Probability Distribution Function Pratical Implementation

Probability Mass Funcation

In [11]:
```
import seaborn as sns
import statistics
import matplotlib.pyplot as plt
import random
import pandas as pd
```

```
# if Single Dice is Rolled for 1000 times
## Chances of getting numbers
```

In [2]:
```
L=[]
for i in range(1000):
    L.append(random.randint(1,6))
```

In [5]:
```
L[:10]
```

Out[5]:
```
[6, 6, 2, 4, 6, 5, 1, 6, 6, 1]
```

```
# if Two Dice is Rolled for 1000 times
## Chances of getting numbers
```

In [6]:
```
L = []
for i in range(10000):
    a = random.randint(1,6)
    b = random.randint(1,6)

    L.append(a + b)
```

In [9]:
```
L[:5]
```

Out[9]:
```
[10, 8, 8, 5, 8]
```

In [12]:
```
pd.Series(L).value_counts()
```

Out[12]:
```
7     1641
8     1410
6     1387
9     1109
5     1093
10     874
4      841
3      571
11     568
2      254
12     252
dtype: int64
```

In [13]:
```
pd.Series(L).value_counts().sum()
```
Out[13]:
```
10000
```

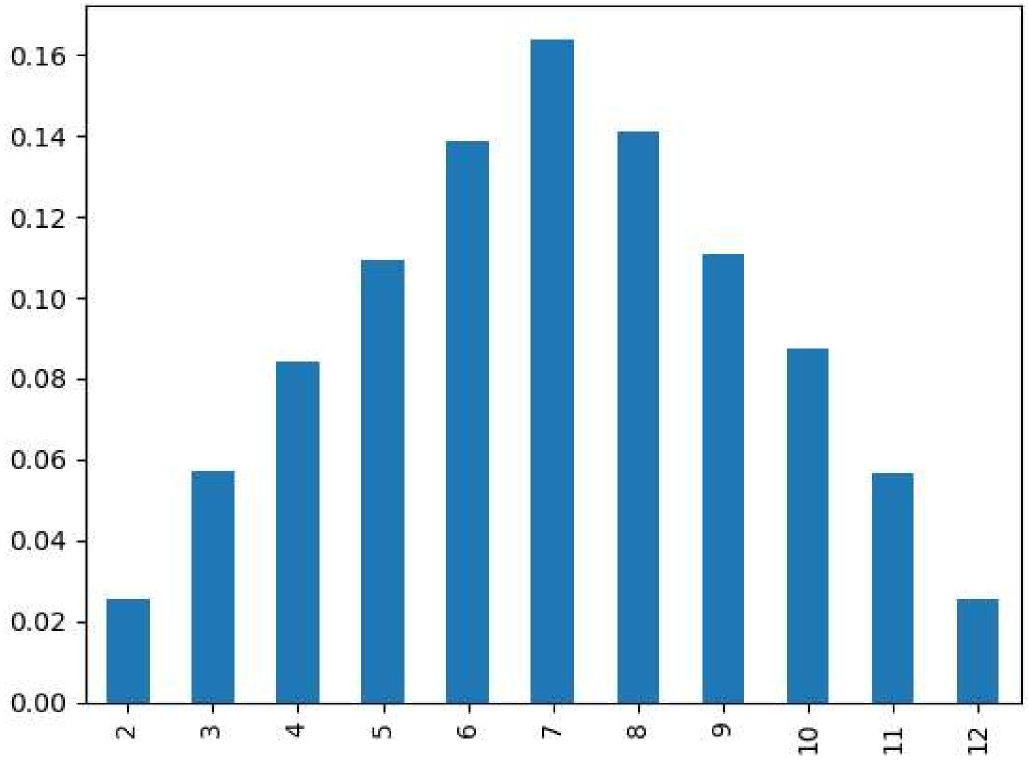
In [14]:
```
pd.Series(L).value_counts()/pd.Series(L).value_counts().sum()
```
Out[14]:
```
7     0.1641
8     0.1410
6     0.1387
9     0.1109
5     0.1093
10    0.0874
4     0.0841
3     0.0571
11    0.0568
2     0.0254
12    0.0252
dtype: float64
```

In [15]:
```
(pd.Series(L).value_counts()/pd.Series(L).value_counts().sum()).sort_index()
```
Out[15]:
```
2     0.0254
3     0.0571
4     0.0841
5     0.1093
6     0.1387
7     0.1641
8     0.1410
9     0.1109
10    0.0874
11    0.0568
12    0.0252
dtype: float64
```

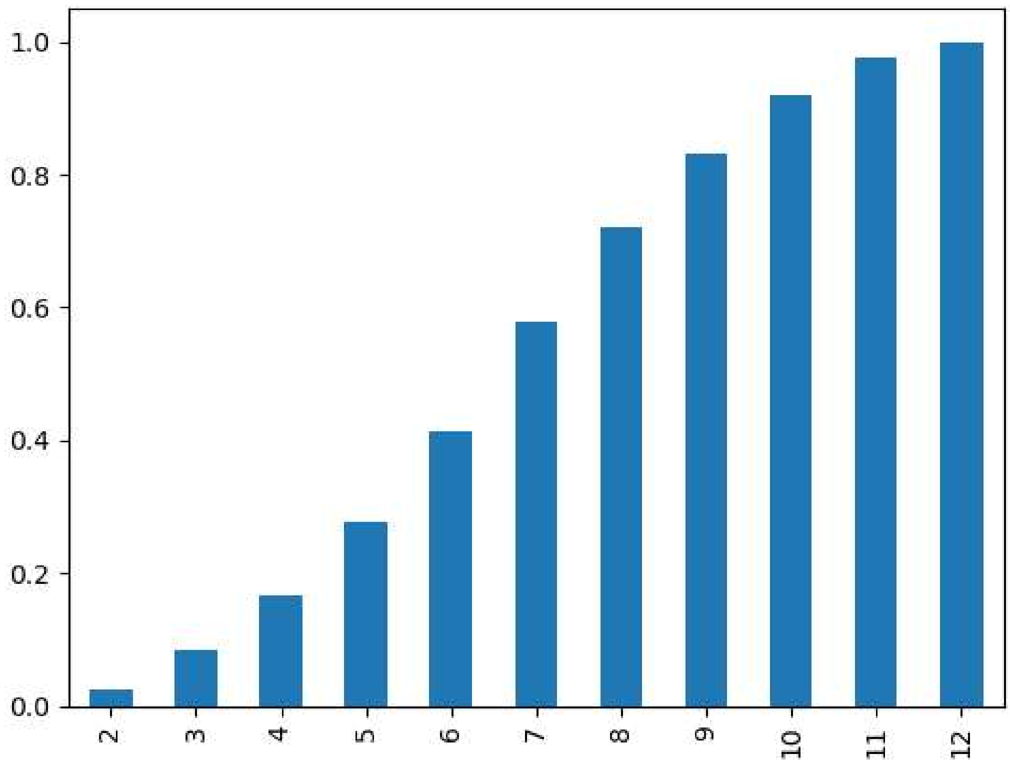
In [20]:
```
s = (pd.Series(L).value_counts()/pd.Series(L).value_counts().sum()).sort_index(
```

In [17]:
```
import numpy as np
np.cumsum(s)
```
Out[17]:
```
2     0.0254
3     0.0825
4     0.1666
5     0.2759
6     0.4146
7     0.5787
8     0.7197
9     0.8306
10    0.9180
11    0.9748
12    1.0000
dtype: float64
```

In [18]:
```
s.plot(kind='bar')
```
Out[18]: <Axes: >



In [19]:
```
np.cumsum(s).plot(kind='bar')
```
Out[19]: <Axes: >



In [ ]:
```

```

Probability Density Function

Parametric Density Estimation

In [35]:
```python
from numpy.random import normal

sample = normal(loc=50, scale=5,size=1000)
```

In [36]:
```python
sample
```

```
        55.53784104, 41.06298964, 46.33982049, 52.51954708, 45.0652093 ,
        48.62159399, 48.62449045, 47.56946958, 52.82961673, 41.6542016 ,
        44.4041443 , 46.52092917, 48.10721169, 50.39475177, 43.35602488,
        43.90552686, 59.61282112, 53.36249785, 55.5479493 , 51.63614222,
        52.40556726, 47.82609111, 49.68469971, 53.82622874, 49.26679037,
        51.35737457, 43.87720568, 54.28859473, 44.36625562, 43.92146206,
        49.25264189, 43.21754818, 55.25474992, 39.85602295, 50.27533111,
        46.7137934 , 50.90531791, 50.91395094, 32.875293  , 45.65092132,
        49.56763655, 57.36486996, 60.37704073, 46.68007223, 50.28255209,
        51.24170621, 55.91010554, 54.75528883, 45.82384594, 47.79932419,
        59.68112534, 52.6925637 , 48.90908203, 46.51047187, 52.27375947,
        45.14220822, 43.26596485, 52.06796534, 47.48795095, 44.74766323,
        56.70652037, 46.06053169, 60.07096964, 47.07644697, 52.45681659,
        49.30851503, 49.2073696 , 52.09024624, 55.80086697, 53.73898743,
        50.02330752, 55.25752193, 55.00881331, 56.18729407, 42.86896557,
        50.12429547, 43.74067699, 45.99803273, 49.26293691, 50.82826809,
        49.56305183, 54.00313897, 52.52011805, 52.19746119, 53.45362296,
        46.89485811, 50.98696345, 42.0030731 , 55.87449577, 46.69220442,
        47.7954326 , 57.67551738, 43.65415679, 52.09899674, 51.69887299,
        54.43743353, 45.08208384, 53.53044071, 48.41544376, 51.02919982,
```

In [37]:
```python
sample.mean()
```

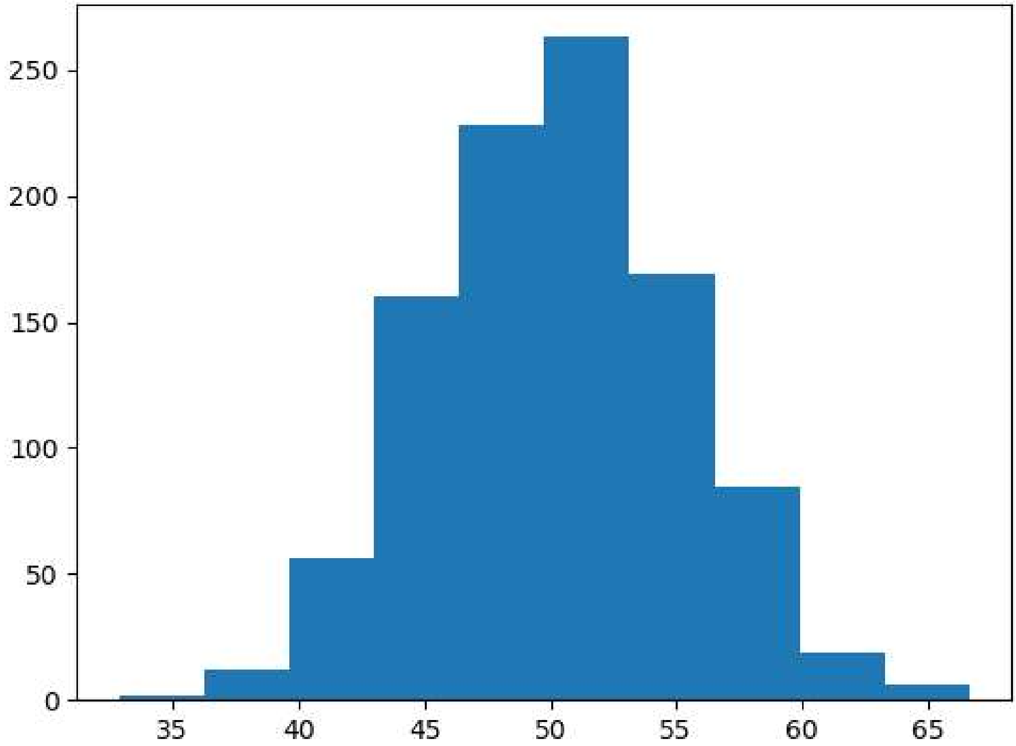
Out[37]: 50.19120634821261

In [38]:
```python
sample.std()
```

Out[38]: 5.015112308145442

In [39]:
```python
# plot histogram to understand the distribution of data
plt.hist(sample,bins=10)
```

Out[39]:
```
(array([  2.,  12.,  56., 160., 228., 263., 169.,  85.,  19.,   6.]),
 array([32.875293  , 36.25253122, 39.62976944, 43.00700766, 46.38424587,
        49.76148409, 53.13872231, 56.51596053, 59.89319875, 63.27043697,
        66.64767519]),
 <BarContainer object of 10 artists>)
```



In [40]:
```python
# calculate sample mean and sample std dev
sample_mean = sample.mean()
sample_std = sample.std()
```

In [41]:
```python
# fit the distribution with the above parameters

from scipy.stats import norm
dist = norm(60, 12)
```

In [42]:
```python
values = np.linspace(sample.min(),sample.max(),100)
```

In [43]:
```python
sample.max()
```

Out[43]: 66.6476751889642

In [44]:
```python
sample.min()
```

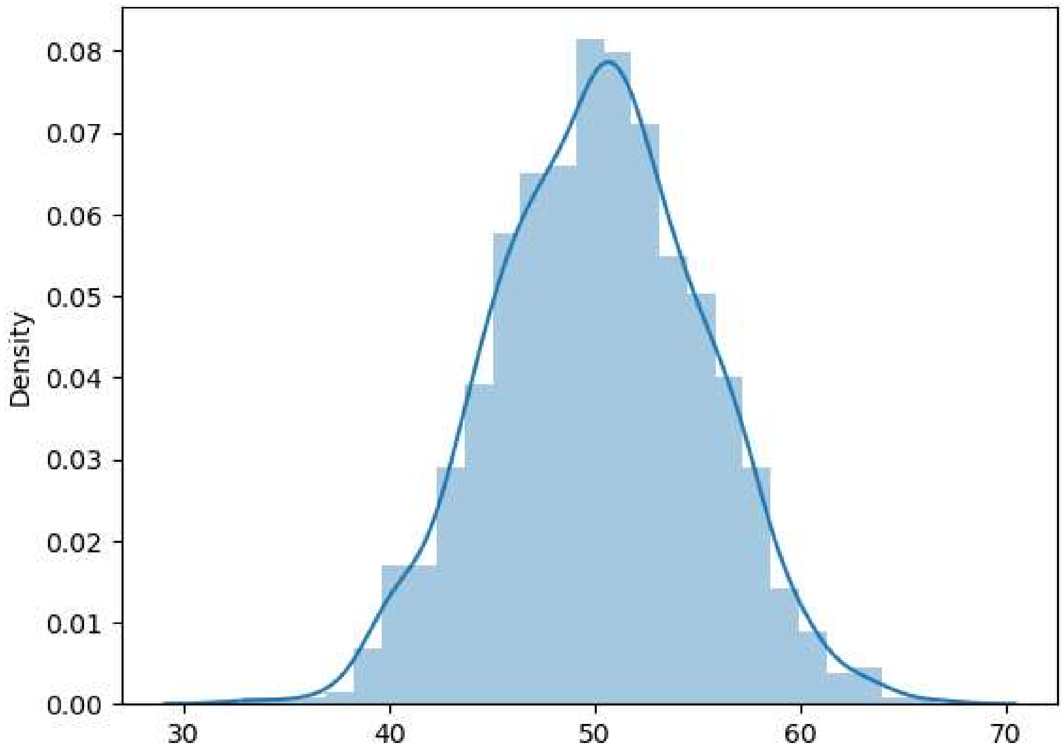
Out[44]: 32.875292998372515

In [45]:
```python
probabilities = [dist.pdf(value) for value in values]
```

In [48]:
```
import seaborn as sns
sns.distplot(sample)
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_1036\1482356190.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(sample)
```

Out[48]: <Axes: ylabel='Density'>



Non Parametric Density Estimation

In [50]:
```
# generate a sample
sample1 = normal(loc=20, scale=5, size=300)
sample2 = normal(loc=40, scale=5, size=700)
sample = np.hstack((sample1, sample2))
```

In [51]:

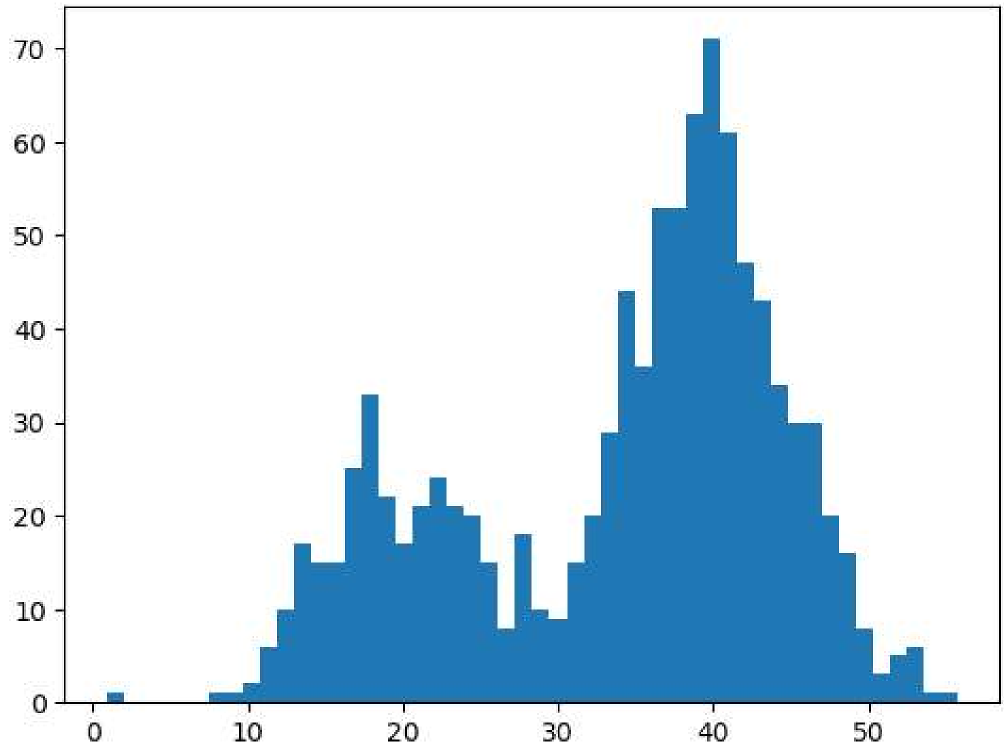
```python
sample
```

Out[51]:

```
array([20.41045957, 12.0382574 , 16.03287492, 23.10769314, 17.57787548,
       15.71468691, 10.73017185, 18.10793421, 34.54842875, 23.61207937,
       23.06398232, 33.21385954, 31.42199103, 18.15115907, 14.39267612,
       23.01723844, 16.72531329, 22.7479508 , 20.54766286, 11.27897051,
       12.57185908, 16.29530424, 24.3885556 , 25.64790734, 26.50786067,
       21.68583296, 13.16376936, 25.02906137, 20.98540971, 19.20899069,
        0.93130587, 18.3865535 , 22.4236775 , 15.59937169, 24.0952057 ,
       17.77275375, 27.21446458, 22.46982336, 21.6802293 , 14.58759596,
       26.08685596, 23.22634176, 32.77386257, 17.99125039, 23.99227005,
       16.66672854, 18.41892568, 22.38432422, 14.35320614, 17.11321381,
       26.39153716, 12.43467592, 28.51614946, 19.62464048, 19.21623312,
       22.85711389, 15.4342385 , 28.33630387, 23.47483379, 23.47110714,
       16.24920748, 13.05379279, 20.90186187, 21.41474249, 17.20936164,
       24.84464369, 25.14538515, 14.80473615, 25.53213377, 22.60762225,
       24.03231706, 17.83261338, 17.65728141, 17.45870642, 14.60050083,
       22.15655513, 21.00837888, 13.3889863 , 17.40401711, 13.28078503,
       24.20206307, 15.85812547, 21.47776474, 18.35996672, 16.83157174,
       25.81440276, 20.37963616, 21.18456873, 14.92178011, 18.29183026,
       20.1779268 , 17.0353535 , 10.12601813, 18.32643261, 27.70326994,
```

In [52]:
```python
# plot histogram bins=50
plt.hist(sample,bins=50)
```

Out[52]:
```
(array([ 1.,  0.,  0.,  0.,  0.,  0.,  1.,  1.,  2.,  6., 10., 17., 15.,
        15., 25., 33., 22., 17., 21., 24., 21., 20., 15.,  8., 18., 10.,
         9., 15., 20., 29., 44., 36., 53., 53., 63., 71., 61., 47., 43.,
        34., 30., 30., 20., 16.,  8.,  3.,  5.,  6.,  1.,  1.]),
 array([ 0.93130587,  2.02772532,  3.12414478,  4.22056423,  5.31698368,
         6.41340314,  7.50982259,  8.60624205,  9.7026615 , 10.79908096,
        11.89550041, 12.99191987, 14.08833932, 15.18475878, 16.28117823,
        17.37759769, 18.47401714, 19.5704366 , 20.66685605, 21.76327551,
        22.85969496, 23.95611441, 25.05253387, 26.14895332, 27.24537278,
        28.34179223, 29.43821169, 30.53463114, 31.6310506 , 32.72747005,
        33.82388951, 34.92030896, 36.01672842, 37.11314787, 38.20956733,
        39.30598678, 40.40240623, 41.49882569, 42.59524514, 43.6916646 ,
        44.78808405, 45.88450351, 46.98092296, 48.07734242, 49.17376187,
        50.27018133, 51.36660078, 52.46302024, 53.55943969, 54.65585915,
        55.7522786 ]),
 <BarContainer object of 50 artists>)
```



In [53]:
```python
from sklearn.neighbors import KernelDensity

model = KernelDensity(bandwidth=5, kernel='gaussian')

# convert data to a 2D array
sample = sample.reshape((len(sample), 1))

model.fit(sample)
```

Out[53]:

▾ KernelDensity

KernelDensity(bandwidth=5)

In [54]:

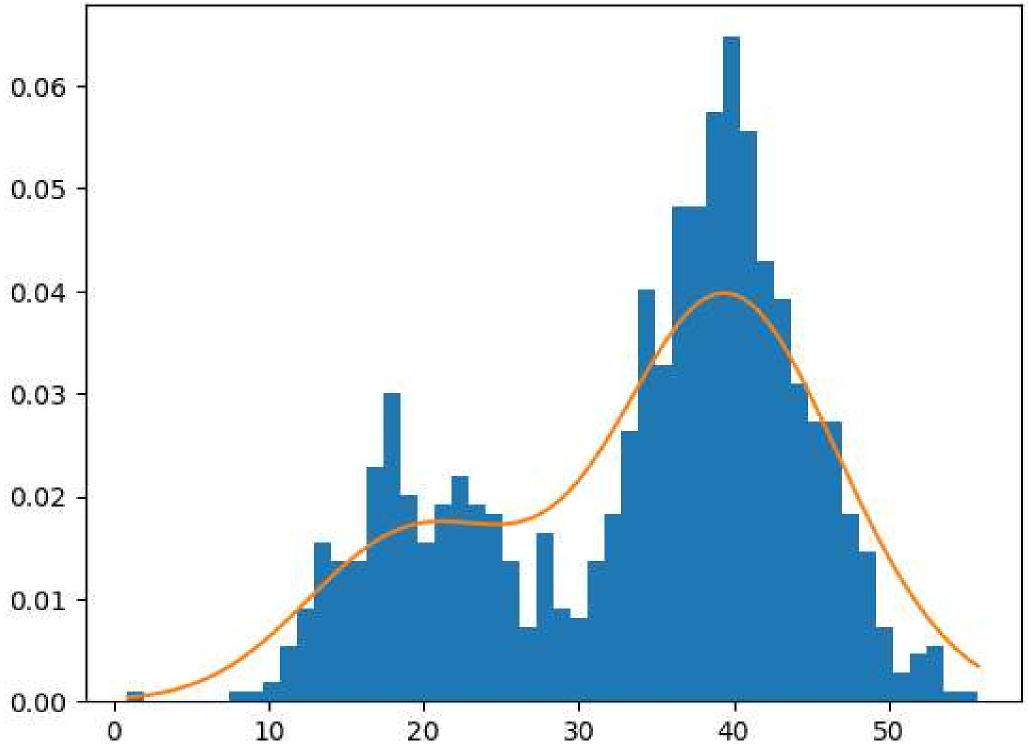
```
values = np.linspace(sample.min(),sample.max(),100)
values = values.reshape((len(values), 1))
```

In [55]:

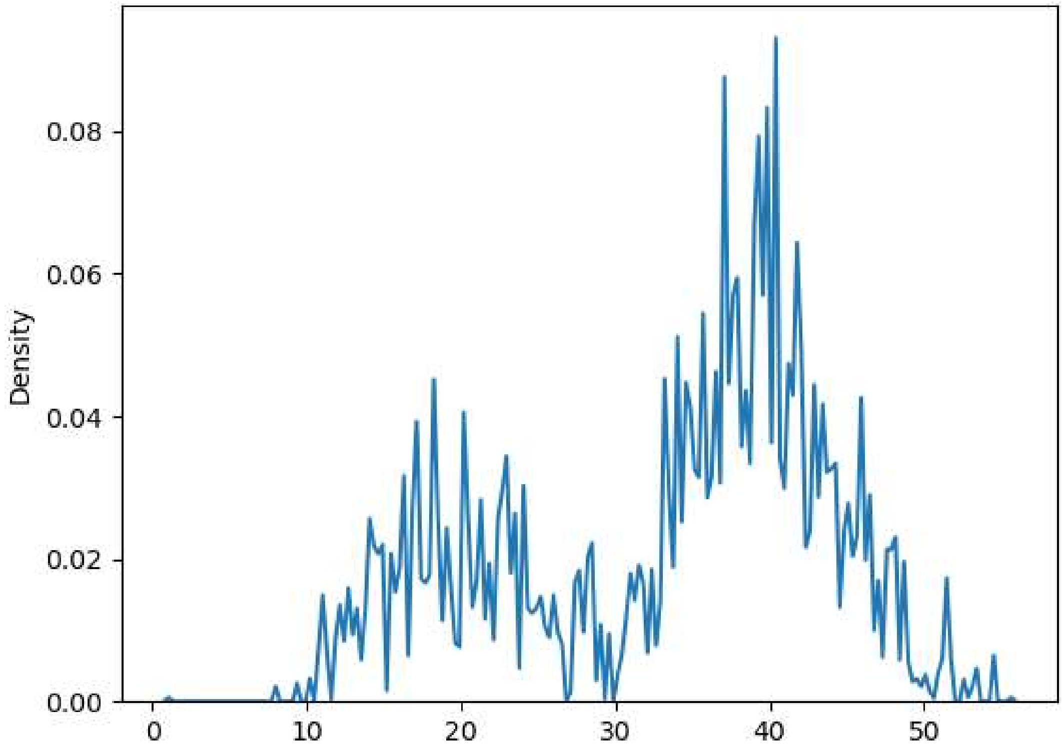
```
probabilities = model.score_samples(values)
probabilities = np.exp(probabilities)
```

In [56]:

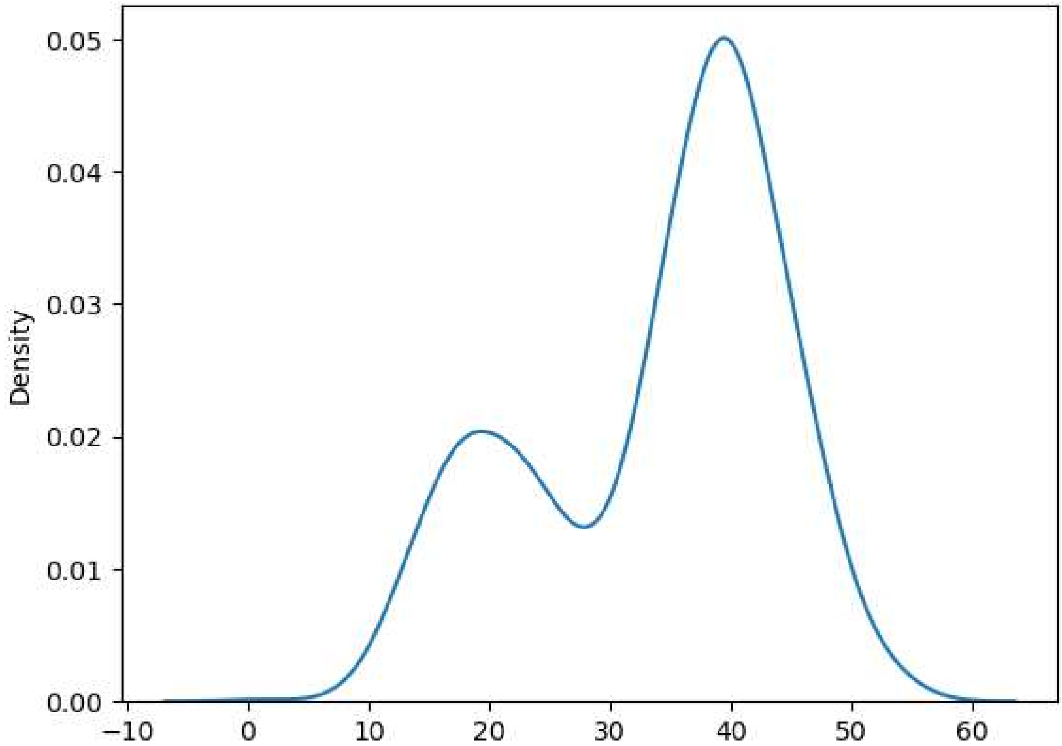
```
plt.hist(sample, bins=50, density=True)
plt.plot(values[:], probabilities)
plt.show()
```



In [57]:
```
sns.kdeplot(sample.reshape(1000),bw_adjust=0.02)
```
Out[57]: <Axes: ylabel='Density'>



In [59]:
```
sns.kdeplot(sample.reshape(1000),bw_adjust=1)
```
Out[59]: <Axes: ylabel='Density'>



Performing PMF and PDF on iris Dataset

In [61]:
```
df = sns.load_dataset('iris')
```

In [62]:
```
df.head()
```

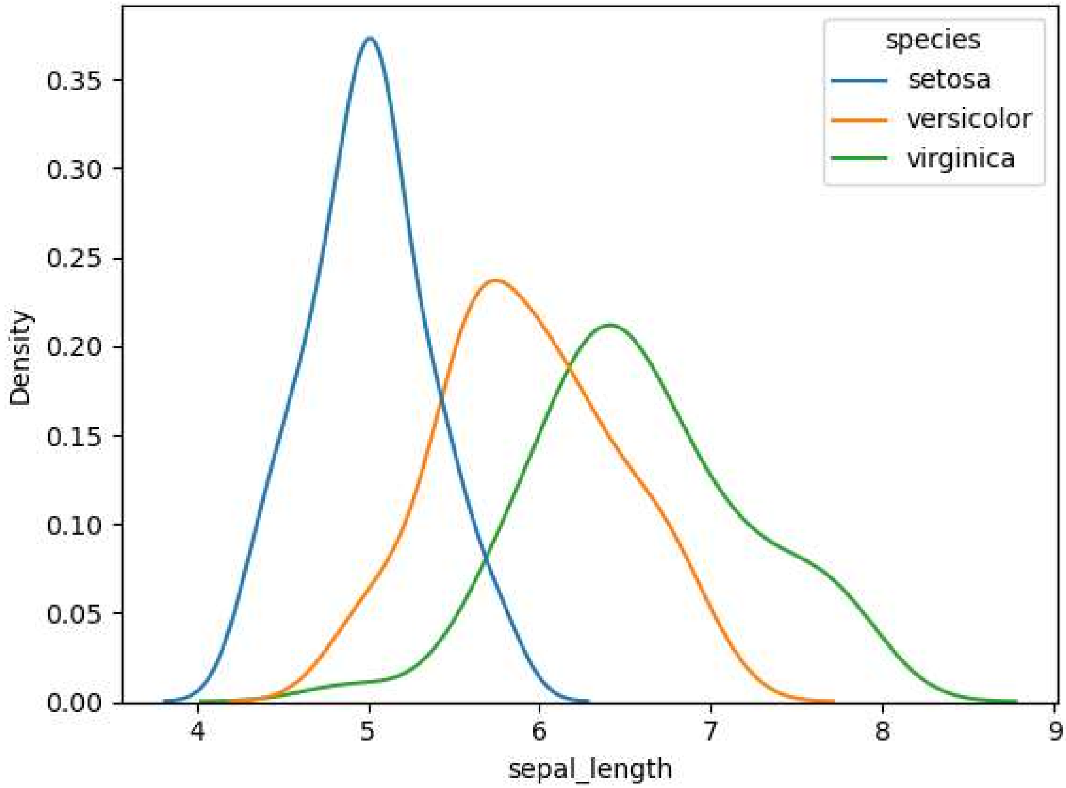
Out[62]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

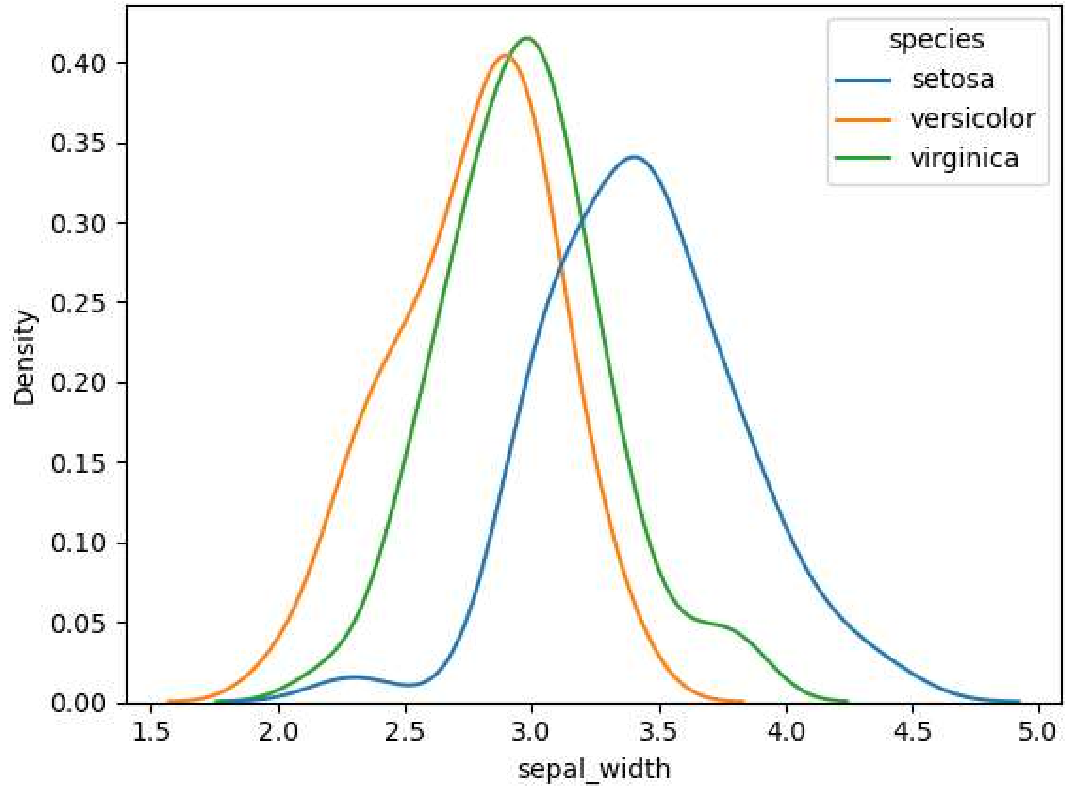
In [ ]:
```
## PDF on Continuous Variable
```

In [63]:
```
sns.kdeplot(data=df,x='sepal_length',hue='species')
```

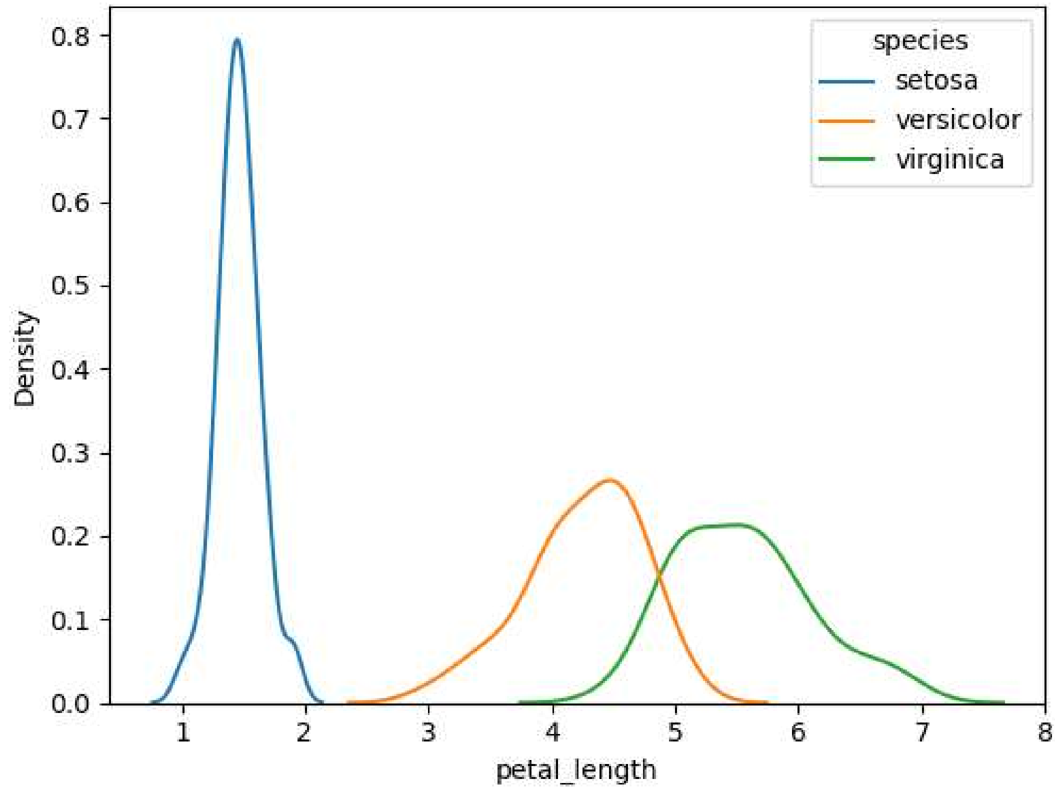
Out[63]: <Axes: xlabel='sepal_length', ylabel='Density'>



In [64]:
```
sns.kdeplot(data=df,x='sepal_width',hue='species')
```
Out[64]: <Axes: xlabel='sepal_width', ylabel='Density'>

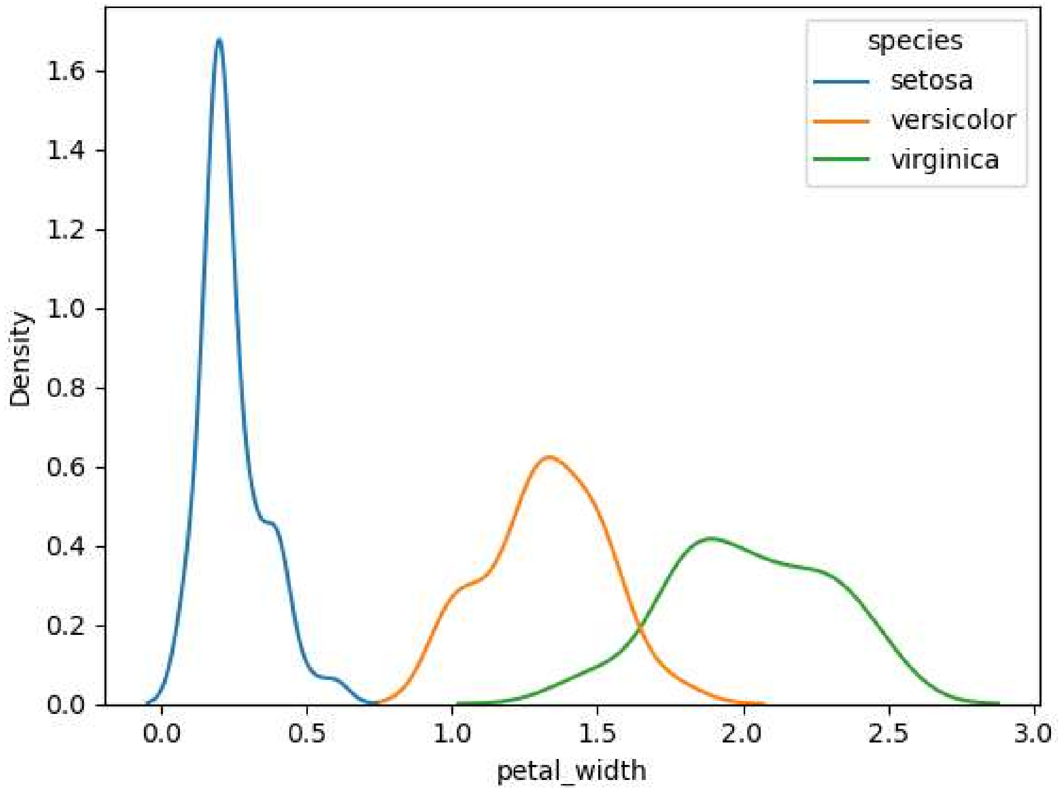


In [65]:
```
sns.kdeplot(data=df,x='petal_length',hue='species')
```
Out[65]: <Axes: xlabel='petal_length', ylabel='Density'>



In [66]:
```
1  sns.kdeplot(data=df,x='petal_width',hue='species')
```

Out[66]: <Axes: xlabel='petal_width', ylabel='Density'>



In [69]:
```
1  ## PMF on Descrete Variable
```

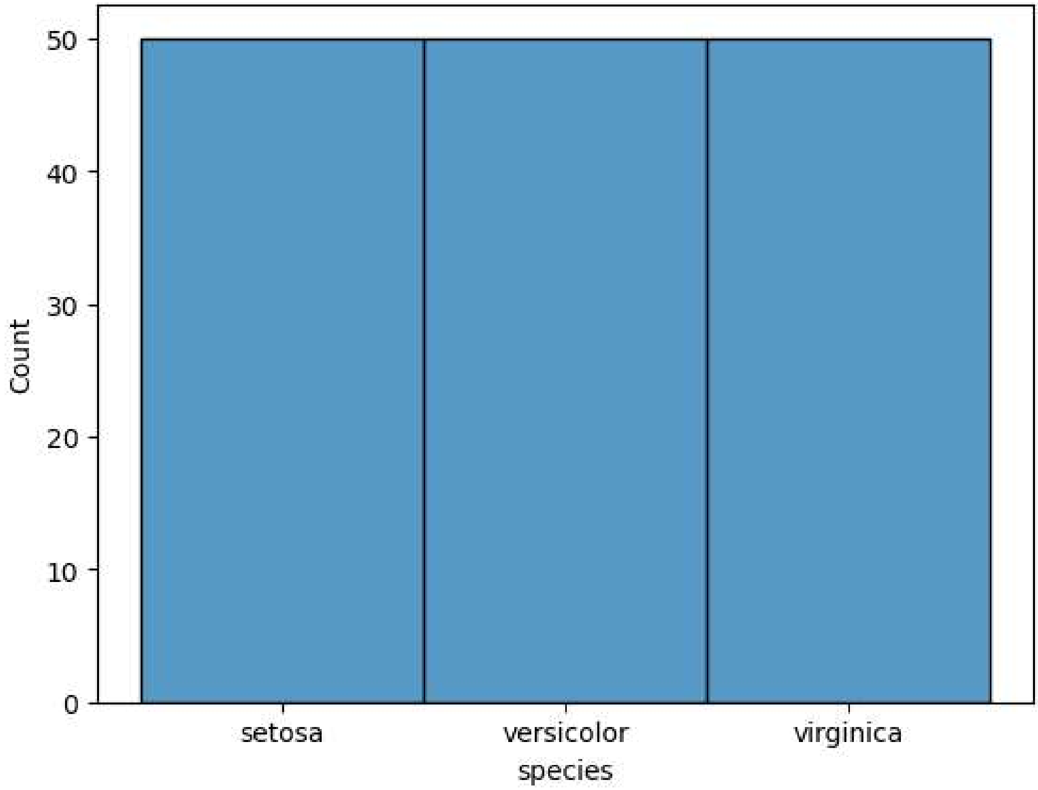
In [70]:
```
1  df.columns
```

Out[70]:
```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

In [71]:
```
1  df['species']
```

Out[71]:
```
0         setosa
1         setosa
2         setosa
3         setosa
4         setosa
         ...
145    virginica
146    virginica
147    virginica
148    virginica
149    virginica
Name: species, Length: 150, dtype: object
```

In [72]:
```
sns.histplot(df['species'])
```
Out[72]: <Axes: xlabel='species', ylabel='Count'>



In [ ]:
```

```