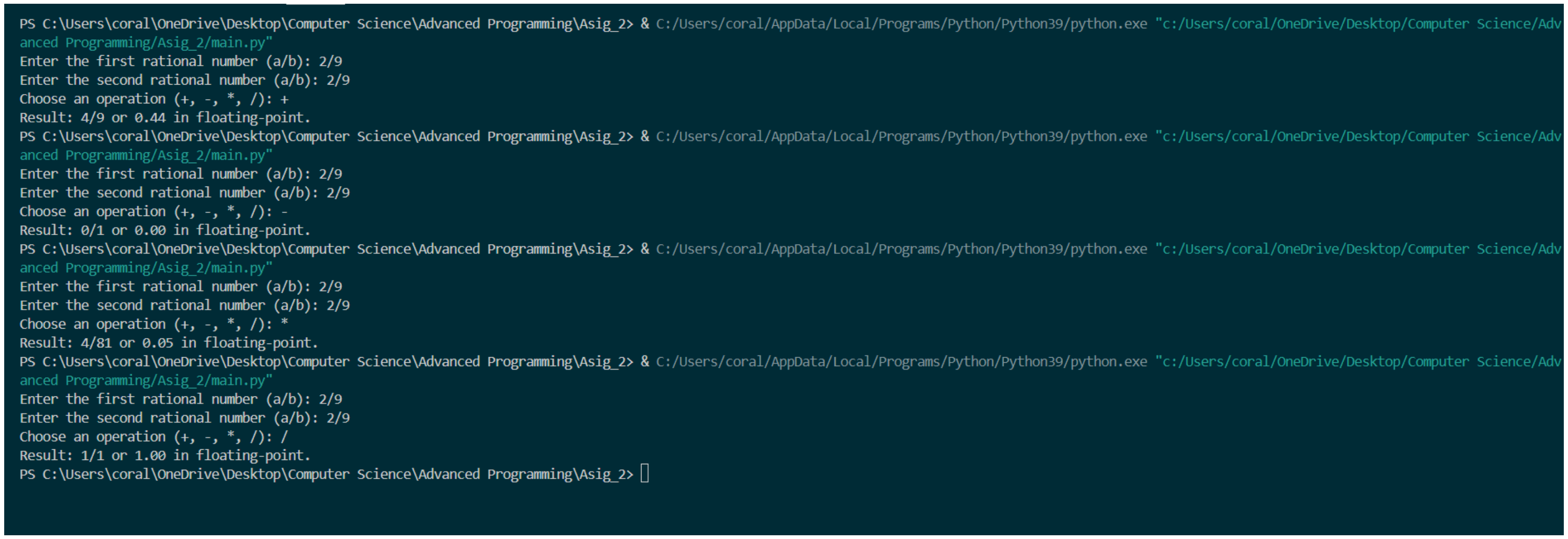
```python
"""
Author: Coral S. Schmidt Montilla
Student Number: 148830
Filename: main.py

This application performs arithmetic operations with Rational numbers:
    In this file, main interacts with the Rational class to perform arithmetic
operations and display the results.
"""

from Rational import Rational

def get_rational_input(prompt):
    # Function to get user input for rational numbers.
    while True:
        try:
            numerator, denominator = map(int, input(prompt).split('/'))
            return Rational(numerator, denominator)
        except ValueError:
            print("Invalid input. Please enter in the format
numerator/denominator.")

def main():

    r1 = get_rational_input("Enter the first rational number (a/b): ")
    r2 = get_rational_input("Enter the second rational number (a/b): ")

    operation_input = input("Choose an operation (+, -, *, /): ")

    result = None
    if operation_input == '+':
        result = r1.addition(r2)

    elif operation_input == '-':
        result = r1.subtraction(r2)

    elif operation_input == '*':
        result = r1.multiplication(r2)

    elif operation_input == '/':
        if r2.numerator == 0:
            print("Error: Division by zero is not allowed.")
            return
        result = r1.division(r2)
    else:
        print("Error: Invalid operation.")
        return

    print(f"Result: {result} or {result.rational_to_float(2)} in floating-
point.")

if __name__ == "__main__":
    main()

"""
Author: Coral S. Schmidt Montilla
Student Number: 148830
Filename: Rational.py

This application provides a class called Rational for performing arithmetic with
fractions:
    The class Rational handles addition, subtraction, multiplication, and
division of
    rational numbers. It also reduces fractions to their simplest form and
supports
    conversion to floating-point format.
"""

class Rational:
    def __init__(self, numerator=0, denominator=1):

        self.set_fraction(numerator, denominator)

    def set_fraction(self, numerator, denominator):

        if denominator == 0:
            print("Error: Denominator cannot be zero.")
            return

        self.numerator = numerator
        self.denominator = denominator
        self.simplest_form()

    def simplest_form(self):

        g = self.greatest_common_denominator(self.numerator, self.denominator)
        self.numerator //= g
        self.denominator //= g
    @staticmethod
    def greatest_common_denominator(a, b):

        while b != 0:
            a, b = b, a % b
        return a

    def __str__(self):

        return f"{self.numerator}/{self.denominator}"

    def rational_to_float(self, precision=None):

        if self.denominator == 0:
            return "Undefined"

        result = self.numerator / self.denominator

        if precision is not None:
            return f"{result:.{precision}f}"

        return str(result)

    def addition(self, other):

        new_numerator = self.numerator * other.denominator + other.numerator *
self.denominator
        new_denominator = self.denominator * other.denominator

        return Rational(new_numerator, new_denominator)

    def subtraction(self, other):

        new_numerator = self.numerator * other.denominator - other.numerator *
self.denominator
        new_denominator = self.denominator * other.denominator

        return Rational(new_numerator, new_denominator)

    def multiplication(self, other):

        new_numerator = self.numerator * other.numerator
        new_denominator = self.denominator * other.denominator

        return Rational(new_numerator, new_denominator)
    def division(self, other):

        if other.numerator == 0:
            print("Error: Attempt to divide by a fraction with a numerator of
zero.")
            return

        new_numerator = self.numerator * other.denominator
        new_denominator = self.denominator * other.numerator

        return Rational(new_numerator, new_denominator)
```

Output:



```
PS C:\Users\coral\OneDrive\Desktop\Computer Science\Advanced Programming\Asig_2> & C:/Users/coral/AppData/Local/Programs/Python/Python39/python.exe "c:/Users/coral/OneDrive/Desktop/Computer Science/Adv
anced Programming/Asig_2/main.py"
Enter the first rational number (a/b): 2/9
Enter the second rational number (a/b): 2/9
Choose an operation (+, -, *, /): +
Result: 4/9 or 0.44 in floating-point.
PS C:\Users\coral\OneDrive\Desktop\Computer Science\Advanced Programming\Asig_2> & C:/Users/coral/AppData/Local/Programs/Python/Python39/python.exe "c:/Users/coral/OneDrive/Desktop/Computer Science/Adv
anced Programming/Asig_2/main.py"
Enter the first rational number (a/b): 2/9
Enter the second rational number (a/b): 2/9
Choose an operation (+, -, *, /): -
Result: 0/1 or 0.00 in floating-point.
PS C:\Users\coral\OneDrive\Desktop\Computer Science\Advanced Programming\Asig_2> & C:/Users/coral/AppData/Local/Programs/Python/Python39/python.exe "c:/Users/coral/OneDrive/Desktop/Computer Science/Adv
anced Programming/Asig_2/main.py"
Enter the first rational number (a/b): 2/9
Enter the second rational number (a/b): 2/9
Choose an operation (+, -, *, /): *
Result: 4/81 or 0.05 in floating-point.
PS C:\Users\coral\OneDrive\Desktop\Computer Science\Advanced Programming\Asig_2> & C:/Users/coral/AppData/Local/Programs/Python/Python39/python.exe "c:/Users/coral/OneDrive/Desktop/Computer Science/Adv
anced Programming/Asig_2/main.py"
Enter the first rational number (a/b): 2/9
Enter the second rational number (a/b): 2/9
Choose an operation (+, -, *, /): /
Result: 1/1 or 1.00 in floating-point.
PS C:\Users\coral\OneDrive\Desktop\Computer Science\Advanced Programming\Asig_2> 
```