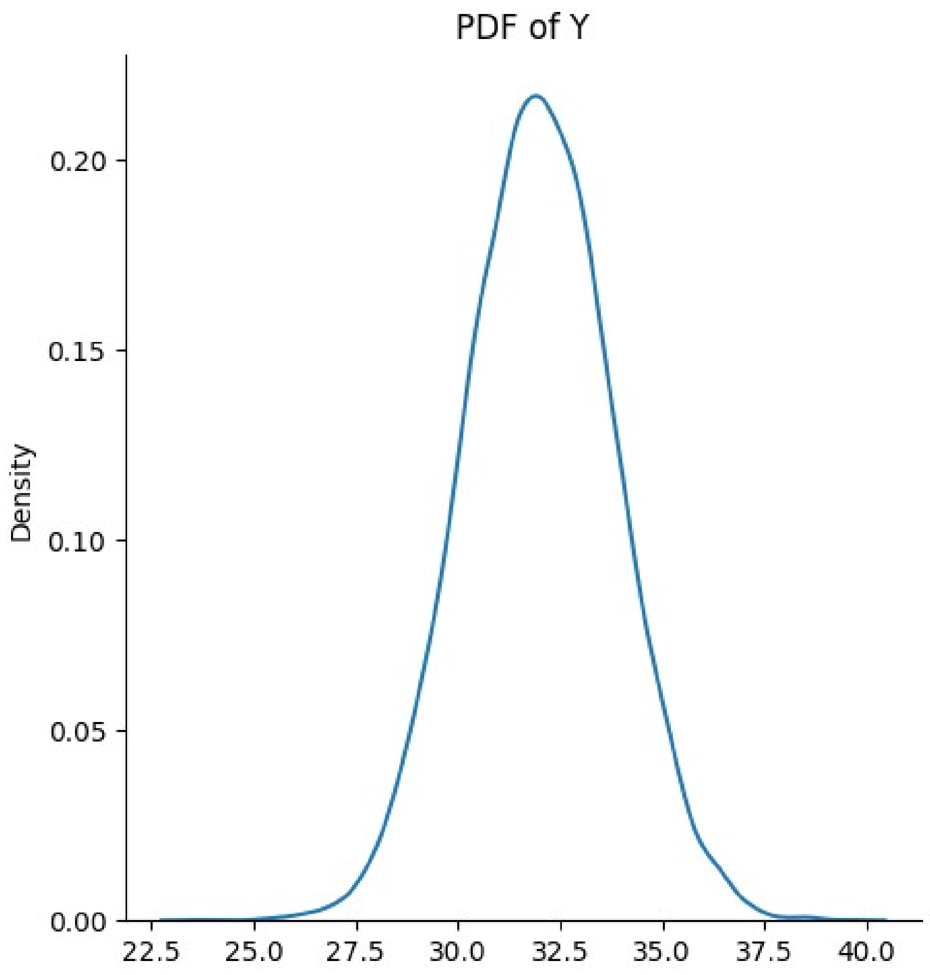
The following code generates 10,000 samples of standard Gaussian random variable X and plots an approximate PDF of X (based on its 10,000 samples) using the displot() function of Seaborn library.

Modify the code to plot the approximate PDF of $Y=1.8X+32$ and $Z=X^2+X+32$. Comment on the PDF shapes of Y and Z, compared to X.

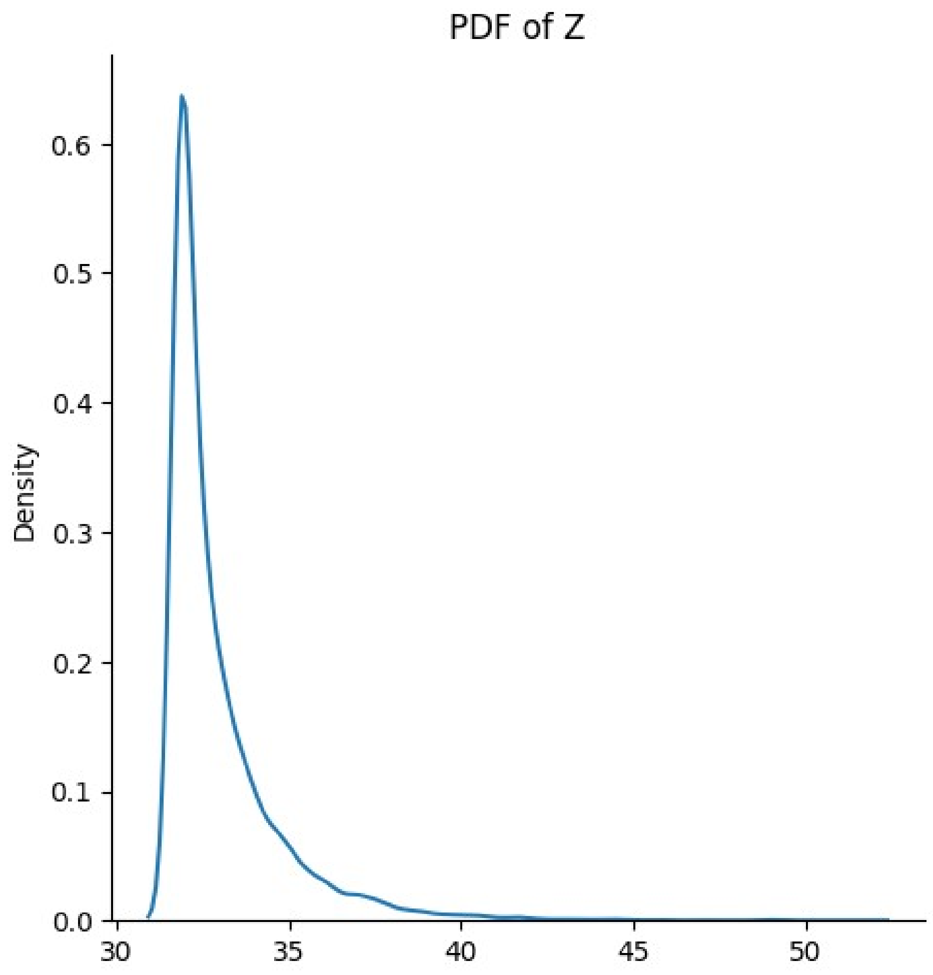
```
sample_size = 10000

x_sample = np.random.normal(0, 1, sample_size)

sns.displot(data=x_sample,kind="kde").set(title='PDF of X')

<seaborn.axisgrid.FacetGrid at 0x79110d4b0850>
```



```python
# Mohit Rai mr06638, Moiz Zulfiqar mz08229

sample_size = 10000

x_sample = np.random.normal(0, 1, sample_size)
y_sample = 1.8 * x_sample + 32

sns.displot(data=y_sample, kind="kde").set(title='PDF of Y')
plt.show()
```



```
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
sample_size = 10000

x_sample = np.random.normal(0, 1, sample_size)
z_sample = x_sample ** 2 + x_sample + 32

sns.displot(data=z_sample, kind="kde").set(title='PDF of Z')
plt.show()
```



Comments

1. ****For Y = 1.8X + 32:****

- **PDF Shape:** With 1.8 scaling and 32 shifting, Y is a linear transformation of X, so even though its parameters are different, its PDF shape will still resemble a Gaussian distribution. In particular, the distribution will be stretched by a factor of 1.8 along the x-axis, and the mean of Y will be moved to the right by 32 units relative to the mean of X.

- **Effect of Linear Transformation:** The distribution's spread and central tendency are impacted by the linear transformation. The positive shift will cause the mean of Y to be significantly higher than the mean of X, and the scaling factor will affect the distribution's spread. The distribution will be stretched and have a wider spread if 1.8 is more than 1. The distribution will be compressed and have a narrower spread if 1.8 is less than 1.

1. ****For Z = X^2 + X + 32:****

- **PDF Shape:** Z, in contrast to Y, is a non-linear transformation of X because of the quadratic component. Consequently, Z's PDF shape won't be Gaussian. Rather, it is likely to show non-linear behaviour, which might involve asymmetry, multiple peaks, or valleys. Although there will be a noticeable deviation from a Gaussian shape, the distribution will still likely have a central tendency around the mean.

- **Effect of Non-linear Transformation:** There are deviations from linearity in the distribution due to the curvature introduced by the quadratic term. Complex interactions within the distribution can arise from the quadratic term's ability to either amplify or dampen the influence of the linear term. Depending on the transformation's parameters, the distribution may display traits like skewness or heavy tails.

To summarise, Z is subjected to a non-linear transformation that essentially modifies the distribution, potentially resulting in more intricate and non-Gaussian shapes, whereas Y is subjected to a linear transformation that maintains the Gaussian shape with modified parameters.

[24 points] Task 03 - Sum of Independent and Identically-Distributed Random Variables

Suppose $X_1, X_2, \cdots, X_n$ are independent random variables with the same underlying distribution. In this case, we say that the X_i are independent and identically distributed or i.i.d. In particular, the X_i, all have the same mean μ and standard deviation σ.

Let S_n be the sum of n i.i.d random variables:

$$S_n = X_1 + X_2 + \cdots + X_n$$

[6 points] Part A: Uniform Random Variable

In this part, we consider X_is to be continuous uniform random variables. The following code generates 10,000 samples of X_1 random variable and plots an approximate PDF of X_1 (based on its 10,000 samples) using the displot() function of Seaborn library.

Modify the given code to also generate 10,000 samples of $X_2, X_3, \cdots, X_n$, and plot the approximate PDF of S_n using the displot() function of Seaborn library for the following values of n: 1,2,3,5,10,50,100.

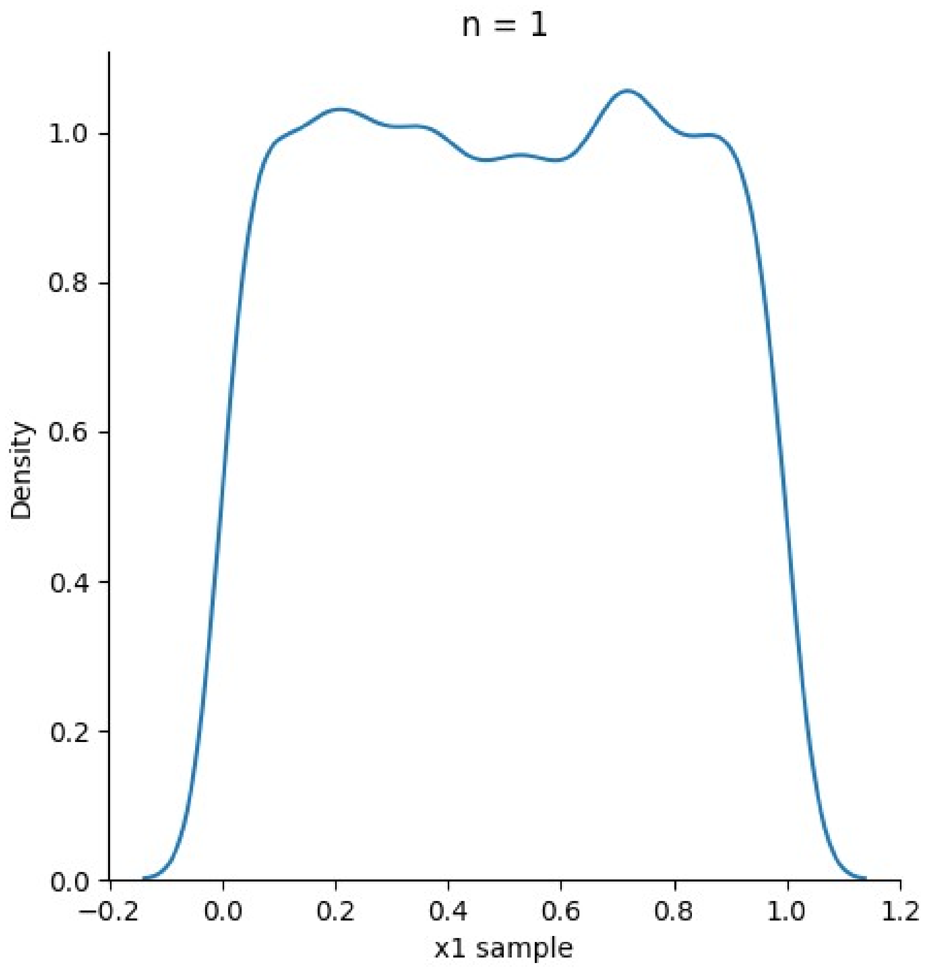
For example, for n=2, you have to plot the approximate PDF of $S_2 = X_1 + X_2$ and for n=3, you have to plot the approximate PDF of $S_3 = X_1 + X_2 + X_3$

Comment on the evolution of PDF shape of S_n as n increases.

```
sample_size = 10000

df = pd.DataFrame()
x1_sample = np.random.uniform(0, 1, sample_size)
```

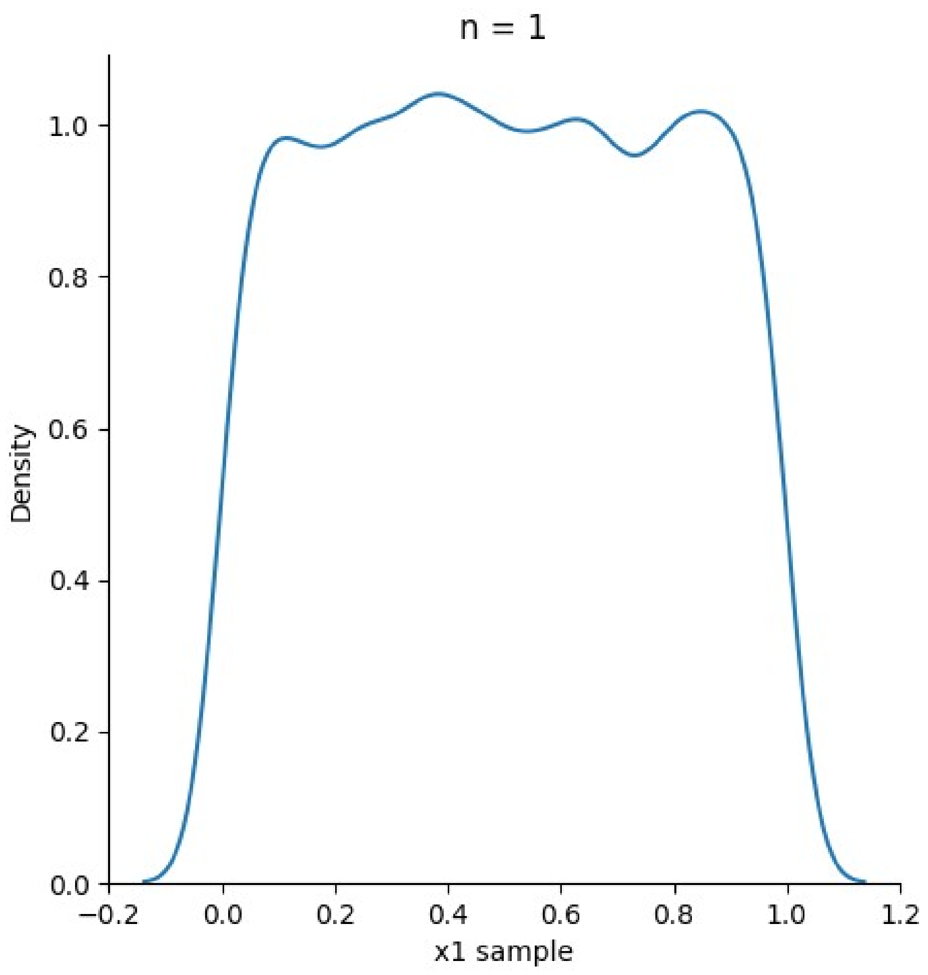
```
i = 1
col = f'x{i} sample'
df[col] = x1_sample

sns.displot(data=df[col], kind="kde").set(title='n = 1')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4e911f490>
```



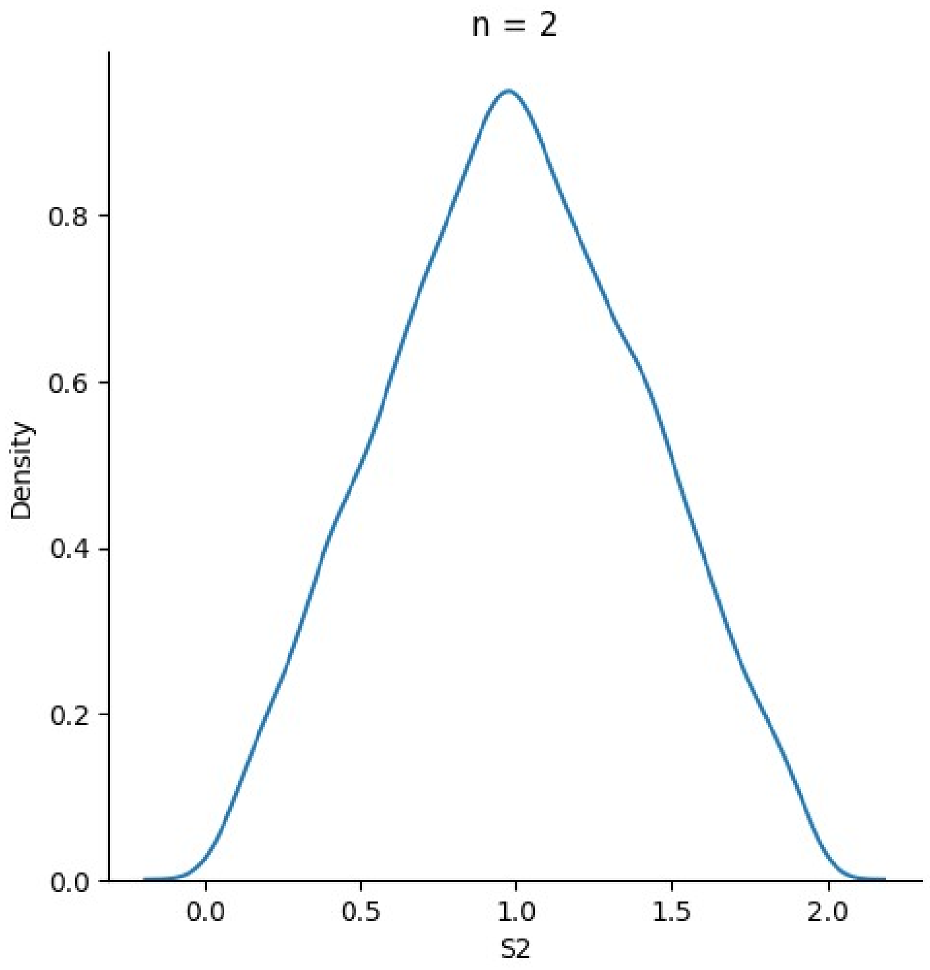
```
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n=1
sample_size = 10000

# Create all samples
samples = {f'x{i} sample': np.random.uniform(0, 1, sample_size) for i
in range(1, 2)}

# Create DataFrame
df = pd.DataFrame(samples)
```

```
# Plot the approximate PDF of S1
sns.displot(data=df['x1 sample'], kind="kde").set(title='n = 1')

<seaborn.axisgrid.FacetGrid at 0x7fa4e6043520>
```



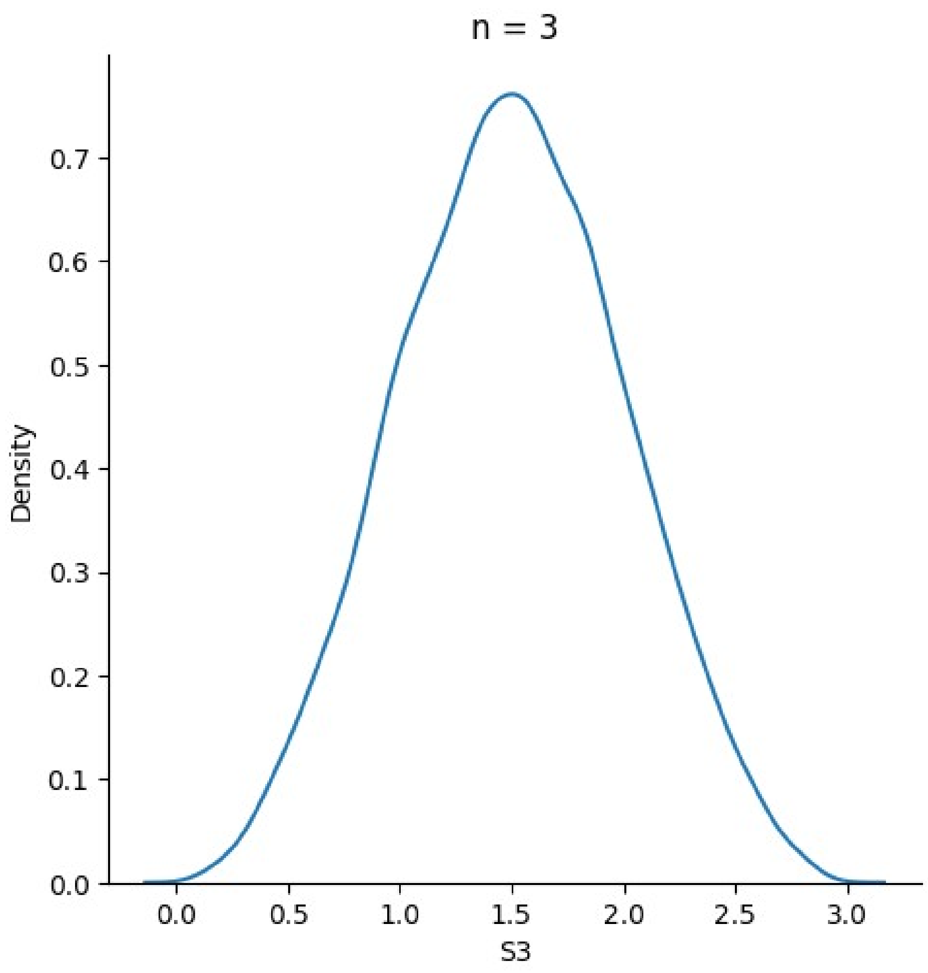
```
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n=2
# Create all samples
samples = {f'x{i} sample': np.random.uniform(0, 1, sample_size) for i
in range(1, 3)}

# Create DataFrame
df = pd.DataFrame(samples)

# Calculate S2
df['S2'] = df.sum(axis=1)

# Plot the approximate PDF of S2
sns.displot(data=df['S2'], kind="kde").set(title='n = 2')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4ed20e650>
```



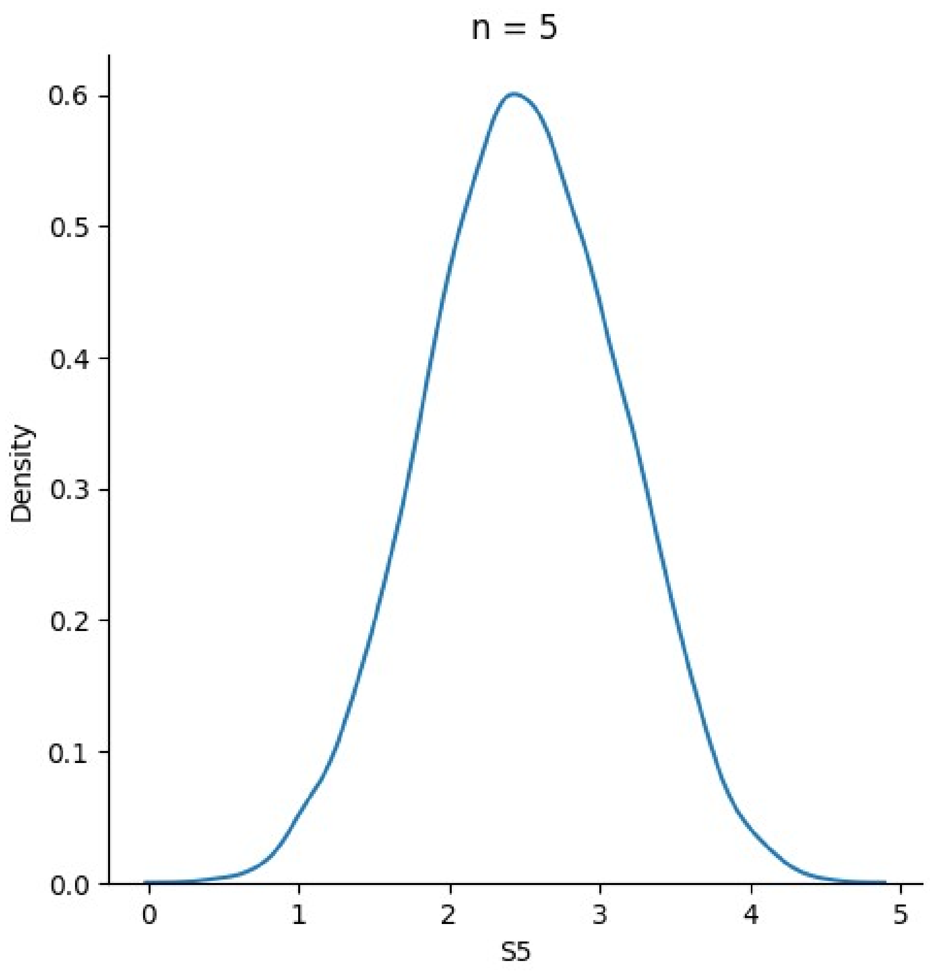
```
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n=3
# Create all samples
samples = {f'x{i} sample': np.random.uniform(0, 1, sample_size) for i
in range(1, 4)}

# Create DataFrame
df = pd.DataFrame(samples)

# Calculate S3
df['S3'] = df.sum(axis=1)

# Plot the approximate PDF of S3
sns.displot(data=df['S3'], kind="kde").set(title='n = 3')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4e6284580>
```



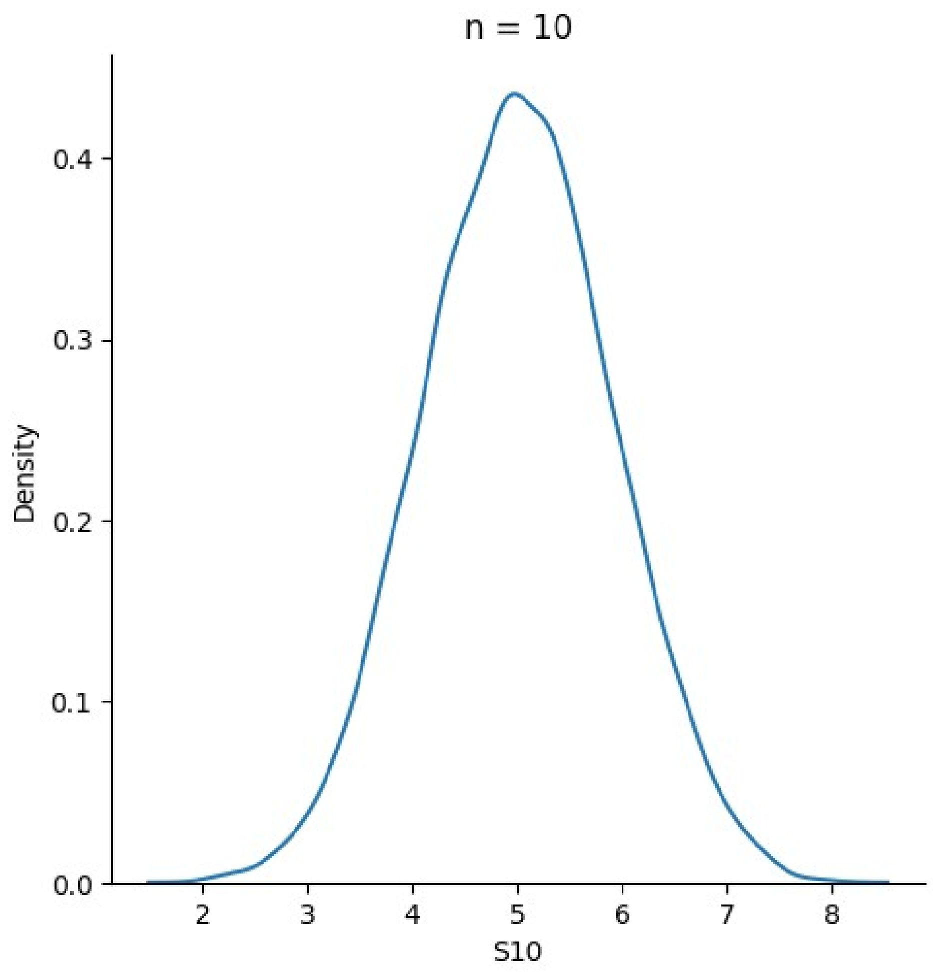
```
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n=5
# Create all samples
samples = {f'x{i} sample': np.random.uniform(0, 1, sample_size) for i
in range(1, 6)}

# Create DataFrame
df = pd.DataFrame(samples)

# Calculate S5
df['S5'] = df.sum(axis=1)

# Plot the approximate PDF of S5
sns.displot(data=df['S5'], kind="kde").set(title='n = 5')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4e90e2980>
```



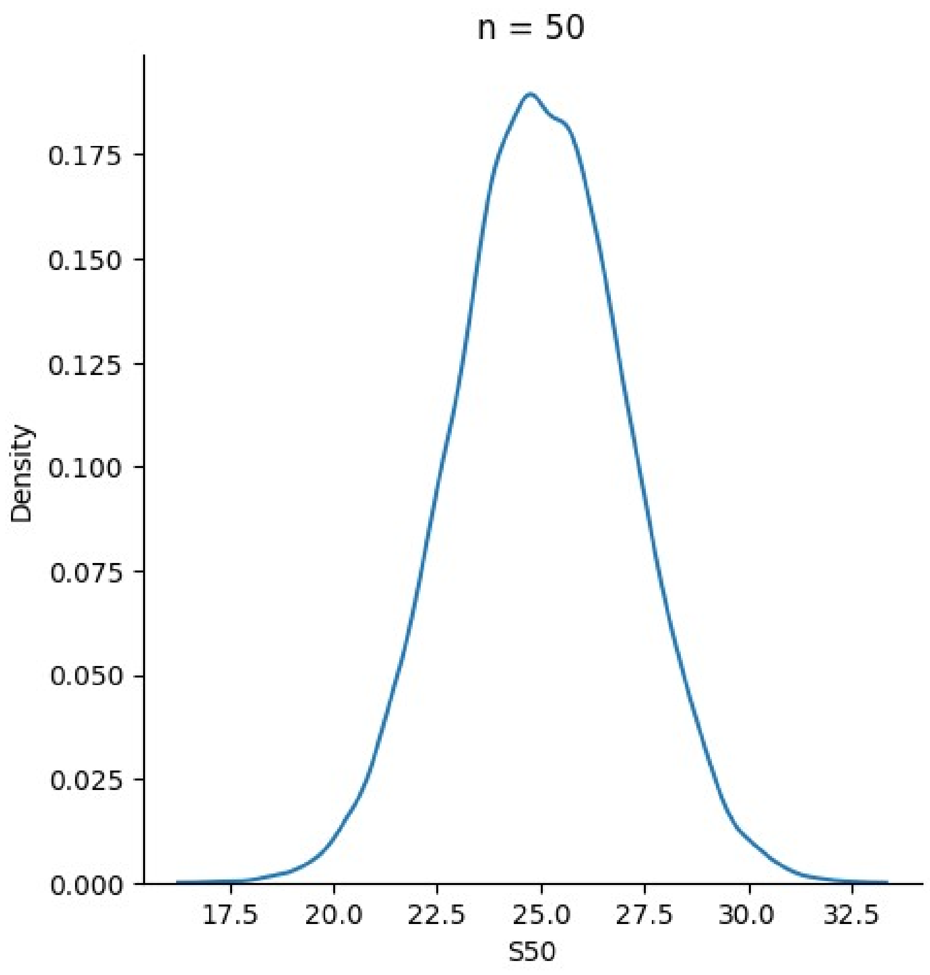
```
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n=10
# Create all samples
samples = {f'x{i} sample': np.random.uniform(0, 1, sample_size) for i
in range(1, 11)}

# Create DataFrame
df = pd.DataFrame(samples)

# Calculate S10
df['S10'] = df.sum(axis=1)

# Plot the approximate PDF of S10
sns.displot(data=df['S10'], kind="kde").set(title='n = 10')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4e8b85750>
```



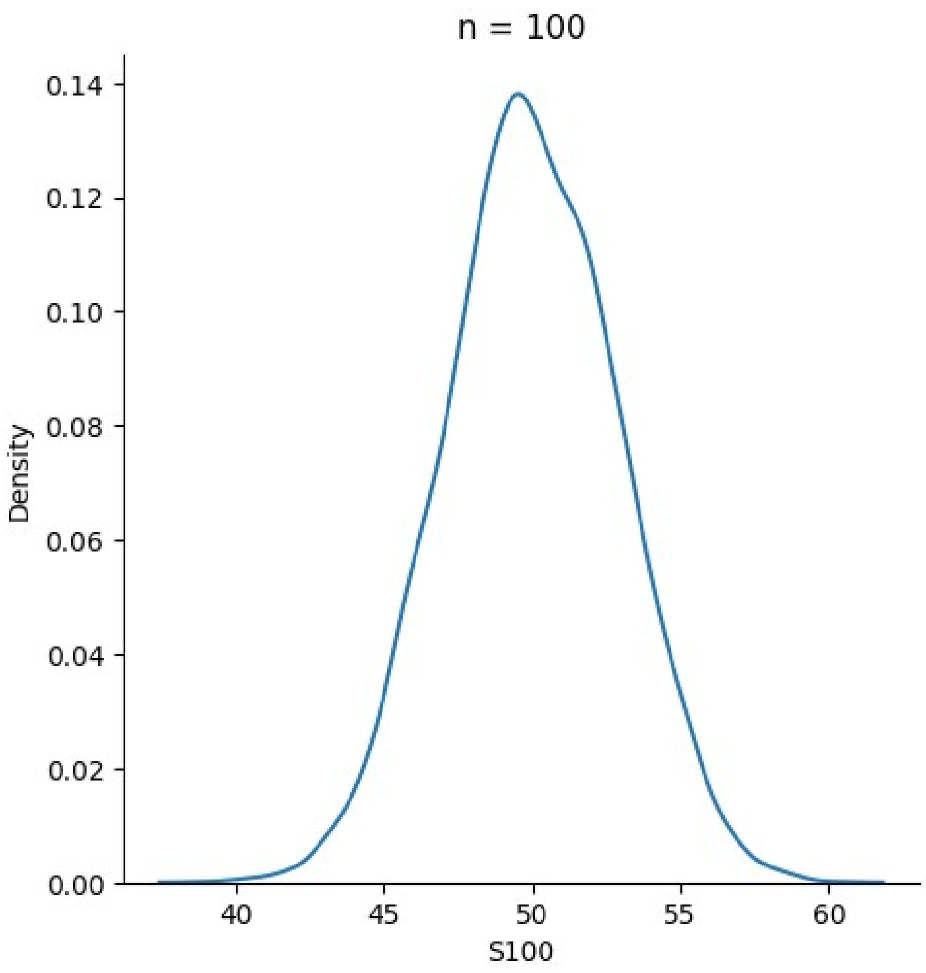
```
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n=50
# Create all samples
samples = {f'x{i} sample': np.random.uniform(0, 1, sample_size) for i
in range(1, 51)}

# Create DataFrame
df = pd.DataFrame(samples)

# Calculate S50
df['S50'] = df.sum(axis=1)

# Plot the approximate PDF of S50
sns.displot(data=df['S50'], kind="kde").set(title='n = 50')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4e85ab910>
```



```
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n=100
# Create all samples
samples = {f'x{i} sample': np.random.uniform(0, 1, sample_size) for i
in range(1, 101)}

# Create DataFrame
df = pd.DataFrame(samples)

# Calculate S100
df['S100'] = df.sum(axis=1)

# Plot the approximate PDF of S100
sns.displot(data=df['S100'], kind="kde").set(title='n = 100')
# The caution that I was observing had to do with how pandas manages
DataFrame memory. A DataFrame may get fragmented if
# I add a lot of columns to it one at a time, as in a loop.
# This indicates that the data is dispersed and not kept continuously
in memory.
# Performance problems may arise as a result.
# I made all of the samples before creating the DataFrame in order to
get around this warning and enhance efficiency.
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4e8ada7d0>
```



Comments Regarding the evolution of Sn's PDF form as n grows, I can see that Sn's distribution resembles a normal distribution more and more as n grows. This is because of the Central Limit Theorem, which asserts that, regardless of the initial distribution of the variables, the sum of a large number of independent, identically distributed random variables will roughly follow a normal distribution. Since each Xi in this instance is an independent uniform random variable with an identical distribution, as n rises, their aggregate Sn tends to resemble a normal distribution. This explains why, for bigger n (e.g., n=50 or n=100), the PDF's shape becomes more symmetrical from being flat for small n (e.g., n=1).

[6 points] Part B: Exponential Random Variable

In this part, we consider X_is to be exponential random variables. The following code generates 10,000 samples of X_1 random variable and plots an approximate PDF of X_1 (based on its 10,000 samples) using the displot() function of Seaborn library.

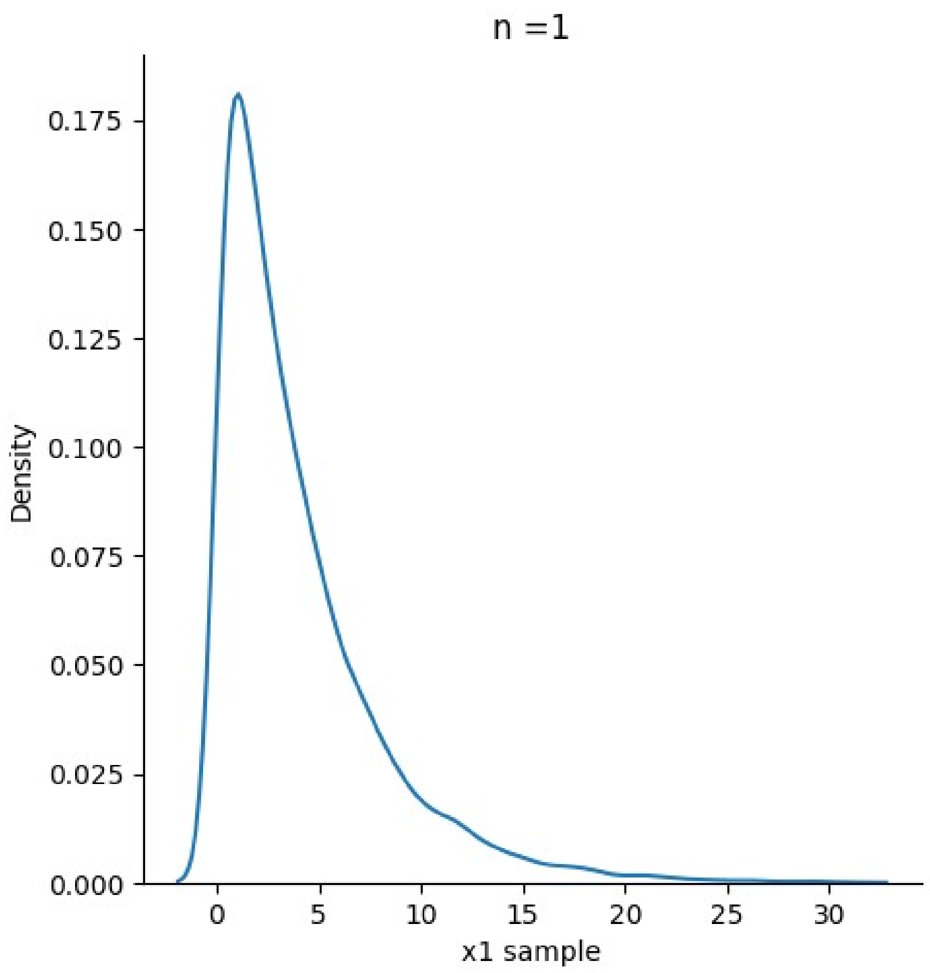
Modify the given code to generate 10,000 samples of $X_2, X_3, \cdots, X_n$, and plot the approximate PDF of S_n using the displot() function of Seaborn library for the following values of n: 1,2,3,5,10,50,100.

For example, for n=2, you have to plot the approximate PDF of $S_2 = X_1 + X_2$ and for n=3, you have to plot the approximate PDF of $S_3 = X_1 + X_2 + X_3$

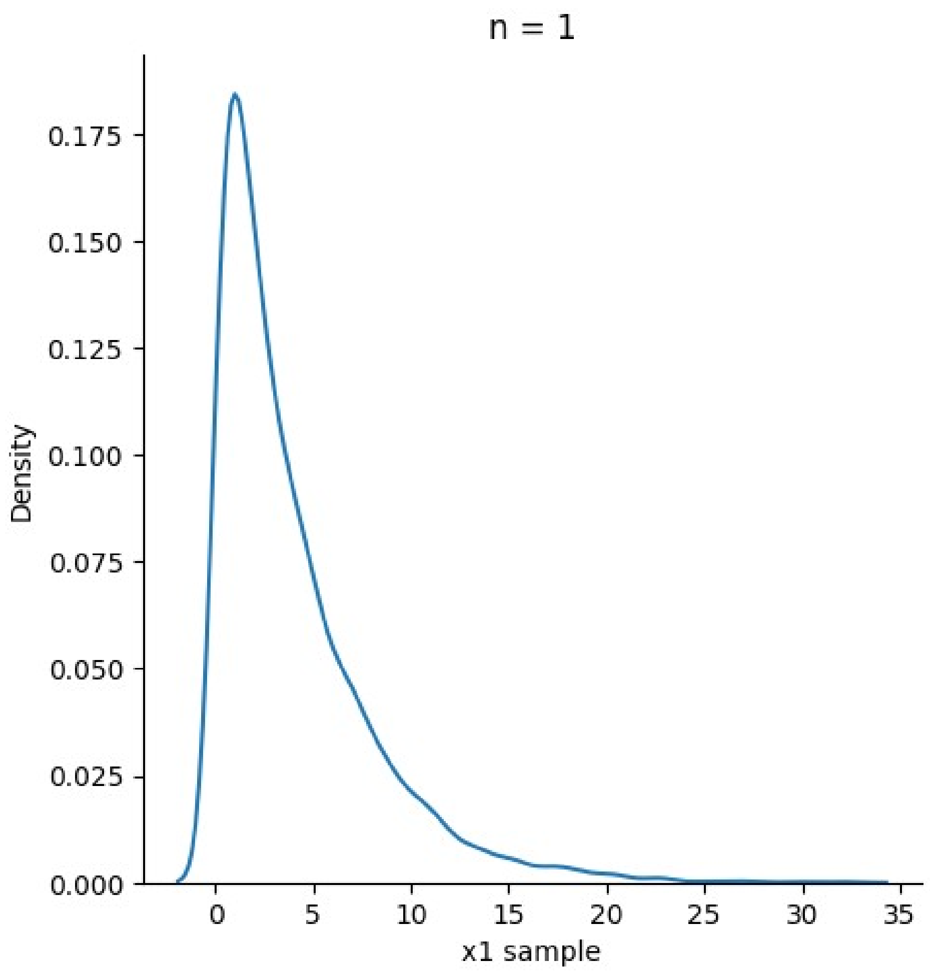
Comment on the evolution of PDF shape of S_n as n increases.

```
sample_size = 10000

rate = 0.25 #Rate of exponential random variables

df = pd.DataFrame()
x1_sample = np.random.exponential((1/rate),sample_size)

i = 1
col = f'x{i} sample'
df[col] = x1_sample

sns.displot(data=df[col], kind="kde").set(title='n =1')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4e607d720>
```

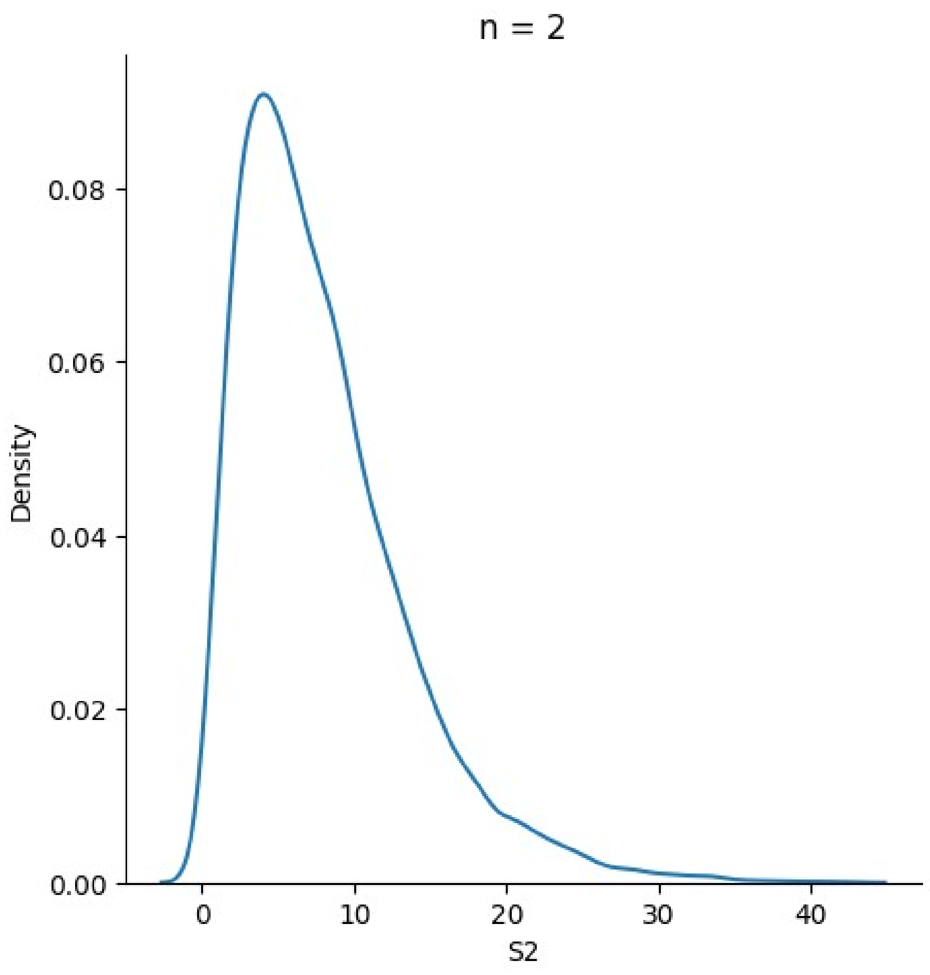


```
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n=1
sample_size = 10000
rate = 0.25 #Rate of exponential random variables

# For n=1
df = pd.DataFrame()
x1_sample = np.random.exponential((1/rate),sample_size)
df['x1 sample'] = x1_sample
sns.displot(data=df['x1 sample'], kind="kde").set(title='n = 1')

<seaborn.axisgrid.FacetGrid at 0x7fa4e6462b60>
```



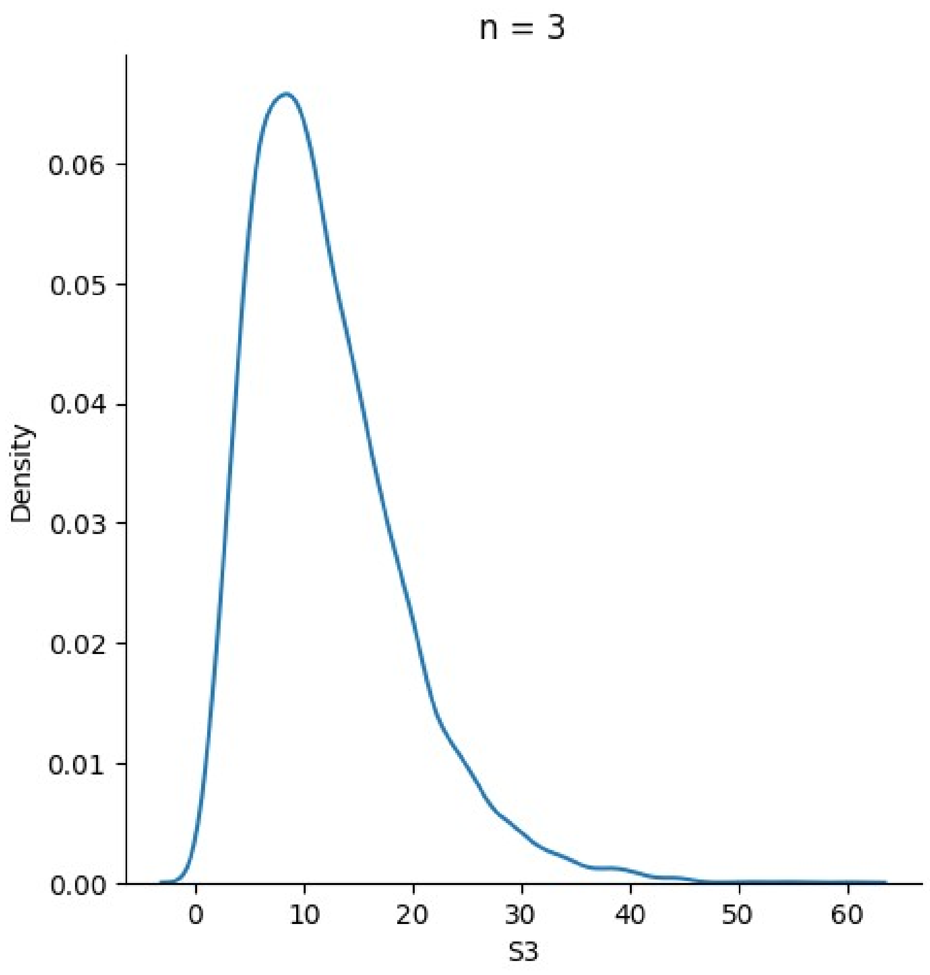
```
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n=2
df = pd.DataFrame()
for i in range(1, 3):
    x_sample = np.random.exponential((1/rate),sample_size)
    col = f'x{i} sample'
    df[col] = x_sample

# Calculate S2
df['S2'] = df.sum(axis=1)

# Plot the approximate PDF of S2
sns.displot(data=df['S2'], kind="kde").set(title='n = 2')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4ed34d5a0>
```



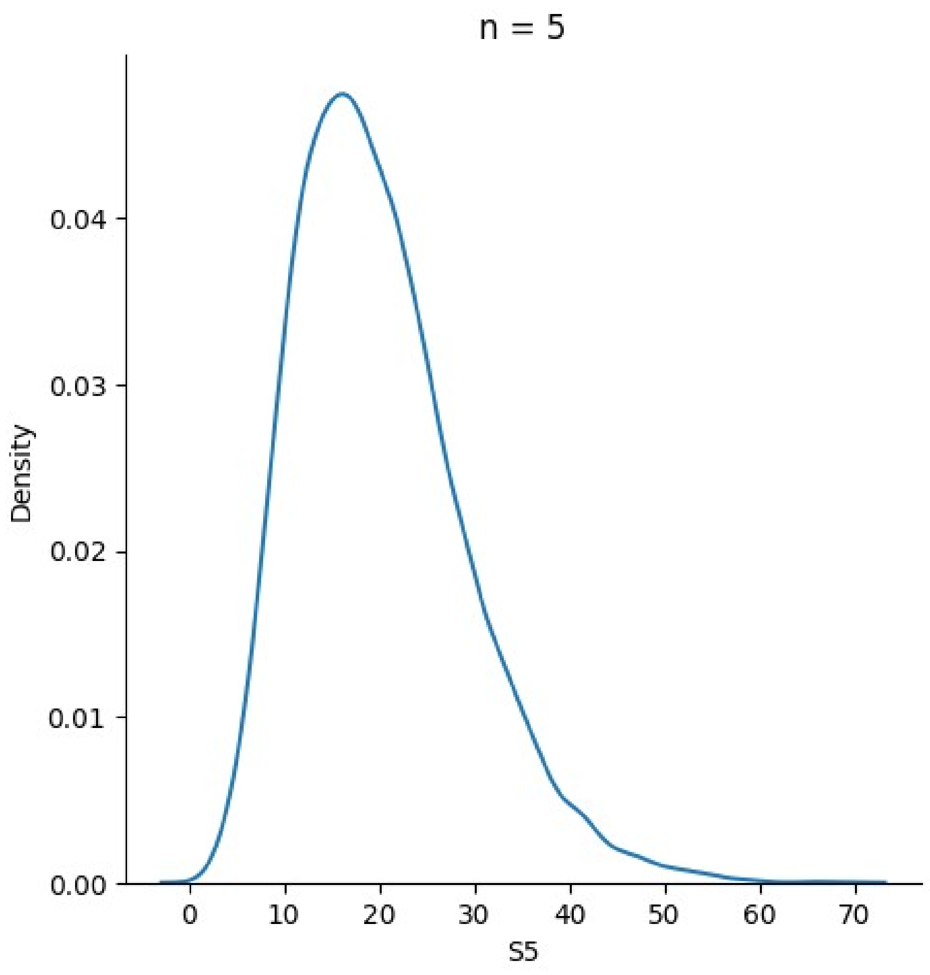
```
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n=3
df = pd.DataFrame()
for i in range(1, 4):
    x_sample = np.random.exponential((1/rate),sample_size)
    col = f'x{i} sample'
    df[col] = x_sample

# Calculate S3
df['S3'] = df.sum(axis=1)

# Plot the approximate PDF of S3
sns.displot(data=df['S3'], kind="kde").set(title='n = 3')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4e8e03ca0>
```



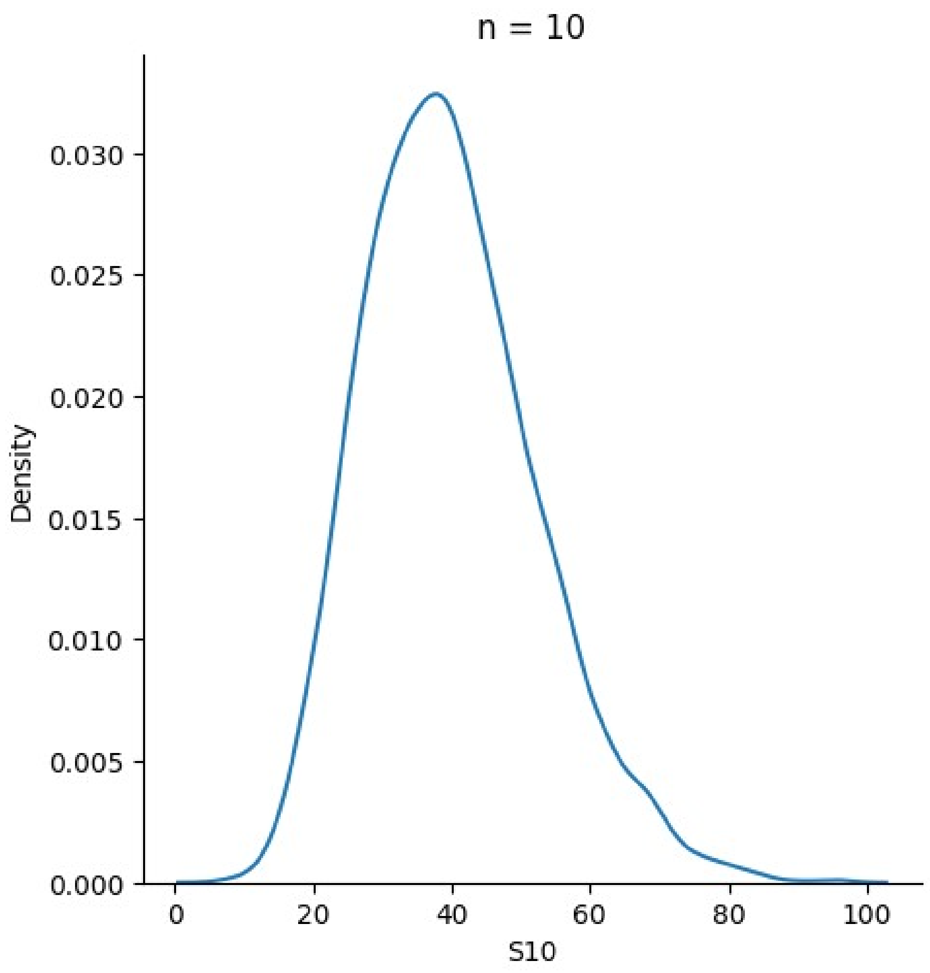
```
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n=5
df = pd.DataFrame()
for i in range(1, 6):
    x_sample = np.random.exponential((1/rate),sample_size)
    col = f'x{i} sample'
    df[col] = x_sample

# Calculate S5
df['S5'] = df.sum(axis=1)

# Plot the approximate PDF of S5
sns.displot(data=df['S5'], kind="kde").set(title='n = 5')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4e8f3df60>
```



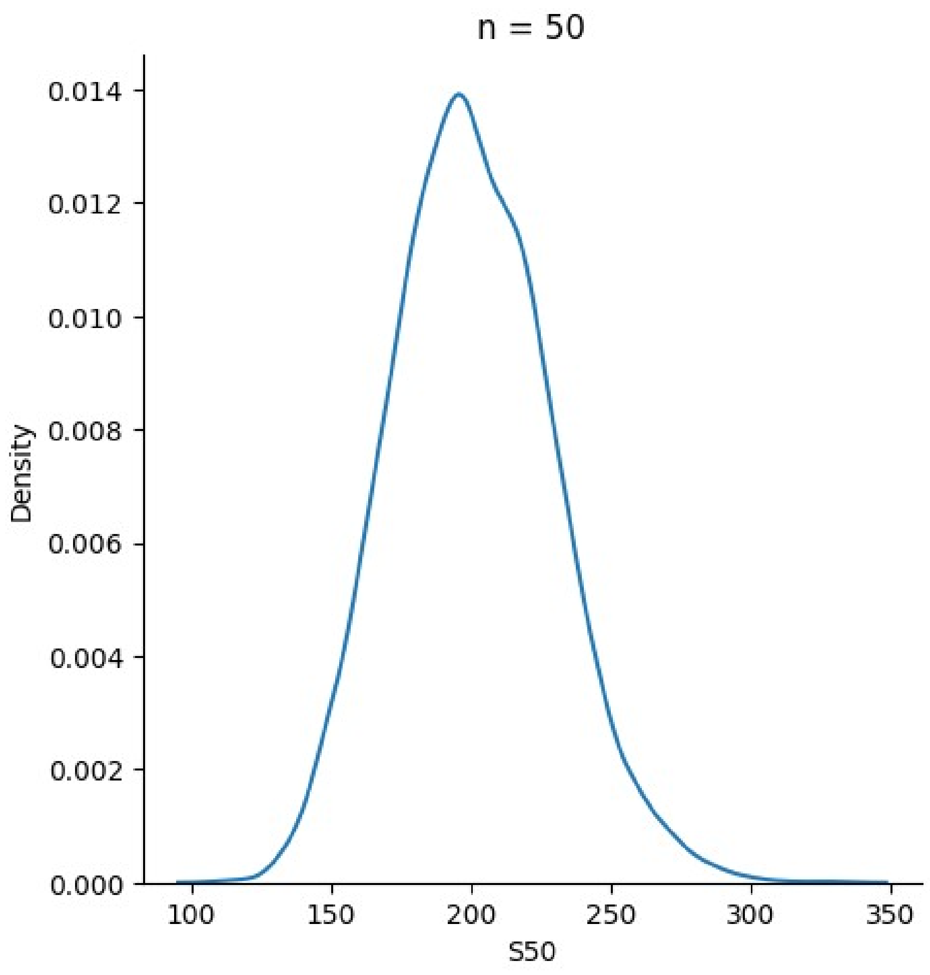
```
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n = 10
df = pd.DataFrame()
for i in range(1, 11):
    x_sample = np.random.exponential((1/rate),sample_size)
    col = f'x{i} sample'
    df[col] = x_sample

# Calculate S10
df['S10'] = df.sum(axis=1)

# Plot the approximate PDF of S10
sns.displot(data=df['S10'], kind="kde").set(title='n = 10')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4e5d4ebc0>
```



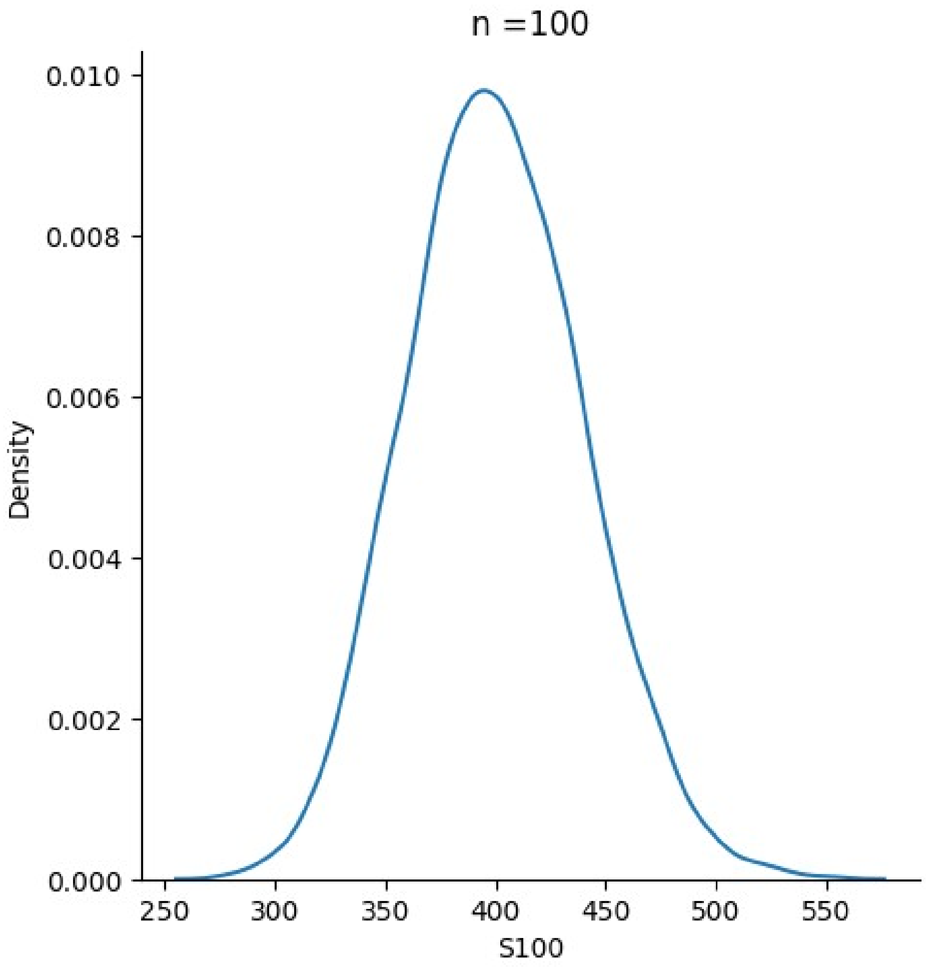
```
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n = 50
df = pd.DataFrame()
for i in range(1, 51):
    x_sample = np.random.exponential((1/rate),sample_size)
    col = f'x{i} sample'
    df[col] = x_sample

# Calculate S50
df['S50'] = df.sum(axis=1)

# Plot the approximate PDF of S50
sns.displot(data=df['S50'], kind="kde").set(title='n = 50')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4e9005ed0>
```



```
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n = 100
sample_size = 10000
rate = 0.25 #Rate of exponential random variables
n = 100

# Create all samples
samples = {f'x{i} sample': np.random.exponential((1/rate),
sample_size) for i in range(1, n+1)}

# Create DataFrame
df = pd.DataFrame(samples)

# Calculate Sn
df['S100'] = df.sum(axis=1)

# Plot the approximate PDF of S100
sns.displot(data=df['S100'], kind="kde").set(title='n = 100')

# The caution that I was observing had to do with how pandas manages
```

```
DataFrame memory. A DataFrame may get fragmented if
# I add a lot of columns to it one at a time, as in a loop.
# This indicates that the data is dispersed and not kept continuously
in memory.
# Performance problems may arise as a result.
# I made all of the samples before creating the DataFrame in order to
get around this warning and enhance efficiency.

<seaborn.axisgrid.FacetGrid at 0x7fa4e8514250>
```



Comments Regarding the evolution of Sn's PDF form as n grows, I can see that Sn's distribution resembles a normal distribution more and more as n grows. This is because of the Central Limit Theorem, which asserts that, regardless of the initial distribution of the variables, the sum of a large number of independent, identically distributed random variables will roughly follow a normal distribution. Since each Xi in this instance is an independent exponential random variable with an identical distribution, as n rises, their aggregate Sn tends to resemble a normal distribution. This is why, for greater n (e.g., n=50 or n=100), the PDF's shape becomes more symmetric and stops being right-skewed for small n (e.g., n=1).

[6 points] Part C: Binomial Random Variable

In this part, we consider X_is to be binomial random variables. The following code generates 10,000 samples of X_1 random variable and plots an approximate PDF of X_1 (based on its 10,000 samples) using the displot() function of Seaborn library.

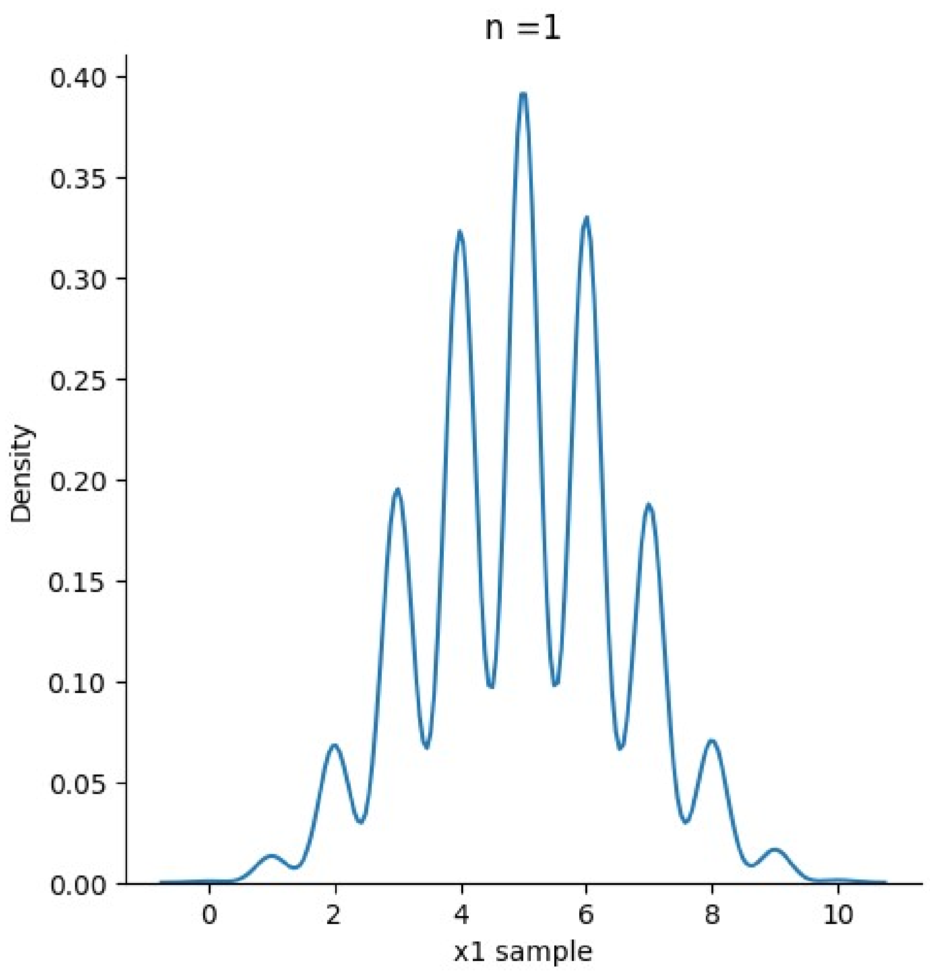
Modify the given code to generate 10,000 samples of $X_2, X_3, \cdots, X_n$, and plot the approximate PDF of S_n using the displot() function of Seaborn library for the following values of n: 1,2,3,5,10,50,100.

For example, for n=2, you have to plot the approximate PDF of $S_2 = X_1 + X_2$ and for n=3, you have to plot the approximate PDF of $S_3 = X_1 + X_2 + X_3$

Comment on the evolution of PDF shape of S_n as n increases.

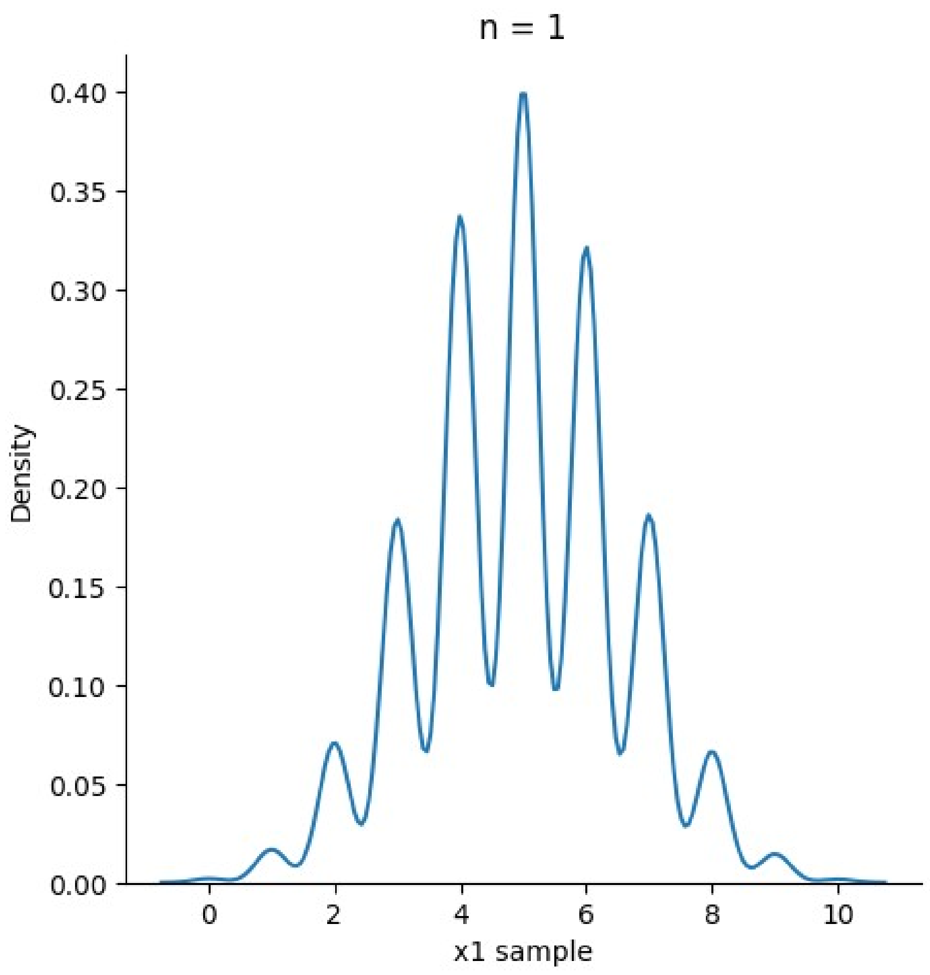
```
sample_size = 10000

df = pd.DataFrame()
x1_sample = np.random.binomial(10,0.5,size=sample_size)

i = 1
col = f'x{i} sample'
df[col] = x1_sample

sns.displot(data=df[col], kind="kde").set(title='n = 1')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4e82a57b0>
```



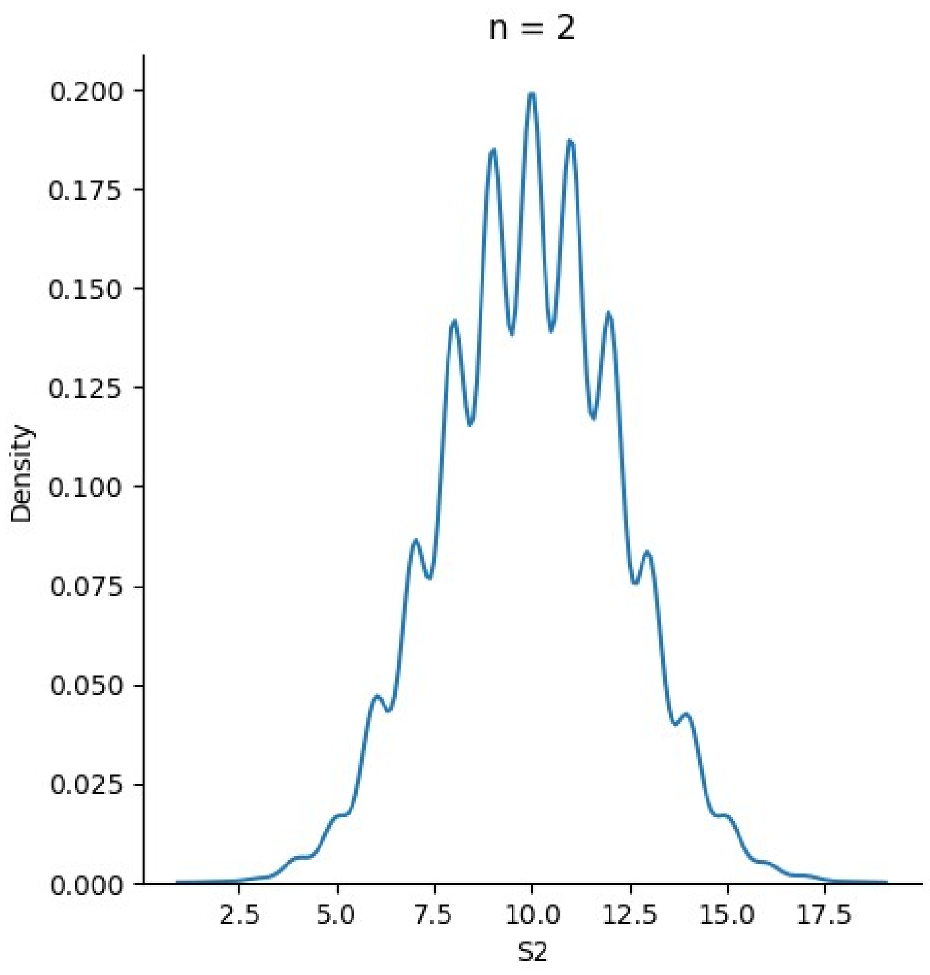
```
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n=1
sample_size = 10000
p = 0.5 #Probability of success

# Create all samples
samples = {f'x{i} sample': np.random.binomial(10, p, sample_size) for
i in range(1, 2)}

# Create DataFrame
df = pd.DataFrame(samples)

# Plot the approximate PDF of S1
sns.displot(data=df['x1 sample'], kind="kde").set(title='n = 1')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4e5cacb20>
```



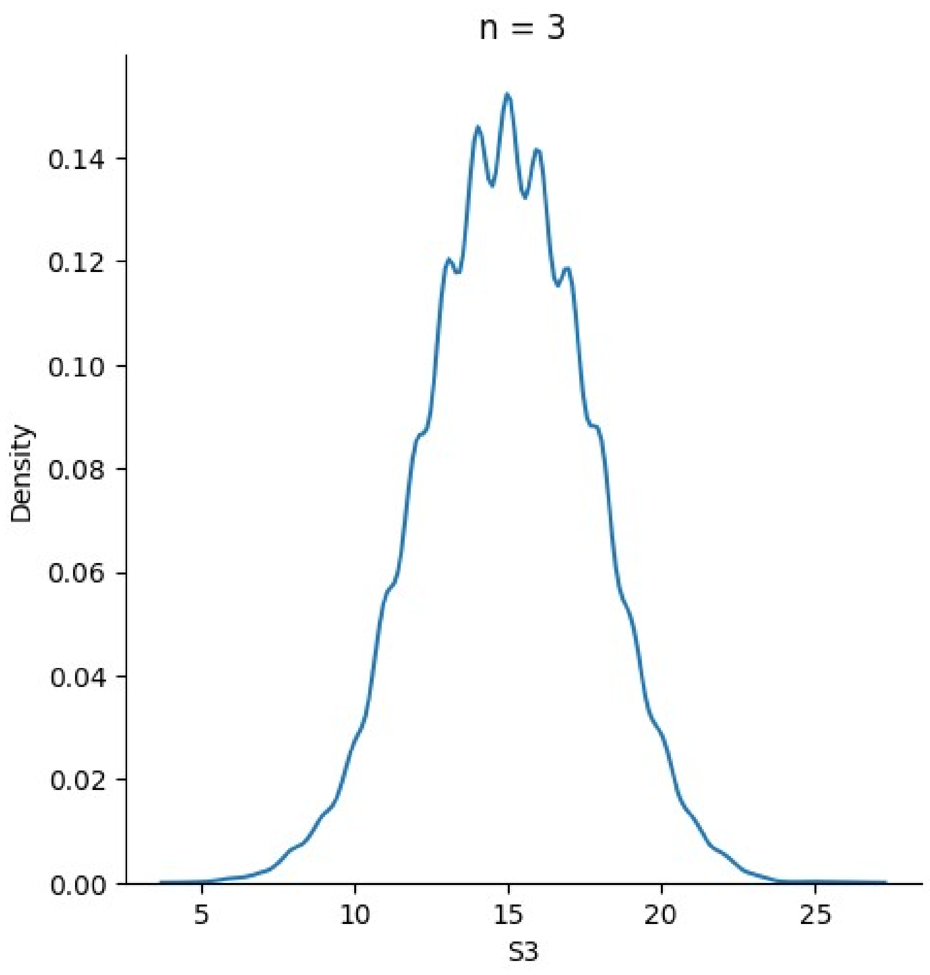
```
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n=2
# Create all samples
samples = {f'x{i} sample': np.random.binomial(10, p, sample_size) for
i in range(1, 3)}

# Create DataFrame
df = pd.DataFrame(samples)

# Calculate S2
df['S2'] = df.sum(axis=1)

# Plot the approximate PDF of S2
sns.displot(data=df['S2'], kind="kde").set(title='n = 2')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4e5caf4f0>
```



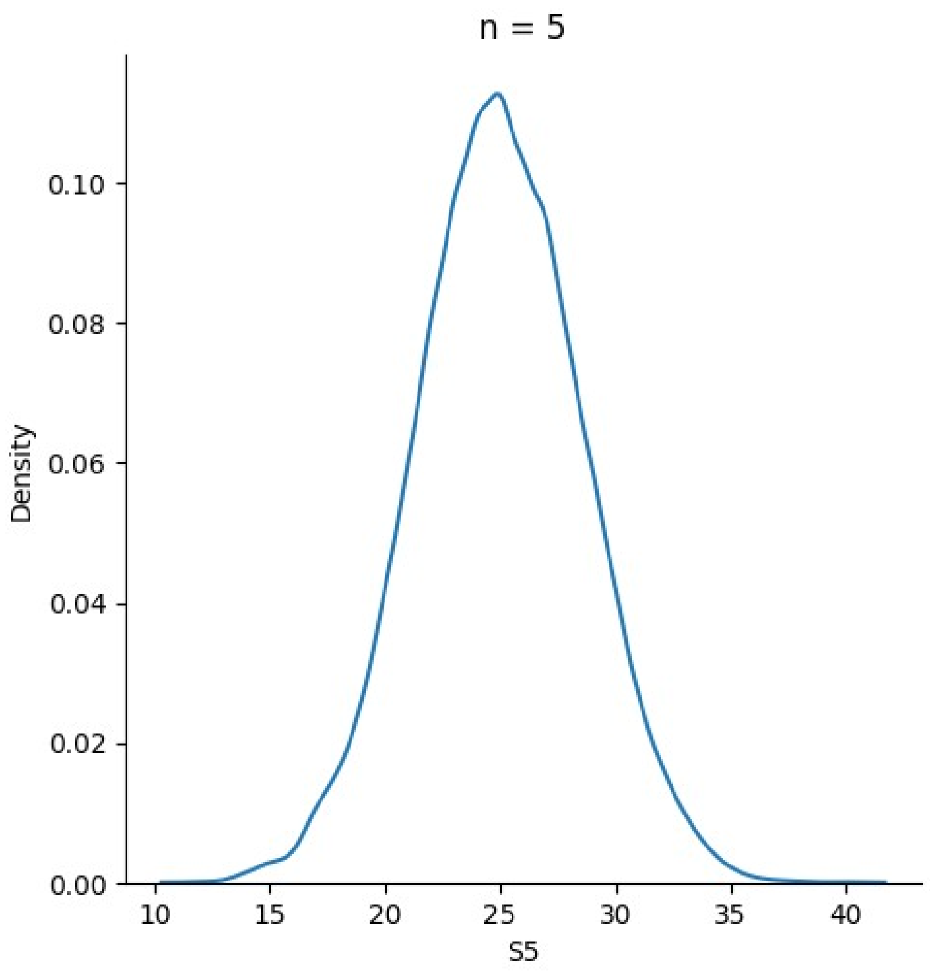
```python
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n=3
# Create all samples
samples = {f'x{i} sample': np.random.binomial(10, p, sample_size) for
i in range(1, 4)}

# Create DataFrame
df = pd.DataFrame(samples)

# Calculate S3
df['S3'] = df.sum(axis=1)

# Plot the approximate PDF of S3
sns.displot(data=df['S3'], kind="kde").set(title='n = 3')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4e8b43340>
```



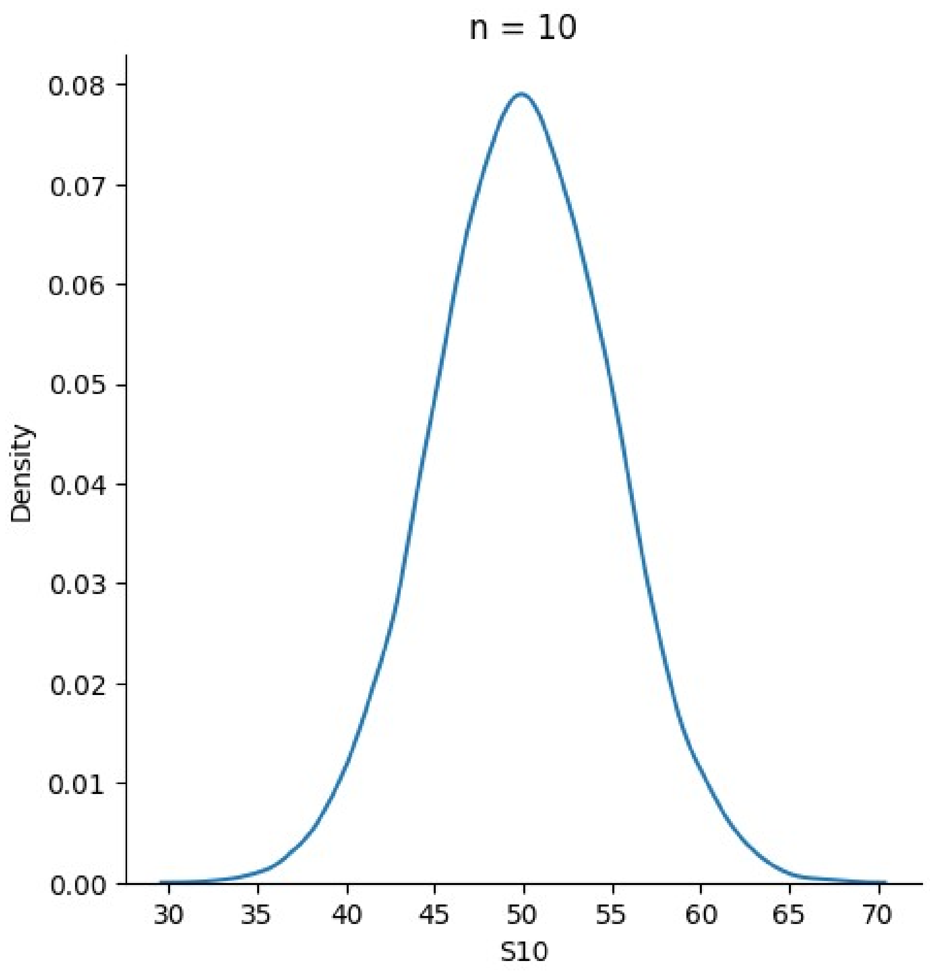
```
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n=5
# Create all samples
samples = {f'x{i} sample': np.random.binomial(10, p, sample_size) for
i in range(1, 6)}

# Create DataFrame
df = pd.DataFrame(samples)

# Calculate S5
df['S5'] = df.sum(axis=1)

# Plot the approximate PDF of S5
sns.displot(data=df['S5'], kind="kde").set(title='n = 5')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4e5a56c50>
```



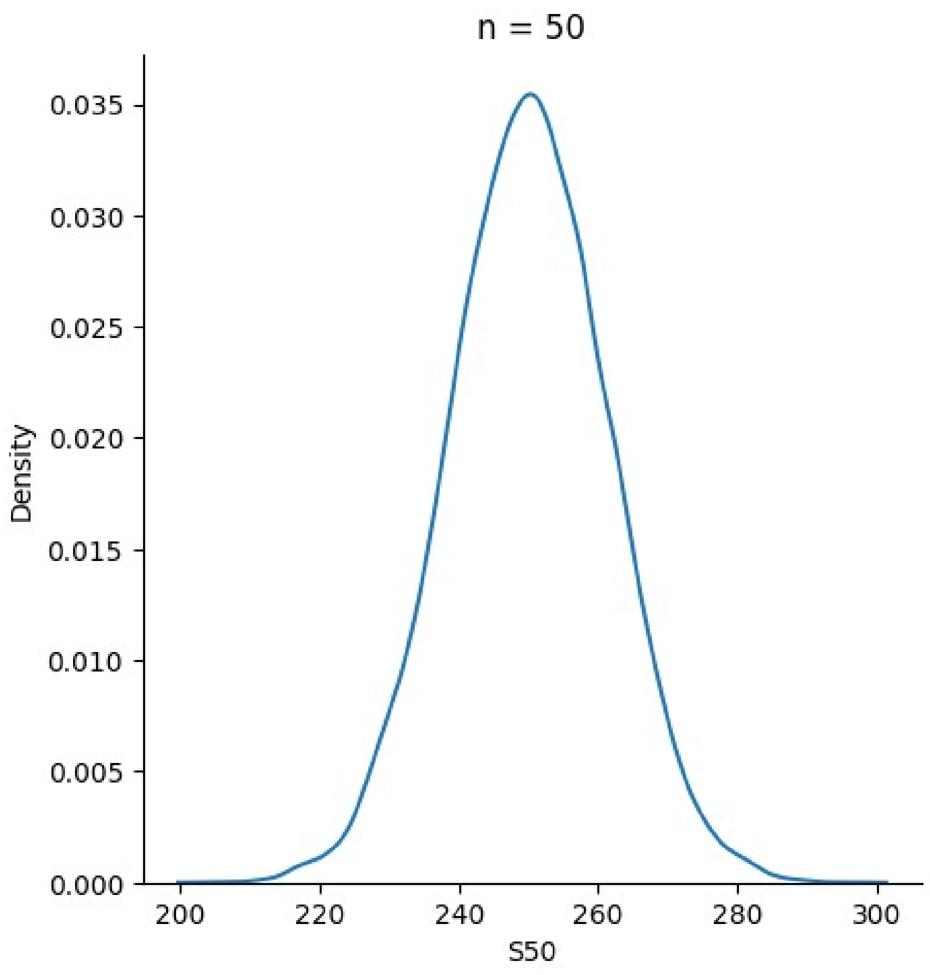
```
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n=10
# Create all samples
samples = {f'x{i} sample': np.random.binomial(10, p, sample_size) for
i in range(1, 11)}

# Create DataFrame
df = pd.DataFrame(samples)

# Calculate S10
df['S10'] = df.sum(axis=1)

# Plot the approximate PDF of S10
sns.displot(data=df['S10'], kind="kde").set(title='n = 10')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4e5ae5a20>
```



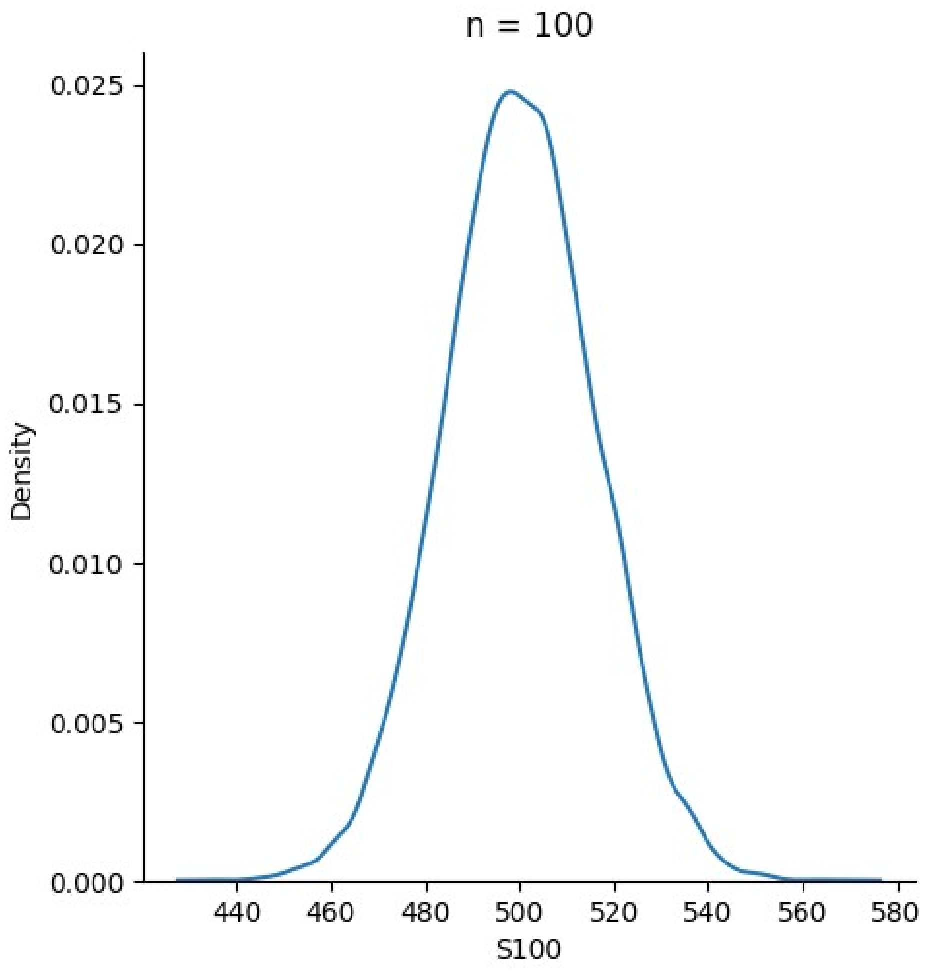
```python
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n=50
# Create all samples
samples = {f'x{i} sample': np.random.binomial(10, p, sample_size) for
i in range(1, 51)}

# Create DataFrame
df = pd.DataFrame(samples)

# Calculate S50
df['S50'] = df.sum(axis=1)

# Plot the approximate PDF of S50
sns.displot(data=df['S50'], kind="kde").set(title='n = 50')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4e596e3b0>
```



```
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n=100
# Create all samples
samples = {f'x{i} sample': np.random.binomial(10, p, sample_size) for
i in range(1, 101)}

# Create DataFrame
df = pd.DataFrame(samples)

# Calculate S100
df['S100'] = df.sum(axis=1)

# Plot the approximate PDF of S100
sns.displot(data=df['S100'], kind="kde").set(title='n = 100')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4e59f14e0>
```



Comments

I am creating n binomial random variables in my scenario, each with the same number of trials (10), and the same success chance (0.5). I then calculate Sn by adding these variables. I am adding more and more of these binomial random variables as n grows.Because of the Central Limit Theorem (CLT), the Probability Density Function (PDF) of Sn will change in shape as n grows. According to the CLT, regardless of the initial distribution's shape, the total of many independent and identically distributed (i.i.d.) random variables tends towards a normal distribution when appropriately normalised.Therefore, the PDF of Sn for small n (n=1,2,3) will resemble the sum of several binomial distributions, which might not be bell-shaped or symmetric. However, the PDF of Sn will begin to resemble a normal distribution more and more as n increases (e.g., n=5,10,50,100). This is because the CLT begins to operate because I am summing a greater number of i.i.d. binomial random variables.

[6 points] Part D: Poisson Random Variable

In this part, we consider X_is to be poisson random variables. The following code generates 10,000 samples of X_1 random variable and plots an approximate PDF of X_1 (based on its 10,000 samples) using the displot() function of Seaborn library.

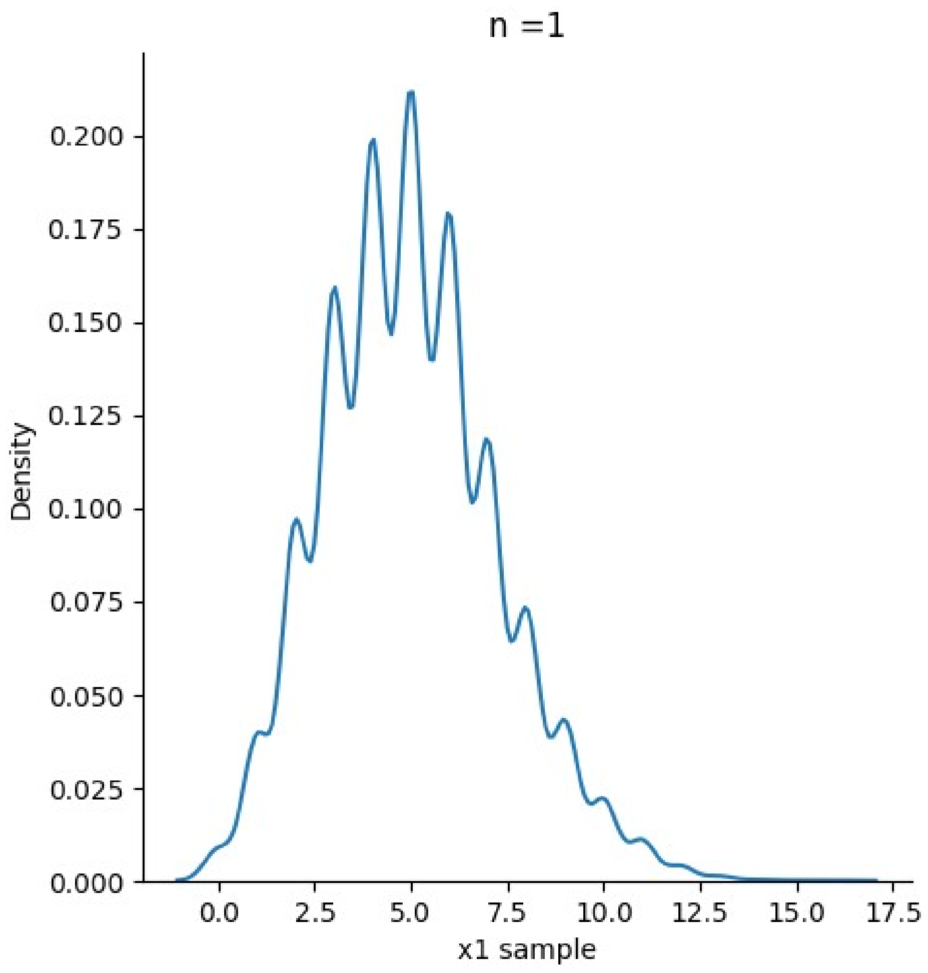
Modify the given code to generate 10,000 samples of $X_2, X_3, \cdots, X_n$, and plot the approximate PDF of S_n using the displot() function of Seaborn library for the following values of n: 1,2,3,5,10,50,100.

For example, for n=2, you have to plot the approximate PDF of $S_2 = X_1 + X_2$ and for n=3, you have to plot the approximate PDF of $S_3 = X_1 + X_2 + X_3$

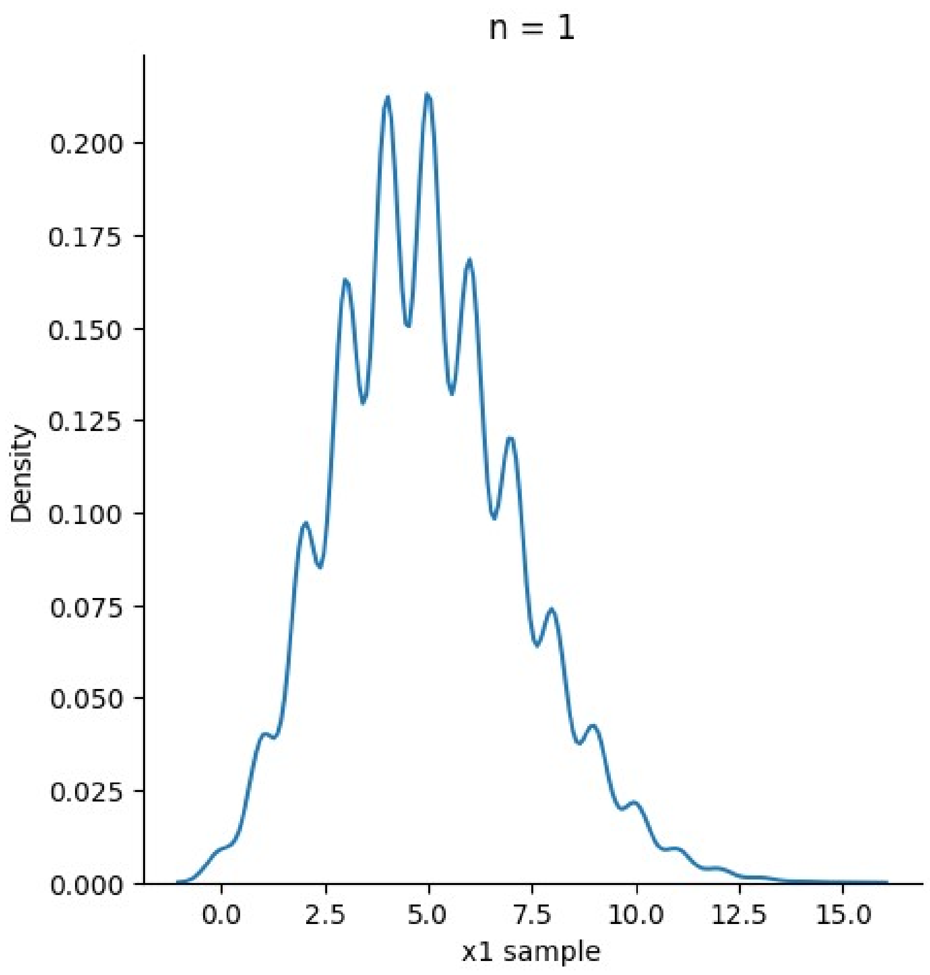
Comment on the evolution of PDF shape of S_n as n increases.

```
sample_size = 10000

df = pd.DataFrame()
x1_sample = np.random.poisson(5,size=sample_size)

i = 1
col = f'x{i} sample'
df[col] = x1_sample

sns.displot(data=df[col], kind="kde").set(title='n = 1')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4e59367a0>
```

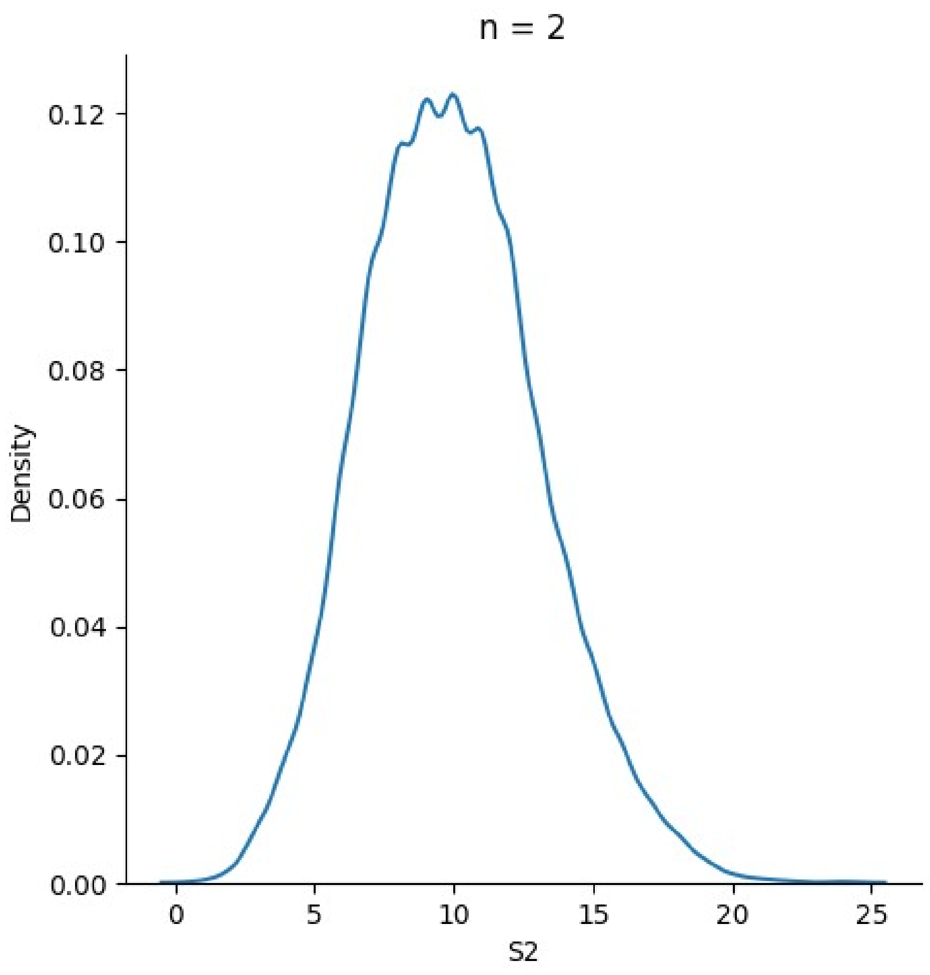


```
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n=1
sample_size = 10000
lam = 5 # lambda parameter of Poisson distribution

# Create all samples
samples = {f'x{i} sample': np.random.poisson(lam, sample_size) for i
in range(1, 2)}

# Create DataFrame
df = pd.DataFrame(samples)

# Plot the approximate PDF of S1
sns.displot(data=df['x1 sample'], kind="kde").set(title='n = 1')

<seaborn.axisgrid.FacetGrid at 0x7fa4e5914fa0>
```



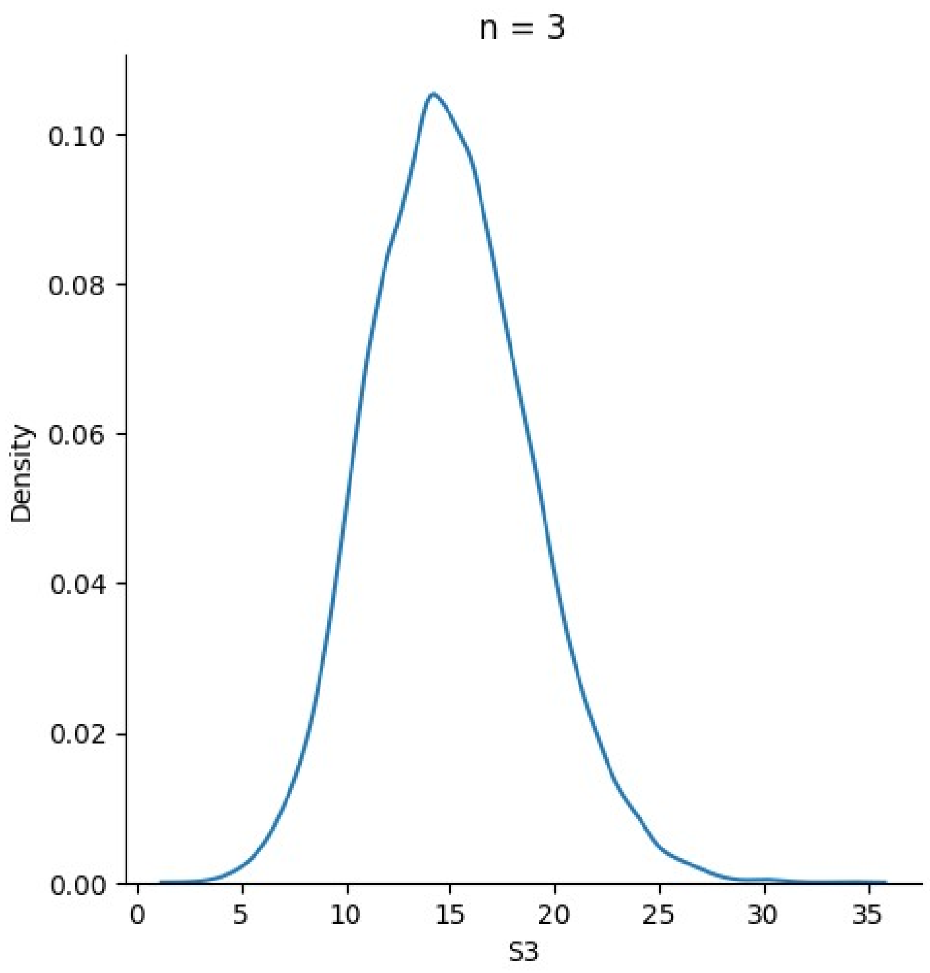
```
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n=2
# Create all samples
samples = {f'x{i} sample': np.random.poisson(lam, sample_size) for i
in range(1, 3)}

# Create DataFrame
df = pd.DataFrame(samples)

# Calculate S2
df['S2'] = df.sum(axis=1)

# Plot the approximate PDF of S2
sns.displot(data=df['S2'], kind="kde").set(title='n = 2')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4e58f6d70>
```



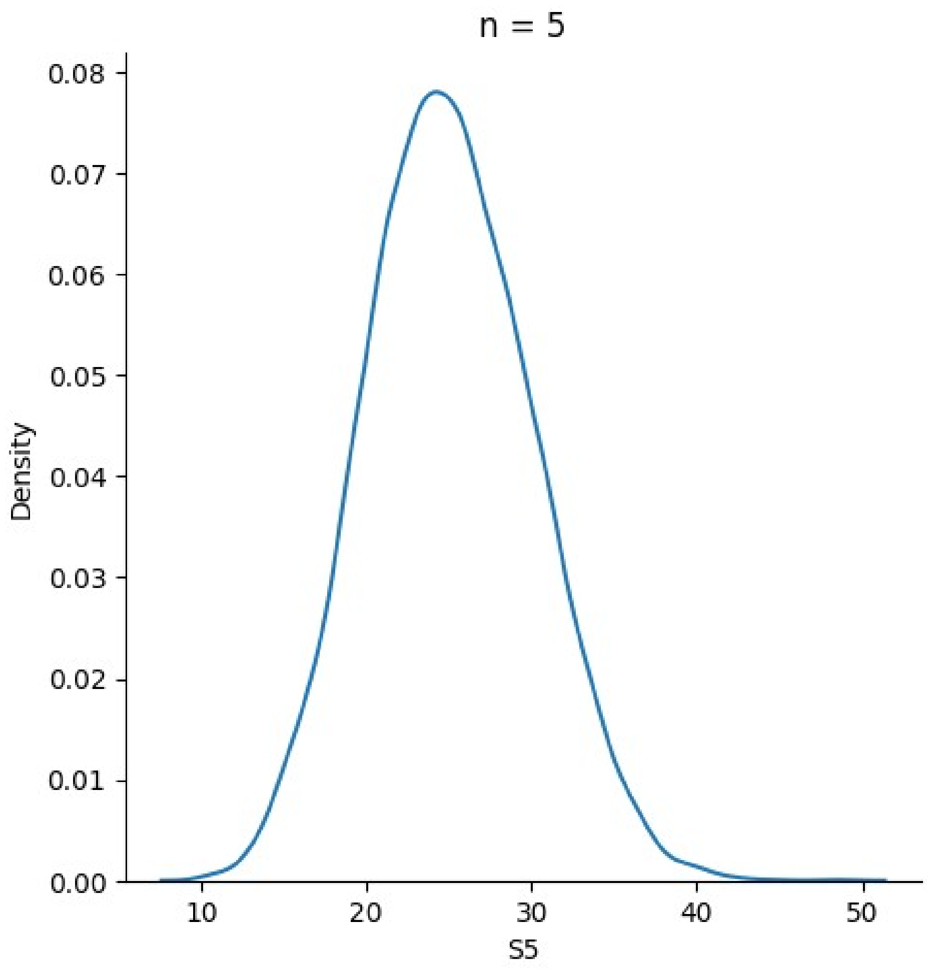
```python
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n=3
# Create all samples
samples = {f'x{i} sample': np.random.poisson(lam, sample_size) for i
in range(1, 4)}

# Create DataFrame
df = pd.DataFrame(samples)

# Calculate S3
df['S3'] = df.sum(axis=1)

# Plot the approximate PDF of S3
sns.displot(data=df['S3'], kind="kde").set(title='n = 3')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4e57f9030>
```



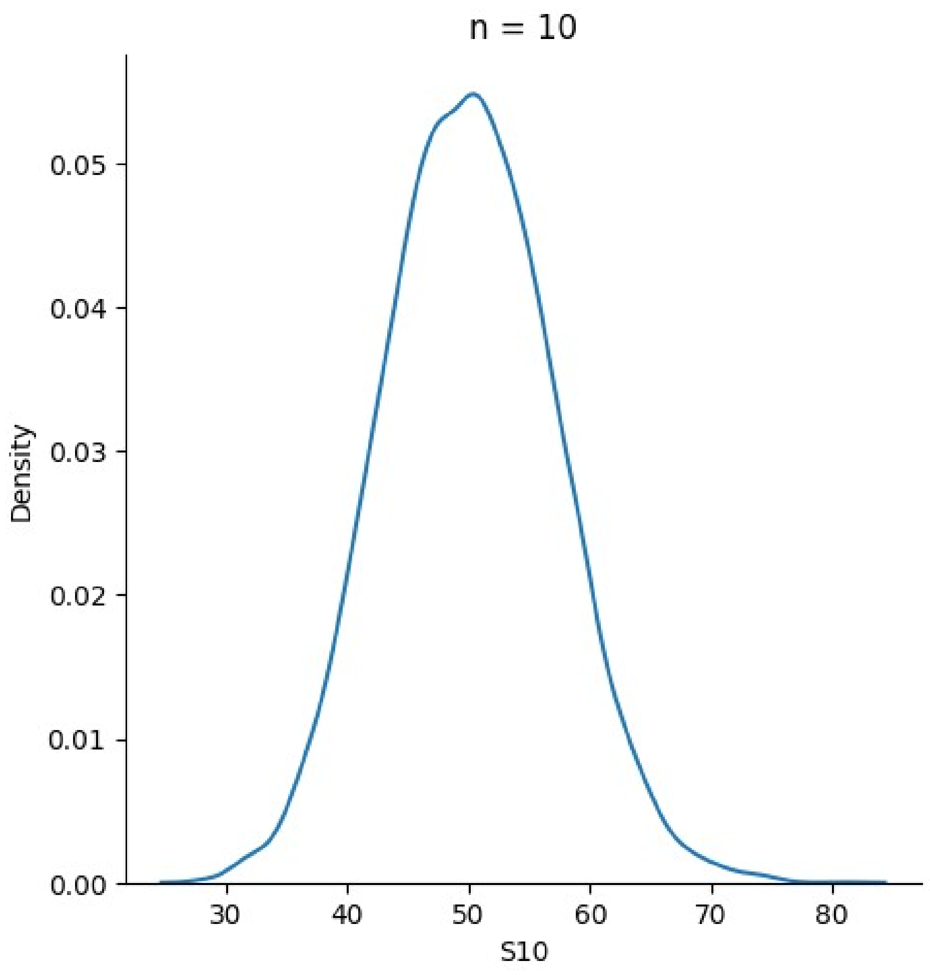
```
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n=5
# Create all samples
samples = {f'x{i} sample': np.random.poisson(lam, sample_size) for i
in range(1, 6)}

# Create DataFrame
df = pd.DataFrame(samples)

# Calculate S5
df['S5'] = df.sum(axis=1)

# Plot the approximate PDF of S5
sns.displot(data=df['S5'], kind="kde").set(title='n = 5')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4e567d720>
```



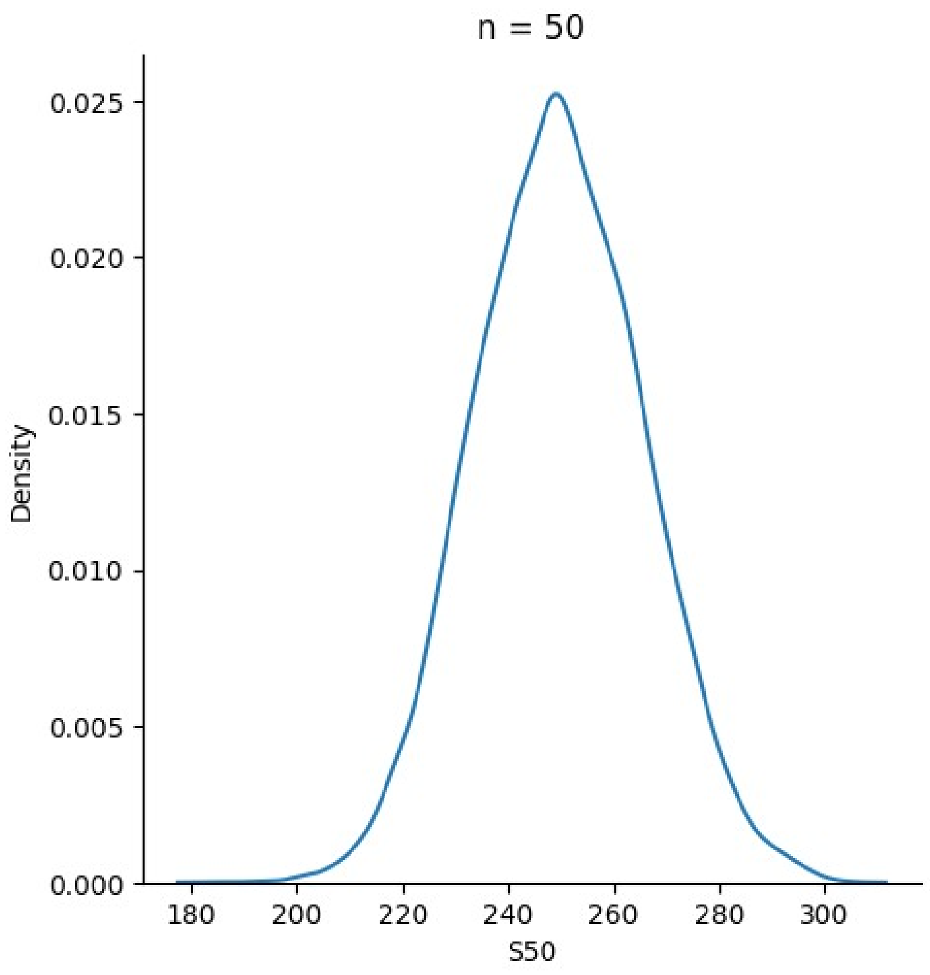
```
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n=10
# Create all samples
samples = {f'x{i} sample': np.random.poisson(lam, sample_size) for i
in range(1, 11)}

# Create DataFrame
df = pd.DataFrame(samples)

# Calculate S10
df['S10'] = df.sum(axis=1)

# Plot the approximate PDF of S10
sns.displot(data=df['S10'], kind="kde").set(title='n = 10')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4e58d2320>
```

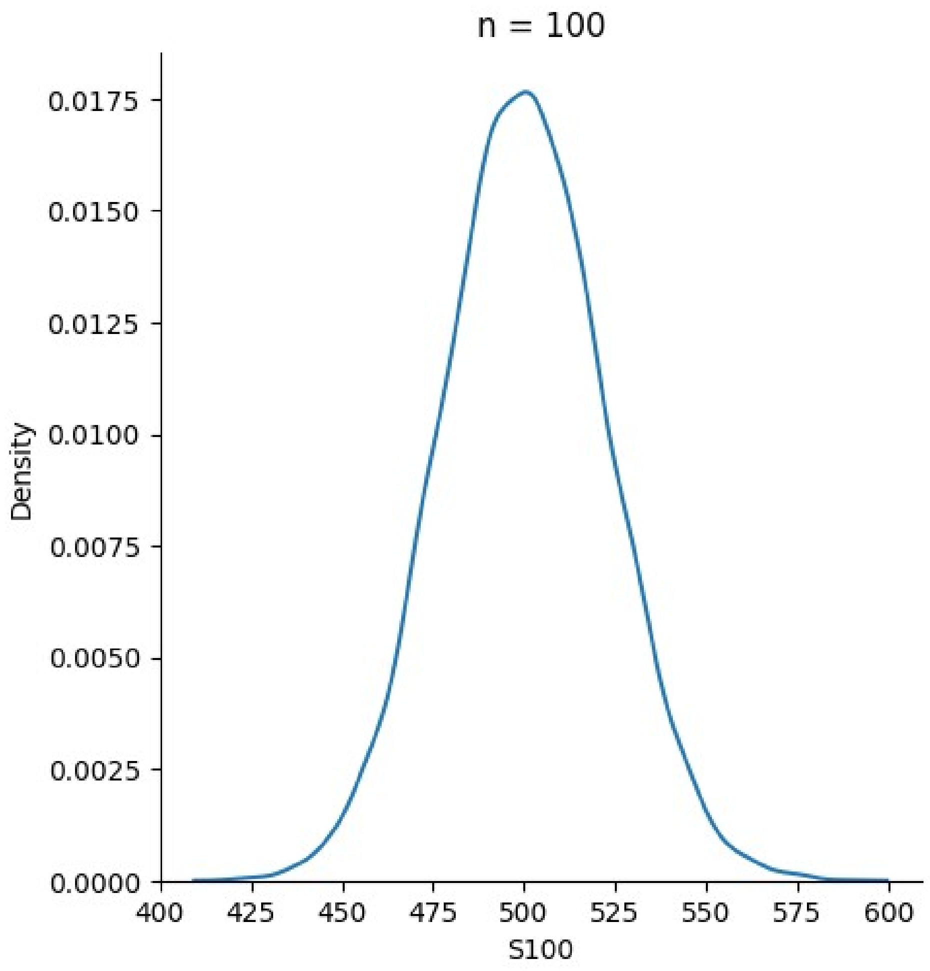


```
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n=50
# Create all samples
samples = {f'x{i} sample': np.random.poisson(lam, sample_size) for i
in range(1, 51)}

# Create DataFrame
df = pd.DataFrame(samples)

# Calculate S50
df['S50'] = df.sum(axis=1)

# Plot the approximate PDF of S50
sns.displot(data=df['S50'], kind="kde").set(title='n = 50')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fa4e5604580>
```



```
# Mohit Rai mr06638, Moiz Zulfiqar mz08229
# n=100
# Create all samples
samples = {f'x{i} sample': np.random.poisson(lam, sample_size) for i
in range(1, 101)}

# Create DataFrame
df = pd.DataFrame(samples)

# Calculate S100
df['S100'] = df.sum(axis=1)

# Plot the approximate PDF of S100
sns.displot(data=df['S100'], kind="kde").set(title='n = 100')

<seaborn.axisgrid.FacetGrid at 0x7fa4e565e980>
```



Comments Regarding the evolution of Sn's PDF form as n grows, you can see that Sn's distribution resembles a normal distribution more and more as n grows. This is because of the Central Limit Theorem, which asserts that, regardless of the initial distribution of the variables, the sum of a large number of independent, identically distributed random variables will roughly follow a normal distribution. Since each Xi in this instance is an independent Poisson random variable with an identical distribution, as n rises, their aggregate Sn moves closer to a normal distribution. For small n (e.g., n=1), the PDF's shape is Poisson; for bigger n (e.g., n=50 or n=100), it becomes more symmetrical.