Descriptive statistics: Available in the data

Deals with presentaion and collection of data.

Measures of Central Tendency

Mean Mode Median

In [51]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [53]:

```
df=pd.DataFrame(dict(age=np.random.randint(18,31,size=6)))
print(df)
```

```
   age
0   27
1   20
2   26
3   28
4   23
5   29
```

Mean

In [54]:

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

Out[54]:

25.5

Median

In [55]:

```
df['age'].median()
```

Out[55]:

26.5

Mode

In [56]:

```
df['age'].mode()
```

Out[56]:

```
0    20
1    23
2    26
3    27
4    28
5    29
dtype: int32
```

Measures of variability

range Quartile Variance Standard Deviation

Range

In [14]:

```
df['age'].max() - df['age'].min()
```

Out[14]:

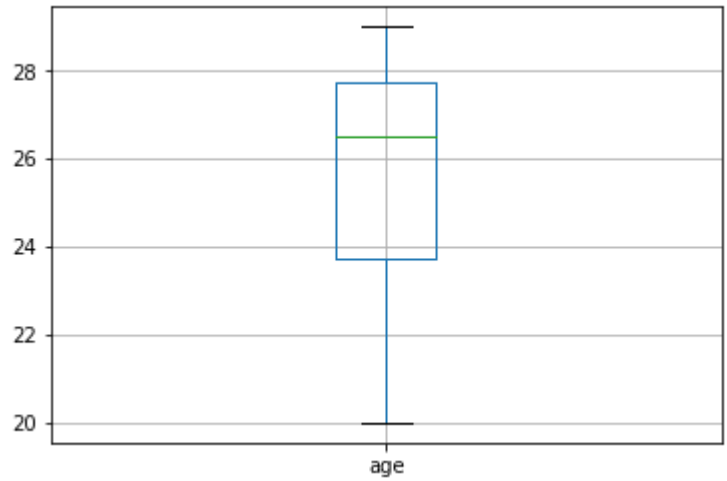
```
9
```

In [57]:

```
df.boxplot(column='age', return_type='axes')
```

Out[57]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x217eb4991d0>
```



variance

In [58]:

```
df['age'].var()
```

Out[58]:

11.5

Standard Deviation

In [59]:

```
df['age'].std()
```

Out[59]:

3.391164991562634

Skewness

Skewness usually describes lack of symmetry from the mean.

A perfectly symmetric data will have a skewness of 0. Ex: Normal Distribution.

Negative values for the skewness indicate data that are skewed left, the left tail is long relative to the right tail.

Positive values for the skewness indicate data that are skewed right, the right tail is long relative to the left tail.

In [60]:

```
df['age'].skew()
```

Out[60]:

-0.9231148559263125

Kurtosis

Kurtosis: kurtosis is a measure of the "tailedness" of the probability distribution.

Distributions having higher kurtosis have fatter tails or more extreme values (a phenomenon called 'leptokurtosis') and those with lower kurtosis have fatter middles or fewer extreme values called 'platykurtosis'.

Positive kurtosis means more peak towards the normal distribution and

negative kurtosis means less peak towards the normal distribution

In [61]:

```
df['age'].kurt()
```

Out[61]:

-0.17164461247637064

Inferential statistics: Which are not available and can be inferred from the data

In [62]:

```
np.random.seed(7)
#Create a population
population=np.random.randint(10,20,1000)
# population mean
mean_population=population.mean()
print(mean_population)
```

14.491

In [63]:

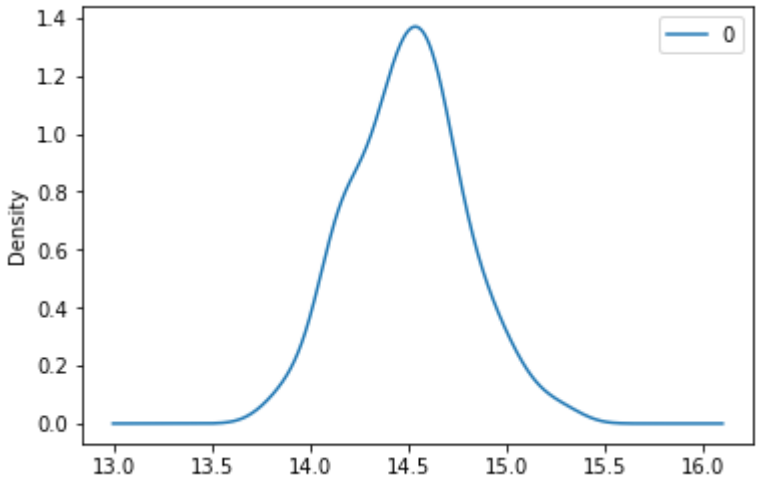
```
sample_means=[] # A list is created to store the sample estimates
#generate samples from the population
for x in range(200):
    sample=np.random.choice(a=population, size=100)
    sample_means.append(sample.mean())
```

In [64]:

```
#plot the sample means
df1=pd.DataFrame(sample_means)
df1.plot(kind='density')
```

Out[64]:

<matplotlib.axes._subplots.AxesSubplot at 0x217eb4ce6d8>



Confidence Interval

Confidence intervals usually covers 95% of the data and gives us the lower limit and upper limit.

Confidence interval= (Sample_mean-margin_Error, Sample_mean+margin_Error)

Margin Error= crtical_val * (std) /sqrt(n)

In [65]:

```
import scipy.stats as stats
z_critical= stats.norm.ppf(q=0.975)  # percent point function
t_critical= stats.t.ppf(q=0.975, df=24)
margin_error = z_critical * (np.std(sample_means) / np.sqrt(200))
```

In [66]:

```
#Lower limit of the confidence interval
lower_limit= np.mean(sample_means) - margin_error
print(lower_limit)
```

```
14.454827128111752
```

In [68]:

```
#Upper limit of the confidence interval
upper_limit= np.mean(sample_means) + margin_error
print(upper_limit)
```

```
14.534872871888247
```