# CS 172 – Lab 3: Operator Overloading

## Fraction Objects

Creating objects with overloaded operators allows you to use the object like a built-in data type. In this lab, you will create a `Fraction` class. By overloading its operators, you can easily use your class to solve mathematical problems related to fractions.

## Provided Code: The `Fraction` Class

You are provided stub/template/starter for a fraction module that contains a `Fraction` class. Much is already implemented for you. Your job will be to implement the operators described below.

**Note**: all these operators can be used in a static context and return a new `Fraction` object. In addition, all operators other than the power operator have `Fraction` objects as their operands. The power object has its first operand be a `Fraction` object, and the second one is an Integer.

## Milestone #1: The `__add__`, `__sub__`, and `__mul__` methods

Overload the `__add__`, `__sub__`, and `__mul__` methods. Here is how these operations work with Fractions:

Add:

$$\frac{a}{b} + \frac{x}{y} = \frac{ay + xb}{by}$$

Subtract:

$$\frac{a}{b} - \frac{x}{y} = \frac{a}{b} + \frac{-x}{y}$$

Multiply:

$$\frac{a}{b} * \frac{x}{y} = \frac{ax}{by}$$

Once you have implemented the methods, test them using hardcoded values in **main.py**. For example, you could add these lines:

```
f1 = Fraction(3, 5)
f2 = Fraction(2, 3)
print(f1 + f2) # this should display 19/15
print(f2 - f1) # this should display 1/15
print(f1 * f2) # this should display 2/5
```

If your code is working correctly at this point, then this is a good time to submit your code for up to 24 points! You will get partial lab credit for completing this part, even if you do not complete the rest of the lab.

## Milestone #2: The `__truediv__` and `__pow__` methods

Overload the `__truediv__` and `__pow__` methods. Here is how these operations work with Fractions:

Divide:

$$\frac{a}{b} / \frac{x}{y} = \frac{ay}{bx}$$

Power (when k > 0 and k is an integer):

$$\left(\frac{a}{b}\right)^{-k} = \left(\frac{b}{a}\right)^{k}$$

$$\left(\frac{a}{b}\right)^{0} = 1$$

$$\left(\frac{a}{b}\right)^{k} = \frac{a}{b} * \left(\frac{a}{b}\right)^{k-1}$$

Once you have implemented the methods, test them using hardcoded values in **main.py**. For example, you could add these lines:

```
f1 = Fraction(3, 5)
f2 = Fraction(2, 3)
print(f1 / f2)  # this should display 9/10
print(f1 ** 0)  # this should display 1
print(f1 ** -2) # this should display 25/9
print(f1 ** 3)  # this should display 27/125
```

If your code is working correctly at this point, then this is a good time to submit your code for up to 16 more points! You will get partial lab credit for completing this part, even if you do not complete the rest of the lab.

## Milestone #3: The `H(n)` function

You have been provided with a **main.py** file template. Open this file and implement the `H(n)` function.

Remember that a summation symbol just tells you to add all the values in a range.

$$\sum_{k=1}^{5} k = 1 + 2 + 3 + 4 + 5 = 15$$

Harmonic Series:

$$H(n) = \sum_{k=1}^{n} \frac{1}{k}$$

Test the function with a hardcoded value. For example, use `H(10)` and print the result (you should get `7381/2520`). Comment out any tests from Milestone 1 and Milestone 2 you don't need anymore.

If your code is working correctly at this point, then this is a good time to submit your code for up to 10 more points! You will get partial lab credit for completing this part, even if you do not complete the rest of the lab.

## Milestone #4: The `T(n)` and `Z(n)` functions

Implement the `T(n)` and `Z(n)` functions:

Two:

$$T(n) = \sum_{k=0}^{n} \left(\frac{1}{2}\right)^{k}$$

Zero:

$$Z(n) = 2 - \sum_{k=0}^{n} \left(\frac{1}{2}\right)^{k}$$

Test the functions with hardcoded values. For example, use `T(10)` should return `2047/1024` and `Z(10)` should return `1/1024`.

If your code is working correctly at this point, then this is a good time to submit your code for up to 20 more points! You will get partial lab credit for completing this part, even if you do not complete the rest of the lab.

## Milestone #5: The `R(n)` function

Implement the `R(n, b)` function:

Partial Riemann Zeta:

$$R(n,b) = \sum_{k=1}^{n} \left(\frac{1}{k}\right)^{b}$$

Test the function with hardcoded values. For example, use `R(10, 10)` should return `41352057490642308398789372291260 9/41310970629609628851240960000000`.

If your code is working correctly at this point, then this is a good time to submit your code for up to 10 more points! You will get partial lab credit for completing this part, even if you do not complete the rest of the lab.

## Milestone #6: Main Script, part 1

Your program should ask for `n` as input. Replace all your hardcoded examples with the values the user entered. Verify that the input is a valid number. If it is not, ask repeatedly until a valid number is given. Compute each of the functions for the given input `n`. When testing the Partial Riemann Zeta function, we will test it as `R(n, n)`

If your code is working correctly at this point, then this is a good time to submit your code for up to 8 more points! You will get partial lab credit for completing this part, even if you do not complete the rest of the lab.

## Milestone #7: Main Script, part 2

Once you have been given a valid input, print out the values of each of the functions in the order `H()`, `T()`, `Z()`, `R()`. Use print statements to make sure everything looks like the examples below.

Add comments and make sure the style expectations are met. Delete any tests you no longer need. See below execution trace for a sample layout.

Once you reach this point you are ready to make your final submission for up to 12 more points.

## Example Execution Trace

```
Welcome to Fun with Fractions!
Enter Number of iterations (integer>0):
Bad Input
Enter Number of iterations (integer>0):
10
H(10)=7381/2520
H(10)~=2.92896825
T(10)=2047/1024
T(10)~=1.99902344
Z(10)=1/1024
Z(10)~=0.00097656
R(10,10)=413520574906423083987893722912609/413109706296096288512409600000000
R(10,10)~=1.00099458
```

## Fun Facts

Each of these series approaches a value when `n` = infinity.

$$H(\infty) = \infty$$

$$T(\infty) = 2$$

$$Z(\infty) = 0$$

$$R(\infty, b) = \zeta(b)$$

This series only converges to simple values on even `b`.

$$\zeta(2) = \frac{\pi^2}{6}$$

$$\zeta(4) = \frac{\pi^4}{90}$$

$$\zeta(6) = \frac{\pi^6}{945}$$

$$\zeta(8) = \frac{\pi^8}{9450}$$

## Scoring

The score for the assignment is determined as follows:

- 8 points: `__add__` method is implemented correctly.
- 8 points: `__sub__` method is implemented correctly.
- 8 points: `__mul__` method is implemented correctly.
- 8 points: `__truediv__` method is implemented correctly.
- 8 points: `__pow__` method is implemented correctly.
- 10 points: `H(n)` function is implemented correctly.
- 10 points: `T(n)` function is implemented correctly.
- 10 points: `Z(n)` function is implemented correctly.
- 10 points: `R(n, b)` function is implemented correctly.
- 8 points: Input validation.
- 6 points: Program is complete and runs without problems.
- 6 points: Compliance and style: program meets the lab requirements, good variable names are used, and the program is appropriately commented (there is at the very least a header comment).

**Note:** Please make sure you write both lab partners' names and user IDs (in the form abc123) and program purpose in the header comment of the file you submit for this assignment.