Name: Akhilesh Singh Roll No: 186

Lab. Assignment 5

PYTHON

1. Write a Python program that takes two numbers as input from the user, performs addition, subtraction, multiplication, and division, and displays the results.

Code:

```
# Get input from the user
n1 = float(input("Enter the first number: "))
n2 = float(input("Enter the second number: "))

# Perform operations
addition = n1 + n2
subtraction = n1 - n2
multiplication = n1 * n2

# Handle division with check for division by zero
if n2 != 0:
    division = n1 / n2
else:
    division = "undefined (cannot be divided by zero)"

# Display the results
print(f"Addition: {n1} + {n2} = {addition}")
print(f"Subtraction: {n1} - {n2} = {subtraction}")
print(f"Multiplication: {n1} * {n2} = {multiplication}")
print(f"Division: {n1} / {n2} = {division}")
```

Output:

```
Enter the first number: 20
Enter the second number: 10
Addition: 20.0 + 10.0 = 30.0
Subtraction: 20.0 - 10.0 = 10.0
Multiplication: 20.0 * 10.0 = 200.0
Division: 20.0 / 10.0 = 2.0
```

2. Implement a program that takes a string input from the user and performs the following operations:

- Convert the string to uppercase.
- Reverse the string.
- Count the number of vowels in the string.

Code:

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

# Convert the string to uppercase
uppercase_string = input_string.upper()

# Reverse the string
reversed_string = input_string[::-1]

# Count the number of vowels in the string
vowels = "aeiouAEIOU"
vowel_count = sum(1 for char in input_string if char in vowels)

# Display the results
print(f"Uppercase: {uppercase_string}")
print(f"Reversed: {reversed_string}")
print(f"Number of vowels: {vowel_count}")
```

Output:

```
Enter a string: bond, james bond
Uppercase: BOND, JAMES BOND
Reversed: dnob semaj ,dnob
Number of vowels: 4
```

3. Write a Python program that defines a custom function to calculate the factorial of a given number. Include error handling to manage invalid inputs (e.g., negative numbers or non-integer inputs).

Code:

```
# Define a function to calculate the factorial
def calculate_factorial(n):
    # Check if the number is negative
    if n < 0:
        raise ValueError("Factorial is not defined for negative
numbers.")
    # Base case for 0 factorial
    elif n == 0:
        return 1
    # Recursive case for positive integers
    else:
        factorial = 1
        for i in range(1, n + 1):
            factorial *= i
        return factorial

# Input handling with error checking
def get_user_input():
    while True:
        try:
            # Take user input
            user_input = input("Enter a number to calculate its
factorial: ")

            # Try to convert the input to an integer
            number = int(user_input)

            # Calculate and display the factorial
            result = calculate_factorial(number)
            print(f"The factorial of {number} is: {result}")
            break  # Exit loop after successful calculation

        except ValueError as e:
            # Handle invalid input (non-integer or negative number)
            print(f"Error: {e}. Please enter a valid positive
integer.")
```

```
# Run the input handling function
get_user_input()
```

Output:

```
Enter a number to calculate its factorial: 5
The factorial of 5 is: 120
```

Error Handling

```
Enter a number to calculate its factorial: -3
Error: Factorial is not defined for negative numbers.. Please enter a valid positive integer.
Enter a number to calculate its factorial: abcd
Error: invalid literal for int() with base 10: 'abcd'. Please enter a valid positive integer.
Enter a number to calculate its factorial: 6
The factorial of 6 is: 720
```

4. Implement a program that includes a function to compute the roots of a quadratic equation. Use exception handling to manage cases where the roots are complex (i.e., when the discriminant is negative).

Code:

```
import cmath  # For handling complex numbers

# Function to compute the roots of the quadratic equation ax^2 + bx
+ c = 0
def calculate_roots(a, b, c):
    # Calculate the discriminant (b^2 - 4ac)
    discriminant = b**2 - 4*a*c

    # Check if discriminant is negative and handle complex roots
    if discriminant < 0:
        raise ValueError("The equation has complex roots because the
discriminant is negative.")

    # Calculate the two roots using the quadratic formula
    root1 = (-b + cmath.sqrt(discriminant)) / (2 * a)
    root2 = (-b - cmath.sqrt(discriminant)) / (2 * a)

    return root1, root2

# Input handling with error checking
def get_user_input():
    while True:
        try:
            # Take user input for the coefficients a, b, c
            a = float(input("Enter the coefficient a: "))
            b = float(input("Enter the coefficient b: "))
            c = float(input("Enter the coefficient c: "))

            # Check if a is zero, since the equation wouldn't be
quadratic
            if a == 0:
                raise ValueError("The coefficient 'a' cannot be zero
for a quadratic equation.")

            # Calculate and display the roots
            root1, root2 = calculate_roots(a, b, c)
```

```
            print(f"The roots of the quadratic equation are: {root1}
and {root2}")
            break  # Exit loop after successful calculation

        except ValueError as e:
            # Handle exceptions, such as invalid inputs or complex
roots
            print(f"Error: {e}. Please enter valid coefficients.")

# Run the input handling function
get_user_input()
```

Output:

```
Enter the coefficient a: 1
Enter the coefficient b: -6
Enter the coefficient c: 8
The roots of the quadratic equation are: (4+0j) and (2+0j)
```

Exception Handling

```
Enter the coefficient a: 1
Enter the coefficient b: 2
Enter the coefficient c: 5
Error: The equation has complex roots because the discriminant is negative.. Please enter valid coefficients.
Enter the coefficient a:
```