Fração

Neste exercício, definiremos uma classe chamada `Rational` para representar números racionais como frações.

O método `__init__` de sua classe deve receber dois inteiros: o primeiro deve representar o numerador e o segundo, o denominador.

Sua classe deve fornecer os seguintes métodos:

- um método `get_numerator()`, que retorna o numerador como um `int`.
- um método `get_denominator()`, que retorna o denominador como um `int`.
- um método `to_float()`, que retorna este número representado como um `float`.
- um método `reciprocal()`, que retorna o recíproco desse número como uma instância de `Rational`.
- um método `reduce()`, que retorna um número `Rational` equivalente, mas reduzido aos termos mais baixos.
  - Note que você precisará achar o máximo divisor comum dos dois números para fazer isso. Talvez valha a pena definir uma função separada para isso. Idealmente você deveria passar algum tempo tentando descobrir um algoritmo para fazer isso sozinho, mas se isso se provar difícil, você pode usar o algoritmo Euclidiano (https://sites.math.rutgers.edu/~greenfie/gs2004/euclid.html) para obter o MDC.
- um método `__add__(other)`, que retorna um número que representa a soma da instância original e `other`:
  - se `other` for uma instância de `Rational` ou `int`, retorne uma nova instância de `Rational`.
  - se `other` for uma instância de `float`, retorna um `float`.
  - caso contrário, retorna `None` (indicando um erro) Observe que você pode usar a função integrada do Python `isinstance` para verificar se um determinado objeto é uma instância de uma determinada classe.
- um método `__mul__(other)`, que retorna um número que representa o resultado da multiplicação da instância original e `other`, seguindo as mesmas regras de tipo que `__add__`.
- um método `__truediv__(other)`, que retorna um número que representa o resultado da divisão da instância original por `other`, seguindo as mesmas regras de tipo que `__add__`.
- um método `__sub__(other)`, que deve se comportar de forma análoga para adicionar, mas deve realizar a subtração, seguindo as mesmas regras de tipo que `__add__`.

Não se esqueça de que cada definição de método exigirá um argumento a mais do que o listado acima (o argumento que normalmente chamamos de `self`).

Em todos os casos, esses métodos não devem modificar a instância original de `Rational`; em vez disso, eles devem retornar novos objetos.

Observe também que, como usamos nomes especiais para esses métodos, eles substituirão alguns dos comportamentos padrão do Python:

- `m1 + o` executará `m1.__add__(o)`.

- `m1 - o` executará `m1.__sub__(o)`.
- `m1 * o` executará `m1.__mul__(o)`.
- `m1 / o` executará `m1.__truediv__(o)`.

Seu código não deve usar `import`.

Submissão

Quando estiver pronto(depois de ter simulado manualmente e testado em sua própria máquina e estiver convencido de que seu programa fará a coisa certa), faça upload do seu arquivo Python no **Problema 4.3** no Gradescope. Lembre-se de nomear seu arquivo `p4_3.py`.