DAY – 4

DSA0410 – Fundamentals of Data Science

Lab Experiments:

Name: B Harish Balaji

Reg no: 192424386

Slot: D

16. Scenario: You are working on a project that involves analyzing customer reviews for a product.

You have a dataset containing customer reviews, and your task is to develop a Python program that

calculates the frequency distribution of words in the reviews.

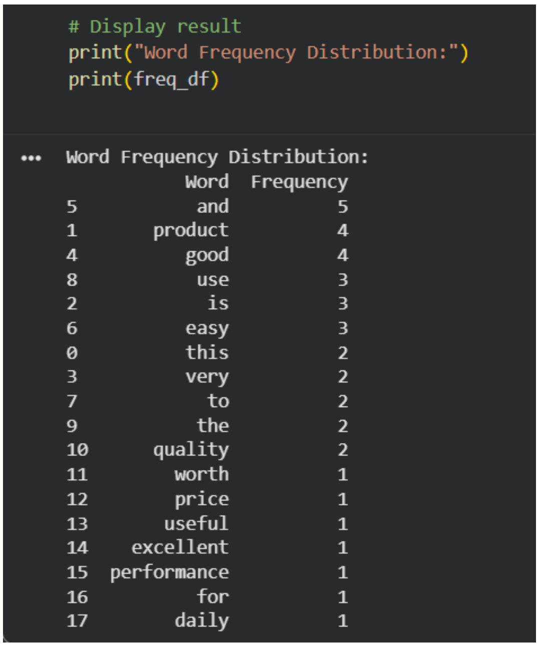
Question: Develop a Python program to calculate the frequency distribution of words in the customer reviews dataset?

CODE:

```
import pandas as pd
import re
from collections import Counter

# Step 1: Create customer reviews dataset
reviews = [
    "This product is very good and easy to use",
    "The product quality is good and worth the price",
    "Easy to use and very useful product",
    "Good quality and excellent performance",
    "This product is easy and good for daily use"
]
reviews_df = pd.DataFrame({
    "Review": reviews
})
```

```
# Step 2: Combine all reviews into one text

all_reviews = " ".join(reviews_df["Review"]).lower()


# Step 3: Remove punctuation and tokenize words

words = re.findall(r'\b\w+\b', all_reviews)


word_frequency = Counter(words)


freq_df = pd.DataFrame(word_frequency.items(), columns=["Word",
"Frequency"]) \

          .sort_values(by="Frequency", ascending=False)


print("Word Frequency Distribution:")

print(freq_df)
```

sample output:



```
# Display result
print("Word Frequency Distribution:")
print(freq_df)
```

```
Word Frequency Distribution:
           Word  Frequency
5           and          5
1       product          4
4          good          4
8           use          3
2            is          3
6          easy          3
0          this          2
3          very          2
7            to          2
9           the          2
10      quality          2
11        worth          1
12        price          1
13       useful          1
14    excellent          1
15  performance          1
16          for          1
17        daily          1
```

17. Scenario: You are a data analyst working for a marketing research company. Your team has

collected a large dataset containing customer feedback from various social media platforms. The

dataset consists of thousands of text entries, and your task is to develop a Python program to

analyze the frequency distribution of words in this dataset. Your program should be able to perform

the following tasks:

- Load the dataset from a CSV file (data.csv) containing a single column named "feedback"with each row representing a customer comment.
- Preprocess the text data by removing punctuation, converting all text to lowercase, and eliminating any stop words (common words like "the," "and," "is," etc. that don't carry significant meaning).
- Calculate the frequency distribution of words in the preprocessed dataset.
- Display the top N most frequent words and their corresponding frequencies, where N is provided as user input.
- Plot a bar graph to visualize the top N most frequent words and their frequencies.

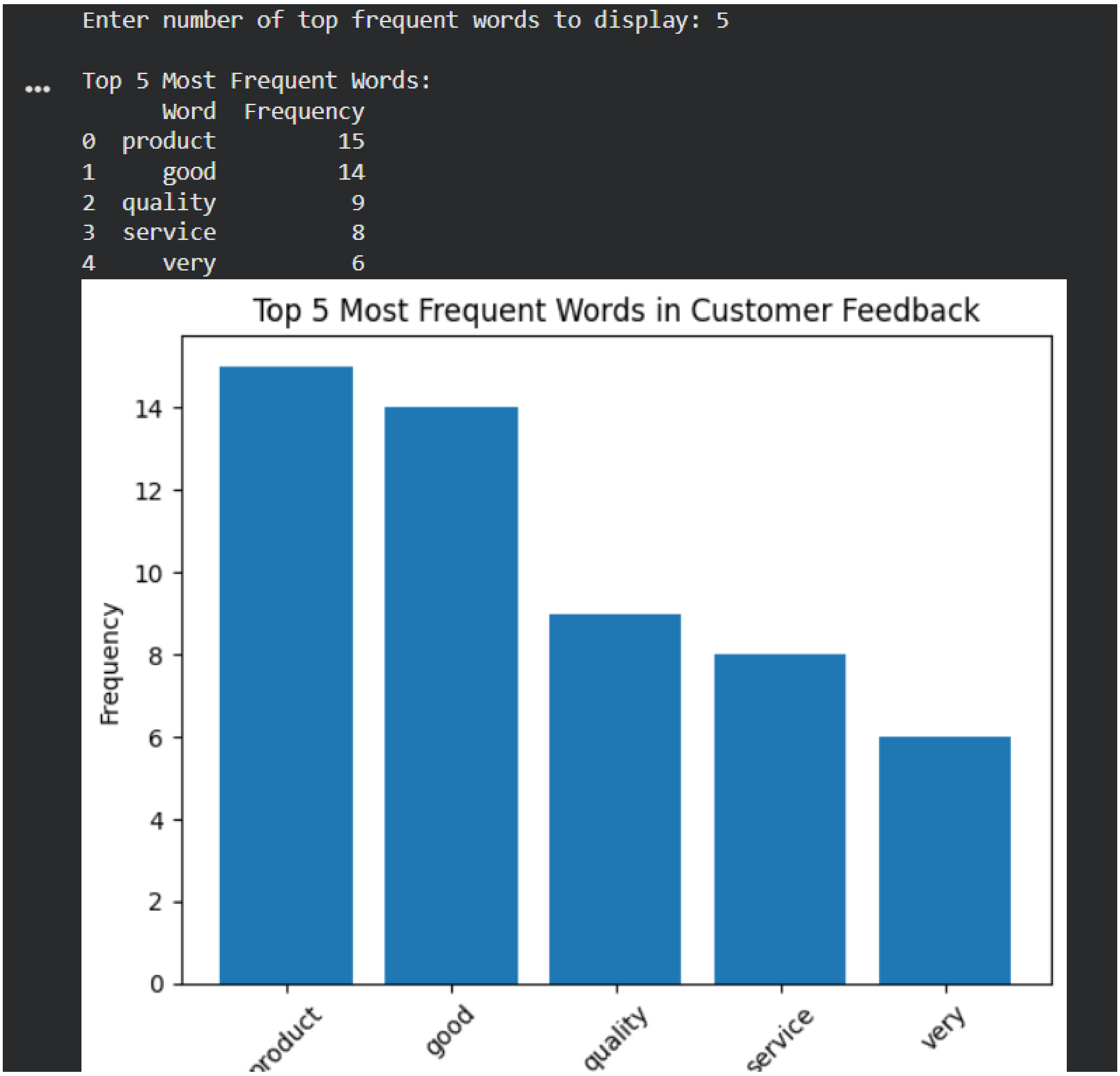
Question: Create a Python program that fulfills these requirements and helps your team gain insights from the customer feedback data.

CODE:

```
import pandas as pd

import re
import matplotlib.pyplot as plt
from collections import Counter

# ------------------------------
# Step 1: Load Dataset
# ------------------------------
df = pd.read_csv("data.csv")

# ------------------------------
# Step 2: Text Preprocessing
# ------------------------------

# Define stop words
stop_words = {
    "the", "and", "is", "in", "to", "of", "a", "for", "on",
    "with", "this", "that", "it", "as", "are", "was", "were",
    "be", "by", "an", "at", "from"
}

# Combine all feedback into one text
text = " ".join(df["feedback"].astype(str)).lower()

# Remove punctuation and tokenize
words = re.findall(r'\b\w+\b', text)
```

```
# Remove stop words
filtered_words = [word for word in words if word not in stop_words]

# ------------------------------
# Step 3: Frequency Distribution
# ------------------------------
word_freq = Counter(filtered_words)

# Step 4: User Input for Top N Words

N = int(input("Enter number of top frequent words to display: "))

top_words = word_freq.most_common(N)

# Convert to DataFrame
freq_df = pd.DataFrame(top_words, columns=["Word", "Frequency"])

print("\nTop", N, "Most Frequent Words:")
print(freq_df)

# Step 5: Bar Plot Visualization
plt.figure()
plt.bar(freq_df["Word"], freq_df["Frequency"])
plt.xlabel("Words")
plt.ylabel("Frequency")
plt.title("Top " + str(N) + " Most Frequent Words in Customer Feedback")
plt.xticks(rotation=45)
plt.show()
```

sample output:



```
Enter number of top frequent words to display: 5

Top 5 Most Frequent Words:
      Word  Frequency
0  product         15
1     good         14
2  quality          9
3  service          8
4     very          6
```

18. Suppose a hospital tested the age and body fat data for 18 randomly selected adults with the

following result.

Question:

Calculate the mean, median and standard deviation of age and %fat using Pandas.

Draw the boxplots for age and %fat.

Draw a scatter plot and a q-q plot based on these two variables

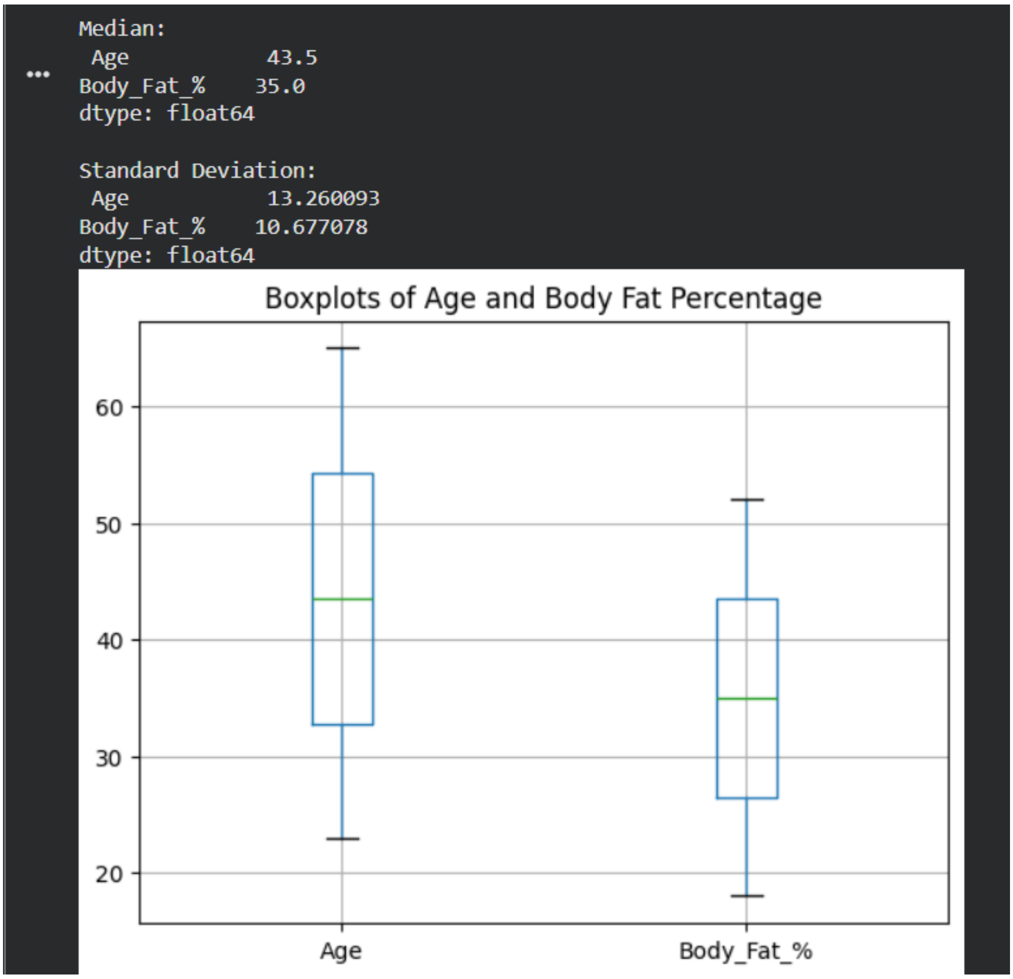
Code:

```
import pandas as pd

import matplotlib.pyplot as plt

from scipy import stats


# Create dataset

data = {

   "Age": [23,25,28,30,32,35,38,40,42,45,48,50,52,55,58,60,62,65],

   "Body_Fat_%": [18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52]

}


df = pd.DataFrame(data)


# Mean, Median, Standard Deviation
```

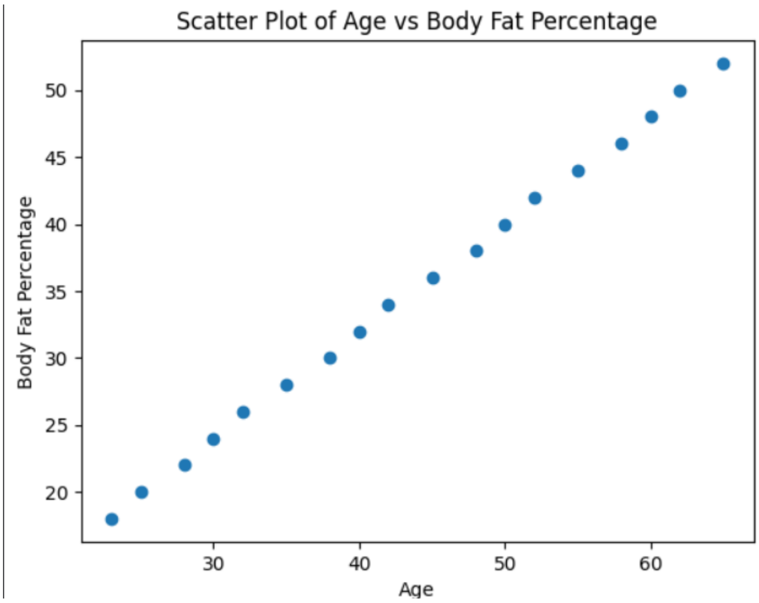
```
print("Mean:\n", df.mean())
print("\nMedian:\n", df.median())
print("\nStandard Deviation:\n", df.std())

# ------------------------------
# Boxplots
# ------------------------------
plt.figure()
df.boxplot(column=["Age", "Body_Fat_%"])
plt.title("Boxplots of Age and Body Fat Percentage")
plt.show()

# ------------------------------
# Scatter Plot
# ------------------------------
plt.figure()
plt.scatter(df["Age"], df["Body_Fat_%"])
plt.xlabel("Age")
plt.ylabel("Body Fat Percentage")
plt.title("Scatter Plot of Age vs Body Fat Percentage")
plt.show()

# Q-Q Plots
# ------------------------------
```

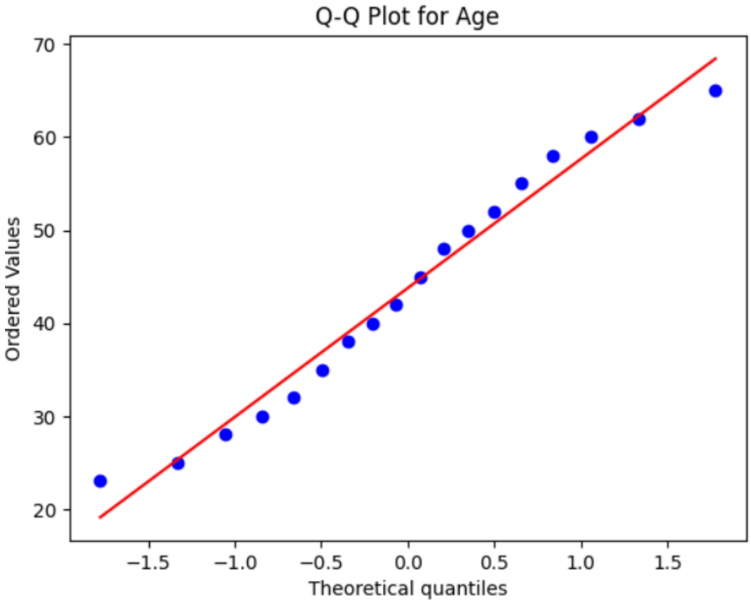
```
plt.figure()

stats.probplot(df["Age"], plot=plt)

plt.title("Q-Q Plot for Age")

plt.show()


plt.figure()

stats.probplot(df["Body_Fat_%"], plot=plt)

plt.title("Q-Q Plot for Body Fat Percentage")

plt.show()
```

sample output:



```
Median:
 Age           43.5
Body_Fat_%    35.0
dtype: float64

Standard Deviation:
 Age           13.260093
Body_Fat_%    10.677078
dtype: float64
```





19. Scenario:

You are a medical researcher investigating the effectiveness of a new drug in reducing blood

pressure. You conduct a clinical trial with a sample of 50 patients who were randomly assigned to

receive either the new drug or a placebo. After measuring their blood pressure levels at the end of the trial, you obtain the data for both groups. Now, you want to determine the confidence intervals

for the mean reduction in blood pressure for both the drug and placebo groups.

Question:

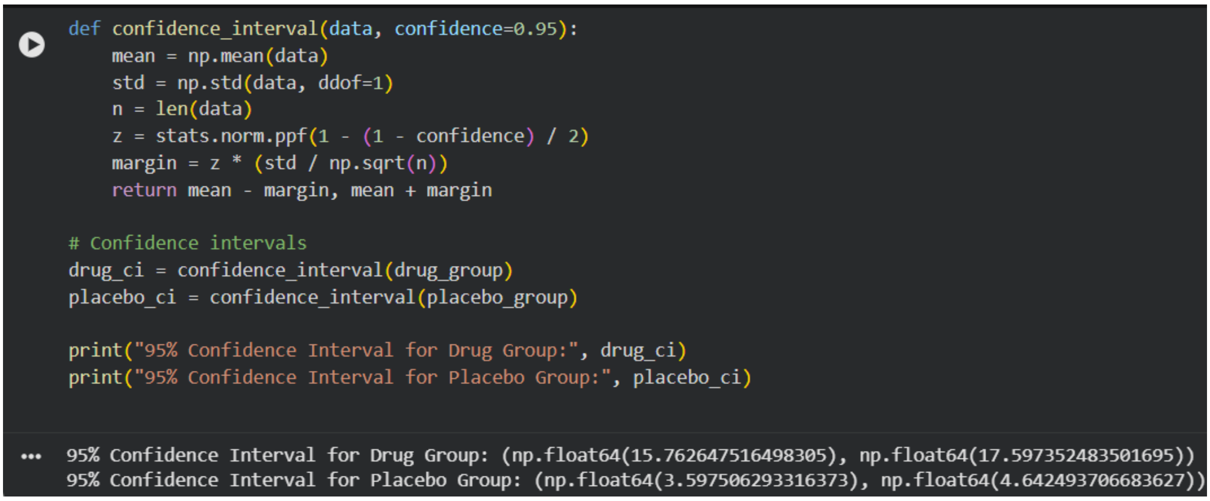
"What is the 95% confidence interval for the mean reduction in blood pressure for patients who

received the new drug? Also, what is the 95% confidence

CODE:

```
import numpy as np

from scipy import stats


# Sample size

n = 25


# Blood pressure reduction data (mmHg)

drug_group = np.array([

   12, 15, 14, 16, 18, 17, 13, 19, 20, 16,

   14, 15, 17, 18, 16, 19, 21, 20, 18, 17,

   16, 15, 14, 18, 19

])


placebo_group = np.array([
   2, 3, 4, 5, 6, 3, 4, 5, 6, 4,
   3, 2, 5, 6, 4, 3, 5, 6, 4, 3,
   2, 4, 5, 6, 3
])

# Function to calculate 95% confidence interval
def confidence_interval(data, confidence=0.95):
   mean = np.mean(data)
   std = np.std(data, ddof=1)
   n = len(data)
   z = stats.norm.ppf(1 - (1 - confidence) / 2)
   margin = z * (std / np.sqrt(n))
   return mean - margin, mean + margin

# Confidence intervals
drug_ci = confidence_interval(drug_group)
placebo_ci = confidence_interval(placebo_group)

print("95% Confidence Interval for Drug Group:", drug_ci)
print("95% Confidence Interval for Placebo Group:", placebo_ci)
```

sample output:



```
def confidence_interval(data, confidence=0.95):
    mean = np.mean(data)
    std = np.std(data, ddof=1)
    n = len(data)
    z = stats.norm.ppf(1 - (1 - confidence) / 2)
    margin = z * (std / np.sqrt(n))
    return mean - margin, mean + margin

# Confidence intervals
drug_ci = confidence_interval(drug_group)
placebo_ci = confidence_interval(placebo_group)

print("95% Confidence Interval for Drug Group:", drug_ci)
print("95% Confidence Interval for Placebo Group:", placebo_ci)
```

```
95% Confidence Interval for Drug Group: (np.float64(15.762647516498305), np.float64(17.597352483501695))
95% Confidence Interval for Placebo Group: (np.float64(3.597506293316373), np.float64(4.642493706683627))
```

20. Scenario:

You are a data scientist working for an e-commerce company. The marketing team has conducted

an A/B test to evaluate the effectiveness of two different website designs (A and B) in terms of

conversion rate. They randomly divided the website visitors into two groups, with one group

experiencing design A and the other experiencing design B. After a week of data collection, you

now have the conversion rate data for both groups. You want to determine whether there is a

statistically significant difference in the mean conversion rates between the two website designs.

Question:

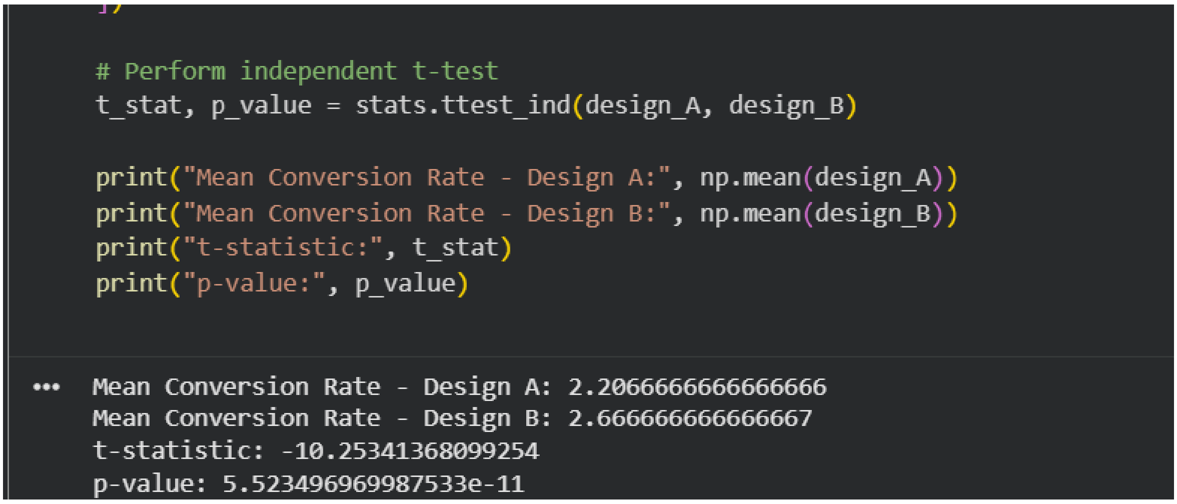
"Based on the data collected from the A/B test, is there a statistically significant difference in the mean conversion rates between website design A and website design B?"

CODE:

```
import numpy as np

from scipy import stats


# Conversion rate data (%)

design_A = np.array([

    2.1, 2.3, 2.0, 2.4, 2.2, 2.1, 2.3, 2.2, 2.4, 2.1,

    2.0, 2.3, 2.2, 2.4, 2.1

])


design_B = np.array([

    2.6, 2.7, 2.5, 2.8, 2.6, 2.7, 2.5, 2.8, 2.6, 2.7,

    2.6, 2.8, 2.7, 2.6, 2.8

])


# Perform independent t-test

t_stat, p_value = stats.ttest_ind(design_A, design_B)


print("Mean Conversion Rate - Design A:", np.mean(design_A))

print("Mean Conversion Rate - Design B:", np.mean(design_B))

print("t-statistic:", t_stat)
```

print("p-value:", p_value)

sample output:



```
])

# Perform independent t-test
t_stat, p_value = stats.ttest_ind(design_A, design_B)

print("Mean Conversion Rate - Design A:", np.mean(design_A))
print("Mean Conversion Rate - Design B:", np.mean(design_B))
print("t-statistic:", t_stat)
print("p-value:", p_value)
```

```
Mean Conversion Rate - Design A: 2.2066666666666666
Mean Conversion Rate - Design B: 2.6666666666666667
t-statistic: -10.25341368099254
p-value: 5.523496969987533e-11
```