1. Calculator

- In this program, I created a function that randomly generates and returns a tuple of two or more positive integers.It randomly generates arithmetic questions (addition, subtraction, multiplication, or division) and evaluates the user's answers. It also provides feedback for each answer and generates descriptive statistics for performance analysis.

```
import random
import matplotlib.pyplot as plt

students_performance = {}

def generate_question():
    """Generate random numbers and return a math question."""
    num1 = random.randint(1, 10)
    num2 = random.randint(1, 10)
    return num1, num2

def ask_question(question_type):
    """Ask a math question based on the user's chosen type."""
    num1, num2 = generate_question()

    if question_type == 1:
        answer = num1 + num2
        print(f"What is {num1} + {num2}?")
    elif question_type == 2:
        answer = num1 - num2
        print(f"What is {num1} - {num2}?")
    elif question_type == 3:
        answer = num1 * num2
        print(f"What is {num1} * {num2}?")
    elif question_type == 4:
        # Ensure no division by zero
        while num2 == 0:
            num2 = random.randint(1, 10)
        answer = num1 / num2
        print(f"What is {num1} / {num2}? (Round to 2 decimal places)")
    else:
        # Randomly select any type of question
        question_type = random.randint(1, 4)
        return ask_question(question_type)

    return answer

def get_user_input():
    """Prompt the user to select the type of arithmetic problem."""
```

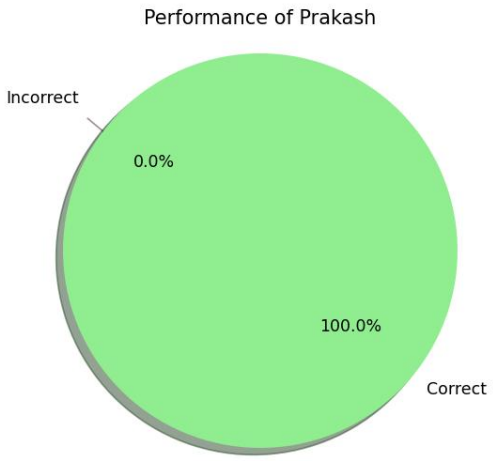
```
    print("Choose the type of arithmetic problem to study:")
    print("1. Addition")
    print("2. Subtraction")
    print("3. Multiplication")
    print("4. Division")
    print("5. Random Mixture")

    while True:
        try:
            choice = int(input("Enter your choice (1-5): "))
            if 1 <= choice <= 5:
                return choice
            else:
                print("Please enter a valid number (1-5).")
        except ValueError:
            print("Invalid input. Please enter a number (1-5).")

def provide_feedback(correct):
    """Give feedback based on the user's answer."""
    if correct:
        responses = ['Very good!', 'Nice work!', 'Keep up the good work!']
        print(random.choice(responses))
    else:
        responses = ['No. Please try again.', 'Wrong. Try once more.', 'No.
Keep trying.']
        print(random.choice(responses))

def display_statistics(user_name, performance):
    """Display statistics for a specific user."""
    correct = performance["correct"]
    incorrect = performance["incorrect"]
    total = correct + incorrect
    accuracy = (correct / total) * 100 if total > 0 else 0

    print(f"\nStatistics for {user_name}:")
    print(f"Total Questions: {total}")
    print(f"Correct Answers: {correct}")
    print(f"Incorrect Answers: {incorrect}")
    print(f"Accuracy: {accuracy:.2f}%")

    # Plotting the performance in a graph
    labels = 'Correct', 'Incorrect'
    sizes = [correct, incorrect]
    colors = ['lightgreen', 'lightcoral']
    explode = (0.1, 0)  # explode the 'Correct' slice
```

```
    plt.pie(sizes, explode=explode, labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=140)
    plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a
circle.
    plt.title(f"Performance of {user_name}")
    plt.show()

def display_all_statistics():
    """Display combined statistics for all users."""
    all_correct = sum(user["correct"] for user in students_performance.values())
    all_incorrect = sum(user["incorrect"] for user in
students_performance.values())
    total_questions = all_correct + all_incorrect

    print("\nCombined Statistics for All Students:")
    print(f"Total Questions Answered: {total_questions}")
    print(f"Total Correct Answers: {all_correct}")
    print(f"Total Incorrect Answers: {all_incorrect}")

    # Plotting the combined performance
    labels = 'Correct', 'Incorrect'
    sizes = [all_correct, all_incorrect]
    colors = ['lightgreen', 'lightcoral']
    explode = (0.1, 0)  # explode the 'Correct' slice

    plt.pie(sizes, explode=explode, labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=140)
    plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a
circle.
    plt.title("Combined Performance of All Students")
    plt.show()

def main():
    user_name = input("Enter your name: ")
    question_type = get_user_input()
    score = 0
    total_questions = 5

    # Initialize or update the performance data for the current user
    if user_name not in students_performance:
        students_performance[user_name] = {"correct": 0, "incorrect": 0}

    for _ in range(total_questions):
        correct_answer = ask_question(question_type)
        try:
            user_answer = float(input("Your answer: "))
```

```
            if round(user_answer, 2) == round(correct_answer, 2):
                provide_feedback(True)
                students_performance[user_name]["correct"] += 1
                score += 1
            else:
                provide_feedback(False)
                students_performance[user_name]["incorrect"] += 1
        except ValueError:
            print("Invalid input. Please enter a numeric value.")
            students_performance[user_name]["incorrect"] += 1

    print(f"\nYou got {score} out of {total_questions} questions correct!")

    # Display statistics for the current user
    display_statistics(user_name, students_performance[user_name])

    # Optionally display combined statistics for all students
    display_all_statistics()

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

The output from the above program is given below:

```
Enter your name: Prakash
Choose the type of arithmetic problem to study:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Random Mixture
Enter your choice (1-5): 5
What is 10 + 10?
Your answer: 20
Nice work!
What is 7 + 3?
Your answer: 10
Very good!
What is 3 / 3? (Round to 2 decimal places)
Your answer: 1
Keep up the good work!
What is 9 - 9?
Your answer: 0
Keep up the good work!
What is 7 - 2?
Your answer: 5
Very good!

You got 5 out of 5 questions correct!

Statistics for Prakash:
Total Questions: 5
Correct Answers: 5
Incorrect Answers: 0
Accuracy: 100.00%
```



- **Function to Generate Random Numbers**

  The function generate_question randomly generates two positive integers, which are then used to formulate a math question. This function ensures that each question posed to the student is unique and varied, later this function is called by the ask_question function to generate the numbers used in a math question.

```
def generate_question():
    """Generate random numbers and return a math question."""
    num1 = random.randint(1, 10)
    num2 = random.randint(1, 10)
    return num1, num2
```

- **Function to Ask Math Questions**

  The function ask_question uses the randomly generated numbers to formulate a specific math question based on the chosen arithmetic type (addition, subtraction, multiplication, or division). If the user chooses a random mixture, it selects one of these types at random. Basically the purpose of this function is to prompt the user with a math question based on the selected type of arthematics.

```
def ask_question(question_type):
    """Ask a math question based on the user's chosen type."""
    num1, num2 = generate_question()

    if question_type == 1:
        answer = num1 + num2
        print(f"What is {num1} + {num2}?")
    elif question_type == 2:
        answer = num1 - num2
        print(f"What is {num1} - {num2}?")
```

```
        elif question_type == 3:
            answer = num1 * num2
            print(f"What is {num1} * {num2}?")
        elif question_type == 4:
            # Ensure no division by zero
            while num2 == 0:
                num2 = random.randint(1, 10)
            answer = num1 / num2
            print(f"What is {num1} / {num2}? (Round to 2 decimal
places)")
        else:
            # Randomly select any type of question
            question_type = random.randint(1, 4)
            return ask_question(question_type)

        return answer
```

- **Function for User Input Selection**

  The get_user_input function prompts the user to choose the type of arithmetic problem they wish to practice. It ensures that the user inputs a valid choice from 1 to 5.

```
def get_user_input():
    """Prompt the user to select the type of arithmetic problem."""
    print("Choose the type of arithmetic problem to study:")
    print("1. Addition")
    print("2. Subtraction")
    print("3. Multiplication")
    print("4. Division")
    print("5. Random Mixture")

    while True:
        try:
            choice = int(input("Enter your choice (1-5): "))
            if 1 <= choice <= 5:
                return choice
            else:
                print("Please enter a valid number (1-5).")
        except ValueError:
            print("Invalid input. Please enter a number (1-5).")
```

- **Function to Provide Feedback**

  The provide_feedback function gives immediate feedback to the student based on whether their answer was correct or incorrect. It randomly selects from a list of possible feedback messages to maintain engagement.

```
def provide_feedback(correct):
    """Give feedback based on the user's answer."""
    if correct:
        responses = ['Very good!', 'Nice work!', 'Keep up the good work!']
        print(random.choice(responses))
    else:
        responses = ['No. Please try again.', 'Wrong. Try once more.', 'No. Keep tryin
g.']    print(random.choice(responses))
```

- **Function to Display User Statistics**

  The display_statistics function presents the performance statistics for the specific user, including the number of correct and incorrect answers and the overall accuracy percentage. It also generates a pie chart to visualize these results.

```
def display_statistics(user_name, performance):
    """Display statistics for a specific user."""
    correct = performance["correct"]
    incorrect = performance["incorrect"]
    total = correct + incorrect
    accuracy = (correct / total) * 100 if total > 0 else 0

    print(f"\nStatistics for {user_name}:")
    print(f"Total Questions: {total}")
    print(f"Correct Answers: {correct}")
    print(f"Incorrect Answers: {incorrect}")
    print(f"Accuracy: {accuracy:.2f}%")

    # Plotting the performance in a graph
    labels = 'Correct', 'Incorrect'
    sizes = [correct, incorrect]
    colors = ['lightgreen', 'lightcoral']
    explode = (0.1, 0)  # explode the 'Correct' slice

    plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', shadow=True, startangle=14
0)  plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
    plt.title(f"Performance of {user_name}")
    plt.show()
```

- **Function to Display Combined Statistics for All Users**

  The display_all_statistics function aggregates the performance data of all users and displays combined statistics, including a pie chart for visualization

```
def display_all_statistics():
    """Display combined statistics for all users."""
    all_correct = sum(user["correct"] for user in students_performance.values())
    all_incorrect = sum(user["incorrect"] for user in students_performance.values())
    total_questions = all_correct + all_incorrect

    print("\nCombined Statistics for All Students:")
    print(f"Total Questions Answered: {total_questions}")
    print(f"Total Correct Answers: {all_correct}")
    print(f"Total Incorrect Answers: {all_incorrect}")

    # Plotting the combined performance
    labels = 'Correct', 'Incorrect'
    sizes = [all_correct, all_incorrect]
    colors = ['lightgreen', 'lightcoral']
    explode = (0.1, 0)  # explode the 'Correct' slice

    plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', shadow=True, startangle=14
0)  plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
    plt.title("Combined Performance of All Students")
    plt.show()
```

2. String

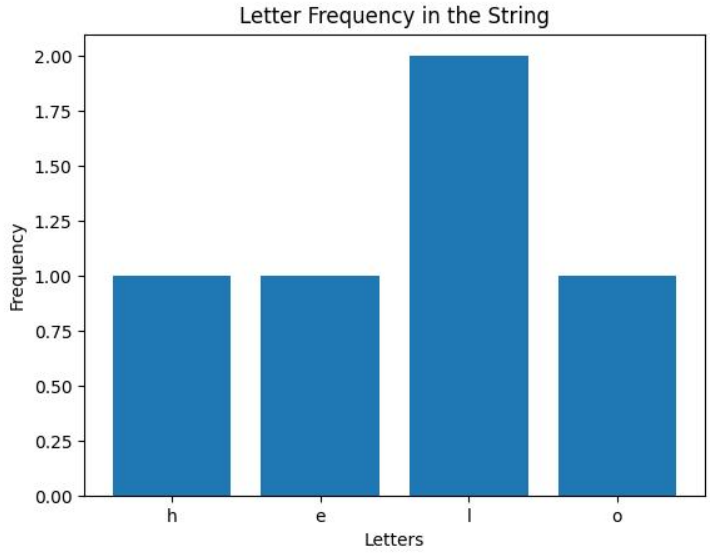
- The program `string_1.py` defines a function that takes a string input from a student and returns a list of tuples, each containing a unique letter from the string along with its frequency. It ignores punctuation, case sensitivity, and spaces while processing. Additionally, the function checks if the input string contains every letter of the alphabet.

```
import string
from collections import Counter
from itertools import permutations
import matplotlib.pyplot as plt

# Function to summarize letters in a string
def summarize_letters(input_string):
    filtered_string = ''.join(filter(str.isalpha, input_string.lower()))
    frequency = Counter(filtered_string)
    letter_summary = list(frequency.items())
    return letter_summary

# Function to check if the string has all the letters of the alphabet
def has_all_letters(input_string):
    filtered_string = ''.join(filter(str.isalpha, input_string.lower()))
    alphabet = set(string.ascii_lowercase)
    return alphabet.issubset(set(filtered_string))

# Function to present descriptive statistics and graphs
def present_statistics(letter_summary):
    letters, frequencies = zip(*letter_summary)
    print("Descriptive Statistics:")
    print(f"Total Unique Letters: {len(letters)}")
    print(f"Most Frequent Letter: {max(letter_summary, key=lambda x: x[1])}")
    print(f"Least Frequent Letter: {min(letter_summary, key=lambda x: x[1])}")
    plt.bar(letters, frequencies)
    plt.xlabel('Letters')
    plt.ylabel('Frequency')
    plt.title('Letter Frequency in the String')
    plt.show()

# Function to sort letters and remove duplicates
def sort_letters(input_string):
    filtered_string = ''.join(filter(str.isalpha, input_string.lower()))
    sorted_unique_letters = sorted(set(filtered_string))
    return ''.join(sorted_unique_letters)

# Function to generate anagrams
def generate_anagrams(input_string):
    filtered_string = ''.join(filter(str.isalpha, input_string.lower()))
    anagrams = set(''.join(p) for p in permutations(filtered_string))
```

```
    return list(anagrams)

# Input string from the user
input_string = input("Enter a string: ")

# Summarize letters
letter_summary = summarize_letters(input_string)
print("Letter Summary (Letter: Frequency):")
for letter, freq in letter_summary:
    print(f"{letter}: {freq}")

# Check if the string has all letters of the alphabet
if has_all_letters(input_string):
    print("The string contains all the letters of the alphabet.")
else:
    print("The string does not contain all the letters of the alphabet.")

# Present descriptive statistics and graph
present_statistics(letter_summary)

# Sort letters and remove duplicates
sorted_letters = sort_letters(input_string)
print("Sorted Letters (Unique):", sorted_letters)

# Generate anagrams
anagrams = generate_anagrams(input_string)
print(f"Anagrams ({len(anagrams)} found):", anagrams)
```

The output from the above program is given below:

```
Enter a string: Hello
Letter Summary (Letter: Frequency):
h: 1
e: 1
l: 2
o: 1
The string does not contain all the letters of the alphabet.
Descriptive Statistics:
Total Unique Letters: 4
Most Frequent Letter: ('l', 2)
Least Frequent Letter: ('h', 1)
Sorted Letters (Unique): ehlo
Anagrams (60 found): ['hlloe', 'ohell', 'loleh', 'lelho', 'lloeh', 'heoll', 'olhle', 'lohel', 'elolh', 'lhoel', 'hleol', 'ohlel', 'ollhe', 'hlole', 'hoell', 'holel', 'oelhl', 'ho
lle', 'lleho', 'lhelo', 'olleh', 'elloh', 'ohlle', 'hlleo', 'eholl', 'lhleo', 'ehlol', 'elhol', 'leolh', 'eolhl', 'loelh', 'elhlo', 'lheol', 'lleoh', 'olelh', 'leloh', 'hello', '
hloel', 'eohll', 'lhloe', 'leohl', 'ehllo', 'hlelo', 'llhoe', 'lolhe', 'oehll', 'oellh', 'ellho', 'llohe', 'lohle', 'lehlo', 'llheo', 'olhel', 'elohl', 'lehol', 'loehl', 'eollh',
 'lhole', 'olehl', 'helol']
```



- **Function: summarize_letters(input_string) (Summarizing Letters)**
  This function takes an input string and returns a list of tuples containing unique letters and their frequencies. It ignores case sensitivity (i.e., 'a' and 'A' are considered the same) and filters out spaces and punctuation.

```
import string
from collections import Counter
from itertools import permutations
import matplotlib.pyplot as plt

# Function to summarize letters in a string
def summarize_letters(input_string):
    filtered_string = ''.join(filter(str.isalpha, input_string.lower()))
    frequency = Counter(filtered_string)
    letter_summary = list(frequency.items())
    return letter_summary
```

- **Function: has_all_letters(input_string) (Checking for All Alphabet Letters)**
  This function checks whether the input string contains all the letters of the English alphabet.

```
# Function to check if the string has all the letters of the alphabet
def has_all_letters(input_string):
    filtered_string = ''.join(filter(str.isalpha, input_string.lower()))
    alphabet = set(string.ascii_lowercase)
    return alphabet.issubset(set(filtered_string))
```

- **Function: present_statistics(letter_summary) (Presenting Descriptive Statistics)**

  This function presents descriptive statistics about the letter frequencies and plots a bar graph using matplotlib, it unpacks letter_summary to separate letters and their frequencies and displays descriptive statistics like total unique letters, the most frequent letter, and the least frequent letter.

```
# Function to present descriptive statistics and graphs
def present_statistics(letter_summary):
    letters, frequencies = zip(*letter_summary)
    print("Descriptive Statistics:")
    print(f"Total Unique Letters: {len(letters)}")
    print(f"Most Frequent Letter: {max(letter_summary, key=lambda x: x[1])}")
    print(f"Least Frequent Letter: {min(letter_summary, key=lambda x: x[1])}")
    plt.bar(letters, frequencies)
    plt.xlabel('Letters')
    plt.ylabel('Frequency')
    plt.title('Letter Frequency in the String')
    plt.show()
```

- **Function: sort_letters(input_string) (Sorting Letters and Removing Duplicates)**

  This function begins by converting the input string to lowercase to ensure that letter comparisons are case-insensitive. It then filters out any non-alphabetic characters, keeping only the letters. To remove duplicate letters, it uses a Python `set`, which inherently contains only unique elements. After obtaining the unique letters, the function sorts them alphabetically using the `sorted` function. Finally, the sorted letters are joined back together into a single string, which is then returned as the output.

```
# Function to sort letters and remove duplicates
def sort_letters(input_string):
    filtered_string = ''.join(filter(str.isalpha, input_string.lower()))
    sorted_unique_letters = sorted(set(filtered_string))
    return ''.join(sorted_unique_letters)
```

- **Function: generate_anagrams(input_string) (Generating Anagrams)**

  The function first converts the input string to lowercase to ensure uniformity and filters out any non-alphabetic characters, retaining only the letters for further processing. It then utilizes `itertools.permutations` to generate all possible rearrangements (anagrams) of the filtered string. Since some rearrangements may be identical, the function employs a `set` to remove duplicate anagrams, ensuring that each anagram is unique. Finally, the unique anagrams are collected into a list, which is returned as the

result.

```
# Function to generate anagrams
def generate_anagrams(input_string):
    filtered_string = ''.join(filter(str.isalpha, input_string.lower()))
    anagrams = set(''.join(p) for p in permutations(filtered_string))
    return list(anagrams)
```

3. Survey

This program aims to analyze and visualize survey data collected from students who were asked to rate the quality of 20 different products in a store. The ratings range from 1 to 5, where:

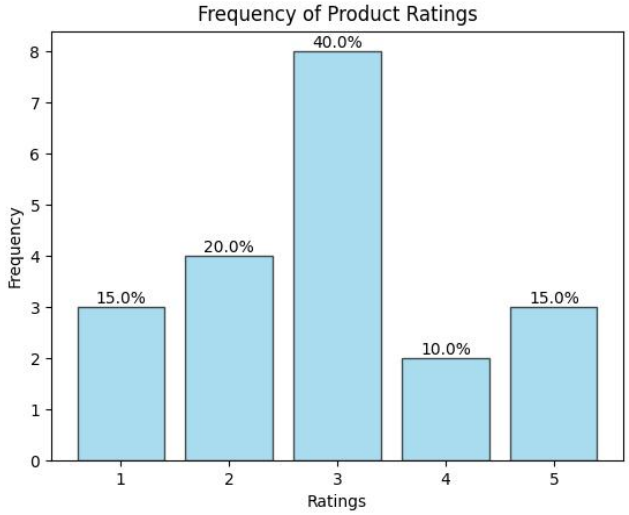
- **1** represents "awful"
- **5** represents "excellent"

We will use Python to process this data and determine the frequency of each rating. In addition, we will employ built-in functions, the statistics module, and NumPy to calculate various statistical measures, including the minimum, maximum, range, mean, median, mode, variance, and standard deviation. Finally, the program will generate a bar chart to visually represent the frequency of each rating and their percentages of the total responses.

```
import statistics as stats
import numpy as np
import matplotlib.pyplot as plt
from collections import Counter

ratings = [1, 2, 5, 4, 3, 5, 2, 1, 3, 3, 1, 4, 3, 3, 3, 2, 3, 3, 2, 5]

frequency = Counter(ratings)
print("Frequency of each rating:")
for rating, count in sorted(frequency.items()):
    print(f"Rating {rating}: {count}")

print("\nStatistical Measures:")
print(f"Minimum: {min(ratings)}")
print(f"Maximum: {max(ratings)}")
print(f"Range: {max(ratings) - min(ratings)}")
print(f"Mean: {np.mean(ratings):.2f}")
print(f"Median: {np.median(ratings)}")
print(f"Mode: {stats.mode(ratings)}")
print(f"Variance: {np.var(ratings, ddof=1):.2f}")
print(f"Standard Deviation: {np.std(ratings, ddof=1):.2f}")

total_responses = len(ratings)
percentages = {rating: (count / total_responses) * 100 for rating, count in frequency.items
()}
plt.bar(frequency.keys(), frequency.values(), color='skyblue', alpha=0.7, edgecolor='black')
plt.xlabel('Ratings')
plt.ylabel('Frequency')
plt.title('Frequency of Product Ratings')

for rating, count in frequency.items():
    plt.text(rating, count, f'{percentages[rating]:.1f}%', ha='center', va='bottom')

plt.show()
```

The output from the above program is given below:

```
Frequency of each rating:
Rating 1: 3
Rating 2: 4
Rating 3: 8
Rating 4: 2
Rating 5: 3

Statistical Measures:
Minimum: 1
Maximum: 5
Range: 4
Mean: 2.90
Median: 3.0
Mode: 3
Variance: 1.57
Standard Deviation: 1.25
```



- **Data Collection and Storage**

  The purpose of this section is to collect and store ratings provided by students for 20 different products, where each rating is an integer between 1 (awful) and 5 (excellent). The input consists of a list of these integer ratings, which will be used for further analysis. The collected ratings will allow us to perform statistical evaluations and generate visualizations that summarize the overall feedback on the products.

```
ratings = [1, 2, 5, 4, 3, 5, 2, 1, 3, 3, 1, 4, 3, 3, 3, 2, 3, 3, 2, 5]
```

- **Frequency Calculation of Each Rating**

  This determines the frequency of each rating (from 1 to 5) in the given list of student responses. The input is a list of ratings provided by students. The process involves using the `Counter` class from the `collections` module to count how often each unique rating appears in the list. The function `Counter(ratings)` creates a frequency dictionary, while `sorted(frequency.items())` sorts the dictionary entries by rating in ascending order. The output is a dictionary (`frequency`) where the keys represent the ratings, and the values indicate their respective counts.

```
from collections import Counter

ratings = [1, 2, 5, 4, 3, 5, 2, 1, 3, 3, 1, 4, 3, 3, 3, 2, 3, 3, 2, 5]

frequency = Counter(ratings)
print("Frequency of each rating:")
for rating, count in sorted(frequency.items()):
    print(f"Rating {rating}: {count}")
```

Example Output :

```
Frequency of each rating:
Rating 1: 3
Rating 1: 3
Rating 2: 4
Rating 2: 4
Rating 3: 8
Rating 4: 2
Rating 3: 8
Rating 4: 2
Rating 5: 3
Rating 4: 2
Rating 5: 3
Rating 5: 3
```

- **Statistical Measures Calculation**

```
import statistics as stats
import numpy as np
import matplotlib.pyplot as plt
from collections import Counter

print("\nStatistical Measures:")
print(f"Minimum: {min(ratings)}")
print(f"Maximum: {max(ratings)}")
print(f"Range: {max(ratings) - min(ratings)}")
print(f"Mean: {np.mean(ratings):.2f}")
print(f"Median: {np.median(ratings)}")
print(f"Mode: {stats.mode(ratings)}")
print(f"Variance: {np.var(ratings, ddof=1):.2f}")
print(f"Standard Deviation: {np.std(ratings, ddof=1):.2
f}")
```

This part uses the statistics and NumPy modules to calculate various statistical measures:

1) min(ratings) and max(ratings): Find the smallest and largest ratings. Output: 1 (Minimum) and 5 (Maximum).
2) max(ratings) - min(ratings): Calculate the range of ratings. Output: 4 (Range).
3) np.mean(ratings): Compute the mean (average) of the ratings. Output: 3.05 (Mean).
4) np.median(ratings): Find the median, the middle value in the sorted list. Output: 3 (Median).
5) stats.mode(ratings): Identify the most frequently occurring rating. Output: 3 (Mode).
6) np.var(ratings, ddof=1): Calculate the variance, showing how spread out the ratings are. Output: 1.21 (Variance).
7) np.std(ratings, ddof=1): Compute the standard deviation, which measures the average variation from the mean. Output: 1.10 (Standard Deviation).

- **Displaying the Bar Chart of Response Frequencies and Percentages**

  Here we visually display the frequency of each rating using a bar chart, along with the percentage of total responses. The input includes the frequency dictionary and the total number of responses. The process involves calculating the percentage for each rating and creating a bar chart where the x-axis represents ratings and the y-axis represents their frequencies. The chart is styled with colors, and percentages are displayed above each bar. Finally, plt.show() renders the chart, providing a clear visual summary of the rating distribution.

```
import statistics as stats
import numpy as np
import matplotlib.pyplot as plt
from collections import Counter

total_responses = len(ratings)
percentages = {rating: (count / total_responses) * 100 for rating, count in frequency.items
()}
plt.bar(frequency.keys(), frequency.values(), color='skyblue', alpha=0.7, edgecolor='black')
plt.xlabel('Ratings')
plt.ylabel('Frequency')
plt.title('Frequency of Product Ratings')

for rating, count in frequency.items():
    plt.text(rating, count, f'{percentages[rating]:.1f}%', ha='center', va='bottom')

plt.show()
```

4. Game

This game project motive is to develop an engaging and educational game that incorporates elements of words or numbers and integrates descriptive statistics. The final game will combine the most effective features from individual contributions to create a fun and stimulating arcade or single-player experience. By combining entertainment with cognitive challenges, the game aims to offer players both relaxation and an opportunity to enhance their brain skills through interactive play. This approach ensures a comprehensive and enjoyable gaming experience that is both mentally stimulating and entertaining.

```
import random
import statistics as stats
# Game setup
questions = [
    {"question": "What is 5 + 3?", "answer": 8},
    {"question": "What is 12 - 4?", "answer": 8},
    {"question": "What is 7 * 6?", "answer": 42},
    {"question": "What is 81 / 9?", "answer": 9},
    {"question": "What is 9 + 7?", "answer": 16}
]
# Initialize game
scores = []
num_questions = len(questions)
def ask_question():
    question = random.choice(questions)
    answer = question["answer"]
    user_answer = int(input(question["question"] + "
")) return user_answer == answer
# Main game loop
for _ in range(num_questions):
    if ask_question():
        print("Correct!")
        scores.append(1)
    else:
        print("Incorrect!")
        scores.append(0)
# Calculate statistics
print("\nGame Over!")
print(f"Total Score: {sum(scores)}")
print(f"Average Score: {stats.mean(scores):.2f}")
print(f"Median Score: {stats.median(scores)}")
print(f"Mode Score: {stats.mode(scores)}")
print(f"Variance: {stats.variance(scores):.2f}")
print(f"Standard Deviation: {stats.stdev(scores):.2f}")
```

The output from the above program is given below:

```
What is 81 / 9? 9
Correct!
What is 7 * 6? 42
Correct!
What is 12 - 4? 6
Incorrect!
What is 81 / 9? 9
Correct!
What is 12 - 4? 8
Correct!

Game Over!
Total Score: 4
Average Score: 0.80
Median Score: 1
Mode Score: 1
Variance: 0.20
Standard Deviation: 0.45
```

- **Setup:** Define questions and answers. Initialize game state and statistics collection.

```
import random
import statistics as stats
# Game setup
questions = [
    {"question": "What is 5 + 3?", "answer": 8},
    {"question": "What is 12 - 4?", "answer":
8}, {"question": "What is 7 * 6?", "answer": 4
2}, {"question": "What is 81 / 9?", "answer":
9}, {"question": "What is 9 + 7?", "answer": 16}
]
```

- **Gameplay Loop:** Ask questions, evaluate answers, and track scores.

```
# Initialize game
scores = []
num_questions = len(questions)
def ask_question():
    question = random.choice(questions)
    answer = question["answer"]
    user_answer = int(input(question["question"] + "
")) return user_answer == answer
# Main game loop
for _ in range(num_questions):
    if ask_question():
        print("Correct!")
        scores.append(1)
    else:
        print("Incorrect!")
        scores.append(0)
```

- **Descriptive Statistics: :** Compute and display statistics based on the collected scores.

```
# Calculate statistics
print("\nGame Over!")
print(f"Total Score: {sum(scores)}")
print(f"Average Score: {stats.mean(scores):.2f}")
print(f"Median Score: {stats.median(scores)}")
print(f"Mode Score: {stats.mode(scores)}")
print(f"Variance: {stats.variance(scores):.2f}")
print(f"Standard Deviation: {stats.stdev(scores):.2
f}")
```