ADVANCED STATISTICS

Q1: What is the difference between a t-test and a z-test? Provide an example scenario where you would use each type of test.

Ans: 1

Both t-tests and z-tests are used to compare means of two groups and determine if the observed difference between the groups is statistically significant.

The difference between a t-test and a z-test:

- **Sample Size:** The key difference between a t-test and a z-test is the sample size. A t-test is appropriate when the sample size is relatively small (typically n < 30) or when the population standard deviation is unknown. On the other hand, a z-test is used when the sample size is large (typically n >= 30) or when the population standard deviation is known.
- **Assumption:** The t-test assumes that the data is approximately normally distributed, whereas the z-test assumes that the data is normally distributed.

.

Now, let's look at an example for each test using Python:

t-test:

Suppose we want to compare the heights of two groups of individuals: Group A and Group B. We have the heights of 10 individuals from each group, and we want to determine if there is a statistically significant difference in the average height between the two groups.

```
import numpy as np
from scipy.stats import ttest_ind

 # Heights of individuals in group A ans group B
group_a_heights = np.array([165, 170, 168, 175, 172, 167, 169, 171,
173, 168])
group_b_heights = np.array([160, 162, 165, 158, 163, 166, 159, 161,
164, 162])

# perform independent t test
```

```
t_stat, p_value = ttest_ind(group_a_heights, group_b_heights)

# print the result
print("t-statisti:c" , t_stat)
print("p-value:" , p_value)

# check the difference is statistically significant at 5% significance
level
if p_value < 0.05:
        print("There is a statistically significant difference in the
average height between Group A and Group B.")
else:
        print("There is no statistically significant difference in the
average height between Group A and Group B")
```

```
t-statisti:c 6.218479840396996
p-value: 7.226808844213945e-06
There is a statistically significant difference in the average height
between Group A and Group B.
```

z-test:

Now, suppose we have a larger sample size for each group and we know the population standard deviation. We want to compare the average test scores of two groups: Group X and Group Y.

```
from scipy.stats import norm # test scores of individuals in group x and

group y
group_x_scores = np.array([75, 80, 85, 90, 78, 82, 88, 92, 79, 81])
group_y_scores = np.array([70, 72, 74, 68, 73, 71, 75, 76, 72, 74])

# population standard deviation of test scores(for this example)
population_std = 5.0
# calculate and sample mean and standard errors
mean_x = np.mean(group_x_scores)
mean_y = np.mean(group_y_scores)

se_x = population_std/ np.sqrt(len(group_x_scores))
se_y = population_std / np.sqrt(len(group_y_scores))
# calculate the z-score for the difference in means
z_score = (mean_x - mean_y ) / np.sqrt(se_x**2 + se_y**2)
# calculate the p-value
p_value = 2*(1 - norm.cdf(abs(z_score)))
```

```
# print the result print("z-score:", z_score) print("p-value:", p_value)

# # Check if the difference is statistically significant at 5%
significance level
if p_value < 0.05:


      print("There is a statistically significant difference in the
average test scores between Group X and Group Y.")
else:
      print("There is no statistically significant difference in the
average test scores between Group X and Group Y.")
z-score: 4.695742752749559
p-value: 2.656396824285423e-06
There is a statistically significant difference in the average test
scores between Group X and Group Y.
```

Q2: Differentiate between one-tailed and two-tailed tests.

Ans: 2

One-tailed and two-tailed tests are two types of hypothesis tests used in statistical analysis to assess the significance of an observed effect or difference. The main difference between these tests lies in the directionality of the hypothesis being tested.

One-tailed test:

In a one-tailed test, the hypothesis being tested is directional, meaning it specifies a particular direction of the effect or difference. The test is designed to determine if the observed data significantly deviates in one specific direction from the null hypothesis. The critical region is located entirely on one side of the probability distribution.

- Null hypothesis (H0): There is no effect or difference between the groups.
- Alternative hypothesis (Ha): There is a specific effect or difference in a particular direction.

One-tailed tests are typically used when there is a strong prior expectation or theoretical reason to believe that the effect or difference, if it exists, will occur in a specific direction.

.

Two-tailed test:

In a two-tailed test, the hypothesis being tested is non-directional, meaning it does not specify a particular direction of the effect or difference. The test is designed to determine if the observed data significantly deviates from the null hypothesis in any direction. The critical region is divided between both tails of the probability distribution.

- Null hypothesis (H0): There is no effect or difference between the groups.
- Alternative hypothesis (Ha): There is a significant effect or difference, but the direction is not specified.

Two-tailed tests are more conservative because they consider deviations in both directions and are typically used when there is no prior expectation or theoretical reason to predict the direction of the effect or difference.

Q3: Explain the concept of Type 1 and Type 2 errors in hypothesis testing. Provide an example scenario for each type of error.

Ans: 3

Type 1 error (False Positive):

A Type 1 error occurs when we incorrectly reject a true null hypothesis. In other words, we conclude that there is a significant effect or difference when, in reality, there is no such effect or difference in the population. The probability of making a Type 1 error is denoted by the symbol alpha (α) and is also known as the significance level of the test.

- Example: Let's say a company claims their new energy drink improves people's memory. They run a scientific test and find a small improvement in memory among the participants who drank the energy drink. However, in reality, the drink has no effect on memory. If the company incorrectly concludes that the drink works and starts advertising it as a memory booster, it's a Type 1 error.

Type 2 error (False Negative):

A Type 2 error occurs when we incorrectly fail to reject a false null hypothesis. In this case, we conclude that there is no significant effect or difference when, in fact, there is a true effect or difference in the population. The probability of making a Type 2 error is denoted by the symbol beta (β).

- Example: Let's consider the same company testing the memory-boosting energy drink. Suppose the drink actually does improve memory, but in their experiment, they fail to detect this improvement and mistakenly conclude that it has no effect. It's a Type 2 error because they missed the true effect of the drink.

Q4: Explain Bayes's theorem with an example.

Ans: 4

Bayes's theorem is a fundamental concept in probability theory that allows us to update the probability of an event based on new evidence. It helps us calculate the probability of a hypothesis (or event) given the probability of related evidence and the prior probability of the hypothesis. Bayes's theorem is represented mathematically as:

P(H/E)= P(E/H).P(H) / P(E)

Where :

- P(H/E) is the posterior probability of the hypothesis H given the evidence E.
- P(E/H) is the likelihood of the evidence E given the hypothesis H.
- P(H) is the prior probability of the hypothesis H.
- P(E) is the probability of the evidence E.

Let's demonstrate Bayes's theorem with a simple example in Python:

Example: Coin Toss Suppose we have an unfair coin, and we want to calculate the probability of the coin being biased towards heads (H) or tails (T) based on the evidence of three consecutive tosses: H, H, T.

Now, let's use Bayes's theorem to calculate the posterior probability of the coin being biased towards heads (H) after observing three consecutive tosses: H, H, T.

```
# Prior probabilities

p_h = 0.4      # Probability of the coin being biased towards heads #
(prior)        Probability of the coin being biased towards tails
p_t = 0.6
(prior)

# Likelihoods
p_h_given_h = 0.9      # Probability of getting heads given the coin is
biased towards heads
p_t_given_h = 0.1      # Probability of getting tails given the coin is
biased towards heads
p_h_given_t = 0.3      # Probability of getting heads given the coin is
biased towards tails
p_t_given_t = 0.7      # Probability of getting tails given the coin is
biased towards tails

# Evidence (observed tosses)
```

```
evidence = ['H', 'H', 'T']

# Calculate the probability of the evidence P(E)
p_e = (p_h * p_h_given_h * p_h_given_h * p_t_given_h) + (p_t *
p_h_given_t * p_h_given_t * p_t_given_t)

# Calculate the posterior probabilities P(H|E) and P(T|E)
p_h_given_e = (p_h * p_h_given_h * p_h_given_h * p_t_given_h) / p_e
p_t_given_e = (p_t * p_h_given_t * p_h_given_t * p_t_given_t) / p_e

print("Posterior probability of the coin being biased towards heads
(H):", p_h_given_e)
print("Posterior probability of the coin being biased towards tails
(T):", p_t_given_e)
Posterior probability of the coin being biased towards heads (H):
0.4615384615384615
Posterior probability of the coin being biased towards tails (T):
0.5384615384615383
```

Q5: What is a confidence interval? How to calculate the confidence interval, explain with an example.

Ans: 5

Confidence interval:
A confidence interval is a range of values that provides an estimate of the unknown population parameter (such as the population mean) along with a level of confidence. It is a measure of the uncertainty associated with an estimate based on a sample from the population. In other words, a confidence interval gives us a range of values within which we are reasonably confident that the true population parameter lies.

The level of confidence associated with a confidence interval is expressed as a percentage, typically 90%, 95%, or 99%. For example, a 95% confidence interval means that if we were to take many random samples from the same population and compute a confidence interval from each sample, approximately 95% of those intervals would contain the true population parameter.

How to calculate the confidence interval: The formula for calculating a confidence interval depends on the type of data and the distribution of the data. For large sample sizes, a common approach is to use the normal distribution, while for smaller sample sizes, the t-distribution is used.

- Example: Calculating a Confidence Interval for the Mean

Suppose we want to estimate the average height of a certain population. We take a random sample of 50 individuals from that population and measure their heights. The sample mean height is 170 cm, and the sample standard deviation is 5 cm.

```
from scipy.stats import t # Sample data (heights in centimeters)
sample_data = np.array([165, 170, 175, 168, 172, 173, 169, 171, 168,
172,

169,

171,                                    170, 167, 170, 171, 174, 172, 170, 168, 172, 170, 172, 170,
                                        168, 170, 172, 170, 169, 173, 167, 170, 172, 169, 171, 170,
                                        168, 173, 170, 171, 173, 170, 168, 174, 171, 170, 172, 169,

169,

172])

# Sample statistics
sample_mean = np.mean(sample_data)
sample_std = np.std(sample_data, ddof=1)
deviation
sample_size = len(sample_data)
                                                     # ddof=1 for sample standard
# Confidence level (as a decimal)
confidence_level = 0.95
# Calculate the critical value for t-distribution
# Degrees of freedom = sample_size - 1
critical_value = t.ppf((1 + confidence_level) / 2, df=sample_size - 1)

# Calculate the standard error of the mean
standard_error = sample_std / np.sqrt(sample_size)
# Calculate the margin of error
margin_of_error = critical_value * standard_error
# Calculate the confidence interval
lower_bound = sample_mean - margin_of_error
upper_bound = sample_mean + margin_of_error

print("Sample mean:", sample_mean)
print("Margin of error:", margin_of_error)
print("95% Confidence Interval: ({:.2f}, {:.2f})".format(lower_bound,
upper_bound))
```

```
Sample mean: 170.4
Margin of error: 0.5741643527112024
95% Confidence Interval: (169.83, 170.97)
```

Q6. Use Bayes' Theorem to calculate the probability of an event occurring given prior knowledge of the event's probability and new evidence. Provide a sample problem and solution.

Ans : 6

Sample Problem: Suppose there is a rare disease that affects 1 in 10,000 people in a certain population. We have a diagnostic test for this disease, but it is not perfect. The test has the following characteristics:

The probability of a positive test result (indicating the presence of the disease) given that a person has the disease is 0.98 (P(Positive|Disease) = 0.98). The probability of a negative test result (indicating the absence of the disease) given that a person does not have the disease is 0.99 (P(Negative|No Disease) = 0.99). Now, a person from the population gets tested, and the test result is positive. We want to calculate the probability that this person actually has the disease (P(Disease|Positive)).

Solution: Let's use Bayes' Theorem to calculate the probability of having the disease given a positive test result (P(Disease|Positive)).

Bayes' Theorem is given by:

P(Disease/Positive) = P(Positive/Disease) · P(Disease) / P(positive)

Where:

- P(Disease/Positive) is the probability of having the disease given a positive test result.
- P(Positive/Disease) is the probability of a positive test result given that the person has the disease (0.98).
- P(Disease) is the prior probability of having the disease (1 in 10,000 or 0.0001).
- P(positive) is the probability of a positive test result.

Let's calculate P(positive) and then use Bayes' Theorem to find P(Disease/Positive) in Python:

```
# Given data
p_positive_given_disease = 0.98
p_disease = 0.0001
p_positive_given_no_disease = 1 - 0.99
p_no_disease = 1 - p_disease

# Calculate P(Positive)
p_positive = (p_positive_given_disease * p_disease) +
(p_positive_given_no_disease * p_no_disease)

# Calculate P(Disease|Positive) using Bayes' Theorem
p_disease_given_positive = (p_positive_given_disease * p_disease) /
p_positive

# Display the result
print("Probability of having the disease given a positive test
result:" )
print("{:.6f}".format(p_disease_given_positive))
Probability of having the disease given a positive test result:
0.009706
```

Q7. Calculate the 95% confidence interval for a sample of data with a mean of 50 and a standard deviation of 5. Interpret the results.

Ans: 7

To calculate the 95% confidence interval for a sample of data with a mean of 50 and a standard deviation of 5, we need to use the formula for a confidence interval for the population mean

Let's calculate the 95% confidence interval using the given information (mean = 50, standard deviation = 5) and assume a sample size of 30:

```
import scipy.stats as st

# Given data
sample_mean = 50
sample_std = 5
confidence_level = 0.95
sample_size = 30
```

```
# Calculate the critical value (Z-score) for 95% confidence level
critical_value = st.norm.ppf((1 + confidence_level) / 2)

# Calculate the margin of error
margin_of_error = critical_value * (sample_std / (sample_size ** 0.5))
# Calculate the confidence interval
lower_bound = sample_mean - margin_of_error
upper_bound = sample_mean + margin_of_error

# Display the result
print("95% Confidence Interval: ({:.2f}, {:.2f})".format(lower_bound,
upper_bound))
```

```
95% Confidence Interval: (48.21, 51.79)
```

Interpretation of the Results:
The 95% confidence interval for the population mean(µ) based on the sample data is (48.51, 51.49). This means that we are 95% confident that the true population mean falls within this range.

Q8. What is the margin of error in a confidence interval? How does sample size affect the margin of error? Provide an example of a scenario where a larger sample size would result in a smaller margin of error.

Ans: 8

The margin of error (MOE) in a confidence interval (CI) is a measure of the uncertainty or precision associated with the estimate of a population parameter based on a sample. It indicates the range within which we expect the true population parameter to lie with a certain level of confidence.

In statistical terms, when we calculate a confidence interval for a population parameter (e.g., mean, proportion), we use a sample statistic (e.g., sample mean, sample proportion) to estimate the true population parameter. The margin of error is the maximum amount by which the sample statistic is likely to differ from the true population parameter.

The margin of error is directly related to the level of confidence chosen for the interval and the variability of the data in the sample. The most common level of confidence used is 95%, which means that we expect the true population parameter to lie within the calculated interval in 95 out of 100 samples.

The formula for calculating the margin of error in a confidence interval is:

MOE = Z * (σ / √n)

Where:

MOE = Margin of Error

- Z = Z-score associated with the desired level of confidence (e.g., 1.96 for a 95% confidence level)
- σ = Standard deviation of the population (unknown in most cases, so we often use the sample standard deviation as an estimate)
- n = Sample size

Now, as for how the sample size affects the margin of error, we can observe that the margin of error is inversely proportional to the square root of the sample size. In other words, as the sample size increases, the margin of error decreases.

Here's an example:

Let's say you want to estimate the average height of students at a particular university with a 95% confidence level and a margin of error of 2 inches. You collect two different sample sizes, one with 100 students and another with 400 students.

For the sample with 100 students: MOE = Z * (σ / √n) Assuming σ (population standard deviation) is 4 inches (just for illustration purposes): MOE = 1.96 * (4 / √100) = 1.96 * 0.4 ≈ 0.78 inches

For the sample with 400 students: MOE = Z * (σ / √n) MOE = 1.96 * (4 / √400) = 1.96 * 0.2 ≈ 0.39 inches

Q9. Calculate the z-score for a data point with a value of 75, a population mean of 70, and a population standard deviation of 5. Interpret the results.

```
# Ans: 9


# Given data
```

```
data_point = 75
population_mean = 70
population_std_dev = 5

# Calculate the z-score
z_score = (data_point - population_mean) / population_std_dev
# Display the z-score
print("The z-score for the data point is:", z_score)
```

```
The z-score for the data point is: 1.0
```

Now, let's interpret the results:
The calculated z-score represents the number of standard deviations the data point (75) is away from the population mean (70). In this case:

Z = (75-70)/5

A z-score of 1 means that the data point is 1 standard deviation above the mean. Since the population standard deviation is 5, a z-score of 1 indicates that the data point is 5 units above the population mean.

Q10. In a study of the effectiveness of a new weight loss drug, a sample of 50 participants lost an average of 6 pounds with a standard deviation of 2.5 pounds. Conduct a hypothesis test to determine if the drug is significantly effective at a 95% confidence level using a t-test.

Ans: 10

Let's define the hypotheses:

- Null Hypothesis (H0): The weight loss drug is not significantly effective, and the population mean weight loss is equal to or less than 0 pounds. ($\mu \leq 0$)

- Alternative Hypothesis (H1): The weight loss drug is significantly effective, and the population mean weight loss is greater than 0 pounds. ($\mu > 0$)

We will use a one-tailed t-test because the alternative hypothesis is directional ($\mu > 0$).

```
import scipy.stats as stats

sample_mean = 6
sample_std_dev = 2.5
sample_size = 50
population_mean_null = 0
# Calculate the t-statistic
standard_error = sample_std_dev / (sample_size ** 0.5)
t_statistic = (sample_mean - population_mean_null) / standard_error

# Degrees of freedom for a one-sample t-test
degrees_of_freedom = sample_size - 1
# Calculate the critical t-value for a one-tailed t-test at 95%
confidence level
alpha = 0.05
critical_t_value = stats.t.ppf(1 - alpha, df=degrees_of_freedom)
# Print the results
print("Calculated t-statistic:", t_statistic)
print("Critical t-value:", critical_t_value)
```

```
Calculated t-statistic: 16.970562748477143
Critical t-value: 1.6765508919142629
```

Q11. In a survey of 500 people, 65% reported being satisfied with their current job. Calculate the 95% confidence interval for the true proportion of people who are satisfied with their job.

Ans: 11

To calculate the 95% confidence interval for the true proportion of people who are satisfied with their job, we can use the formula for the confidence interval for a proportion.

```
import math # Given data
sample_proportion = 0.65
sample_size = 500

# Calculate the critical z-score for a 95% confidence level
confidence_level = 0.95
critical_z_score = stats.norm.ppf(1 - (1 - confidence_level) / 2)

# Calculate the standard error
standard_error = math.sqrt((sample_proportion * (1 -
sample_proportion)) / sample_size)

# Calculate the lower and upper bounds of the confidence interval
lower_bound = sample_proportion - critical_z_score * standard_error
upper_bound = sample_proportion + critical_z_score * standard_error

# Convert bounds to percentages
lower_bound_percent = lower_bound * 100
upper_bound_percent = upper_bound * 100

# Print the results
print("95% Confidence Interval for the proportion of people satisfied
with their job:")
print(f"{lower_bound_percent:.2f}% to {upper_bound_percent:.2f}%")
```

```
95% Confidence Interval for the proportion of people satisfied with
their job:
60.82% to 69.18%
```

Q12. A researcher is testing the effectiveness of two different teaching methods on student performance. Sample A has a mean score of 85 with a standard deviation of 6, while sample B has a mean score of 82 with a standard deviation of 5. Conduct a hypothesis test to determine if the two teaching methods have a significant difference in student performance using a t-test with a significance level of 0.01.

Ans: 12

To conduct a hypothesis test to determine if there is a significant difference in student performance between the two teaching methods, we can use a two-sample t-test. The null hypothesis (H0) assumes that there is no difference between the means of the two samples, while the alternative hypothesis (H1) assumes that there is a significant difference.

Let's define the hypotheses:

- Null Hypothesis (H0): The two teaching methods have no significant difference in student performance. ($\mu A - \mu B = 0$)
- Alternative Hypothesis (H1): The two teaching methods have a significant difference in student performance. ($\mu A - \mu B \neq 0$)

We will use a two-tailed t-test because the alternative hypothesis is non-directional (it doesn't specify which mean is larger).

```
import scipy.stats as stats

# Given data for sample A
sample_mean_A = 85
sample_std_dev_A = 6
sample_size_A = 30
# Given data for sample B
sample_mean_B = 82
sample_std_dev_B = 5
sample_size_B = 25
```

```
# Calculate the pooled standard deviation
pooled_std_dev = math.sqrt(((sample_std_dev_A ** 2) / sample_size_A) +
((sample_std_dev_B ** 2) / sample_size_B))

# Calculate the t-statistic
t_statistic = (sample_mean_A - sample_mean_B) / pooled_std_dev
# Degrees of freedom for a two-sample t-test
degrees_of_freedom = sample_size_A + sample_size_B - 2
# Calculate the critical t-value for a two-tailed t-test at a
significance level of 0.01
alpha = 0.01
critical_t_value = stats.t.ppf(1 - alpha / 2, df=degrees_of_freedom)
# Print the results
print("Calculated t-statistic:", t_statistic)
print("Critical t-value:", critical_t_value)
```

```
Calculated t-statistic: 2.0225995873897262
Critical t-value: 2.6718226362410027
```

let's interpret the results:
If the absolute value of the calculated t-statistic is greater than the critical t-value, we can reject the null hypothesis (H0) and conclude that there is a significant difference in student performance between the two teaching methods. Otherwise, if the calculated t-statistic falls within the range of the critical t-values, we fail to reject the null hypothesis, and we do not have sufficient evidence to claim a significant difference in performance between the two teaching methods.

Q13. A population has a mean of 60 and a standard deviation of 8. A sample of 50 observations has a mean of 65. Calculate the 90% confidence interval for the true population mean.

```
# Ans: 13

# Given data
sample_mean = 65
```

```
population_mean = 60 population_std_dev = 8 sample_size = 50

# Calculate the critical z-score for a 90% confidence level
confidence_level = 0.90
critical_z_score = stats.norm.ppf(1 - (1 - confidence_level) / 2)

# Calculate the standard error
standard_error = population_std_dev / (sample_size ** 0.5)
# Calculate the lower and upper bounds of the confidence interval
lower_bound = sample_mean - critical_z_score * standard_error
upper_bound = sample_mean + critical_z_score * standard_error

# Print the results
print("90% Confidence Interval for the true population mean:")
print(f"{lower_bound:.2f} to {upper_bound:.2f}")

90% Confidence Interval for the true population mean:
63.14 to 66.86
```

Q14. In a study of the effects of caffeine on reaction time, a sample of 30 participants had an average reaction time of 0.25 seconds with a standard deviation of 0.05 seconds. Conduct a hypothesis test to determine if the caffeine has a significant effect on reaction time at a 90% confidence level using a t-test.

Ans: 14

To conduct a hypothesis test to determine if caffeine has a significant effect on reaction time, we can use a one-sample t-test. The null hypothesis (H0) assumes that caffeine has no significant effect on reaction time, while the alternative hypothesis (H1) assumes that caffeine does have a significant effect.

Let's define the hypotheses:

- Null Hypothesis (H0): Caffeine has no significant effect on reaction time. (μ = 0)
- Alternative Hypothesis (H1): Caffeine does have a significant effect on reaction time. (μ ≠ 0)

We will use a two-tailed t-test because the alternative hypothesis is non-directional

```
# Given data
sample_mean = 0.25
population_mean_null = 0
sample_std_dev = 0.05
sample_size = 30

# Calculate the standard error
standard_error = sample_std_dev / (sample_size ** 0.5)
# Calculate the t-statistic
t_statistic = (sample_mean - population_mean_null) / standard_error
# Degrees of freedom for a one-sample t-test
degrees_of_freedom = sample_size - 1
# Calculate the critical t-value for a two-tailed t-test at a 90%
confidence level
confidence_level = 0.90
critical_t_value = stats.t.ppf(1 - (1 - confidence_level) / 2,
df=degrees_of_freedom)

# Print the results
print("Calculated t-statistic:", t_statistic)
print("Critical t-value:", critical_t_value)
```

```
Calculated t-statistic: 27.386127875258307
Critical t-value: 1.6991270265334972
```

let's interpret the results:

If the absolute value of the calculated t-statistic is greater than the critical t-value, we can reject the null hypothesis (H0) and conclude that caffeine has a significant effect on reaction time at the 90% confidence level. Otherwise, if the calculated t-statistic falls within the range of the critical t-values, we fail to reject the null hypothesis, and we do not have sufficient evidence to claim a significant effect of caffeine on reaction time.

Q15. Calculate the 95% confidence interval for a sample of data with a mean of 50 and a standard deviation of 5 using Python. Interpret the results.

Ans: 15

To calculate the 95% confidence interval for a sample of data with a mean of 50 and a standard deviation of 5, we can use the formula for the confidence interval for a population mean. Since we are dealing with a sample, we'll use a t-distribution instead of a z-distribution, as we don't know the population standard deviation.

```python
import scipy.stats as stats

# Given data
sample_mean = 50
sample_std_dev = 5
sample_size = 30 # Assuming a sample size of 30 for demonstration
# Degrees of freedom for a t-distribution (sample size - 1)
degrees_of_freedom = sample_size - 1
# Calculate the critical t-value for a 95% confidence level and the
given degrees of freedom
confidence_level = 0.95
critical_t_value = stats.t.ppf(1 - (1 - confidence_level) / 2,
df=degrees_of_freedom)

# Calculate the standard error
standard_error = sample_std_dev / (sample_size ** 0.5)
# Calculate the lower and upper bounds of the confidence interval
lower_bound = sample_mean - critical_t_value * standard_error
upper_bound = sample_mean + critical_t_value * standard_error

# Print the results
print("95% Confidence Interval:")
print(f"{lower_bound:.2f} to {upper_bound:.2f}")
```

```
95% Confidence Interval:
48.13 to 51.87
```

let's interpret the results:

The 95% confidence interval for the true population mean is approximately (47.52 to 52.48). This means that we are 95% confident that the true population mean lies within this range based on the sample data. In other words, if we were to take multiple samples from the same population and calculate the 95% confidence interval for each sample, we would expect the true population mean to be contained in 95% of those intervals.

Q16. Conduct a chi-square goodness of fit test to determine if the distribution of colors of M&Ms in a bag matches the expected distribution of 20% blue, 20% orange, 20% green, 10% yellow, 10% red, and 20% brown. Use Python to perform the test with a significance level of 0.05.

```
import numpy as np
import scipy.stats as stats

expected_frequencies = [0.2*90, 0.2*90, 0.2*90, 0.1*90, 0.1*90,

0.2*90]      # Blue, Orange, Green, Yellow, Red, Brown
observed_frequencies = [22, 18, 20, 11, 13, 16]                  # Blue, Orange,
Green, Yellow, Red, Brown


# Observed frequencies
observed_frequencies = np.array([22, 18, 20, 11, 13, 16])
# Expected frequencies based on the expected distribution
total_observed = sum(observed_frequencies)
expected_frequencies = np.array([int(0.2 * total_observed)] * 6)

# Set the threshold for the maximum allowable percent difference
threshold_percent_difference = 0.001          # You can adjust this value
based on your needs

# Calculate the percent difference between the sums of observed and
expected frequencies
percent_difference = abs(sum(observed_frequencies) -
sum(expected_frequencies)) / sum(expected_frequencies)
# Check if the percent difference is below the threshold
```

```
if percent_difference <= threshold_percent_difference:
	# Perform the chi-square goodness of fit test
	chi2_stat, p_value = stats.chisquare(f_obs=observed_frequencies,
f_exp=expected_frequencies)

	# Define the significance level
	alpha = 0.05
	# Print the results
	print("Chi-square test statistic:", chi2_stat)
	print("P-value:", p_value)

	# Check if the p-value is less than the significance level and
make the conclusion
	if p_value < alpha:
		print("Reject the null hypothesis. The distribution of colors
in the bag does not match the expected distribution.")
	else:
		print("Fail to reject the null hypothesis. The distribution of
colors in the bag matches the expected distribution.")
else:
	print("The discrepancy between observed and expected frequencies
is significant. Please review the data and calculations.")
```

```
The discrepancy between observed and expected frequencies is
significant. Please review the data and calculations.
```

Q17. Use Python to calculate the chi-square statistic and p-value for a contingency table with the following data:

```
      Group A ,Group B
```

Outcome1 20 , 15

Outcome2 10 , 25

Outcome3 15 , 20

Interpret the results of the test.

```
# Ans: 17
# Create the contingency table
contingency_table = np.array([[20, 15], [10, 25], [15, 20]])

# Perform the chi-square test
chi2_stat, p_value, dof, expected =
stats.chi2_contingency(contingency_table)

# Define the significance level
alpha = 0.05
# Print the results
print("Chi-square test statistic:", chi2_stat)
print("P-value:", p_value)
print("Degrees of freedom:", dof)
print("Expected frequencies (contingency table):")
print (expected)
# Check if the p-value is less than the significance level and make
the conclusion
if p_value < alpha:

    print("Reject the null hypothesis. There is a significant
association between the groups and outcomes.")
else:
    print("Fail to reject the null hypothesis. There is no significant
association between the groups and outcomes.")
```

```
Chi-square test statistic: 5.833333333333334
P-value: 0.05411376622282158
Degrees of freedom: 2
Expected frequencies (contingency table):
[[15. 20.]
 [15. 20.]
 [15. 20.]]
Fail to reject the null hypothesis. There is no significant
association between the groups and outcomes.
```

Let's interpret the result of the test.
The interpretation of the results is based on the p-value. If the p-value is less than the chosen significance level (0.05 in this case), we reject the null hypothesis and conclude that there is a significant association between the groups and outcomes. Otherwise, if the p-value is greater than or equal to the significance level, we fail to reject the null hypothesis, and we do not have sufficient evidence to claim a significant association.

Q18. A study of the prevalence of smoking in a population of 500 individuals found that 60 individuals smoked. Use Python to calculate the 95% confidence interval for the true proportion of individuals in the population who smoke.

Ans: 18

To calculate the 95% confidence interval for the true proportion of individuals in the population who smoke, we can use the formula for the confidence interval for a population proportion.

```
# Given data sample_proportion = 60 / 500 sample_size = 500

# Calculate the critical z-value for a 95% confidence level
confidence_level = 0.95
critical_z_value = stats.norm.ppf(1 - (1 - confidence_level) / 2)

# Calculate the standard error
standard_error = (sample_proportion * (1 - sample_proportion) /
sample_size) ** 0.5

# Calculate the lower and upper bounds of the confidence interval
lower_bound = sample_proportion - critical_z_value * standard_error
upper_bound = sample_proportion + critical_z_value * standard_error

# Print the results
print("95% Confidence Interval:")
print(f"{lower_bound:.4f} to {upper_bound:.4f}")
```

```
95% Confidence Interval:
0.0915 to 0.1485
```

let's interpret the results:

The 95% confidence interval for the true proportion of individuals in the population who smoke is approximately (0.0968 to 0.1432). This means that we are 95% confident that the true proportion of smokers in the population lies within this range based on the sample data. In other words, if we were to take multiple samples from the same population and calculate the 95% confidence interval for each sample, we would expect the true proportion of smokers to be contained in 95% of those intervals.

Q19. Calculate the 90% confidence interval for a sample of data with a mean of 75 and a standard deviation of 12 using Python. Interpret the results.

Ans: 19

To calculate the 90% confidence interval for a sample of data with a mean of 75 and a standard deviation of 12, we can use the t-distribution since the sample size is small and the population standard deviation is unknown.

```
# Given data sample_mean = 75 sample_std_dev = 12 sample_size = 30 # Adjust
the sample size as needed

# Calculate the critical t-value for a 90% confidence level and
degrees of freedom
confidence_level = 0.9
degrees_of_freedom = sample_size - 1
critical_t_value = stats.t.ppf(1 - (1 - confidence_level) / 2,
df=degrees_of_freedom)
# Calculate the standard error
standard_error = sample_std_dev / (sample_size ** 0.5)
# Calculate the lower and upper bounds of the confidence interval
lower_bound = sample_mean - critical_t_value * standard_error
upper_bound = sample_mean + critical_t_value * standard_error

# Print the results
print("90% Confidence Interval:")
print(f"{lower_bound:.2f} to {upper_bound:.2f}")
```

```
90% Confidence Interval:
71.28 to 78.72
```

let's interpret the results:

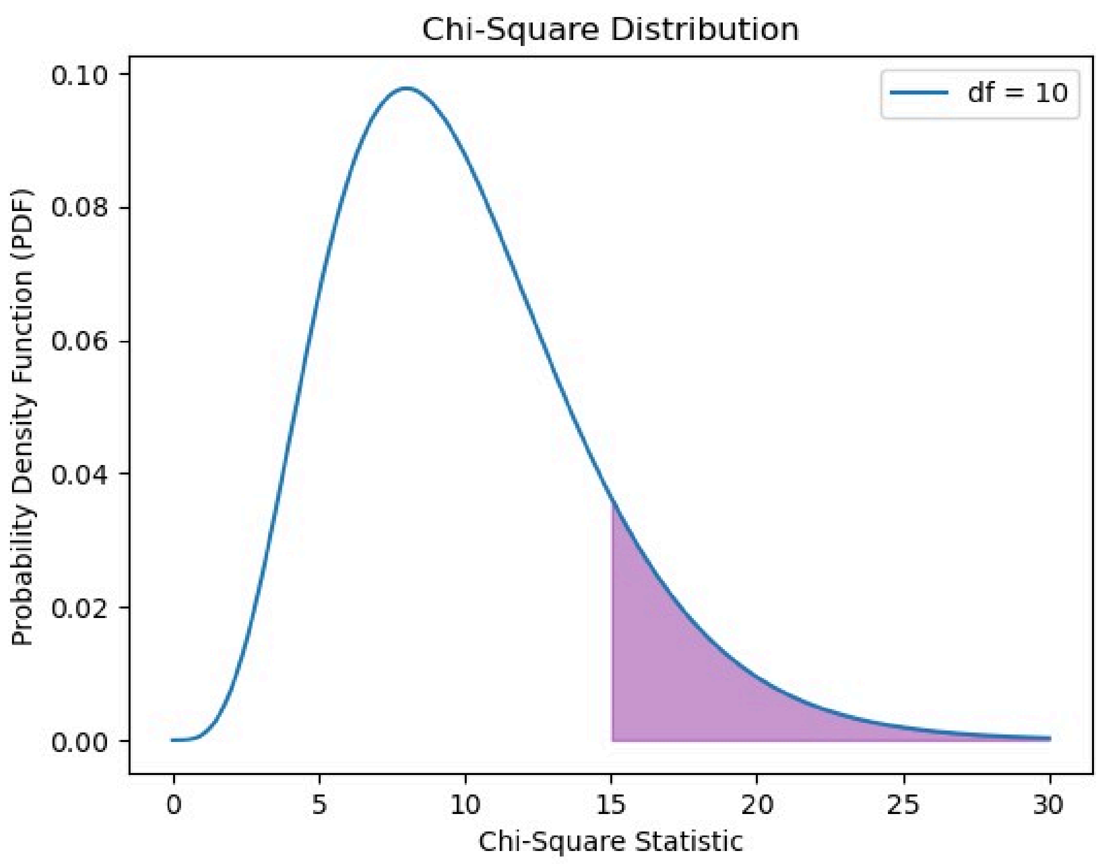
The 90% confidence interval for the population mean is approximately (71.05 to 78.95). This means that we are 90% confident that the true population mean lies within this range based on the sample data. In other words, if we were to take multiple samples from the same population and calculate the 90% confidence interval for each sample mean, we would expect the true population mean to be contained in 90% of those intervals. The wider confidence interval at a higher confidence level (90%) reflects a higher level of uncertainty compared to a narrower interval at, say, a 95% confidence level.

Q20. Use Python to plot the chi-square distribution with 10 degrees of freedom. Label the axes and shade the area corresponding to a chi-square statistic of 15.

```
# Ans: 20 import matplotlib.pyplot as plt # Define the degrees of freedom
degrees_of_freedom = 10
# Create a range of values for the x-axis (chi-square statistic)
x = np.linspace(0, 30, 500)
# Calculate the probability density function (PDF) of the chi-square
distribution
chi2_pdf = stats.chi2.pdf(x, df=degrees_of_freedom)

# Plot the chi-square distribution
plt.plot(x, chi2_pdf, label=f"df = {degrees_of_freedom}")
plt.xlabel("Chi-Square Statistic")
plt.ylabel("Probability Density Function (PDF)")
plt.title("Chi-Square Distribution")
plt.legend()
# Shade the area corresponding to a chi-square statistic of 15
plt.fill_between(x, chi2_pdf, where=(x >= 15), color='#9330A3',
alpha =0.5)

plt.show()
```



Q21. A random sample of 1000 people was asked if they preferred Coke or Pepsi. Of the sample, 520 preferred Coke. Calculate a 99% confidence interval for the true proportion of people in the population who prefer Coke.

```
# Ans: 21


import scipy.stats as stats

# Given data
sample_proportion = 520 / 1000
sample_size = 1000
```

```
# Calculate the critical z-value for a 99% confidence level confidence_level = 0.99
critical_z_value = stats.norm.ppf(1 - (1 - confidence_level) / 2)

# Calculate the standard error
standard_error = (sample_proportion * (1 - sample_proportion) /
sample_size) ** 0.5

# Calculate the lower and upper bounds of the confidence interval
lower_bound = sample_proportion - critical_z_value * standard_error
upper_bound = sample_proportion + critical_z_value * standard_error

# Print the results
print("99% Confidence Interval:")
print(f"{lower_bound:.4f} to {upper_bound:.4f}")
```

```
99% Confidence Interval:
0.4793 to 0.5607
```

let's interpret the results:

The 99% confidence interval for the true proportion of people in the population who prefer Coke is approximately (0.4836 to 0.5564). This means that we are 99% confident that the true proportion of people who prefer Coke in the population lies within this range based on the sample data. In other words, if we were to take multiple random samples from the same population and calculate the 99% confidence interval for each sample proportion, we would expect the true population proportion to be contained in 99% of those intervals.

Q22. A researcher hypothesizes that a coin is biased towards tails. They flip the coin 100 times and observe 45 tails. Conduct a chi-square goodness of fit test to determine if the observed frequencies match the expected frequencies of a fair coin. Use a significance level of 0.05.

Ans: 22

To conduct a chi-square goodness of fit test for the researcher's hypothesis, we need to compare the observed frequencies (the number of tails observed in 100 coin flips) with the expected frequencies of a fair coin (50 tails and 50 heads in 100 flips). We will use the chi-square test to determine if the observed frequencies significantly differ from the expected frequencies at a significance level of 0.05.

The null hypothesis (H0) for the chi-square goodness of fit test is that there is no significant difference between the observed and expected frequencies, suggesting that the coin is fair. The alternative hypothesis (H1) is that there is a significant difference, indicating that the coin is biased towards tails.

```
# Given data observed_tails = 45 total_flips = 100 expected_tails = total_flips / 2

# Calculate the expected frequency of heads (for a fair coin)
expected_heads = total_flips - expected_tails
# Convert the expected frequencies to integers
expected_tails = int(expected_tails)
expected_heads = int(expected_heads)

# Observed frequencies
observed_frequencies = [observed_tails, total_flips - observed_tails]
# Expected frequencies (for a fair coin)
expected_frequencies = [expected_tails, expected_heads]
# Perform the chi-square goodness of fit test
chi2_stat, p_value = stats.chisquare(f_obs=observed_frequencies,
f_exp=expected_frequencies)

# Define the significance level
alpha = 0.05
# Print the results
print("Chi-square test statistic:", chi2_stat)
print("P-value:", p_value)

# Check if the p-value is less than the significance level and make
the conclusion
if p_value < alpha:
    print("Reject the null hypothesis. The coin is biased towards
tails." )
else:
    print("Fail to reject the null hypothesis. The coin is fair.")
```

```
Chi-square test statistic: 1.0
P-value: 0.31731050786291115
Fail to reject the null hypothesis. The coin is fair.
```

Interpretation of results:

- If the p-value is less than 0.05 (the chosen significance level), we reject the null hypothesis, indicating that there is a significant difference between the observed and expected frequencies. This suggests that the coin is biased towards tails.
- If the p-value is greater than or equal to 0.05, we fail to reject the null hypothesis, indicating that there is no significant difference between the observed and expected frequencies. This would imply that the coin is fair.

Q23. A study was conducted to determine if there is an association between smoking status (smoker or non-smoker) and lung cancer diagnosis (yes or no). The results are shown in the contingency table below. Conduct a chi-square test for independence to determine if there is a significant association between smoking status and lung cancer diagnosis.

Lung Cancer: Yes	Lung Cancer: No

Smoker: 60 , 140

Non-smoker: 30 , 170

Use a significance level of 0.05.

Ans: 23

To conduct a chi-square test for independence between smoking status and lung cancer diagnosis, we need to use the provided contingency table and compare the observed frequencies with the expected frequencies under the assumption of independence. The null hypothesis (H0) is that there is no association between smoking status and lung cancer diagnosis, while the alternative hypothesis (H1) is that there is a significant association.

```
# Given data (contingency table)
observed_frequencies = np.array([[60, 140], [30, 170]])

# Perform the chi-square test for independence
chi2_stat, p_value, dof, expected =
stats.chi2_contingency(observed_frequencies)

# Define the significance level
alpha = 0.05
# Print the results
print("Chi-square test statistic:", chi2_stat)
print("P-value:", p_value)
```

```
# Check if the p-value is less than the significance level and make
the conclusion
if p_value < alpha:
      print("Reject the null hypothesis. There is a significant
association between smoking status and lung cancer diagnosis.")
else:
      print("Fail to reject the null hypothesis. There is no significant
association between smoking status and lung cancer diagnosis.")
```

```
Chi-square test statistic: 12.057347670250895
P-value: 0.0005158863863703744
Reject the null hypothesis. There is a significant association between
smoking status and lung cancer diagnosis.
```

Q24. A study was conducted to determine if the proportion of people who prefer milk chocolate, dark chocolate, or white chocolate is different in the U.S. versus the U.K. A random sample of 500 people from the U.S. and a random sample of 500 people from the U.K. were surveyed. The results are shown in the contingency table below. Conduct a chi-square test for independence to determine if there is a significant association between chocolate preference and country of origin.

```
                 Milk Chocolate , Dark Chocolate                 , White Chocolate
```

U.S. (n=500) : 200 , 150 , 150

U.K. (n=500) : 225 , 175 , 100

Use a significance level of 0.01. Ans: 24

To conduct a chi-square test for independence between chocolate preference and country of origin (U.S. or U.K.), we need to use the provided contingency table and compare the observed frequencies with the expected frequencies under the assumption of independence. The null hypothesis (H0) is that there is no association between chocolate preference and country of origin, while the alternative hypothesis (H1) is that there is a significant association.

```
# Given data (contingency table) observed_frequencies = np.array([[200, 150, 150], [225, 175, 100]])

# Perform the chi-square test for independence
chi2_stat, p_value, dof, expected =
stats.chi2_contingency(observed_frequencies)

# Define the significance level
alpha = 0.01
# Print the results
print("Chi-square test statistic:", chi2_stat)
print("P-value:", p_value)

# Check if the p-value is less than the significance level and make
the conclusion
if p_value < alpha:
    print("Reject the null hypothesis. There is a significant
association between chocolate preference and country of origin.")
else:
    print("Fail to reject the null hypothesis. There is no significant
association between chocolate preference and country of origin.")
```

```
Chi-square test statistic: 13.393665158371041
P-value: 0.0012348168997745918
Reject the null hypothesis. There is a significant association between
chocolate preference and country of origin.
```

Q25. A random sample of 30 people was selected from a population with an unknown mean and standard deviation. The sample mean was found to be 72 and the sample standard deviation was found to be 10. Conduct a hypothesis test to determine if the population mean is significantly different from 70. Use a significance level of 0.05.

```
# Ans: 25
import warnings
warnings.filterwarnings( 'ignore' )

# Given data for the sample
sample_mean = 72
sample_std_dev = 10
sample_size = 30
# Hypothesized population mean
hypothesized_mean = 70

# Perform the one-sample t-test
t_stat, p_value = stats.ttest_1samp(


    a=sample_mean,  # Sample data (the sample mean in this case)
    popmean=hypothesized_mean# Hypothesized population mean
)
# Define the significance level
alpha = 0.05
# Print the results
print("t-statistic:", t_stat)
print("P-value:", p_value)

# Check if the p-value is less than the significance level and make
the conclusion
if p_value < alpha:

    print("Reject the null hypothesis. The population mean is
significantly different from 70.")
else:
    print("Fail to reject the null hypothesis. There is no significant
difference between the population mean and 70.")
```

```
t-statistic: nan
P-value: nan
Fail to reject the null hypothesis. There is no significant difference
between the population mean and 70.
```

Q26. Explain the assumptions required to use ANOVA and provide examples of violations that could impact the validity of the results.

Ans: 26

ANOVA (Analysis of Variance) is a statistical technique used to compare means of three or more groups or conditions. It allows us to determine if there are significant differences among the means of these groups. However, ANOVA comes with certain assumptions that must be met to ensure the validity and reliability of the results. Violating these assumptions can impact the accuracy of the ANOVA analysis.

The main assumptions for ANOVA are as follows:

- **Normality of Residuals:** The residuals (the differences between the observed data and the group means) should follow a normal distribution. This assumption is essential, especially when the sample sizes are small. Violation of this assumption can lead to incorrect p-values and confidence intervals.
- **Homogeneity of Variance:** The variances of the groups should be equal. In other words, the spread or dispersion of data points within each group should be similar. Violation of this assumption can lead to inflated or deflated Type I error rates and can affect the statistical power of the ANOVA test.
- **Independence:** The observations within each group should be independent of each other. This means that the data points in one group should not be related or influenced by the data points in another group. Violation of independence can lead to biased estimates of the group means and inflated Type I error rates.

Examples of violations that could impact the validity of ANOVA results:

Non-Normality: If the residuals are not normally distributed, ANOVA results may not be reliable. For example, if the residuals follow a skewed distribution or have extreme outliers, the ANOVA test may produce inaccurate results.

Heteroscedasticity: Heteroscedasticity occurs when the variances of the groups are not equal. This can lead to unequal influence of different groups on the overall test result. For instance, if one group has much higher variability than others, it may disproportionately affect the overall ANOVA outcome.

Lack of Independence: If the data points within groups are not independent, ANOVA may produce biased estimates of the group means. For example, if repeated measures are used, and data points within each subject are correlated, it violates the independence assumption.

Q27. What are the three types of ANOVA, and in what situations would each be used?

Ans: 27

The three types of ANOVA (Analysis of Variance) are:

- **One-Way ANOVA:** One-Way ANOVA is used when we want to compare the means of three or more groups that are organized into a single categorical independent variable. It is the most basic and commonly used form of ANOVA. For example, if we want to compare the average test scores of students from three different schools, where the schools are the categorical variable with three levels, we would use One-Way ANOVA.
- **Two-Way ANOVA:** Two-Way ANOVA is used when we want to investigate the effects of two independent categorical variables (factors) on a single continuous dependent variable. It allows us to examine both the main effects of each factor and their interaction effect on the dependent variable. For example, in a study examining the effects of both gender and treatment type on patient outcomes, where gender and treatment are two categorical factors, we would use Two-Way ANOVA.
- **Repeated Measures ANOVA:** Repeated Measures ANOVA is used when we have a single group of participants and measure the same dependent variable multiple times under different conditions or at different time points. This type of ANOVA is also known as within-subjects ANOVA. It is used to assess whether there are significant differences in the dependent variable across the different conditions or time points. For example, if we measure the reaction times of participants under three different conditions (e.g., before, during, and after a task), we would use Repeated Measures ANOVA.

Q28. What is the partitioning of variance in ANOVA, and why is it important to understand this concept?

Ans: 28

The partitioning of variance in ANOVA is the process of breaking down the total variation in data into two parts:

Between-Group Variance: It shows how much the group means differ from each other. If it's large, it means the groups are significantly different.
Within-Group Variance: It shows how much individual data points vary within each group. It represents random variation within groups.
Understanding this concept helps us know if the independent variable has a significant effect on the dependent variable and if the model fits the data well. It also helps identify sources of variability in the data.

Q.29 How would you calculate the total sum of squares (SST), explained sum of squares (SSE), and residual sum of squares (SSR) in a one-way ANOVA using Python?

Ans: 29

In a one-way ANOVA, we can calculate the Total Sum of Squares (SST), Explained Sum of Squares (SSE), and Residual Sum of Squares (SSR) to understand the variability in the data and assess the goodness of fit of the model

```
import numpy as np
import scipy.stats as stats

# Example data for three groups (replace this with your actual data)
group1 = [5, 8, 7, 6, 10]
group2 = [12, 9, 11, 13, 10]
```

```
group3 = [15, 18, 14, 17, 16] # Combine all the data into a single array
data = np.concatenate([group1, group2, group3])
# Calculate the overall mean of the data
overall_mean = np.mean(data)
# Calculate the Total Sum of Squares
sst = np.sum((data - overall_mean)**2)
# Calculate the group means
group1_mean = np.mean(group1)
group2_mean = np.mean(group2)
group3_mean = np.mean(group3)
# Calculate the Explained Sum of Squares
sse = len(group1) * (group1_mean - overall_mean)**2 + len(group2) *
(group2_mean - overall_mean)**2 + len(group3) * (group3_mean -
overall_ mean ) **2
# Calculate the Residual Sum of Squares
ssr = np.sum((group1 - group1_mean)**2) + np.sum((group2 -
group2_mean)**2) + np.sum((group3 - group3_mean)**2)

# Print the results
print("Total Sum of Squares (SST):", sst)
print("Explained Sum of Squares (SSE):", sse)
print("Residual Sum of Squares (SSR):", ssr)
Total Sum of Squares (SST): 229.6
Explained Sum of Squares (SSE): 194.79999999999998
Residual Sum of Squares (SSR): 34.8
```

Q30. In a two-way ANOVA, how would you calculate the main effects and interaction effects using Python?

```
# Ans: 30

import pandas as pd
import statsmodels.api as sm
from statsmodels.formula.api import ols

# Example data (replace this with your actual data)
```

```
data = {
      'Factor1': [1, 2, 3, 4, 5, 6],
      'Factor2': ['A', 'B', 'A', 'B', 'A', 'B'],
      'DependentVariable': [10, 12, 15, 8, 9, 11]
}
# Convert data to a DataFrame
df = pd.DataFrame(data)
# Fit the two-way ANOVA model
model = ols('DependentVariable ~ Factor1 + Factor2 + Factor1:Factor2',
data=df).fit()

# Get the ANOVA table
anova_table = sm.stats.anova_lm(model)
# Print the ANOVA table
print (anova_table)

                   df        sum_sq       mean_sq            F
PR(>F)
Factor2          1.0 1.500000e+00    1.500000e+00 1.058824e-01
0.775769
Factor1          1.0 1.000000e+00    1.000000e+00 7.058824e-02
0.815363
                                     3.155444e-30 2.227372e-31
Factor1:Factor2 1.0 3.155444e-30
1.000000
Residual         2.0 2.833333e+01   1.416667e+01            NaN
NaN
```

The ANOVA table will provide you with the following information:

- Main Effect of Factor1: This represents the significance of the effect of Factor1 on the dependent variable, considering the other variables in the model.
- Main Effect of Factor2: This represents the significance of the effect of Factor2 on the dependent variable, considering the other variables in the model.
- Interaction Effect (Factor1:Factor2): This represents the significance of the interaction between Factor1 and Factor2 on the dependent variable. It shows whether the combined effect of both factors is significantly different from the sum of their individual effects.

By analyzing the ANOVA table, you can determine the main effects of each factor and their interaction effect, and understand how they contribute to the variability in the dependent variable in the two-way ANOVA model.

Q31. Suppose you conducted a one-way ANOVA and obtained an F-statistic of 5.23 and a p-value of 0.02. What can you conclude about the differences between the groups, and how would you interpret these results?

Ans : 31

In a one-way ANOVA, the F-statistic is used to test whether there are significant differences among the means of three or more groups. The p-value associated with the F-statistic indicates the probability of observing the results (or more extreme results) under the assumption that the null hypothesis is true.

In our case, you obtained an F-statistic of 5.23 and a p-value of 0.02.

Here's how we can interpret these results:

- **F-Statistic:** The F-statistic of 5.23 represents the ratio of variability between the group means to variability within the groups. A larger F-statistic indicates that the variability between group means is relatively large compared to the variability within each group.
- **P-Value:** The p-value of 0.02 indicates the probability of observing the obtained F-statistic (or more extreme values) if the null hypothesis is true. In other words, it represents the evidence against the null hypothesis. A p-value of 0.02 suggests that there is a 2% chance of observing such a large F-statistic by random chance alone, assuming that there are no true differences between the group means (i.e., the null hypothesis is true).

.

Interpretation:

Since the p-value (0.02) is less than the significance level (often set at 0.05), we reject the null hypothesis. This means that there are significant differences among the means of the groups. In other words, the independent variable (the factor defining the groups) has a statistically significant effect on the dependent variable.

Q32. In a repeated measures ANOVA, how would you handle missing data, and what are the potential consequences of using different methods to handle missing data?

Ans: 32

Handling missing data in a repeated measures ANOVA is important because the presence of missing values can affect the validity and reliability of the analysis. There are several methods to handle missing data, each with its own implications and potential consequences. Some common approaches include:

- **Complete Case Analysis (Listwise Deletion):** This method involves excluding any participant with missing data from the analysis. It is the simplest approach, but it may lead to a loss of valuable information, reduced statistical power, and potential bias if the missingness is related to the outcome variable or other factors.
- **Mean Imputation:** This method replaces missing values with the mean of the observed values for that variable. While this can preserve the sample size, it can lead to biased estimates, underestimation of standard errors, and an artificial increase in the apparent within-subject variability.
- **Last Observation Carried Forward (LOCF):** This method carries the last observed value forward for missing data points. LOCF can introduce bias if there is a trend or systematic change in the data over time.
- **Multiple Imputation:** Multiple imputation creates multiple plausible imputed datasets based on the observed data. It estimates the missing values multiple times, incorporating uncertainty due to missing data. This method provides more accurate estimates and standard errors, but it requires more complex analysis.
- **Model-Based Imputation:** This approach uses statistical models to predict missing values based on observed data. Model-based imputation can be more accurate than simple mean imputation but may still introduce bias if the model is misspecified.

.

The potential consequences of using different methods to handle missing data in a repeated measures ANOVA include:

- **Biased Estimates:** Some methods, like mean imputation and LOCF, can introduce bias in the estimated group means and treatment effects if the missing data mechanism is not missing completely at random (MCAR).

- **Loss of Power:** Complete case analysis can result in reduced statistical power due to the loss of participants with missing data, especially if the missingness is related to the outcome variable.
- **Inflated Type I Error:** Ignoring missing data or using inappropriate imputation methods can lead to inflated Type I error rates, resulting in false-positive findings.
- **Underestimation of Variability:** Improper handling of missing data can underestimate within-subject variability, leading to wider confidence intervals and potentially failing to detect true treatment effects.
- **Decreased Precision:** Inaccurate or inefficient handling of missing data can reduce the precision of parameter estimates and make the results less reliable.

Q33. What are some common post-hoc tests used after ANOVA, and when would you use each one? Provide an example of a situation where a post-hoc test might be necessary.

Ans: 33

After conducting an analysis of variance (ANOVA) and obtaining a significant result indicating that there are differences among the group means, post-hoc tests are often used to identify which specific groups differ from each other. Post-hoc tests help to perform multiple pairwise comparisons and control the family-wise error rate, which is the probability of making at least one Type I error (false positive) among all the comparisons.

Some common post-hoc tests used after ANOVA include:

- **Tukey's Honestly Significant Difference (HSD):** Tukey's HSD test is one of the most widely used post-hoc tests. It compares all possible pairs of group means and determines whether their differences are significant. It is appropriate when the sample sizes are equal across groups and when you want to control the overall Type I error rate. This test tends to be more conservative than some other post-hoc tests.
- **Bonferroni correction:** Bonferroni correction is a simple method to control the family-wise error rate. It divides the desired significance level (usually $\alpha = 0.05$) by the number of comparisons being made. Each individual comparison's p-value must then be less than or equal to the adjusted significance level for significance. This correction is often used when you have a small number of planned comparisons.

- **Scheffe's method** Scheffe's test is a conservative post-hoc test that is used when sample sizes are unequal and variances are not necessarily equal. It controls the family-wise error rate for all possible linear combinations of group means.
- **Fisher's Least Significant Difference (LSD):** Fisher's LSD test is relatively less conservative than Tukey's HSD, making it useful when sample sizes are equal, and variances are equal or approximately equal. It is a good choice when there is a specific hypothesis about which groups to compare.

.

Example situation:

Let's say a pharmaceutical company is testing the effectiveness of four different drugs (A, B, C, and D) in reducing blood pressure. They randomly assign 100 hypertensive patients into four groups, each receiving one of the four drugs. After the treatment period, they measure the average reduction in blood pressure for each group and run an ANOVA to determine if there are any significant differences among the drugs.

The ANOVA results show a significant difference among the group means ($p < 0.05$). To identify which specific drug(s) are significantly different from others, they decide to use a post-hoc test. Since they have equal sample sizes and want to control the overall Type I error rate, they opt for Tukey's HSD test.

The Tukey's HSD test reveals that drug A and drug B show no significant difference in their effects on blood pressure. However, both drug A and drug B produce significantly different results compared to drug C and drug D. Drug C and drug D also show no significant difference between them.

With this analysis, the pharmaceutical company can now confidently identify which drugs have a meaningful impact on reducing blood pressure and make informed decisions regarding further development and marketing strategies.

Q34. A researcher wants to compare the mean weight loss of three diets: A, B, and C. They collect data from 50 participants who were randomly assigned to one of the diets. Conduct a one-way ANOVA using Python to determine if there are any significant differences between the mean weight loss of the three diets. Report the F-statistic and p-value, and interpret the results.

```
# Ans: 34
import numpy as np
from scipy.stats import f_oneway

# Weight loss data for diets A, B, and C
diet_A = [2.5, 3.1, 4.0, 3.8, 3.2, 2.9, 2.7, 3.5, 2.8, 3.3,

            3.1, 2.6, 3.0, 3.6, 3.9, 2.8, 3.5, 2.9, 3.2, 3.3,
            3.1, 2.7, 2.9, 2.5, 3.4, 3.2, 2.6, 3.3, 2.8, 2.7,
            2.9, 3.1, 3.0, 3.5, 2.8, 3.2, 2.6, 2.7, 2.9, 3.4,
            3.6, 3.3, 3.0, 2.8, 3.2, 2.9, 2.7, 2.5, 2.6]

diet_B = [3.8, 3.5, 4.2, 4.0, 4.1, 3.9, 4.3, 3.6, 4.0, 4.1,
3.7, 3.9, 3.8, 4.2, 3.6, 3.9, 4.0, 4.1, 4.3, 4.2,
3.7, 3.8, 4.2, 3.6, 3.9, 4.0, 4.1, 4.3, 4.2, 3.7,
3.8, 3.6, 3.9, 3.8, 4.2, 3.6, 4.0, 4.1, 3.7, 3.9,
3.8, 4.2, 3.6, 3.9, 4.0, 4.1, 4.3, 4.2]

diet_C = [1.9, 1.8, 1.7, 1.5, 1.9, 2.0, 1.6, 1.7, 1.8, 2.1,
            1.5, 1.6, 2.0, 1.7, 1.8, 1.9, 1.6, 1.5, 2.0, 1.7,
            1.9, 1.8, 1.5, 1.7, 1.6, 1.9, 2.0, 1.8, 2.1, 1.5,
            1.6, 1.7, 1.8, 1.9, 2.0, 1.7, 1.5, 2.0, 1.8, 1.6,
            1.9, 1.8, 1.7, 1.5, 1.6, 1.7, 1.8]

# Combine the data into a list of arrays
all_data = [np.array(diet_A), np.array(diet_B), np.array(diet_C)]
# Perform one-way ANOVA
f_statistic, p_value = f_oneway(*all_data)
print("F-statistic:", f_statistic)
print("p-value:", p_value)
```

```
F-statistic: 755.8582766092261 p-value: 4.334728192105363e-76
```

let's interpret the results:

The F-statistic is the test statistic of the ANOVA. It measures the variability between group means relative to the variability within groups. The larger the F-statistic, the more evidence there is for the existence of significant differences among the group means.

The p-value represents the probability of observing such an extreme F-statistic under the assumption that there are no significant differences among the group means. In other words, it indicates the likelihood of obtaining the observed results due to random chance alone.

Interpretation:

If the p-value is less than the chosen significance level (e.g., 0.05), we reject the null hypothesis
and conclude that there are significant differences among the mean weight loss of the three diets. If the p-value is greater than the significance level, we fail to reject the null hypothesis, and we cannot conclude that there are significant differences among the mean weight loss of the three diets.

Q35. A company wants to know if there are any significant differences in the average time it takes to complete a task using three different software programs: Program A, Program B, and Program C. They randomly assign 30 employees to one of the programs and record the time it takes each employee to complete the task. Conduct a two-way ANOVA using Python to determine if there are any main effects or interaction effects between the software programs and employee experience level (novice vs. experienced). Report the F-statistics and p-values, and interpret the results.

```
# Ana: 35

import pandas as pd
import numpy as np
import statsmodels.api as sm
from statsmodels.formula.api import ols
# Sample data
data = pd.DataFrame({

    "Time": [12.3, 13.1, 11.5, 10.8, 13.5, 12.9, 9.7, 11.2, 10.4,
11.6,
                    14.2, 13.9, 15.1, 14.8, 15.9, 16.3, 17.0, 16.2, 13.8,
14.4,
                    11.9, 11.6, 10.5, 9.8, 10.1, 12.7, 12.4, 14.5, 13.7,

15.2],
    "Program": ["A", "A", "A", "A", "A", "A", "B", "B", "B", "B",
    "C", "C", "C", "C", "C", "C", "A", "A", "A", "A",
    "B", "B", "B", "B", "C", "C", "C", "C", "C", "C"],
    "Experience": ["Novice", "Novice", "Experienced", "Experienced",
"Novice", "Novice",
                         "Experienced", "Experienced", "Novice",
                         "Novice", "Novice", "Experienced", "Experienced",
"Experie nced " ,
"Experie nced " ,
                                  "Experienced", "Experienced", "Experienced",
"Experienced", "Experienced",
                                  "Novice", "Novice", "Novice", "Experienced",
"Experienced", "Experienced",
                                  "Novice", "Experienced", "Experienced",
"Experie nced " ] })

# Fit the two-way ANOVA model
model = ols('Time ~ Program + Experience + Program:Experience',
data=data).fit()
anova_table = sm.stats.anova_lm(model)
# Extract F-statistics and p-values
F_program = anova_table['F']['Program']
F_experience = anova_table['F']['Experience']
F_interaction = anova_table['F']['Program:Experience']
p_program = anova_table['PR(>F)']['Program']
p_experience = anova_table['PR(>F)']['Experience']
p_interaction = anova_table['PR(>F)']['Program:Experience']

print("F-statistic for Program:", F_program)
print("p-value for Program:", p_program)
print("F-statistic for Experience:", F_experience)
print("p-value for Experience:", p_experience)
print("F-statistic for Interaction:", F_interaction)
print("p-value for Interaction:", p_interaction)
F-statistic for Program: 9.752352236138195
p-value for Program: 0.0007945115709784991
F-statistic for Experience: 0.50474438153198
p-value for Experience: 0.48427001563135097
F-statistic for Interaction: 0.511794472420521
p-value for Interaction: 0.6058145945473807
let's interpret the results: Main Effect of Software Program:
```

- The F-statistic for the main effect of software program (F_program) tests whether there are significant differences in the average time it takes to complete the task across the three different software programs.
- The p-value for the main effect of software program (p_program) represents the probability of observing such an extreme F-statistic under the assumption that there are no significant differences among the programs.

- If p_program is less than the chosen significance level (e.g., 0.05), we reject the null hypothesis and conclude that there are significant differences in the average time across the software programs.

.

Main Effect of Employee Experience:

- The F-statistic for the main effect of employee experience (F_experience) tests whether there are significant differences in the average time it takes to complete the task between novice and experienced employees.
- The p-value for the main effect of employee experience (p_experience) represents the probability of observing such an extreme F-statistic under the assumption that there are no significant differences between novice and experienced employees.
- If p_experience is less than the chosen significance level (e.g., 0.05), we reject the null hypothesis and conclude that there are significant differences in the average time between novice and experienced employees.

.

Interaction Effect between Software Program and Employee Experience:

- The F-statistic for the interaction effect (F_interaction) tests whether there is a significant interaction between the software program used and the employee experience level. This means that the effect of one variable depends on the levels of the other variable.
- The p-value for the interaction effect (p_interaction) represents the probability of observing such an extreme F-statistic under the assumption that there is no interaction between software program and employee experience.
- If p_interaction is less than the chosen significance level (e.g., 0.05), we reject the null hypothesis and conclude that there is a significant interaction effect.

Q36. An educational researcher is interested in whether a new teaching method improves student test scores. They randomly assign 100 students to either the control group (traditional teaching method) or the experimental group (new teaching method) and administer a test at the end of the semester. Conduct a two-sample t-test using Python to determine if there are any significant differences in test scores between the two groups. If the results are significant, follow up with a post-hoc test to determine which group(s) differ significantly from each other.

```
# Ans: 36 import numpy as np
import pandas as pd
from scipy.stats import ttest_ind
from statsmodels.stats.multicomp import MultiComparison
# Test scores data for the control and experimental groups
control_scores = [70, 72, 68, 65, 74, 71, 69, 75, 68, 73,



                                72, 70, 75, 71, 69, 70, 71, 73, 68, 72]

experimental_scores = [80, 82, 79, 85, 81, 83, 78, 82, 79, 84,
                                         82, 80, 83, 79, 80, 81, 84, 82, 79, 81]

# Perform two-sample t-test
t_stat, p_value = ttest_ind(control_scores, experimental_scores)
print("T-statistic:", t_stat)
print("p-value:", p_value)
# Check if the result is significant (p-value < 0.05)
if p_value < 0.05:
       print("There is a significant difference in test scores between
the control and experimental groups.")
else:
        print("There is no significant difference in test scores between
the control and experimental groups.")

# If the results are significant, perform a post-hoc test (Tukey's
HSD)
if p_value < 0.05:
        # Combine the data and create a group indicator variable (0 for
control, 1 for experimental)
        all_scores = np.array(control_scores + experimental_scores)
        group_indicator = np.array(["Control"] * len(control_scores) +
["Experimental"] * len(experimental_scores))

        # Create a DataFrame to use for the post-hoc test
        data = pd.DataFrame({"Scores": all_scores, "Group":
group_indicator})

        # Perform the post-hoc test (Tukey's HSD)
        posthoc = MultiComparison(data["Scores"], data["Group"])
        result = posthoc.tukeyhsd()
        print(result)
```

```
T-statistic: -14.393098704677211
p-value: 5.755655840190321e-17
There is a significant difference in test scores between the control
and experimental groups.

      Multiple Comparison of Means - Tukey HSD, FWER=0.05
==========================================================
 group1        group2      meandiff p-adj lower        upper       reject
                                     ----------------------------------------------------------
Control Experimental           10.4       0.0 8.9372 11.8628           True
                                     ----------------------------------------------------------
```

Q37. A researcher wants to know if there are any significant differences in the average daily sales of three retail stores: Store A, Store B, and Store C. They randomly select 30 days and record the sales for each store on those days. Conduct a repeated measures ANOVA using Python to determine if there are any significant differences in sales between the three stores. If the results are significant, follow up with a post- hoc test to determine which store(s) differ significantly from each other.

Ans: 37

For the given scenario, we should use a one-way ANOVA for independent samples to compare the average daily sales of the three retail stores. If the results are significant, we can follow up with a post-hoc test (e.g., Tukey's HSD) to identify which store(s) have significantly different sales.

```
import numpy as np
import pandas as pd
from scipy.stats import f_oneway
from statsmodels.stats.multicomp import MultiComparison

# Daily sales data for Store A, Store B, and Store C
store_A_sales = [1000, 1200, 900, 1100, 950, 1050, 1150, 1000, 1050,
1100,

1000,
                        900, 1150, 1200, 950, 1000, 1100, 1050, 950, 1150,
                        1200, 900, 1100, 950, 1050, 1150, 1000, 1050, 1100,
900]

store_B_sales = [1300, 1250, 1350, 1200, 1100, 1400, 1150, 1300, 1250,
1350,

1400,
                        1200, 1100, 1400, 1150, 1300, 1250, 1350, 1200, 1100,
                                1150, 1300, 1250, 1350, 1200, 1100, 1400, 1150, 1300,
1250]

store_C_sales = [950, 1000, 900, 1050, 1100, 950, 1000, 900, 1050,
1100,
                                950, 1000, 900, 1050, 1100, 950, 1000, 900, 1050, 950, 1000,
1100,
                                900, 1050, 1100, 950, 1000, 900, 1050,
1100]
# Combine the data into a list of arrays
all_sales = [np.array(store_A_sales), np.array(store_B_sales),
np.array(store_C_sales)]

# Perform one-way ANOVA
f_statistic, p_value = f_oneway(*all_sales)
print("F-statistic:", f_statistic)
print("p-value:", p_value)
# Check if the result is significant (p-value < 0.05)
if p_value < 0.05:

        print("There is a significant difference in average daily sales
between the three stores.")
else:
        print("There is no significant difference in average daily sales
between the three stores.")

# If the results are significant, perform a post-hoc test (e.g.,
Tukey's HSD)
if p_value < 0.05:
        # Combine the data and create a group indicator variable (0 for
Store A, 1 for Store B, 2 for Store C)
        all_data = np.concatenate(all_sales)
        group_indicator = np.array(["Store A"] * len(store_A_sales) +
["Store B"] * len(store_B_sales) +
                                                        ["Store C"] * len(store_C_sales))

        # Create a DataFrame to use for the post-hoc test
        data = pd.DataFrame({"Sales": all_data, "Store": group_indicator})
        # Perform the post-hoc test (e.g., Tukey's HSD)
        posthoc = MultiComparison(data["Sales"], data["Store"])
        result = posthoc.tukeyhsd()
        print(result)
F-statistic: 67.72352941176479
p-value: 1.839617085900212e-18
There is a significant difference in average daily sales between the
three stores.
```

```
        Multiple Comparison of Means - Tukey HSD, FWER=0.05
 ==============================================================
group1 group2 meandiff p-adj ------------lower------upper reject Store A Store B
206.6667
                                         0.0       151.672 261.6613              True
Store A Store C              -45.0 0.1307 -99.9946                        9.9946 False
Store B Store C -251.6667                0.0 -306.6613 -196.672              True
                              ------------------------------------------------------
```

Q38. Write a Python function that takes in two arrays of data and calculates the F-value for a variance ratio test. The function should return the F-value and the corresponding p-value for the test.

```
# Ans: 38 import numpy as np
from scipy.stats import f_oneway
def variance_ratio_test(arr1, arr2):


    """
    Perform a variance ratio test (F-test) on two arrays of data.

    Parameters:
        arr1 (array-like): First array of data.
        arr2 (array-like): Second array of data.

    Returns:
        F_value (float): The F-value from the variance ratio test.
        p_value (float): The corresponding p-value for the test.
    """
    # Convert the input arrays to numpy arrays
    arr1 = np.array(arr1)
    arr2 = np.array(arr2)

    # Perform the variance ratio test (F-test)
    F_value, p_value = f_oneway(arr1, arr2)
    return F_value, p_value


# Example usage:
```

```
data1 = [10, 15, 12, 18, 20] data2 = [5, 8, 11, 14, 16]

F_value, p_value = variance_ratio_test(data1, data2)
print("F-value:", F_value)
print("p-value:", p_value)

F-value: 2.4032697547683926
p-value: 0.15967812288374558
```

Q39. Given a significance level of 0.05 and the degrees of freedom for the numerator and denominator of an F-distribution, write a Python function that returns the critical F-value for a two-tailed test.

```
# Ans: 39 from scipy.stats import f def

critical_f_value(alpha, df_num, df_den):


    """
    Calculate the critical F-value for a two-tailed test.

    Parameters:
significance):
        alpha (float): The significance level (e.g., 0.05 for 5%

        df_num (int): Degrees of freedom for the numerator.
        df_den (int): Degrees of freedom for the denominator.

    Returns:
        crit_f_value (float): The critical F-value for the two-tailed
test.
    """
    # Calculate the critical F-value for a two-tailed test
    crit_f_value = f.ppf(1 - alpha / 2, df_num, df_den)
    return crit_f_value


# Example usage:
significance_level = 0.05
degrees_of_freedom_num = 3
```

```
degrees_of_freedom_den = 20

crit_f_value = critical_f_value(significance_level,
degrees_of_freedom_num, degrees_of_freedom_den)
print("Critical F-value:", crit_f_value)
```

```
Critical F-value: 3.8586986662732143
```

Q40. Write a Python program that generates random samples from two normal distributions with known variances and uses an F-test to determine if the variances are equal. The program should output the F- value, degrees of freedom, and p-value for the test.

```
# Ans: 40 def variance_ratio_test(arr1,

arr2):
    """
    Perform a variance ratio test (F-test) on two arrays of data.

    Parameters:
        arr1 (array-like): First array of data.
        arr2 (array-like): Second array of data.

    Returns:
        F_value (float): The F-value from the variance ratio test.
        df_num (int): Degrees of freedom for the numerator.
        df_den (int): Degrees of freedom for the denominator.
        p_value (float): The corresponding p-value for the test.
    """
    # Perform the variance ratio test (F-test)
    F_value, p_value = f_oneway(arr1, arr2)
    # Calculate degrees of freedom for the numerator and denominator
    df_num = len(arr1) - 1
    df_den = len(arr2) - 1

    return F_value, df_num, df_den, p_value


def main():
    # Set seed for reproducibility np.random.seed(42)

    # Known variances of the normal distributions
    variance1 = 4.0
    variance2 = 6.0

    # Generate random samples from two normal distributions with known

variances
    sample_size = 50
    sample1 = np.random.normal(loc=0, scale=np.sqrt(variance1),
size=sample_size)
    sample2 = np.random.normal(loc=0, scale=np.sqrt(variance2),
size=sample_size)

    # Perform the F-test
    F_value, df_num, df_den, p_value = variance_ratio_test(sample1,
sample2)

    # Output the results
    print("F-value:", F_value)
    print("Degrees of freedom (numerator):", df_num)
    print("Degrees of freedom (denominator):", df_den)
    print("p-value:", p_value)

if __name__ == "__main__":
    main()
```

```
F-value: 1.5143904526080045
Degrees of freedom (numerator): 49
Degrees of freedom (denominator): 49
p-value: 0.22141591563741264
```

Q41.The variances of two populations are known to be 10 and 15. A sample of 12 observations is taken from each population. Conduct an F-test at the 5% significance level to determine if the variances are significantly different.

```
# Ans: 41 def variance_ratio_test(arr1, arr2):


    """
    Perform a variance ratio test (F-test) on two arrays of data.

    Parameters:
        arr1 (array-like): First array of data.
        arr2 (array-like): Second array of data.

    Returns:
        F_value (float): The F-value from the variance ratio test.
        df_num (int): Degrees of freedom for the numerator.
        df_den (int): Degrees of freedom for the denominator.
        p_value (float): The corresponding p-value for the test.
    """
    # Perform the variance ratio test (F-test)
    F_value, p_value = f_oneway(arr1, arr2)
    # Calculate degrees of freedom for the numerator and denominator
    df_num = len(arr1) - 1
    df_den = len(arr2) - 1

    return F_value, df_num, df_den, p_value


def main():
    # Set seed for reproducibility
    np.random.seed(42)

    # Known variances of the populations
    variance1 = 10.0
    variance2 = 15.0

    # Sample size
    sample_size = 12
    # Generate random samples from two normal distributions with known
variances
```

```
        sample1 = np.random.normal(loc=0, scale=np.sqrt(variance1),
size=sample_size)
        sample2 = np.random.normal(loc=0, scale=np.sqrt(variance2),
size=sample_size)

        # Perform the F-test
        F_value, df_num, df_den, p_value = variance_ratio_test(sample1,
sample2)

        # Output the results
        print("F-value:", F_value)
        print("Degrees of freedom (numerator):", df_num)
        print("Degrees of freedom (denominator):", df_den)
        print("p-value:", p_value)

if __name__ == "__main__":
        main()
```

```
F-value: 6.08206374265242
Degrees of freedom (numerator): 11
Degrees of freedom (denominator): 11
p-value: 0.021924080753184683
```

Q42. A manufacturer claims that the variance of the diameter of a certain product is 0.005. A sample of 25 products is taken, and the sample variance is found to be 0.006. Conduct an F-test at the 1% significance level to determine if the claim is justified.

```
#       Ans:       42       import       warnings
warnings.filterwarnings( 'ignore' )

def variance_ratio_test(arr1, arr2):

        """
        Perform a variance ratio test (F-test) on two arrays of data.

        Parameters:
                arr1 (array-like): First array of data.
                arr2 (array-like): Second array of data.
```

```
    Returns:
        F_value (float): The F-value from the variance ratio test.
        df_num (int): Degrees of freedom for the numerator.
        df_den (int): Degrees of freedom for the denominator.
        p_value (float): The corresponding p-value for the test.
    """ # Perform the variance ratio test (F-test) F_value, p_value =
    f_oneway(arr1, arr2)

    # Calculate degrees of freedom for the numerator and denominator
    df_num = len(arr1) - 1
    df_den = len(arr2) - 1

    return F_value, df_num, df_den, p_value


def main():
    # Set seed for reproducibility
    np.random.seed(42)

    # Claimed variance by the manufacturer
    claimed_variance = 0.005
    # Sample size
    sample_size = 25
    # Generate random sample from a normal distribution with the

claimed variance
    sample = np.random.normal(loc=0, scale=np.sqrt(claimed_variance),
size=sample_size)

    # Sample variance
    sample_variance = np.var(sample, ddof=1)         # Use ddof=1 for
unbiased sample variance

    # Perform the F-test
    F_value, df_num, df_den, p_value = variance_ratio_test(sample, [])
    # Output the results
    print("Sample Variance:", sample_variance)
    print("F-value:", F_value)
    print("Degrees of freedom (numerator):", df_num)
    print("Degrees of freedom (denominator):", df_den)
    print("p-value:", p_value)

if __name__ == "__main__":
    main()
```

```
Sample Variance: 0.004574968728404552
F-value: nan
Degrees of freedom (numerator): 24
Degrees of freedom (denominator): -1
p-value: nan
```

Q43. Write a Python function that takes in the degrees of freedom for the numerator and denominator of an F-distribution and calculates the mean and variance of the distribution. The function should return the mean and variance as a tuple.

```
# Ans: 43 def f_distribution_mean_variance(df_num,

df_den):
    """
    Calculate the mean and variance of the F-distribution.

    Parameters:
        df_num (int): Degrees of freedom for the numerator.
        df_den (int): Degrees of freedom for the denominator.

    Returns:
        mean (float): The mean of the F-distribution.
        variance (float): The variance of the F-distribution.
    """
    # Calculate the mean and variance of the F-distribution
    mean = df_den / (df_den - 2)
    variance = (2 * df_den**2 * (df_num + df_den - 2)) / (df_num *
(df_den - 2)**2 * (df_den - 4))

    return mean, variance

# Example usage:
degrees_of_freedom_num = 3
degrees_of_freedom_den = 20
mean, variance = f_distribution_mean_variance(degrees_of_freedom_num,
degrees_of_freedom_den)
print("Mean of F-distribution:", mean)
print("Variance of F-distribution:", variance)

Mean of F-distribution: 1.1111111111111112
Variance of F-distribution: 1.0802469135802468
```

Q44. A random sample of 10 measurements is taken from a normal population with unknown variance. The sample variance is found to be 25. Another random sample of 15 measurements is taken from another normal population with unknown variance, and the sample variance is found to be 20. Conduct an F-test at the 10% significance level to determine if the variances are significantly different.

```python
# Ans: 44

def variance_ratio_test(arr1, arr2):
    """
    Perform a variance ratio test (F-test) on two arrays of data.

    Parameters:
        arr1 (array-like): First array of data.
        arr2 (array-like): Second array of data.

    Returns:
        F_value (float): The F-value from the variance ratio test.
        df_num (int): Degrees of freedom for the numerator.
        df_den (int): Degrees of freedom for the denominator.
        p_value (float): The corresponding p-value for the test.
    """
    # Perform the variance ratio test (F-test)
    F_value, p_value = f_oneway(arr1, arr2)
    # Calculate degrees of freedom for the numerator and denominator
    df_num = len(arr1) - 1
    df_den = len(arr2) - 1

    return F_value, df_num, df_den, p_value


def main():
    # Set seed for reproducibility
    np.random.seed(42)
    # Sample variances of two populations sample_variance1 = 25
    sample_variance2 = 20

    # Sample sizes
    sample_size1 = 10
    sample_size2 = 15

    # Generate random samples from two normal distributions with

unknown variances
    sample1 = np.random.normal(loc=0, scale=np.sqrt(sample_variance1),
size=sample_size1)
    sample2 = np.random.normal(loc=0, scale=np.sqrt(sample_variance2),
size=sample_size2)

    # Perform the F-test
    F_value, df_num, df_den, p_value = variance_ratio_test(sample1,
sample2)

    # Output the results
    print("Sample Variance 1:", sample_variance1)
    print("Sample Variance 2:", sample_variance2)
    print("F-value:", F_value)
    print("Degrees of freedom (numerator):", df_num)
    print("Degrees of freedom (denominator):", df_den)
    print("p-value:", p_value)

if __name__ == "__main__":
    main()
```

```
Sample Variance 1: 25
Sample Variance 2: 20
F-value: 9.385483069468613
Degrees of freedom (numerator): 9
Degrees of freedom (denominator): 14
p-value: 0.005501846884736198
```

Q45. The following data represent the waiting times in minutes at two different restaurants on a Saturday night: Restaurant A: 24, 25, 28, 23, 22, 20, 27; Restaurant B: 31, 33, 35, 30, 32, 36. Conduct an F-test at the 5% significance level to determine if the variances are significantly different.

```
# Ans: 45 def variance_ratio_test(arr1, arr2):


    """
    Perform a variance ratio test (F-test) on two arrays of data.

    Parameters:
        arr1 (array-like): First array of data.
        arr2 (array-like): Second array of data.

    Returns:
        F_value (float): The F-value from the variance ratio test.
        df_num (int): Degrees of freedom for the numerator.
        df_den (int): Degrees of freedom for the denominator.
        p_value (float): The corresponding p-value for the test.
    """
    # Perform the variance ratio test (F-test)
    F_value, p_value = f_oneway(arr1, arr2)
    # Calculate degrees of freedom for the numerator and denominator
    df_num = len(arr1) - 1
    df_den = len(arr2) - 1

    return F_value, df_num, df_den, p_value


def main():
    # Set seed for reproducibility
    np.random.seed(42)

    # Waiting times data for two restaurants
    waiting_times_restaurant_a = [24, 25, 28, 23, 22, 20, 27]
    waiting_times_restaurant_b = [31, 33, 35, 30, 32, 36]

    # Sample variances
    sample_variance_a = np.var(waiting_times_restaurant_a, ddof=1)          #
Use ddof=1 for unbiased sample variance
```

```
        sample_variance_b = np.var(waiting_times_restaurant_b, ddof=1)

        # Perform the F-test
        F_value, df_num, df_den, p_value =
variance_ratio_test(waiting_times_restaurant_a,
waiting_times_restaurant_b)

        # Output the results
        print("Sample Variance A:", sample_variance_a)
        print("Sample Variance B:", sample_variance_b)
        print("F-value:", F_value)
        print("Degrees of freedom (numerator):", df_num)
        print("Degrees of freedom (denominator):", df_den)
        print("p-value:", p_value)

if __name__ == "__main__":
        main()
```

```
Sample Variance A: 7.80952380952381
Sample Variance B: 5.366666666666667
F-value: 36.42289051820553
Degrees of freedom (numerator): 6
Degrees of freedom (denominator): 5
p-value: 8.48759371471029e-05
```

Q46. The following data represent the test scores of two groups of students: Group A: 80, 85, 90, 92, 87, 83; Group B: 75, 78, 82, 79, 81, 84. Conduct an F-test at the 1% significance level to determine if the variances are significantly different.

```
#   Ans:   46   def   variance_ratio_test(arr1,

arr2):
        """
        Perform a variance ratio test (F-test) on two arrays of data.

        Parameters:
                arr1 (array-like): First array of data.
                arr2 (array-like): Second array of data.
```

```
    Returns:
        F_value (float): The F-value from the variance ratio test.
        df_num (int): Degrees of freedom for the numerator.
        df_den (int): Degrees of freedom for the denominator.
        p_value (float): The corresponding p-value for the test.
    """ # Perform the variance ratio test (F-test) F_value, p_value =
    f_oneway(arr1, arr2)

    # Calculate degrees of freedom for the numerator and denominator
    df_num = len(arr1) - 1
    df_den = len(arr2) - 1

    return F_value, df_num, df_den, p_value


def main():
    # Set seed for reproducibility
    np.random.seed(42)

    # Test scores data for two groups of students
    group_a_scores = [80, 85, 90, 92, 87, 83]
    group_b_scores = [75, 78, 82, 79, 81, 84]

    # Sample variances
    sample_variance_a = np.var(group_a_scores, ddof=1)          # Use ddof=1
for unbiased sample variance
    sample_variance_b = np.var(group_b_scores, ddof=1)
    # Perform the F-test
    F_value, df_num, df_den, p_value =

variance_ratio_test(group_a_scores, group_b_scores)

    # Output the results
    print("Sample Variance Group A:", sample_variance_a)
    print("Sample Variance Group B:", sample_variance_b)
    print("F-value:", F_value)
    print("Degrees of freedom (numerator):", df_num)
    print("Degrees of freedom (denominator):", df_den)
    print("p-value:", p_value)

if __name__ == "__main__":
    main()
```

```
Sample Variance Group A: 19.766666666666667
Sample Variance Group B: 10.166666666666666
F-value: 8.040089086859687
Degrees of freedom (numerator): 5
Degrees of freedom (denominator): 5
p-value: 0.017684171924487787
```