Question 1: What are Type I and Type II errors in hypothesis testing, and how do they impact decision-making?
Answer-

Type I error = false alarm, rejecting a true null → risk controlled by α.
Type II error = missed detection, failing to reject a false null → risk controlled by β (power = 1 – β).
Decision-making involves balancing these errors based on **consequences** of each type of error in that context. Good experimental design (sample size, significance level) tries to minimize both, but often with a **trade-off**.

Question 2: What is the P-value in hypothesis testing, and how should it be interpreted in the context of the null hypothesis?
Answer-

P-value = the probability of observing your data (or more extreme) **assuming the null hypothesis is true**. A **low p-value** → evidence *against* the null → may reject H_0.A **high p-value** → data are consistent with H_0 → fail to reject H_0, but don't "accept" it as true. Interpret p-values *in context* (with α, effect size, confidence intervals) — they're not the whole picture.

Question 3:Explain the difference between a Z-test and a T-test, including when to use each.
Answer -

Feature	**Z-Test**	**T-Test**
Population Standard Deviation (σ)	Known	Unknown (estimated from sample)
Distribution	Standard normal (Z)	Student's t-distribution
Sample Size	Typically large ((n \ge 30))	Small or large; especially for small samples
Degrees of Freedom (df)	Not used	(df = n-1) (one-sample), or adjusted for two-sample tests
Formula (one-sample)	(Z = \frac{\bar{x} -	(t = \frac{\bar{x} - \mu_0}{s /

\mu_0}{\sigma / \sqrt{n}})   \sqrt{n}})

Z-Test → Known σ, large sample
T-Test → Unknown σ, especially small sample
The t-test is more commonly used in practice because population variance is rarely known.

Question 4:What is a confidence interval, and how does the margin of error influence its width and interpretation?
Answer -

A confidence interval (CI) is a range of values that is likely to contain an unknown population parameter, such as a mean, with a specified level of confidence (e.g., 95%). The margin of error (MOE) is a value that represents the uncertainty of the estimate; it is directly added to and subtracted from the sample statistic to create the CI, making the CI wider as the MOE increases. A larger margin of error leads to a wider, less precise interval, while a smaller margin of error results in a narrower, more precise interval.

Question 5: Describe the purpose and assumptions of an ANOVA test. How does it extend hypothesis testing to more than two groups?
Answer-

An ANOVA test determines if there are statistically significant differences between the means of three or more groups by partitioning variance into "between-group" and "within-group" components. Its assumptions include normality within groups, homogeneity of variances (equal variances across groups), and independent observations. It extends hypothesis testing beyond two groups by comparing the ratio of between-group variance to within-group variance (the F-statistic), which is more efficient and less prone to Type I errors than multiple t-tests.

Question 6: Write a Python program to perform a one-sample Z-test and interpret the result for a given dataset.

Answer -

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

# --- Sample dataset ---
data = np.array([88, 92, 94, 94, 96, 97, 97, 97, 99, 99,
                 105, 109, 109, 109, 110, 112, 112, 113, 114, 115])

# --- Hypothesized population mean ---
mu_0 = 100        # Null hypothesis: population mean = 100
sigma = 15        # Known population standard deviation
alpha = 0.05      # Significance level

# --- Sample statistics ---
n = len(data)
sample_mean = np.mean(data)

# --- Compute Z statistic ---
z_stat = (sample_mean - mu_0) / (sigma / np.sqrt(n))

# --- Compute two-tailed p-value ---
p_value = 2 * (1 - norm.cdf(abs(z_stat)))

# --- Display results ---
print("One-Sample Z-Test Results")
print("------------------------")
print(f"Sample mean = {sample_mean:.2f}")
print(f"Z statistic = {z_stat:.3f}")
print(f"p-value = {p_value:.4f}")

# --- Interpret the result ---
if p_value < alpha:
    print(f"\nSince p-value ({p_value:.4f}) < alpha ({alpha}), reject the null
hypothesis.")
    print("Conclusion: The sample mean is significantly different from the
hypothesized mean.")
else:
    print(f"\nSince p-value ({p_value:.4f}) >= alpha ({alpha}), fail to reject the
null hypothesis.")
    print("Conclusion: There is not enough evidence to say the mean differs from
the hypothesized mean.")
```

Question 7:Simulate a dataset from a binomial distribution (n = 10, p = 0.5) using NumPy and plot the histogram.

Answer -

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

# --- Simulation parameters ---
n = 10        # number of trials per experiment
p = 0.5       # probability of success
size = 1000   # number of simulated experiments

# --- Simulate binomial dataset ---
data = np.random.binomial(n=n, p=p, size=size)

# --- Print basic statistics ---
print("First 20 simulated values:", data[:20])
print("Mean of simulated data:", np.mean(data))
print("Standard deviation:", np.std(data))

# --- Plot histogram ---
plt.hist(data, bins=range(0, n+2), edgecolor='black', density=True)
plt.title("Binomial Distribution Simulation (n=10, p=0.5)")
plt.xlabel("Number of Successes")
plt.ylabel("Relative Frequency")
plt.xticks(range(0, n+1))
plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.show()
```

Question 8: Generate multiple samples from a non-normal distribution and implement the Central Limit Theorem using Python.

Answer -

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

# --- Parameters ---
population_size = 100000
sample_size = 50      # size of each sample
num_samples = 1000    # number of samples to draw

# --- Generate a non-normal population (Exponential distribution) ---
population = np.random.exponential(scale=2.0, size=population_size)

# Plot the population distribution
plt.figure(figsize=(8,4))
plt.hist(population, bins=50, color='skyblue', edgecolor='black', density=True)
plt.title("Population Distribution (Exponential)")
plt.xlabel("Value")
plt.ylabel("Density")
plt.show()

# --- Generate sample means ---
sample_means = []
for _ in range(num_samples):
    sample = np.random.choice(population, size=sample_size, replace=False)
    sample_means.append(np.mean(sample))

sample_means = np.array(sample_means)

# --- Plot the sampling distribution of the mean ---
plt.figure(figsize=(8,4))
plt.hist(sample_means, bins=30, color='salmon', edgecolor='black', density=True)
plt.title(f"Sampling Distribution of the Mean (n={sample_size}, {num_samples}
samples)")
plt.xlabel("Sample Mean")
plt.ylabel("Density")
plt.show()

# --- Print statistics ---
print(f"Population mean: {np.mean(population):.2f}, std:
{np.std(population):.2f}")
print(f"Mean of sample means: {np.mean(sample_means):.2f}, std of sample means:
{np.std(sample_means):.2f}")
```

Question 9: Write a Python function to calculate and visualize the confidence interval for a sample mean.

Answer -

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

def confidence_interval(data, confidence=0.95):
    """
    Calculate and visualize the confidence interval for the sample mean.

    Parameters:
    - data: array-like, sample data
    - confidence: float, confidence level (default 0.95)

    Returns:
    - ci_lower, ci_upper: lower and upper bounds of the confidence interval
    """
    n = len(data)
    mean = np.mean(data)
    std_err = np.std(data, ddof=1) / np.sqrt(n)  # sample standard error
    z_score = norm.ppf(0.5 + confidence / 2)     # two-tailed Z-score
    margin_error = z_score * std_err

    ci_lower = mean - margin_error
    ci_upper = mean + margin_error

    print(f"Sample mean = {mean:.2f}")
    print(f"{confidence*100:.0f}% Confidence Interval = ({ci_lower:.2f},
{ci_upper:.2f})")

    # --- Visualization ---
    plt.figure(figsize=(8,4))
    plt.bar(0, mean, yerr=margin_error, capsize=10, color='skyblue',
edgecolor='black')
    plt.xticks([0], ['Sample Mean'])
    plt.ylabel('Value')
    plt.title(f'{int(confidence*100)}% Confidence Interval for the Sample Mean')
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.show()

    return ci_lower, ci_upper

# --- Example usage ---
data = [88, 92, 94, 94, 96, 97, 97, 97, 99, 99,
        105, 109, 109, 109, 110, 112, 112, 113, 114, 115]

ci_lower, ci_upper = confidence_interval(data, confidence=0.95)

Sample mean = 102.05
```

```
95% Confidence Interval = (98.86, 105.24)
```

Question 10: Perform a Chi-square goodness-of-fit test using Python to compare observed and expected distributions, and explain the outcome.

Answer -

```
import numpy as np
from scipy.stats import chi2

# --- Example Data ---
# Observed frequencies (e.g., counts of outcomes)
observed = np.array([18, 22, 20, 15, 25])

# Expected frequencies (based on theoretical distribution)
expected = np.array([20, 20, 20, 20, 20])

# --- Chi-square test statistic ---
chi_square_stat = np.sum((observed - expected)**2 / expected)

# --- Degrees of freedom ---
df = len(observed) - 1

# --- p-value ---
p_value = 1 - chi2.cdf(chi_square_stat, df)

# --- Display results ---
print("Chi-Square Goodness-of-Fit Test")
print("--------------------------------")
print(f"Chi-square statistic = {chi_square_stat:.3f}")
print(f"Degrees of freedom = {df}")
print(f"p-value = {p_value:.4f}")

# --- Interpretation ---
alpha = 0.05
if p_value < alpha:
    print(f"\nSince p-value ({p_value:.4f}) < alpha ({alpha}), reject the null
hypothesis.")
    print("Conclusion: The observed frequencies differ significantly from the
expected frequencies.")
else:
    print(f"\nSince p-value ({p_value:.4f}) >= alpha ({alpha}), fail to reject the
null hypothesis.")
    print("Conclusion: There is no significant difference between the observed and
expected frequencies.")
```