1.a. Write a Python program to determine whether a given number is a prime number or not. Also enhance the program to find the first N prime numbers, where N is user input

```
def is_prime(num):
    if num <= 1:
        return False
    for i in range(2, int(num ** 0.5) + 1):
        if num % i == 0:
            return False
    return True

def find_first_n_primes(n):
    primes = []
    num = 2  # Starting number to check for prime
    while len(primes) < n:
        if is_prime(num):
            primes.append(num)
        num += 1
    return primes

if __name__ == "__main__":
    # Check if a number is prime
    number = int(input("Enter a number to check if it's prime: "))
    if is_prime(number):
        print(f"{number} is a prime number.")
    else:
        print(f"{number} is not a prime number.")

    # Find first N prime numbers
    n = int(input("Enter how many prime numbers you want to find: "))
    first_n_primes = find_first_n_primes(n)
    print(f"The first {n} prime numbers are: {first_n_primes}")
```

Output 1:

Enter a number to check if it's prime: 9
9 is not a prime number.
Enter how many prime numbers you want to find: 18
The first 18 prime numbers are: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61]

Output 2:

Enter a number to check if it's prime: 17
17 is a prime number.
Enter how many prime numbers you want to find: 17
The first 17 prime numbers are: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59]

1.b Write a Python program to accept a positive integer and determine if it is a perfect square. Also display a message accordingly.

```
import math
# Function to check if a number is a perfect square
def is_perfect_square(num):
    if num < 0:
        return False
    sqrt_num = math.sqrt(num)
    return sqrt_num.is_integer()

# Input from user
number = int(input("Enter a positive integer: "))

# Check if the number is positive
if number <= 0:
    print("Please enter a positive integer.")
else:
    # Check if the number is a perfect square and display the message
    if is_perfect_square(number):
        print(f"{number} is a perfect square.")
    else:
        print(f"{number} is not a perfect square.")
```

Output 1:

Enter a positive integer: -10
Please enter a positive integer.

Output 2:

Enter a positive integer: 10
10 is not a perfect square.

Output 3:

Enter a positive integer: 25
25 is a perfect square.

2. Create a list to represent a shopping cart, containing items (strings).
a. Add items to the cart (using append())
b. Remove an item from the cart (using remove())
c. Create a function add_multiple_items that takes a list of items and uses extend to add them to the shopping cart in one go.
d. Implement a function capitalize_first_letter that modifies an item's name (capitalizing the first letter) before adding it to the cart.

```
# Initialize an empty shopping cart list
shopping_cart = []

# a. Add item to the cart
def add_item(item):
    shopping_cart.append(item)
    print(f"'{item}' added to the cart.")

# b. Remove item from the cart
def remove_item(item):
    if item in shopping_cart:
        shopping_cart.remove(item)
        print(f"'{item}' removed from the cart.")
    else:
        print(f"'{item}' is not in the cart.")

# c. Add multiple items to the cart using extend
def add_multiple_items(items):
    shopping_cart.extend(items)
    print(f"Multiple items added to the cart: {', '.join(items)}")

# d. Capitalize the first letter of the item name before adding it to the cart
def capitalize_first_letter(item):
    item = item.capitalize()
    shopping_cart.append(item)
    print(f"'{item}' added to the cart with the first letter capitalized.")
```

```
# Function to collect multiple items from the user
def get_multiple_items():
    items = []
    num_items = int(input("How many items would you like to add? "))
    for _ in range(num_items):
        item = input("Enter the item: ")
        items.append(item)
    return items
# Menu to choose operations
def display_menu():
    while True:
        print("\nShopping Cart Menu:")
        print("1. Add a single item")
        print("2. Remove an item")
        print("3. Add multiple items")
        print("4. Capitalize and add an item")
        print("5. Exit")

        choice = input("Choose an operation (1-5): ")

        if choice == '1':
            item = input("Enter the item to add: ")
            add_item(item)
        elif choice == '2':
            item = input("Enter the item to remove: ")
            remove_item(item)
        elif choice == '3':
            items = get_multiple_items()
            add_multiple_items(items)
        elif choice == '4':
            item = input("Enter the item to capitalize and add: ")
            capitalize_first_letter(item)
        elif choice == '5':
            print("\nExiting the program.")
            break
```

```
        else:
            print("Invalid choice, please try again.")
# Running the menu
display_menu()
# Print the final shopping cart
print("\nFinal Shopping Cart:", shopping_cart)
```

Output:

```
Shopping Cart Menu:
1. Add a single item
2. Remove an item
3. Add multiple items
4. Capitalize and add an item
5. Exit
Choose an operation (1-5): 1
Enter the item to add: Banana
'Banana' added to the cart.

Shopping Cart Menu:
1. Add a single item
2. Remove an item
3. Add multiple items
4. Capitalize and add an item
5. Exit
Choose an operation (1-5): 1
Enter the item to add: Apple
'Apple' added to the cart.

Shopping Cart Menu:
1. Add a single item
2. Remove an item
3. Add multiple items
4. Capitalize and add an item
5. Exit
Choose an operation (1-5): 2
```

Enter the item to remove: Banana
'Banana' removed from the cart.

Shopping Cart Menu:
1. Add a single item
2. Remove an item
3. Add multiple items
4. Capitalize and add an item
5. Exit
Choose an operation (1-5): 3
How many items would you like to add? 3
Enter the item: Grapes
Enter the item: Kiwi
Enter the item: Guava
Multiple items added to the cart: Grapes, Kiwi, Guava

Shopping Cart Menu:
1. Add a single item
2. Remove an item
3. Add multiple items
4. Capitalize and add an item
5. Exit
Choose an operation (1-5): 4
Enter the item to capitalize and add: strawberry
'Strawberry' added to the cart with the first letter capitalized.

Shopping Cart Menu:
1. Add a single item
2. Remove an item
3. Add multiple items
4. Capitalize and add an item
5. Exit
Choose an operation (1-5): 5
Exiting the program.
Final Shopping Cart: ['Apple', 'Grapes', 'Kiwi', 'Guava', 'Strawberry']

3. Write a Python program that functions as a customizable text analyzer. The program should accept a block of text and provide options to choose the type of analysis(word count, character count, Uppercase count and list, Lowercase count and list). Define separate functions for each option and also display the result in user-friendly format.

```
# Function to count words in the text
def count_words(text):
    words = text.split()
    return len(words)

# Function to count characters in the text
def count_characters(text):
    return len(text)

# Function to count uppercase letters and list them
def count_uppercase(text):
    uppercase_letters = [char for char in text if char.isupper()]
    return len(uppercase_letters), uppercase_letters

# Function to count lowercase letters and list them
def count_lowercase(text):
    lowercase_letters = [char for char in text if char.islower()]
    return len(lowercase_letters), lowercase_letters

# Function to display the menu and get user input for analysis type
def display_menu():
    text = input("Enter the block of text: ")

    while True:
        print("\nText Analysis Menu:")
        print("1. Word Count")
        print("2. Character Count")
        print("3. Uppercase Count and List")
        print("4. Lowercase Count and List")
```

```
    print("5. Exit")

    choice = input("Choose an analysis type (1-5): ")

    if choice == '1':
        word_count = count_words(text)
        print(f"Word Count: {word_count}")
    elif choice == '2':
        char_count = count_characters(text)
        print(f"Character Count: {char_count}")
    elif choice == '3':
        uppercase_count, uppercase_list = count_uppercase(text)
        print(f"Uppercase Count: {uppercase_count}")
        print(f"Uppercase Letters: {', '.join(uppercase_list)}")
    elif choice == '4':
        lowercase_count, lowercase_list = count_lowercase(text)
        print(f"Lowercase Count: {lowercase_count}")
        print(f"Lowercase Letters: {', '.join(lowercase_list)}")
    elif choice == '5':
        print("\nExiting the program.")
        break
    else:
        print("Invalid choice, please try again.")

# Running the menu
display_menu()
```

Output:

```
Enter the block of text: Here is the Text to Analyse
Text Analysis Menu:
1. Word Count
2. Character Count
3. Uppercase Count and List
4. Lowercase Count and List
5. Exit
Choose an analysis type (1-5): 1
Word Count: 6

Text Analysis Menu:
1. Word Count
2. Character Count
3. Uppercase Count and List
4. Lowercase Count and List
5. Exit
Choose an analysis type (1-5): 2
Character Count: 27

Text Analysis Menu:
1. Word Count
2. Character Count
3. Uppercase Count and List
4. Lowercase Count and List
5. Exit
Choose an analysis type (1-5): 3
Uppercase Count: 3
Uppercase Letters: H, T, A

Text Analysis Menu:
1. Word Count
2. Character Count
3. Uppercase Count and List
4. Lowercase Count and List
```

5. Exit
Choose an analysis type (1-5): 4
Lowercase Count: 19
Lowercase Letters: e, r, e, i, s, t, h, e, e, x, t, t, o, n, a, l, y, s, e

Text Analysis Menu:
1. Word Count
2. Character Count
3. Uppercase Count and List
4. Lowercase Count and List
5. Exit
Choose an analysis type (1-5): 5

Exiting the program.

4. Create a tuple containing your name, age, and favorite color. Print the elements individually and access the entire tuple.

```
# Asking the user for input
name = input("Enter your name: ")
age = int(input("Enter your age: "))
favorite_color = input("Enter your favorite color: ")

# Creating a tuple with the user inputs
user_tuple = (name, age, favorite_color)

# Printing individual elements
print("\nName:", user_tuple[0])
print("Age:", user_tuple[1])
print("Favorite Color:", user_tuple[2])

# Accessing and printing the entire tuple
print("\nComplete Tuple:", user_tuple)
```

Output:

```
Enter your name: John Doe
Enter your age: 25
Enter your favorite color: Red

Name: John Doe
Age: 25
Favorite Color: Red

Complete Tuple: ('John Doe', 25, 'Red')
```

5. Write a Python program using NumPy that creates a random m x n integer array and prints the shape of the array (number of rows and columns) and also total number of elements in the array.

a. Implement function fill_with_value (e.g., zeros)

b. Implement a function randomly_generate_numbers (within a range) to populate the array.

```
import numpy as np

def create_array(m, n):
    """Create an m x n integer array."""
    return np.zeros((m, n), dtype=int)

def fill_with_value(array, value):
    """Fill the array with a specific value."""
    array.fill(value)

def randomly_generate_numbers(array, low, high):
    """Populate the array with random integers within the specified range [low, high)."""
    array[:] = np.random.randint(low, high, size=array.shape)

def main():
    # Accept m, n, and the value to be filled from the user
    m = int(input("Enter the number of rows (m): "))
    n = int(input("Enter the number of columns (n): "))
    fill_value = int(input("Enter the value to fill the array with: "))

    # Create an m x n array
    my_array = create_array(m, n)

    # Print initial array shape and total elements
    print(f"\nInitial Shape of the array: {my_array.shape}")
    print(f"Total number of elements: {my_array.size}")
```

```
    # Fill the array with a specific value
    fill_with_value(my_array, fill_value)
    print(f"\nArray after filling with the value {fill_value}:")
    print(my_array)

    # Populate the array with random integers in a range
    randomly_generate_numbers(my_array, 1, 100)
    print("\nArray after populating with random numbers (range 1-100):")
    print(my_array)

# Run the program
if __name__ == "__main__":
    main()
```

Output:

```
Enter the number of rows (m): 3
Enter the number of columns (n): 2
Enter the value to fill the array with: 5

Initial Shape of the array: (3, 2)
Total number of elements: 6

Array after filling with the value 5:
[[5 5]
 [5 5]
 [5 5]]

Array after populating with random numbers (range 1-100):
[[53 24]
 [52 17]
 [18 18]]
```

6. Develop a python program to read and print CSV file in the console. After reading the CSV, calculate summary statistics for numerical columns (mean, median, standard deviation) using pandas functions

```
import pandas as pd

# Create a DataFrame with numeric and non-numeric columns
data = {
   "ID": [1, 2, 3, 4],
   "Name": ["Alice", "Bob", "Charlie", "Diana"],
   "Age": [25, 30, 35, 40],
   "Salary": [50000, 60000, 70000, 80000],
   "Department": ["HR", "Finance", "IT", "Marketing"]
}

# Convert to DataFrame
df = pd.DataFrame(data)

# Save the DataFrame to a CSV file
df.to_csv("employee_data.csv", index=False)

print("CSV file with numeric and non-numeric columns created successfully!")

import pandas as pd

def read_and_summarize_csv(file_path):
   try:
      # Read the CSV file
      data = pd.read_csv(file_path)

      # Print the contents of the CSV file
      print("\nContents of the CSV file:")
      print(data)

      # Select numerical columns
      numerical_data = data.select_dtypes(include=['number'])
```

```
        # Calculate and display summary statistics
        print("\nSummary Statistics for Numerical Columns:")
        print("Mean:")
        print(numerical_data.mean())
        print("\nMedian:")
        print(numerical_data.median())
        print("\nStandard Deviation:")
        print(numerical_data.std())

    except FileNotFoundError:
        print("Error: The file was not found. Please check the file path.")
    except pd.errors.EmptyDataError:
        print("Error: The file is empty. Please provide a valid CSV file.")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")

# Example usage
if __name__ == "__main__":
    file_path = "employee_data.csv"  # Path to the CSV file created earlier
    read_and_summarize_csv(file_path)
```

Output:

```
CSV file with numeric and non-numeric columns created successfully!
Contents of the CSV file:
   ID     Name  Age  Salary Department
0   1    Alice   25   50000         HR
1   2      Bob   30   60000    Finance
2   3  Charlie   35   70000         IT
3   4    Diana   40   80000  Marketing

Summary Statistics for Numerical Columns:
Mean:
ID          2.5
Age         32.5
Salary    65000.0
```

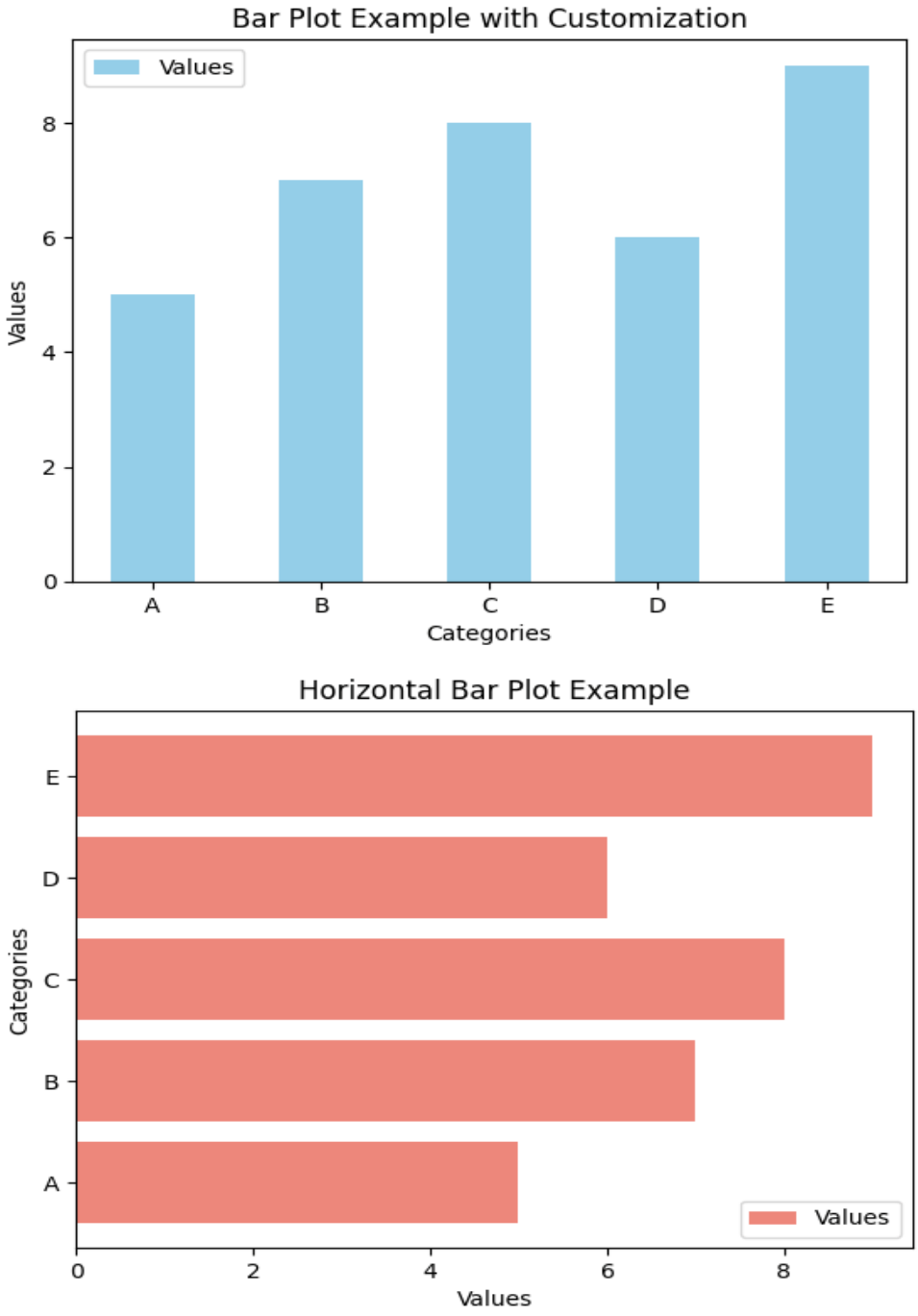
```
dtype: float64

Median:
ID          2.5
Age         32.5
Salary    65000.0
dtype: float64

Standard Deviation:
ID          1.290994
Age         6.454972
Salary    12909.944487
dtype: float64
```

7 Write a Python program to demonstrate how to draw a bar plot using Matplotlib.

```
import matplotlib.pyplot as plt
# Data for the bar plot
categories = ['A', 'B', 'C', 'D', 'E']
values = [5, 7, 8, 6, 9]

# Create a bar plot with customized features
plt.bar(categories, values, color='skyblue', width=0.5, label='Values')  # Colored bars and custom width

# Add labels and title
plt.xlabel('Categories')  # Label for the x-axis
plt.ylabel('Values')      # Label for the y-axis
plt.title('Bar Plot Example with Customization')  # Title of the plot

# Add a legend
plt.legend()

# Show the bar plot
plt.show()

# Horizontal Bar Plot
plt.barh(categories, values, color='salmon', label='Values')  # Horizontal bars and custom color

# Add labels and title for horizontal bar plot
plt.xlabel('Values')      # Label for the x-axis in horizontal bars
plt.ylabel('Categories')  # Label for the y-axis in horizontal bars
plt.title('Horizontal Bar Plot Example')

# Add a legend for horizontal bars
plt.legend()
```

```
# Show the horizontal bar plot
plt.show()
```

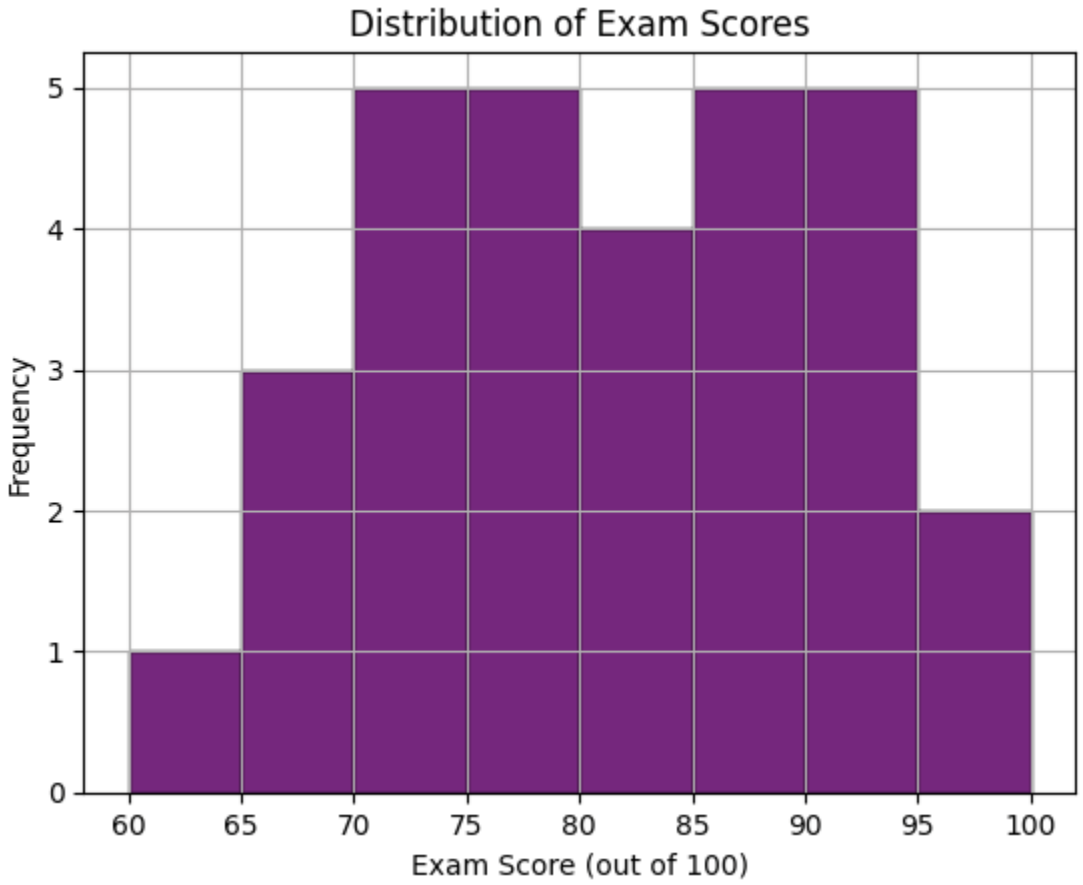
Output:



8 Write a Python program to demonstrate how to draw a histogram plot using Matplotlib.

```
import matplotlib.pyplot as plt
exam_scores = [65, 78, 92, 85, 76, 88, 75, 68, 94, 72, 81, 90, 84, 79, 93, 87, 70, 62, 98, 83, 73, 89, 77, 91, 74, 67, 96, 80, 71, 86]
bins = [60, 65, 70, 75, 80, 85, 90, 95, 100]
plt.hist(exam_scores, bins=bins, color='purple', edgecolor='black')
plt.xlabel('Exam Score (out of 100)')
plt.ylabel('Frequency')
plt.title('Distribution of Exam Scores')
plt.grid(True)
plt.show()
```

Output:



#7. Write a Python program to demonstrate how to draw a bar plot using Matplotlib

```
import matplotlib.pyplot as plt

# Data for the bar plot
categories = ['A', 'B', 'C', 'D', 'E']
values = [5, 7, 8, 6, 9]

# Create the bar plot
plt.bar(categories, values)

# Add labels and title
plt.xlabel('Categories')  # Label for the x-axis
plt.ylabel('Values')      # Label for the y-axis
plt.title('Bar Plot Example')  # Title of the plot

# Show the plot
plt.show()
```