1. **Write a program that takes two numbers as input and performs addition, subtraction, multiplication, and division.**

Answer:

```
# Taking input from user
try:
    num1 = float(input("Enter the first number: "))
    num2 = float(input("Enter the second number: "))

    # Performing operations
    addition = num1 + num2
    subtraction = num1 - num2
    multiplication = num1 * num2
    division = num1 / num2 if num2 != 0 else "Undefined (Division by Zero)"

    # Displaying results
    print(f"Addition: {addition}")
    print(f"Subtraction: {subtraction}")
    print(f"Multiplication: {multiplication}")
    print(f"Division: {division}")

except ValueError:
    print("Please enter valid numbers.")
```

OR

```
# Function to perform operations
def calculate_operations(num1, num2):
    addition = num1 + num2
    subtraction = num1 - num2
    multiplication = num1 * num2
    division = num1 / num2 if num2 != 0 else "Undefined (Division by Zero)"

    return addition, subtraction, multiplication, division

# Taking input from user
try:
    num1 = float(input("Enter the first number: "))
    num2 = float(input("Enter the second number: "))

    # Get results
    add, sub, mul, div = calculate_operations(num1, num2)

    # Display results
```

```
    print(f"Addition: {add}")
    print(f"Subtraction: {sub}")
    print(f"Multiplication: {mul}")
    print(f"Division: {div}")

except ValueError:
    print("Please enter valid numbers.")
```

2. **Write a program to calculate the area of a circle given its radius, rectangle of given breath and length and such regular shapes.**
   Answer:

Using Function:

```
import math

def area_of_circle():
    radius = float(input("Enter the radius of the circle: "))
    return math.pi * radius ** 2

def area_of_rectangle():
    length = float(input("Enter the length of the rectangle: "))
    breadth = float(input("Enter the breadth of the rectangle: "))
    return length * breadth

def area_of_square():
    side = float(input("Enter the side of the square: "))
    return side ** 2

def area_of_triangle():
    base = float(input("Enter the base of the triangle: "))
    height = float(input("Enter the height of the triangle: "))
    return 0.5 * base * height

# Main program
def main():
    print("Select shape to calculate area:")
    print("1. Circle")
    print("2. Rectangle")
    print("3. Square")
    print("4. Triangle")

    choice = input("Enter the number corresponding to your shape (1-4): ")

    if choice == '1':
        area = area_of_circle()
        print(f"The area of the circle is: {area:.2f}")
    elif choice == '2':
        area = area_of_rectangle()
        print(f"The area of the rectangle is: {area:.2f}")
    elif choice == '3':
        area = area_of_square()
        print(f"The area of the square is: {area:.2f}")
    elif choice == '4':
```

```
        area = area_of_triangle()
        print(f"The area of the triangle is: {area:.2f}")
    else:
        print("Invalid choice! Please select a valid shape.")

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

OR

Without using Function

```
import math

# Main program
print("Select shape:")
print("1. Circle")
print("2. Rectangle")
print("3. Square")
print("4. Triangle")

choice = input("Enter the number corresponding to your shape (1-4): ")

if choice == '1':
    # Circle area calculation
    radius = float(input("Enter the radius of the circle: "))
    area_circle = math.pi * radius ** 2
    print(f"The area of the circle is: {area_circle:.2f}")

elif choice == '2':
    # Rectangle area calculation
    length = float(input("Enter the length of the rectangle: "))
    breadth = float(input("Enter the breadth of the rectangle: "))
    area_rectangle = length * breadth
    print(f"The area of the rectangle is: {area_rectangle:.2f}")

elif choice == '3':
    # Square area calculation
    side = float(input("Enter the side of the square: "))
    area_square = side ** 2
    print(f"The area of the square is: {area_square:.2f}")

elif choice == '4':
    # Triangle area calculation
    base = float(input("Enter the base of the triangle: "))
```

```
    height = float(input("Enter the height of the triangle: "))
    area_triangle = 0.5 * base * height
    print(f"The area of the triangle is: {area_triangle:.2f}")

else:
    print("Invalid choice! Please select a valid shape.")
```

3. Write a program to convert temperatures between Celsius and Fahrenheit.

Answer :

```
# Main program for temperature conversion
print("Temperature Conversion")
print("1. Convert Celsius to Fahrenheit")
print("2. Convert Fahrenheit to Celsius")

choice = input("Enter your choice (1 or 2): ")

if choice == '1':
    # Celsius to Fahrenheit conversion
    celsius = float(input("Enter the temperature in Celsius: "))
    fahrenheit = (celsius * 9/5) + 32
    print(f"{celsius:.2f}°C is equal to {fahrenheit:.2f}°F.")

elif choice == '2':
    # Fahrenheit to Celsius conversion
    fahrenheit = float(input("Enter the temperature in Fahrenheit: "))
    celsius = (fahrenheit - 32) * 5/9
    print(f"{fahrenheit:.2f}°F is equal to {celsius:.2f}°C.")

else:
    print("Invalid choice! Please enter 1 or 2.")
```

4. Write a program to check whether a number is odd or even.

Answer:

```
# Input: Ask the user to enter a number
number = int(input("Enter a number: "))

# Check if the number is even or odd
if number % 2 == 0:
    print(f"{number} is an even number.")
else:
    print(f"{number} is an odd number.")
```

5. Write a program to calculate simple interest given principal, rate, and time.

Answer:

```
# Function to calculate simple interest
def calculate_simple_interest(principal, rate, time):
    # Simple Interest formula: SI = (P * R * T) / 100
    simple_interest = (principal * rate * time) / 100
    return simple_interest

# Input values
principal = float(input("Enter the principal amount: "))
rate = float(input("Enter the rate of interest: "))
time = float(input("Enter the time in years: "))

# Calculate Simple Interest
si = calculate_simple_interest(principal, rate, time)

# Output the result
print(f"The Simple Interest is: {si}")
```

Explanation:

- The program uses the simple interest formula: Simple Interest (SI)=P×R×T100\text{Simple Interest (SI)} = \frac{P \times R \times T}{100}Simple Interest (SI)=100P×R×T where:
    - P is the principal amount,
    - R is the rate of interest,
    - T is the time period in years.
- The calculate_simple_interest function performs the calculation, and the result is printed after taking the input values.

6. Write a program to find the maximum of nth

Answer:

To find the maximum of n numbers in Python, we can ask the user for the total number of inputs, and then find the maximum using either Python's built-in max() function or a manual approach using loops.

Approach 1: Using Python's built-in max() function

```
# Function to find the maximum of n numbers
def find_max(numbers):
    return max(numbers)

# Input number of elements
n = int(input("Enter the number of elements: "))

# Input the numbers
numbers = []
for i in range(n):
    num = float(input(f"Enter number {i + 1}: "))
    numbers.append(num)

# Find the maximum
max_number = find_max(numbers)

# Output the result
print(f"The maximum number is: {max_number}")
```

Approach 2: Using a loop to manually find the maximum

```
# Function to find the maximum of n numbers without using built-in max()
def find_max_manual(numbers):
    max_num = numbers[0]
    for num in numbers[1:]:
        if num > max_num:
            max_num = num
    return max_num

# Input number of elements
n = int(input("Enter the number of elements: "))

# Input the numbers
numbers = []
for i in range(n):
    num = float(input(f"Enter number {i + 1}: "))
    numbers.append(num)
```

```
# Find the maximum manually
max_number = find_max_manual(numbers)

# Output the result
print(f"The maximum number is: {max_number}")
```

Explanation:

- **Approach 1**: The program collects n numbers in a list and uses Python's max() function to find the largest number.
- **Approach 2**: A manual approach, iterating through the list and comparing each number to find the maximum value.

Both approaches will give you the maximum of the input numbers.

7. Write a program to perform basic operations on a list (e.g., adding, removing, and searching for elements).ree numbers.

Answer:

```
# Function to display the current list
def display_list(lst):
    print("Current List:", lst)

# Function to add an element to the list
def add_element(lst, element):
    lst.append(element)
    print(f"{element} has been added to the list.")

# Function to remove an element from the list
def remove_element(lst, element):
    if element in lst:
        lst.remove(element)
        print(f"{element} has been removed from the list.")
    else:
        print(f"{element} not found in the list.")

# Function to search for an element in the list
def search_element(lst, element):
    if element in lst:
        print(f"{element} is present in the list at index {lst.index(element)}.")
    else:
        print(f"{element} is not found in the list.")

# Initialize an empty list
numbers = []

# Add three numbers to the list
add_element(numbers, 10)
add_element(numbers, 20)
add_element(numbers, 30)

# Display the list
display_list(numbers)

# Remove a number from the list
remove_element(numbers, 20)

# Display the list after removal
display_list(numbers)
```

```
# Search for a number in the list
search_element(numbers, 10)

# Try searching for a number not in the list
search_element(numbers, 50)
```

Explanation:

1. **Adding an Element**: The add_element function adds an element to the list using the append() method.
2. **Removing an Element**: The remove_element function removes an element using the remove() method, but only if the element exists in the list.
3. **Searching for an Element**: The search_element function searches for an element using Python's in keyword and index() method to return the index if found.
4. **Display List**: The display_list function prints the current state of the list.

The program performs these operations on a list of three numbers (10, 20, 30). It adds elements, removes one, and searches for both existing and non-existing elements.

8. Create a simple calculator that can add, subtract, multiply, and divide two numbers.

Answer:

```
# Function to perform addition
def add(x, y):
    return x + y

# Function to perform subtraction
def subtract(x, y):
    return x - y

# Function to perform multiplication
def multiply(x, y):
    return x * y

# Function to perform division
def divide(x, y):
    if y == 0:
        return "Error! Division by zero."
    return x / y

# Display menu
def display_menu():
    print("Select Operation:")
    print("1. Add")
    print("2. Subtract")
    print("3. Multiply")
    print("4. Divide")

# Main calculator function
def calculator():
    display_menu()

    # Input operation choice
    choice = input("Enter choice (1/2/3/4): ")

    if choice in ['1', '2', '3', '4']:
        # Input numbers
        num1 = float(input("Enter first number: "))
        num2 = float(input("Enter second number: "))

        if choice == '1':
            print(f"The result of {num1} + {num2} = {add(num1, num2)}")

        elif choice == '2':
```

```
            print(f"The result of {num1} - {num2} = {subtract(num1, num2)}")

        elif choice == '3':
            print(f"The result of {num1} * {num2} = {multiply(num1, num2)}")

        elif choice == '4':
            result = divide(num1, num2)
            print(f"The result of {num1} / {num2} = {result}")

    else:
        print("Invalid input! Please select a valid operation.")

# Run the calculator
calculator()
```

Explanation:

1. **Functions for Operations**:
    - add(), subtract(), multiply(), and divide() perform the respective arithmetic operations.
    - The divide() function includes a check to avoid division by zero.
2. **Menu**:
    - The display_menu() function prints the options available to the user.
3. **Calculator Logic**:
    - The program asks the user to choose an operation and then inputs two numbers.
    - Depending on the selected operation, the corresponding function is called to perform the calculation.
4. **Invalid Input Handling**:
    - The program checks if the user's choice is valid (i.e., between 1 and 4). If not, it displays an error message.

This simple calculator can handle the four basic arithmetic operations with two numbers.

9. Write a program to print the Fibonacci sequence up to n terms

Answer:

```
# Function to generate Fibonacci sequence
def fibonacci(n):
    sequence = []
    a, b = 0, 1
    for _ in range(n):
        sequence.append(a)
        a, b = b, a + b
    return sequence

# Input number of terms
n = int(input("Enter the number of terms: "))

# Check if the input is valid
if n <= 0:
    print("Please enter a positive integer.")
else:
    # Generate Fibonacci sequence
    fib_sequence = fibonacci(n)
    # Output the result
    print(f"Fibonacci sequence up to {n} terms: {fib_sequence}")
```

Explanation:

1. **Fibonacci Sequence**:
    - The Fibonacci sequence is a series of numbers where each number is the sum of the two preceding ones, usually starting with 0 and 1. So, the sequence looks like: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...
2. **Function**:
    - The fibonacci(n) function generates the Fibonacci sequence up to n terms. It uses two variables, a and b, to hold the values of consecutive Fibonacci numbers. The sequence is generated by updating these values in a loop.
3. **User Input**:
    - The program takes input from the user, ensuring that a positive integer is provided. If the user enters a non-positive value, it prompts them to enter a valid number.
4. **Result**:
    - The generated Fibonacci sequence is stored in a list and printed out.

10. Write a program to calculate the factorial of a number.
Answer:

```
# Function to calculate factorial
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)

# Input number from user
num = int(input("Enter a number to calculate its factorial: "))

# Check if the input is valid
if num < 0:
    print("Factorial is not defined for negative numbers.")
else:
    # Calculate factorial
    result = factorial(num)
    # Output the result
    print(f"The factorial of {num} is: {result}")
```

Explanation:

1. **Factorial Definition**:
    - The factorial of a non-negative integer n is the product of all positive integers less than or equal to n. It's denoted as n!. For example, 5! = 5 × 4 × 3 × 2 × 1 = 120.
    - By definition, 0! = 1.
2. **Recursive Function**:
    - The factorial() function is implemented recursively. It multiplies n by the factorial of n-1 until n reaches 1 or 0.
3. **User Input**:
    - The program prompts the user to enter a number and checks if the number is non-negative. If the user enters a negative number, it displays an appropriate message.
4. **Result**:
    - The program calculates the factorial and prints the result.

11. Write a program to find the sum of all natural numbers up to n.
Answer:

```
# Function to calculate the sum of natural numbers up to n
def sum_of_natural_numbers(n):
    return n * (n + 1) // 2  # Using the formula n(n+1)/2

# Input number from user
n = int(input("Enter a natural number: "))

# Check if the input is a positive integer
if n <= 0:
    print("Please enter a positive natural number.")
else:
    # Calculate sum of natural numbers
    total_sum = sum_of_natural_numbers(n)
    # Output the result
    print(f"The sum of all natural numbers up to {n} is: {total_sum}")
```

Explanation:

1. **Natural Numbers**:
    - Natural numbers are positive integers starting from 1 (i.e., 1, 2, 3, ...).
2. **Formula**:
    - The sum of the first n natural numbers is given by the formula: S=n(n+1)2S = \frac{n(n + 1)}{2}S=2n(n+1) This formula is derived from the arithmetic series sum formula.
3. **User Input**:
    - The program prompts the user to input a number and checks if it is a positive natural number. If the user enters a non-positive number, it displays an error message.
4. **Result**:
    - The program calculates the sum using the formula and prints the result.

12. Write a program to check if a given string is a palindrome.
Answer:

```
# Function to check if a string is a palindrome
def is_palindrome(s):
    # Convert the string to lowercase and remove spaces for uniformity
    s = s.replace(" ", "").lower()
    # Check if the string is the same when reversed
    return s == s[::-1]

# Input string from user
input_string = input("Enter a string to check if it's a palindrome: ")

# Check if the string is a palindrome
if is_palindrome(input_string):
    print(f'"{input_string}" is a palindrome.')
else:
    print(f'"{input_string}" is not a palindrome.')
```

Explanation:

1. **Palindrome Definition**:
    - A palindrome is a string that reads the same backward as forward. For example, "madam", "racecar", and "level" are palindromes.
2. **Function**:
    - The is_palindrome() function:
        - Removes any spaces and converts the string to lowercase to ensure that it ignores case and spaces.
        - Uses Python's slicing feature (s[::-1]) to reverse the string and compares it to the original string.
3. **User Input**:
    - The program prompts the user to input a string and checks if it is a palindrome by calling the function.
4. **Result**:
    - If the string is the same when reversed, the program prints that it is a palindrome; otherwise, it prints that it is not.

13. Write a program to check if a number is prime.

Answer:

```
# Function to check if a number is prime
def is_prime(n):
    # Check if number is less than 2
    if n <= 1:
        return False
    # Check for factors from 2 to the square root of n
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

# Input number from user
num = int(input("Enter a number to check if it's prime: "))

# Check if the number is prime
if is_prime(num):
    print(f"{num} is a prime number.")
else:
    print(f"{num} is not a prime number.")
```

Explanation:

1. **Prime Number Definition**:
    - A prime number is a number greater than 1 that has no divisors other than 1 and itself. For example, 2, 3, 5, 7, 11 are prime numbers.
2. **Function**:
    - The is_prime() function:
        - Returns False for numbers less than or equal to 1.
        - Checks divisibility by all numbers from 2 up to the square root of n (since a factor larger than the square root would have a corresponding smaller factor).
        - If a divisor is found, it returns False, meaning the number is not prime. Otherwise, it returns True.
3. **User Input**:
    - The program takes a number from the user and checks if it’s prime by calling the is_prime() function.
4. **Result**:
    - The program prints whether the entered number is prime or not.

14. Write a simple number guessing game where the computer selects a random number, and the user has to guess it.

Answer:

```
import random

# Function for the guessing game
def guessing_game():
    # Generate a random number between 1 and 100
    number_to_guess = random.randint(1, 100)
    attempts = 0
    guessed = False

    print("Welcome to the Number Guessing Game!")
    print("I have selected a number between 1 and 100. Try to guess it!")

    # Loop until the user guesses the number
    while not guessed:
        try:
            # Input guess from the user
            user_guess = int(input("Enter your guess: "))
            attempts += 1

            # Check if the guess is too high, too low, or correct
            if user_guess < number_to_guess:
                print("Too low! Try again.")
            elif user_guess > number_to_guess:
                print("Too high! Try again.")
            else:
                print(f"Congratulations! You guessed the correct number {number_to_guess} in
{attempts} attempts.")
                guessed = True
        except ValueError:
            print("Please enter a valid number.")

# Run the game
guessing_game()
```

Explanation:

1. **Random Number Generation**:
    - The program uses the random.randint(1, 100) function to generate a random number between 1 and 100.
2. **Game Loop**:
    - The game repeatedly prompts the user to guess the number until they guess correctly. Each guess is checked:

    - If the guess is too low, the program tells the user to guess higher.
    - If the guess is too high, the program tells the user to guess lower.
    - If the guess is correct, it congratulates the user and exits the loop.
    -

3. **Attempts Tracking**:

- The number of attempts is tracked, and the program informs the user of how many attempts it took to guess the correct number.

4. **Error Handling**:

- The program handles invalid input (non-numeric entries) by catching the ValueError and prompting the user to enter a valid number.

15. Write a program to reverse a given string.
Answer:

```
# Function to reverse a string
def reverse_string(s):
    return s[::-1]

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

# Reverse the string
reversed_string = reverse_string(input_string)

# Output the result
print(f"The reversed string is: {reversed_string}")
```

Explanation:

1. **String Reversal**:
    - The reverse_string() function uses Python's slicing feature (s[::-1]) to reverse the string. This creates a new string that starts from the end and goes to the beginning.
2. **User Input**:
    - The program prompts the user to input a string they want to reverse.
3. **Output**:
    - After reversing the string, the program prints the result.

16. Write a program to count the number of vowels in a given string.
Answer:

```
# Function to count vowels in a string
def count_vowels(s):
    # Define a set of vowels
    vowels = "aeiouAEIOU"
    count = 0

    # Count each vowel in the string
    for char in s:
        if char in vowels:
            count += 1

    return count

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

# Count the vowels in the string
vowel_count = count_vowels(input_string)

# Output the result
print(f"The number of vowels in the given string is: {vowel_count}")
```

Explanation:

1. **Vowel Definition**:
    - Vowels are defined as the characters 'a', 'e', 'i', 'o', 'u' (both uppercase and lowercase).
2. **Function**:
    - The count_vowels() function iterates through each character in the input string and checks if it is a vowel.
    - It maintains a counter that increments whenever a vowel is found.
3. **User Input**:
    - The program prompts the user to enter a string.
4. **Result**:
    - The program counts the vowels in the string and prints the result.

17. Write a program to find the largest element in a list.

Answer:

```
# Function to find the largest element in a list
def find_largest(lst):
    if not lst:  # Check if the list is empty
        return None
    largest = lst[0]  # Assume the first element is the largest
    for num in lst:
        if num > largest:
            largest = num
    return largest

# Input list from user
input_list = input("Enter numbers separated by spaces: ")

# Convert the input string to a list of numbers
number_list = [float(num) for num in input_list.split()]

# Find the largest element
largest_element = find_largest(number_list)

# Output the result
if largest_element is not None:
    print(f"The largest element in the list is: {largest_element}")
else:
    print("The list is empty.")
```

Explanation:

1. **Function**:
    - The find_largest() function checks if the list is empty and returns None if it is.
    - It initializes the variable largest with the first element of the list and iterates through the list to find the largest number.
2. **User Input**:
    - The program prompts the user to enter numbers separated by spaces.
    - It splits the input string and converts it into a list of floating-point numbers.
3. **Result**:
    - The program calls the function to find the largest element and prints the result. If the list is empty, it notifies the user.

18. Write a program to create a dictionary and perform operations like adding, removing, and updating key-value pairs.

Answer:

```
# Function to display the menu
def display_menu():
    print("\nDictionary Operations Menu:")
    print("1. Add a key-value pair")
    print("2. Remove a key-value pair")
    print("3. Update a key-value pair")
    print("4. Display the dictionary")
    print("5. Exit")

# Function to add a key-value pair
def add_key_value(d):
    key = input("Enter the key: ")
    value = input("Enter the value: ")
    d[key] = value
    print(f"Added: {key}: {value}")

# Function to remove a key-value pair
def remove_key_value(d):
    key = input("Enter the key to remove: ")
    if key in d:
        del d[key]
        print(f"Removed: {key}")
    else:
        print("Key not found.")

# Function to update a key-value pair
def update_key_value(d):
    key = input("Enter the key to update: ")
    if key in d:
        value = input("Enter the new value: ")
        d[key] = value
        print(f"Updated: {key}: {value}")
    else:
        print("Key not found.")

# Function to display the dictionary
def display_dictionary(d):
    if d:
        print("Current Dictionary:")
        for key, value in d.items():
            print(f"{key}: {value}")
```

```
    else:
        print("The dictionary is empty.")

# Main function
def main():
    my_dict = {}
    while True:
        display_menu()
        choice = input("Select an option (1-5): ")

        if choice == '1':
            add_key_value(my_dict)
        elif choice == '2':
            remove_key_value(my_dict)
        elif choice == '3':
            update_key_value(my_dict)
        elif choice == '4':
            display_dictionary(my_dict)
        elif choice == '5':
            print("Exiting the program.")
            break
        else:
            print("Invalid choice! Please select a valid option.")

# Run the program
main()
```

Explanation:

1. **Menu Display**:
    - The display_menu() function prints the available operations for the user.
2. **Dictionary Operations**:
    - **Add**: The add_key_value() function prompts the user for a key and value and adds them to the dictionary.
    - **Remove**: The remove_key_value() function prompts for a key and removes it if it exists in the dictionary.
    - **Update**: The update_key_value() function prompts for a key and updates its value if it exists.
    - **Display**: The display_dictionary() function shows the current state of the dictionary or notifies the user if it's empty.
3. **Main Loop**:
    - The main() function initializes an empty dictionary and runs an infinite loop to handle user choices until they choose to exit.

19. **Write a program to read from and write to a text file.**

Answer :

```
# Function to write to a text file
def write_to_file(filename):
    with open(filename, 'w') as file:
        print("Enter text to write to the file (type 'exit' to finish):")
        while True:
            line = input()
            if line.lower() == 'exit':
                break
            file.write(line + '\n')
    print(f"Text written to {filename} successfully.")

# Function to read from a text file
def read_from_file(filename):
    try:
        with open(filename, 'r') as file:
            content = file.read()
            print("Contents of the file:")
            print(content)
    except FileNotFoundError:
        print(f"The file {filename} does not exist.")

# Main function
def main():
    filename = 'example.txt'  # Specify the filename

    # Write to the file
    write_to_file(filename)

    # Read from the file
    read_from_file(filename)

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

Explanation:

1. **Writing to a File**:
    - The write_to_file() function opens a file in write mode ('w'), allowing the user to input multiple lines of text until they type 'exit'.
    - Each line is written to the file with a newline character (\n) appended to separate the lines.
2. **Reading from a File**:

- The read_from_file() function attempts to open the file in read mode ('r').
- It reads the entire content of the file and prints it. If the file does not exist, it catches a FileNotFoundError and displays an error message.

3. **Main Function**:
   - The main() function defines the filename (example.txt) and calls the functions to write to and read from the file.
4. **Execution**:
   - The program executes when run as a script, allowing users to interactively write to a file and then read the content back.

20. Create a simple 'Person' class with attributes 'name' and 'age'. Create objects of this class and print their attributes.

Answer:

```
# Definition of the Person class
class Person:
    def __init__(self, name, age):
        self.name = name  # Assigning the name attribute
        self.age = age    # Assigning the age attribute

    def display_info(self):
        # Method to display the person's information
        print(f"Name: {self.name}, Age: {self.age}")

# Creating objects of the Person class
person1 = Person("Alice", 30)
person2 = Person("Bob", 25)

# Printing the attributes of the created objects
print("Person 1 Information:")
person1.display_info()

print("Person 2 Information:")
person2.display_info()
```

Explanation:

1. **Class Definition**:
    - The Person class is defined with an __init__ method that initializes the name and age attributes when a new object is created.
2. **Method**:
    - The display_info() method is defined to print the name and age of the person in a formatted string.
3. **Object Creation**:
    - Two objects, person1 and person2, are created with different names and ages.
4. **Printing Attributes**:
    - The program calls the display_info() method on each object to print their respective attributes.

21. Create an abstract class Shape with an abstract method area. Create concrete classes Circle and Rectangle that implement the area method.

Answer:

```
from abc import ABC, abstractmethod
import math

# Abstract class Shape
class Shape(ABC):
    @abstractmethod
    def area(self):
        pass

# Concrete class Circle
class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return math.pi * (self.radius ** 2)

# Concrete class Rectangle
class Rectangle(Shape):
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def area(self):
        return self.width * self.height

# Main function to demonstrate the classes
def main():
    # Create an instance of Circle
    circle = Circle(5)
    print(f"The area of the circle with radius {circle.radius} is: {circle.area()}")

    # Create an instance of Rectangle
    rectangle = Rectangle(4, 6)
    print(f"The area of the rectangle with width {rectangle.width} and height
{rectangle.height} is: {rectangle.area()}")

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

Explanation:

1. **Abstract Class**:
    - The Shape class is defined as an abstract class using ABC from the abc module. It contains an abstract method area() which must be implemented by any concrete subclass.
2. **Concrete Classes**:
    - **Circle**:
        - The Circle class inherits from Shape and initializes with a radius. It implements the area() method to calculate the area using the formula π×radius2\pi \times \text{radius}^2π×radius2.
    - **Rectangle**:
        - The Rectangle class also inherits from Shape and initializes with width and height. It implements the area() method to calculate the area using the formula width×height\text{width} \times \text{height}width×height.
3. **Main Function**:
    - In the main() function, instances of Circle and Rectangle are created, and the area for each shape is calculated and printed.