1. Central Limit Theorem

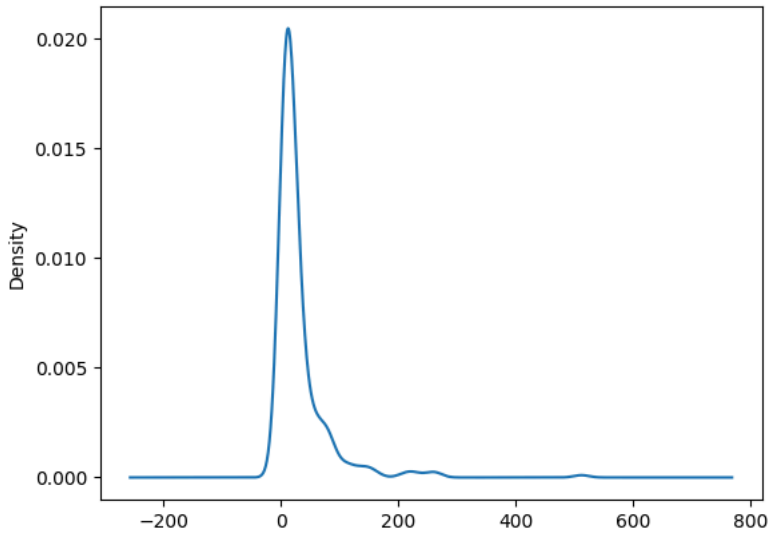
```
import pandas as pd
import numpy as np
import seaborn as sns
```

```
df=sns.load_dataset('titanic')
df.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

```
#plotting distribution of fare column
df['fare'].plot(kind='kde')
```

```
<Axes: ylabel='Density'>
```




```
# sample size = 50 -->  100 times
samples = []
for i in range(120):
    samples.append(df['fare'].sample(50).tolist())
```

```
samples
```

```
[[24.15,
  39.6875,
  106.425,
  153.4625,
  9.35,
  83.1583,
  13.0,
  7.4958,
  7.4958,
  7.8958,
  7.2292,
  24.15,
  7.925,
  13.0,
  26.25,
  9.825,
  26.2833,
```

```
 13.0,
 8.1125,
 7.0458,
 7.8958,
 13.0,
 37.0042,
 13.0,
 21.0,
 110.8833,
 7.75,
 7.75,
 153.4625,
 26.3875,
 7.775,
 73.5,
 31.3875,
 7.8958,
 7.2292,
 13.0,
 86.5,
 56.4958,
 26.0,
 31.0,
 29.125,
 21.0,
 49.5042,
 7.8542,
 19.5,
 14.4542,
 26.0,
 19.2583,
 15.0458,
 46.9],
[31.275,
 113.275,
 79.65,
 46.9,
 14.4542,
 26.0,
 27.9,
 7.3125,
```

```
len(samples)
```

```
120
```

```
np.array(samples).shape
```

```
(120, 50)
```

```
samples=np.array(samples)
```

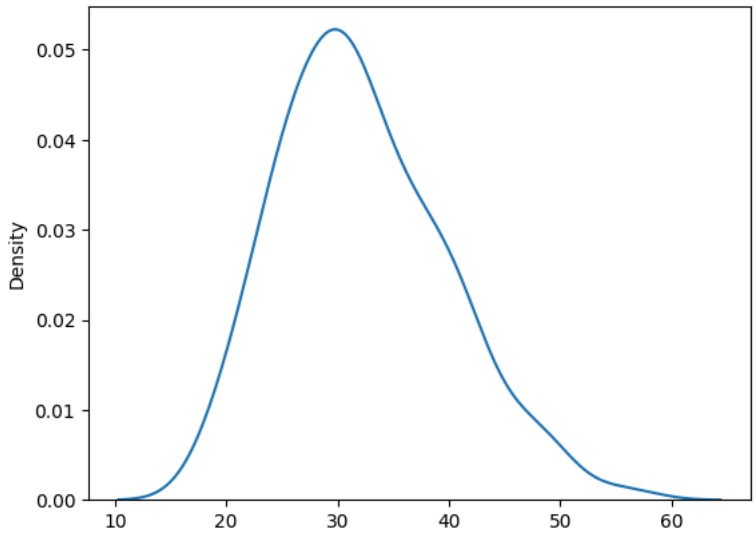
```
samples
```

```
array([[ 24.15  ,  39.6875, 106.425 , ...,  19.2583,  15.0458,  46.9   ],
       [ 31.275 , 113.275 ,  79.65  , ...,   8.6625,  13.    ,  10.5   ],
       [  7.8958,  47.1   ,  14.4542, ...,   9.5875,  26.    ,  50.4958],
       ...,
       [ 22.525 ,   8.05  ,  76.7292, ...,   7.8958,  27.9   ,  13.    ],
       [ 10.5   ,  34.375 ,   8.6625, ..., 153.4625,  83.1583,   9.    ],
       [ 79.65  ,   7.8958,  14.4542, ...,   7.8542,  20.525 ,  56.4958]])
```

```
# mean of each sample
sampling_means=samples.mean(axis=1)
```

```
# ampling distribution of sample mean
sns.kdeplot(sampling_means)
```

```
<Axes: ylabel='Density'>
```




```
# calculating samling mean
sampling_means.mean()
```

```
np.float64(32.2566629)
```

```
df['fare'].mean()
```

```
np.float64(32.204207968574636)
```

Confidence Interval

```
samples=[]
stds=[]
for i in range(10):
  x=df['fare'].sample(30)
  stds.append(x.std())
  samples.append(x.tolist())
```

```
samples=np.array(samples)
```

```
samples.shape
```

```
(10, 30)
```

```
# mean of each sample
sampling_means=samples.mean(axis=1)
```

```
sampling_means
```

```
array([33.66695   , 26.06125   , 30.43138333, 28.03945   , 34.97055333,
       57.08792   , 16.84249667, 29.13903   , 23.56708333, 20.18375   ])
```

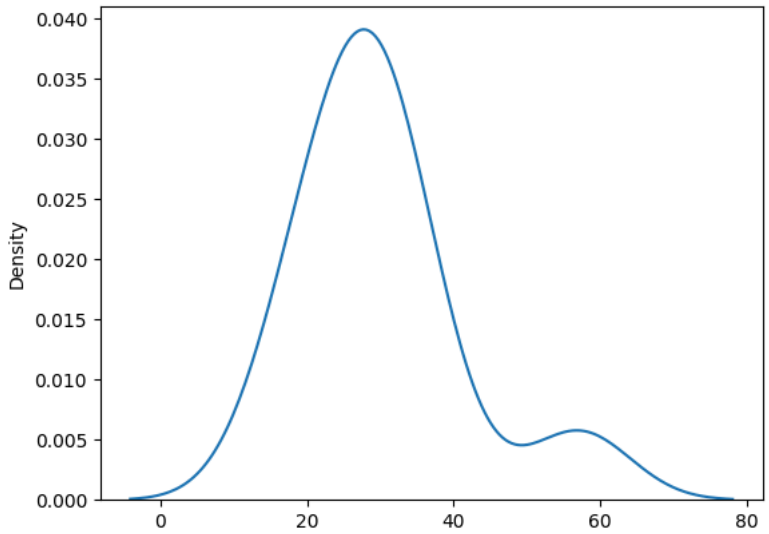
```
sample_stds=np.mean(stds)
```

```
sample_stds
```

```
np.float64(39.60664238400415)
```

```
sns.kdeplot(sampling_means)
```

```
<Axes: ylabel='Density'>
```




```
#t test formula
lower_limit=sampling_means.mean()-2.04*(sample_stds/np.sqrt(30))
upper_limit=sampling_means.mean()+2.04*(sample_stds/np.sqrt(30))
```

```
print("Range : ",(lower_limit,upper_limit))
```

```
Range :  (np.float64(15.24743967335037), np.float64(44.75053365998296))
```

```
df['fare'].mean()
```

```
np.float64(32.204207968574636)
```

```
Start coding or generate with AI.
```