Q7

In [82]:
```python
import pandas as pd
import numpy as np
from scipy import stats
```

In [88]:
```python
df=pd.read_csv("C:\\Users\\chann\\Downloads\\Q7.csv")
```

In [64]:
```python
df.head()
```

Out[64]:

	4.11	Points	Score	Weigh
0	Mazda RX4	3.90	2.620	16.46
1	Mazda RX4 Wag	3.90	2.875	17.02
2	Datsun 710	3.85	2.320	18.61
3	Hornet 4 Drive	3.08	3.215	19.44
4	Hornet Sportabout	3.15	3.440	17.02

In [9]:
```python
import warnings
warnings.filterwarnings('ignore')
```

In [66]:
```python
car_name = 'car_name'
df = df.rename(columns={df.columns[0]: car_name})
```

In [67]:
```python
df.head()
```

Out[67]:

	car_name	Points	Score	Weigh
0	Mazda RX4	3.90	2.620	16.46
1	Mazda RX4 Wag	3.90	2.875	17.02
2	Datsun 710	3.85	2.320	18.61
3	Hornet 4 Drive	3.08	3.215	19.44
4	Hornet Sportabout	3.15	3.440	17.02

In [68]:
```python
df = df.iloc[:, 1:]
```

In [69]:
```python
df.head()
```

Out[69]:

	Points	Score	Weigh
0	3.90	2.620	16.46
1	3.90	2.875	17.02
2	3.85	2.320	18.61
3	3.08	3.215	19.44
4	3.15	3.440	17.02

In [70]:
```python
df.mean()
```

Out[70]:
```
Points     3.596563
Score      3.217250
Weigh     17.848750
dtype: float64
```

In [71]:
```python
df.median()
```

Out[71]:
```
Points     3.69000
Score      3.21725
Weigh     17.82000
dtype: float64
```

In [72]:
```python
df.mode().iloc[0]
```

Out[72]:
```
Points     3.07
Score      3.44
Weigh     17.02
Name: 0, dtype: float64
```

In [73]:
```python
df.var()
```

Out[73]:
```
Points    0.276948
Score     0.927461
Weigh     3.093380
dtype: float64
```

In [74]:
```python
df.std()
```

Out[74]:
```
Points    0.526258
Score     0.963048
Weigh     1.758801
dtype: float64
```

In [75]:
```python
df.max()-df.min()
```

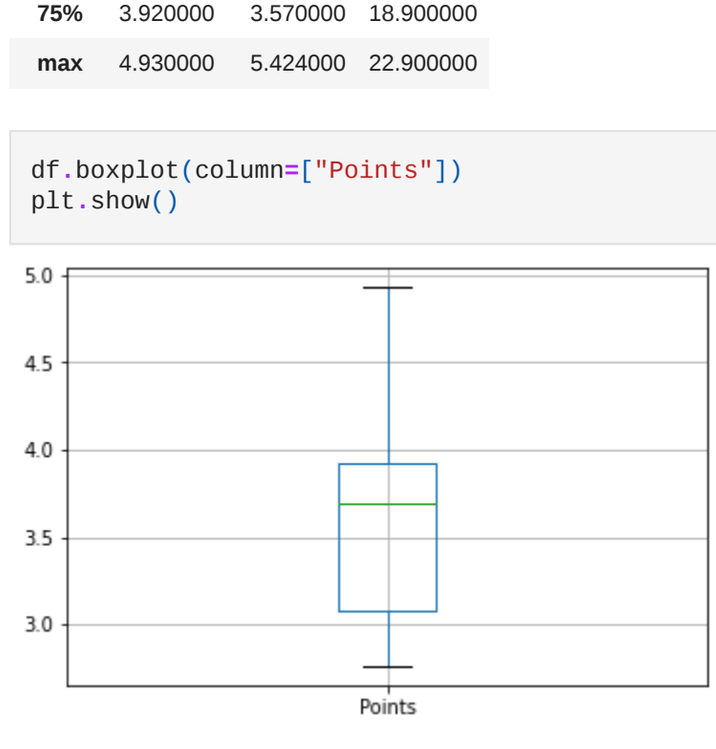
Out[75]:
```
Points    2.170
Score     3.911
Weigh     8.400
dtype: float64
```

In [76]:
```python
df.describe()
```

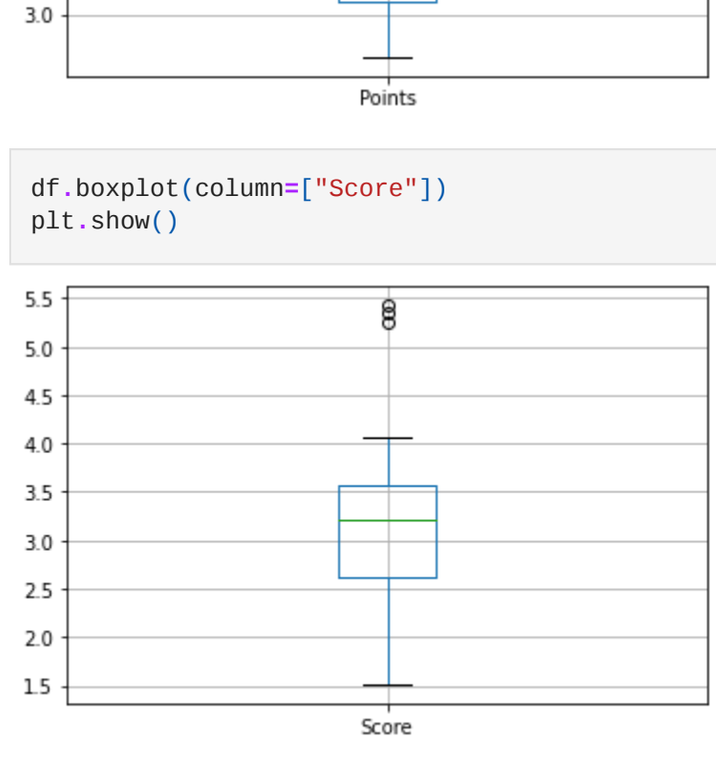
Out[76]:

	Points	Score	Weigh
count	33.000000	33.000000	33.000000
mean	3.596563	3.217250	17.848750
std	0.526258	0.963048	1.758801
min	2.760000	1.513000	14.500000
25%	3.080000	2.620000	16.900000
50%	3.690000	3.217250	17.820000
75%	3.920000	3.570000	18.900000
max	4.930000	5.424000	22.900000

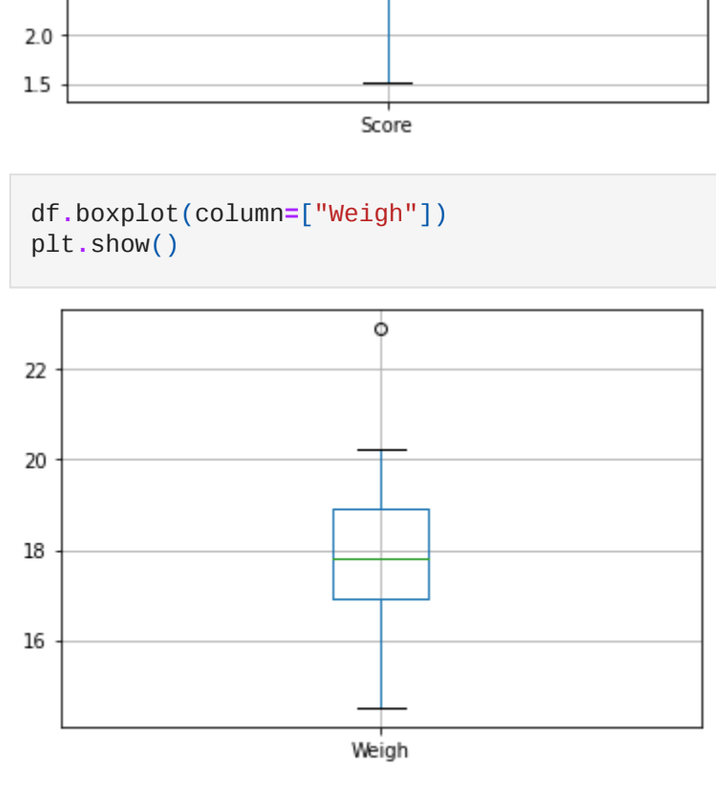
In [93]:
```python
df.boxplot(column=["Points"])
plt.show()
```



In [94]:
```python
df.boxplot(column=["Score"])
plt.show()
```



In [95]:
```python
df.boxplot(column=["Weigh"])
plt.show()
```



Q8

In [ ]:

In [1]:
```python
weights = [108, 110, 123, 134, 135, 145, 167, 187, 199]

sum_weights = sum(weights)

num_weights = len(weights)

expected_value = sum_weights / num_weights

print("Expected Value:", expected_value)
```

```
Expected Value: 145.33333333333334
```

Q9 A

In [2]:
```python
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```python
df=pd.read_csv("C:\\Users\\chann\\Downloads\\Q9_a.csv")
```

In [4]:
```python
df.head()
```

Out[4]:

	Index	speed	dist
0	1	4	2
1	2	4	10
2	3	7	4
3	4	7	22
4	5	8	16

In [5]:
```python
df.skew()
```

Out[5]:
```
Index    0.000000
speed   -0.117510
dist     0.806895
dtype: float64
```

In [6]:
```python
df.kurtosis()
```

Out[6]:
```
Index   -1.200000
speed   -0.508994
dist     0.405053
dtype: float64
```

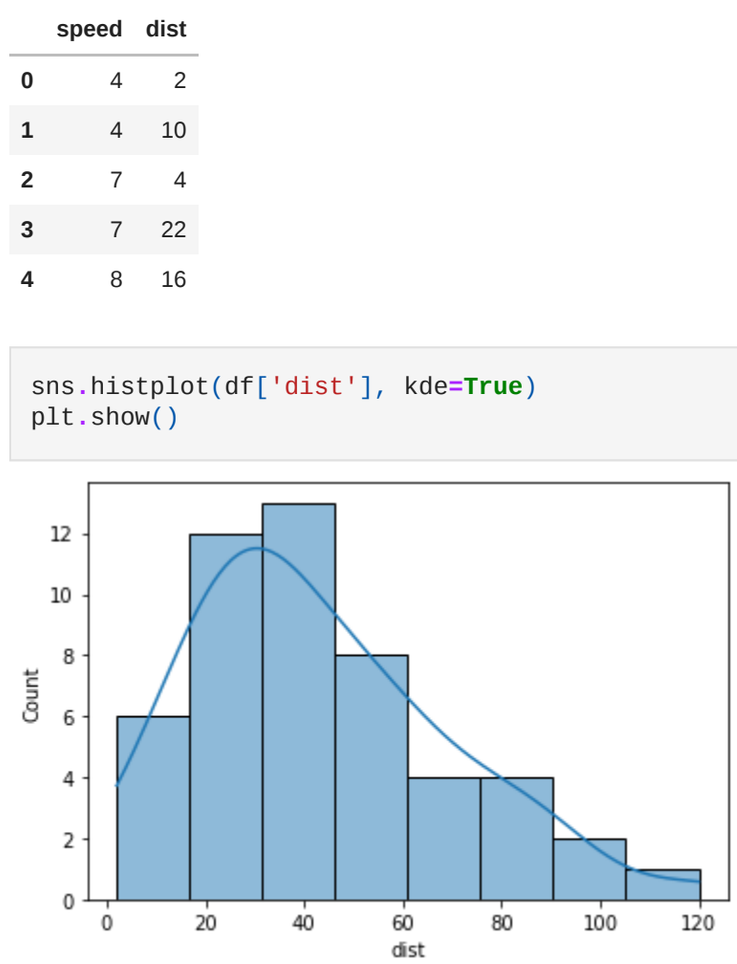
In [7]:
```python
df.drop('Index', axis=1, inplace=True)
```

In [8]:
```python
df.head()
```

Out[8]:

	speed	dist
0	4	2
1	4	10
2	7	4
3	7	22
4	8	16

In [9]:
```python
sns.histplot(df['dist'], kde=True)
plt.show()
```



In [10]:
```python
sns.histplot(df['speed'], kde=True)
plt.show()
```


Q9b

In [11]:
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

In [12]:
```python
df=pd.read_csv("C:\\Users\\chann\\Downloads\\Q9_b.csv")
```

In [13]:
```python
df.head()
```

Out[13]:

	Unnamed: 0	SP	WT
0	1	104.185353	28.762059
1	2	105.461264	30.466833
2	3	105.461264	30.193597
3	4	113.461264	30.632114
4	5	104.461264	29.889149

In [14]:
```python
df.skew()
```

Out[14]:
```
Unnamed: 0    0.000000
SP            1.611450
WT           -0.614753
dtype: float64
```

In [15]:
```python
df.kurtosis()
```

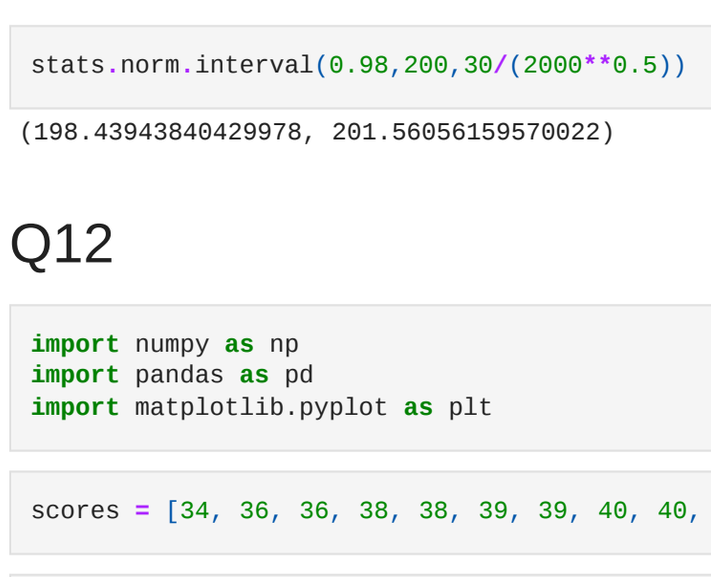
Out[15]:
```
Unnamed: 0   -1.200000
SP            2.977329
WT            0.950291
dtype: float64
```

In [16]:
```python
sns.histplot(df['SP'], kde=True)
```

Out[16]:
```
<AxesSubplot:xlabel='SP', ylabel='Count'>
```

In [17]:
```python
sns.histplot(df['WT'], kde=True)
```

Out[17]:
```
<AxesSubplot:xlabel='WT', ylabel='Count'>
```



Q11

In [18]:
```python
import numpy as np
import pandas as pd
from scipy import stats
from scipy.stats import norm
```

In [19]:
```python
stats.norm.interval(0.94,200,30/(2000**0.5))
```

Out[19]:
```
(198.738325292158, 201.261674707842)
```

In [20]:
```python
stats.norm.interval(0.96,200,30/(2000**0.5))
```

Out[20]:
```
(198.62230334813333, 201.3776966518667)
```

In [21]:
```python
stats.norm.interval(0.98,200,30/(2000**0.5))
```

Out[21]:
```
(198.43943840429978, 201.56056159570022)
```

Q12

In [22]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [23]:
```python
scores = [34, 36, 36, 38, 38, 39, 39, 40, 40, 41, 41, 41, 41, 42, 42, 45, 49, 56]
```

In [24]:
```python
mean = np.mean(scores)
mean
```

Out[24]:
```
41.0
```

In [25]:
```python
median = np.median(scores)
median
```

Out[25]:
```
40.5
```

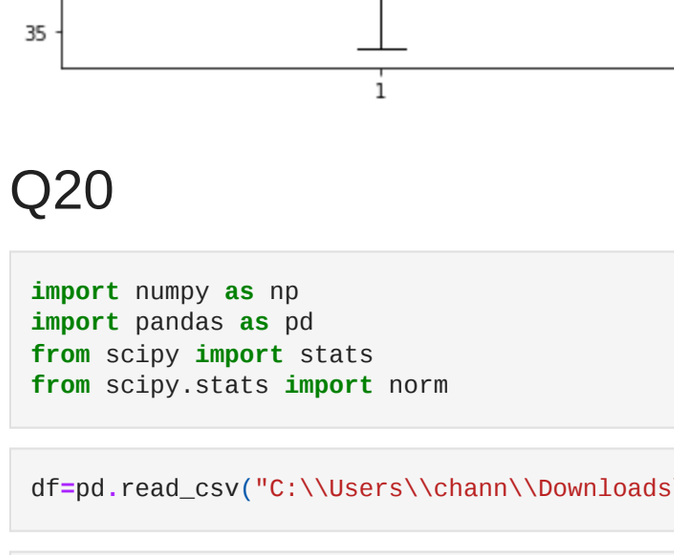
In [26]:
```python
variance = np.var(scores)
variance
```

Out[26]:
```
24.11111111111111
```

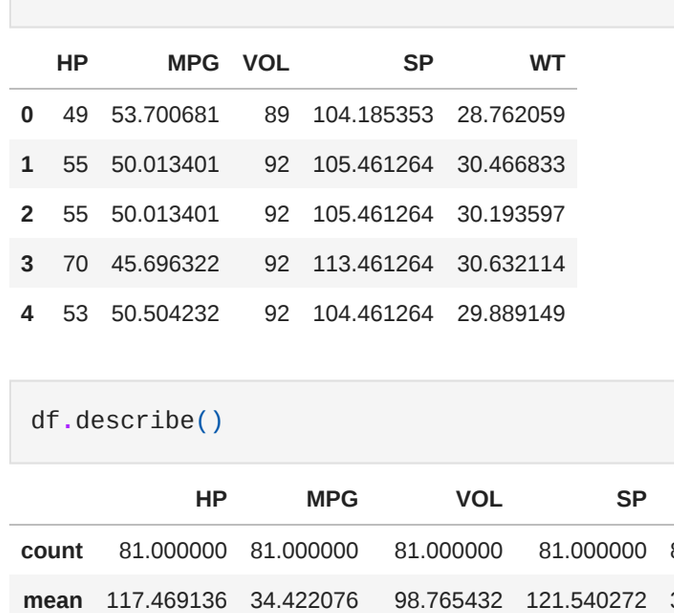
In [27]:
```python
std_dev = np.std(scores)
std_dev
```

Out[27]:
```
4.910306620885412
```

In [30]:
```python
plt.hist(scores,edgecolor="black")
plt.show()
```



In [31]:
```python
plt.boxplot(scores)
plt.show()
```



Q20

In [32]:
```python
import numpy as np
import pandas as pd
from scipy import stats
from scipy.stats import norm
```

In [33]:
```python
df=pd.read_csv("C:\\Users\\chann\\Downloads\\Cars.csv")
```

In [34]:
```python
df.head()
```

Out[34]:

	HP	MPG	VOL	SP	WT
0	49	53.700681	89	104.185353	28.762059
1	55	50.013401	92	105.461264	30.466833
2	55	50.013401	92	105.461264	30.193597
3	70	45.696322	92	113.461264	30.632114
4	53	50.504232	92	104.461264	29.889149

In [35]:
```python
df.describe()
```

Out[35]:

	HP	MPG	VOL	SP	WT
count	81.000000	81.000000	81.000000	81.000000	81.000000
mean	117.469136	34.422076	98.765432	121.540272	32.412577
std	57.113502	9.131445	22.301497	14.181432	7.492813
min	49.000000	12.101263	50.000000	99.564907	15.712859
25%	84.000000	27.856252	89.000000	113.829145	29.591768
50%	100.000000	35.152727	101.000000	118.208698	32.734518
75%	140.000000	39.531633	113.000000	126.404312	37.392524
max	322.000000	53.700681	160.000000	169.598513	52.997752

In [36]:
```python
prob_MPG_grater_then_38 = np.round(1 - stats.norm.cdf(38, loc = df.MPG.mean(), scale = df.MPG.std()),3)
print("p(MPG>38)=",prob_MPG_grater_then_38)
```

```
p(MPG>38)= 0.348
```

In [37]:
```python
prob_MPG_grater_then_40 = np.round(stats.norm.cdf(40, loc = df.MPG.mean(), scale = df.MPG.std()),3)
print("p(MPG<40)=",prob_MPG_grater_then_40)
```

```
p(MPG<40)= 0.729
```

In [40]:
```python
prob_MPG_grater_then_20 = np.round(stats.norm.cdf(20, loc = df.MPG.mean(), scale = df.MPG.std()),3)
print("p(MPG>20)=",prob_MPG_grater_then_20)
```

```
p(MPG>20)= 0.057
```

In [41]:
```python
prob_MPG_less_than_50 = np.round(stats.norm.cdf(50, loc=df["MPG"].mean(), scale=df["MPG"].std()), 3)
print("p(MPG>50)=",prob_MPG_less_than_50)
```

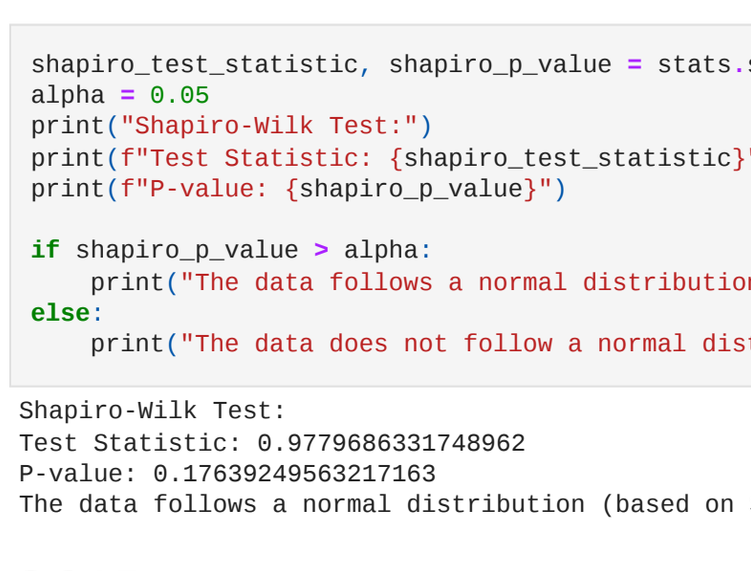
```
p(MPG>50)= 0.956
```

In [42]:
```python
prob_MPG_grater_then_20_and_prob_MPG_less_then_50 = (prob_MPG_less_than_50) - (prob_MPG_grate_then_20 )
print("p(20<MPG<50)=",(prob_MPG_grater_then_20_and_prob_MPG_less_then_50 ))
```

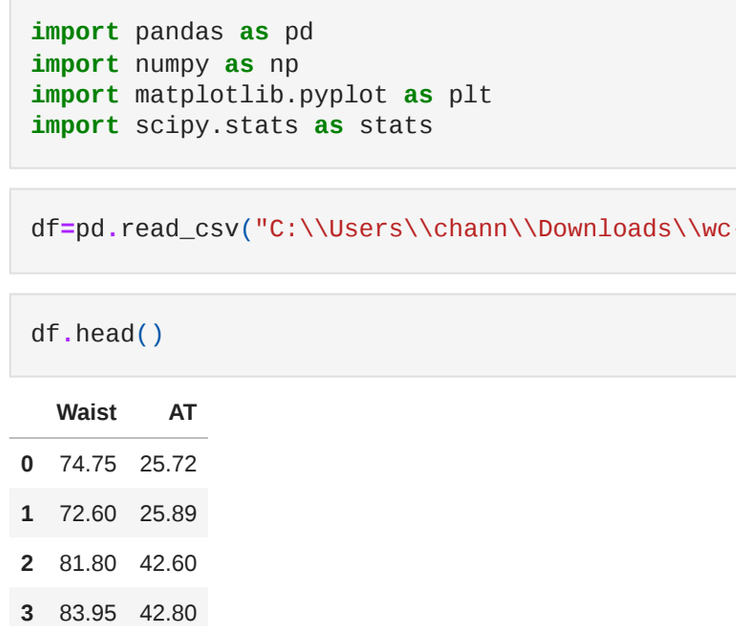
```
p(20<MPG<50)= 0.8989999999999999
```

Q21A

In [43]:
```python
mpg_data = df["MPG"]
plt.hist(mpg_data, bins=20, edgecolor="black", density=True)
plt.xlabel("MPG")
plt.ylabel("Frequency")
plt.title("Histogram of MPG")
plt.show()
```



In [44]:
```python
stats.probplot(mpg_data, plot=plt)
plt.title("Q-Q Plot of MPG")
plt.show()
```



In [45]:
```python
shapiro_test_statistic, shapiro_p_value = stats.shapiro(mpg_data)
alpha = 0.05
print("Shapiro-Wilk Test:")
print(f"Test Statistic: {shapiro_test_statistic}")
print(f"P-value: {shapiro_p_value}")

if shapiro_p_value > alpha:
    print("The data follows a normal distribution (based on Shapiro-Wilk test).")
else:
    print("The data does not follow a normal distribution (based on Shapiro-Wilk test).")
```

```
Shapiro-Wilk Test:
Test Statistic: 0.9779606331748962
P-value: 0.17639249563217163
The data follows a normal distribution (based on Shapiro-Wilk test).
```

Q21B

In [28]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
```

In [29]:
```python
df=pd.read_csv("C:\\Users\\chann\\Downloads\\wc-at.csv")
```

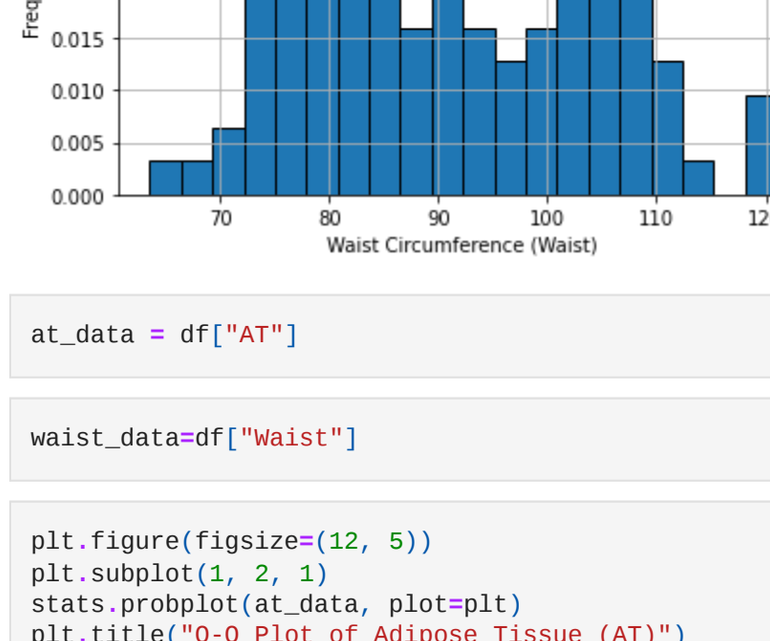
In [30]:
```python
df.head()
```

Out[30]:

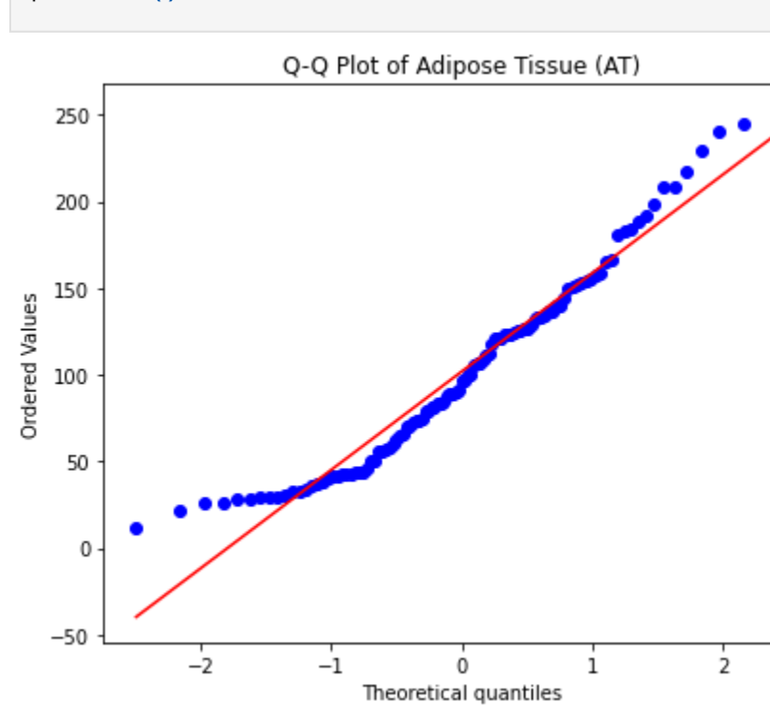
	Waist	AT
0	74.75	25.72
1	72.60	25.89
2	81.80	42.60
3	83.95	42.80
4	74.65	29.84

In [40]:
```python
plt.figure(figsize=(12, 5))
df.hist(column=["AT"],edgecolor="black", bins=20, density=True)
plt.xlabel("Adipose Tissue (AT)")
plt.ylabel("Frequency")
plt.title("Histogram of Adipose Tissue (AT)")

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

```
<Figure size 864x360 with 0 Axes>
```



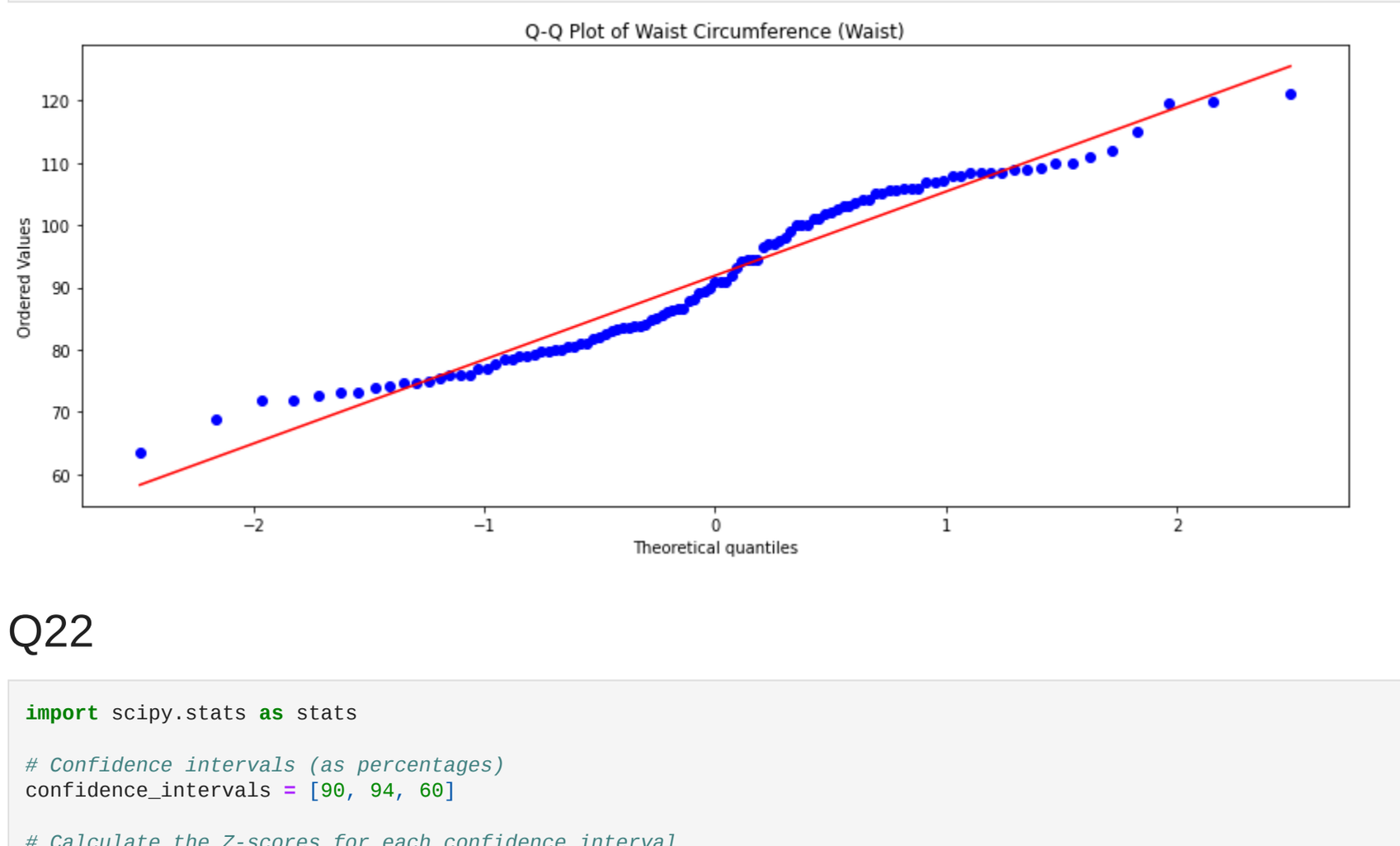
In [32]:
```python
df.hist(column="Waist"],edgecolor="black",bins=20, density=True)
plt.xlabel("Waist Circumference (Waist)")
plt.ylabel("Frequency")
plt.title("Histogram of Waist Circumference (Waist)")

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



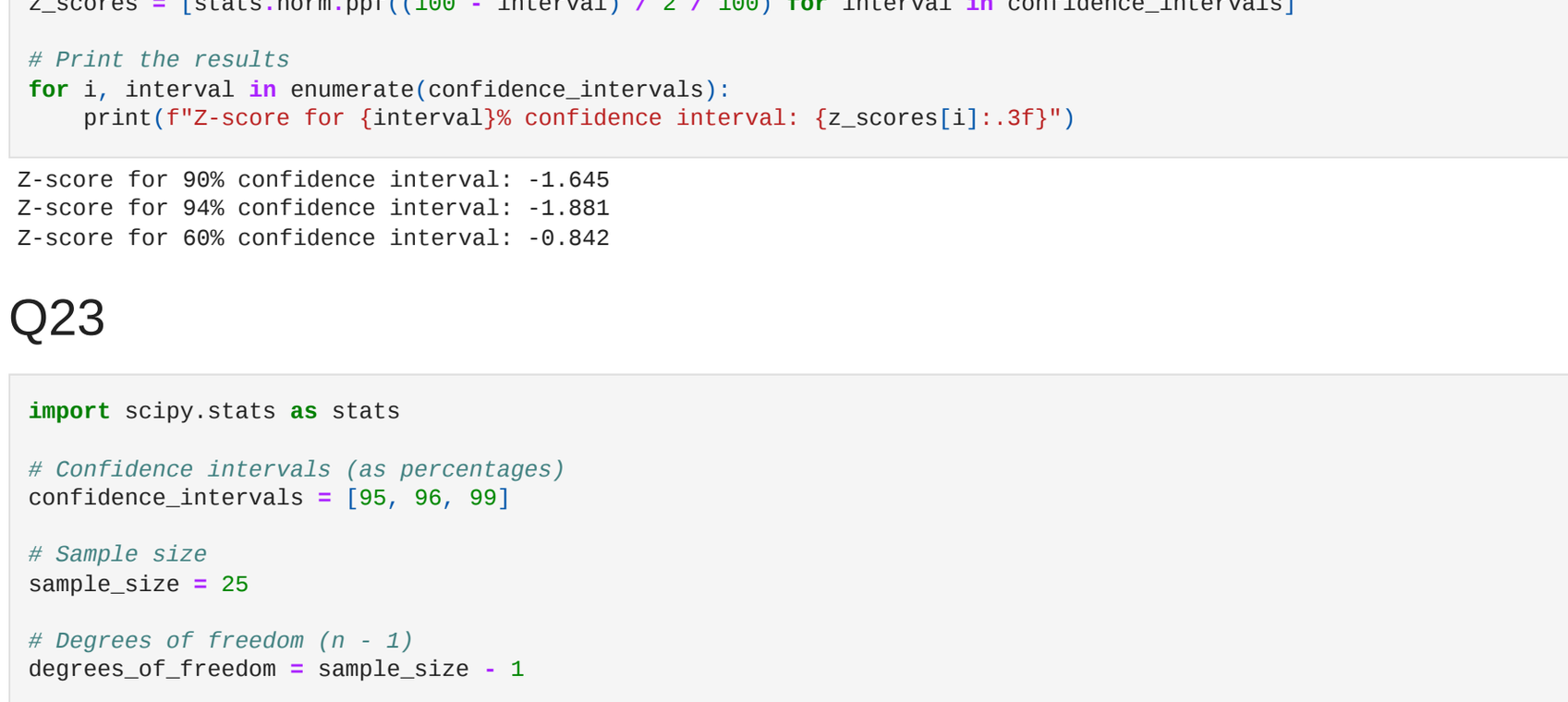
In [41]:
```python
at_data = df["AT"]
```

In [42]:
```python
waist_data=df["Waist"]
```

In [46]:
```python
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
stats.probplot(at_data, plot=plt)
plt.title("Q-Q Plot of Adipose Tissue (AT)")

plt.subplot(1, 2, 2)
stats.probplot(waist_data, plot=plt)
plt.title("Q-Q Plot of Waist Circumference (Waist)")

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



In [45]:



Q22

In [48]:
```python
import scipy.stats as stats

# Confidence intervals (as percentages)
confidence_intervals = [90, 94, 60]

# Calculate the Z-scores for each confidence interval
z_scores = [stats.norm.ppf((100 - interval) / 2 / 100) for interval in confidence_intervals]

# Print the results
for i, interval in enumerate(confidence_intervals):
    print(f"Z-score for {interval}% confidence interval: {z_scores[i]:.3f}")
```

```
Z-score for 90% confidence interval: -1.645
Z-score for 94% confidence interval: -1.881
Z-score for 60% confidence interval: -0.842
```

Q23

In [50]:
```python
import scipy.stats as stats

# Confidence intervals (as percentages)
confidence_intervals = [95, 96, 99]

# Sample size
sample_size = 25

# Degrees of freedom (n - 1)
degrees_of_freedom = sample_size - 1

# Calculate the t-scores for each confidence interval
t_scores = [stats.t.ppf((100 + interval) / 2 / 100, df=degrees_of_freedom) for interval in confidence_intervals]

# Print the results
for i, interval in enumerate(confidence_intervals):
    print(f"t-score for {interval}% confidence interval: {t_scores[i]:.3f}")
```

```
t-score for 95% confidence interval: 2.064
t-score for 96% confidence interval: 2.172
t-score for 99% confidence interval: 2.797
```

Q24

In [53]:
```python
import numpy as np
import scipy.stats as stats
```

In [51]:
```python
x_bar = 260
pop_mean = 270
```

In [54]:
```python
t_value = (260-270)/(90/np.sqrt(18))
t_value
```

Out[54]:
```
-0.4714045207910317
```

In [55]:
```python
1-stats.t.cdf(abs(t_value),df = 17)
```

Out[55]:
```
0.32167253567098353
```

In [ ]: