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

Firm A

```
z = (1850 - 1800) / (100 / np.sqrt(50))
pval = 1 - norm.cdf(z)

print("P-Value:",pval)
```

```
P-Value: 0.00020347600872250293
```

```
alpha=0.01

if pval < alpha:
  print("Reject Ho")
else:
  print("Accept Ho")
```

```
Reject Ho
```

```
z = (1900 - 1800) / (100 / np.sqrt(5))
pval = 1 - norm.cdf(z)

print("P-Value:",pval)

alpha = 0.01

if pval < alpha:
  print("Reject Ho")
else:
  print("Accept Ho")
```

```
P-Value: 0.0126736593387341
Accept Ho
```

Using critical value method

```
#p_value=alpha
#1-norm.cdf(z)=0.01
z_critical = norm.ppf(0.99)

z_critical
```

```
np.float64(2.3263478740408408)
```

```
x = (z_critical*(100/np.sqrt(50))) + 1800
x
```

```
np.float64(1832.8995271426638)
```

Firm B

```
z_critical = norm.ppf(0.99)
x = (z_critical*(100/np.sqrt(5))) + 1800
x
```

```
np.float64(1904.0374397133487)
```

A fitness App claims that its users walk an average of 8,000 steps per day. A random sample of 30 users showed an average of 7,600 steps per day with a standard deviation of 1,200 steps. Conduct a left-tailed Z-test at a 5% significance level to determine if the App's claim is supported. What is the p-value?

```
# Given information
population_mean =  8000 # average steps per day
population_std_dev = 1200  # standard deviation
sample_size = 30  # sample size
sample_mean = 7600  # sample mean
alpha = 0.05 # significance level
```

```
# Calculate the z-score
z_score = (sample_mean - population_mean) / (population_std_dev / np.sqrt(sample_size))

z_score
```

```
np.float64(-1.8257418583505536)
```

```
pval = norm.cdf(z_score)

pval
```

```
np.float64(0.033944577430914516)
```

```
if pval < alpha:
    conclusion = "Reject the null hypothesis."
else:
    conclusion = "Fail to reject the null hypothesis."

print("z-score:", z_score)
print("p-value:", pval)
print(conclusion)
```

```
z-score: -1.8257418583505536
p-value: 0.033944577430914516
Reject the null hypothesis.
```

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

Using Confidence Interval

```
## Ho: u = 1800
## Ha: u > 1800
```

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

```
population_mean = 1800
sample_mean = 1850
pop_std = 100
sample_size = 50
alpha = 0.01
```

```
z = norm.ppf(0.99)
z
```

```
np.float64(2.3263478740408408)
```

```
margin_of_error = z * (pop_std/np.sqrt(sample_size))
margin_of_error
```

```
np.float64(32.899527142663736)
```

```
confidence_interval = (sample_mean - margin_of_error, sample_mean + margin_of_error)
```

```
confidence_interval
```

```
(np.float64(1817.1004728573362), np.float64(1882.8995271426638))
```

```
# Check if the population mean (1800) falls within the confidence interval

if confidence_interval[0] <= population_mean <= confidence_interval[1]:
    print("The population mean falls within the confidence interval. Then we fail to reject the null hypothesis"
else:
    print("The population mean does not fall within the confidence interval. Then we reject the null hypothesis"
```

```
The population mean does not fall within the confidence interval. Then we reject the null hypothesis
```

Firm B

```
population_mean = 1800
sample_mean = 1900
pop_std = 100
sample_size = 5
alpha = 0.01
```

```
z = norm.ppf(0.99)

margin_of_error = z * (pop_std/np.sqrt(sample_size))

confidence_interval = (sample_mean - margin_of_error, sample_mean + margin_of_error)
print(f"Confidence Interval: {confidence_interval}")


# Check if the population mean (1800) falls within the confidence interval

if confidence_interval[0] <= population_mean <= confidence_interval[1]:
    print("The population mean falls within the confidence interval. Then we fail to reject the null hypothesis"
else:
    print("The population mean does not fall within the confidence interval. Then we reject the null hypothesis"
```

```
Confidence Interval: (np.float64(1795.9625602866513), np.float64(2004.0374397133487))
The population mean falls within the confidence interval. Then we fail to reject the null hypothesis
```

```
import numpy as np
from scipy import stats
from statsmodels.stats import power

# Given data
alpha = 0.05  # Significance level (for a two-tailed test)
confidence_level = 1 - (alpha / 2)  # 95% confidence level
sample_size = 30  # Number of chocolate bars in the sample

# Calculate the z-critical value for a 5% significance level (as you did previously)
z_critical = np.abs(round(stats.norm.ppf(1 - alpha/2), 4))

# Calculate the sample mean (average weight of the chocolate bars)
data = [55, 45, 52, 48, 55, 52, 52, 53, 48, 52, 53, 47, 54, 51, 52, 51, 48, 52, 53, 54, 51, 51, 52, 54, 47, 52,
samp_mean = np.mean(data)
samp_std = np.std(data)

# Null hypothesis value (standard weight)
hypo_mean = 50

# Calculate the effect size (difference between sample mean and hypothesized mean)
effect_size = (samp_mean - hypo_mean) / samp_std
print("Effect size:",effect_size)

# Use 'zt_ind_solve_power()' to calculate the power of the z-test
# ratio=0 it implies that the function assumes equal sample sizes in both groups.
# In other words, it assumes that the number of observations in the two groups being compared is the same.
power = power.zt_ind_solve_power(effect_size=effect_size,
                                 nobs1=sample_size,
                                 alpha=alpha,
                                 ratio=0,
                                 alternative='two-sided')

print('Power of the test:', power)
```

```
Effect size: 0.5261336417646574
Power of the test: 0.8216812302268112
```

```
A fast-food restaurant claims that 80% of their customers prefer their new burger over the old one.
In a random sample of 100 customers, 85 said they preferred the new burger.
What is the null and alternative hypothesis?
```

Define the hypotheses

Null Hypothesis (H0): The proportion of satisfied customers equals the target satisfaction level.

Alternative Hypothesis (Ha): The proportion of satisfied customers is not equal to the target satisfaction level.

```
target_satisfaction = 0.70
satisfied_customers = 115
total_customers = 150
```

```
p = target_satisfaction # population proportion
p_hat = satisfied_customers/ total_customers
n = total_customers

Z = (p_hat - p) / np.sqrt((p * (1 - p)) / n)
```

```
p_value = 2 * (1 - stats.norm.cdf(np.abs(Z)))
```

```
p_value
```

```
np.float64(0.07479137758694376)
```

```
alpha = 0.05


if p_value < alpha:
  print("Reject Ho")
else:
  print("Accept Ho")
```

```
Accept Ho
```

You are the manager of an e-commerce website, and you have recently implemented a new web page in hopes of increasing sales.

To evaluate the effectiveness of the new page, you collected data on the conversion rates for both the old and new web pages.

The conversion rate is defined as the proportion of visitors who make a purchase.

For the old web page (Web Page A), you had 1000 visitors, resulting in 50 conversions. For the new web page (Web Page B), you had 500 visitors, resulting in 30 conversions. Now, you want to determine if there is a statistically significant difference in the conversion rates between the old and new web pages

H0: p1=p2 Ha : p1!=p2

```
import statsmodels.api as sm
import numpy as np

# Define the data
conversions = np.array([50, 30])  # Number of conversions for Web Page A and Web Page B # x_1, x_2
visits = np.array([1000, 500])  # Number of visits for Web Page A and Web Page B # n_1, n_2

# Perform the Z-proportions test
z_stat, p_value = sm.stats.proportions_ztest(conversions, visits, alternative='two-sided')

# Print the results
print(f"Z-statistic = {z_stat}")
print(f"P-value = {p_value}")
```

```
Z-statistic = -0.8125338562826986
P-value = 0.4164853677823287
```

```
alpha=0.05
if p_value < alpha :

  print("Reject null")
else:
  print("Accept null")
```

Formula

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

# Step 1: Define the data
# Data for the old web page (Web Page A)
n_1 = 1000 # visits_1
x_1 = 50 # conversions_1

# Data for the new web page (Web Page B)
n_2 = 500 # visits_2
x_2 = 30 # conversions_2

# Step 2: Define the hypotheses
```

```
# Null Hypothesis (H0): Conversion rates are the same.
# Alternative Hypothesis (Ha): Conversion rates are different.
# This is a two-tailed test.
p_1_hat = x_1 / n_1
p_2_hat = x_2 / n_2

# p_hat: is the combined sample proportion for both web pages.
p_hat = (x_1 + x_2) / (n_1 + n_2)

# Step 3: Calculate the test statistic (Z)
Z = (p_1_hat - p_2_hat) / np.sqrt(p_hat * (1 - p_hat) * ((1 / n_1) + (1 / n_2)))

# Step 4: Interpret the test statistic
# Z follows a standard normal distribution. We will calculate the two-tailed p-value next.

# Step 5: Calculate the p-value
p_value = 2 * (1 - stats.norm.cdf(np.abs(Z)))

# Print the results
print(f"Z = {Z}")
print(f"P-value = {p_value}")
```

```
Z = -0.8125338562826986
P-value = 0.4164853677823288
```

```
u=28
sigma=7
n=50

SE=sigma/np.sqrt(50)
```

```
SE
```

```
np.float64(0.9899494936611665)
```

```
probability=norm.cdf(31,loc=28,scale=SE) - norm.cdf(26,loc=28,scale=SE)
```

```
probability
```

```
np.float64(0.9771032071594303)
```

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