Programação com Interfaces Gráfica

Mario Benevides e Paulo Roma

Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brasil

Projeto 1 - Frações

Agenda

Aulas Passadas:

- Introdução a OO e Classes
- Classes
- Exceções

Nesta Aula: Projeto envolvendo estes conceitos

Projeto 1: Objeto Fração

Escreva uma classe para manipular objetos do tipo fração.
Frações são números racionais da forma n/d, onde n e d são inteiros.

- A sua classe deve fazer a sobrecarga dos operadores, $+, +=, *, -, ==, /$, e str.
- O operador de igualdade deve simplificar as frações antes de compará-las (use o MDC).
- Levante uma exceção se algum operador for aplicado a um tipo inválido, ou se for construída uma fração com denominador nulo.

Projeto 1: Dicas

Documentar Bem!!!

- Nós usamos Doxygen.
- Aplicativo para documentar programas.
- Gera documentação em HTML.
- Muito fácil de usar. Diagrama de Classes...
- Baixar: www.doxygen.org/
- Colocar no mesmo diretório arquivo $Doxyfile$. Configurações.
- Executar o $Doxygen$ no mesmo diretório.
- Gera um diretório $html$ com a documentação.

Usamos ingles para documentar.

- Nomes de variáveis.
- Comentários.

Projeto 1: Objeto Fração - Métodos

Métodos Mágicos

- __init__
- __add__
- __iadd__
- __mul__
- __eq__
- __str__
- __truediv__

Dois Métodos

- simplifica
- MDC

Classe Fracao

```
## A simple class for creating fraction objects (rational numbers).
#
class Fracao:
    ## Constructor.
    #
    #  @param num numerator.
    #  @param den denominator
    #
    def __init__(self, num=1, den=1):
        """Constructor"""

        if ( den == 0 ): raise ValueError ("Zero denominator")

        ### Numerator.
        self.num = num
        ### Denominator.
        self.den = den

        if self.den < 0:           # check for a negative denominator
           self.num = -self.num    # change the sign of the numerator
           self.den = -self.den
        self.simplifica()          # simplify the fraction
```

Função *simplifica*

```
## Simplifies this fraction, by dividing either the numerator
#  or the denominator by its gcd.
#
 def simplifica(self):
     max = 1
     if self.num != 0:                    # assert Fracao != 0
        max = mdc(self.num, self.den)     # find the gcd
     if max > 1:                          # reduce this fraction
        self.num //= max                  # integer division
        self.den //= max
     return self
```

Função *mdc*

```
## mdc is a general use function, defined outside the class.
#
#  @param x first integer: numerator.
#  @param y second integer: denominator.
#  @return GCD: Greatest Common Divisor.
#
def mdc(x, y):
    """Greatest Common Divisor (Maximo divisor comum)."""
    while y != 0:
        resto = x % y
        x = y
        y = resto
    return x
```

Exemplo: $x = 30, y = 8 \rightarrow x = 8, y = 6 \rightarrow x = 6, y = 2 \rightarrow x = 2, y = 0$

Operador ==

```
## Operator ==
    #
    def __eq__(self, f):
        a=self.simplifica()
        b=f.simplifica()
        return (a.num == b.num and a.den == b.den)
```

Exemplo:

$$a = \frac{18}{12} \qquad b = \frac{9}{6}$$

$$\downarrow simplifica \qquad \downarrow simplifica$$

$$a = \frac{3}{2} \qquad b = \frac{3}{2}$$

`return (a.num == b.num and a.den == b.den) = true`

Operador +

Revisão: soma de fração

$a + b$

- Exemplo 1: b inteiro

$a = \frac{3}{5}$ e $b = 7$

$$a + b = \frac{3}{5} + 7 = \frac{3 + (5 * 7)}{5} = \frac{38}{5}$$

- Exemplo 1: b é uma fração

$a = \frac{4}{5}$ e $b = \frac{2}{3}$

$$a + b = \frac{4}{5} + \frac{2}{3} = \frac{(4 * 3) + (5 * 2)}{5 * 3} = \frac{22}{15}$$

Operador +

```
## Operator +
    #
    def __add__(self, f):
        if isinstance (f,int):          # check  f is  integer
            num = self.num + f * self.den
            den = self.den
        elif isinstance (f,Fracao):  # check  f is  fraction
            den = self.den * f.den
            num = self.num * f.den + self.den * f.num
        else: raise TypeError ("__add__")
        # returns a copy, and therefore is slower than "+="
        return Fracao(num, den)      # Fraction is simplified
```

- Observação: retorna uma cópia. É menos eficiente.

Operador -

```
## Operator -
    #
    def __sub__(self, f):
        if isinstance (f,int):        # check f is an integer
             num = self.num - f * self.den
             den = self.den
        elif isinstance (f,Fracao):  # check f is a fraction
           den = self.den * f.den
           num = self.num * f.den - self.den * f.num
        else: raise TypeError ("__sub__")
        return Fracao(num, den)
```

- Observação: retorna uma cópia. É menos eficiente.

Operador +=

```
## Operator +=
    #
    def __iadd__(self, f):
        if isinstance (f,int):        # check f is an integer
            self.num += f * self.den
        elif isinstance (f,Fracao):  # check f is a fraction
            self.num = self.num * f.den + self.den * f.num
            self.den *= f.den
        else: raise TypeError ("__iadd__")
        return self.simplifica()
```

- Observação: retorna a própria fração. É mais eficiente.

Operador *

Revisão: multiplicação de fração

$a * b$

- Exemplo:

  $a = \frac{4}{5}$ e $b = \frac{2}{3}$

$$a * b = \frac{4}{5} * \frac{2}{3} = \frac{4 * 2}{5 * 3} = \frac{8}{15}$$

Operador *

```
## Operator *
#
def __mul__(self, f):
    if isinstance (f,int):          # check f is an integer
       num = self.num * f
       den = self.den
    elif isinstance (f,Fracao):  # check f is a fraction
       num = self.num * f.num
       den = self.den * f.den
    else: raise TypeError ("__mul__")
    return Fracao(num, den)
```

- Observação: retorna uma cópia. É menos eficiente.

Operador /

Revisão: divisão de fração

a/b

- Exemplo:

$a = \frac{4}{5}$ e $b = \frac{2}{3}$

$$a \ / \ b = \frac{4}{5} \ / \ \frac{2}{3} = \frac{4 * 3}{5 * 2} = \frac{12}{10} = \frac{6}{5}$$

Operador /

```
## Operator /
#
def __truediv__(self, f):
    if isinstance (f,int):          # check f is an integer
       num = self.num
       den = self.den * f
    elif isinstance (f,Fracao):  # check f is a fraction
       num = self.num * f.den
       den = self.den * f.num
    else: raise TypeError ("__truediv__")
    return Fracao(num, den)
```

- Observação: retorna uma cópia. É menos eficiente.

Operador *str*

```
## Controls how a fraction is printed.
    #
    #  @return a string: numerator/denominator, or
    #   only the numerator, if the denominator is 1, after sir
    #   0, if the numerator is null.
    def __str__(self):
        if self.num == 0:
            return "0"
        elif self.den == 1:
            return str(self.num)
        else:
            return str(self.num)+'/'+ str(self.den)
```

Main - Parte 1

```
## Main program for testing.
#
def main ():
    f = Fracao(15,45)
    g = Fracao(50,75)
    print ("f = 15/45 = %s" % f)
    print ("g = 50/75 = %s" % g)
    print ("f + g = %s" % (f + g))
    h = Fracao (10,28)
    print ("h = 10/28 = %s" % h)
    print ("f * h = %s" % (f * h))
    print ("f + g + h = %s" % (f + g + h))
    print ("f + g * h = %s" % (f + g * h))
    print ("g - f - f = %s" % (g - f - f))
    print ("f * 2 = %s" % (f * 2))
    print ("f + 2 = %s" % (f + 2))
    print ("f / g = %s" % (f / g))
    f += g*2
    print ("f += g*2 = %s" % f)
    f -= g*2
    print ("f -= g*2 = %s" % f)
```

Main - Parte 2

```
try:
    print ("2 + f =" )
    print (2 + f)
except (ValueError,TypeError) as e:
   print ("Exception caught: %s" % e)
print ("f == h %s" % (f==h))
print ("f=g=h")
f=g=h
print ("f == h %s" % (f==h))
try:
   print ("f += \'a\'")
   f += "a"
except (ValueError,TypeError) as e:
   print ("Exception caught: %s" % e)
```

Main - Parte 3

```
try:
        print ("Fracao(2,0) = ")
        print (Fracao(2,0))
    except (ValueError,TypeError) as e:
        print ("Exception caught: %s" % e)
try:
        print ("Fracao(5,3) / Fracao(0,4)")
        print (Fracao(5,3)/Fracao(0,4))
    except Exception as e:
        print ("Exception caught: %s" % e)

if __name__=="__main__":
    sys.exit(main())
```

Main - Executando

```
f = 15/45 = 1/3
g = 50/75 = 2/3
f + g = 1
h = 10/28 = 5/14
f * h = 5/42
f + g + h = 19/14
f + g * h = 4/7
g - f - f = 0
f * 2 = 2/3
f + 2 = 7/3
f / g = 1/2
f += g*2 = 5/3
f -= g*2 = 1/3
2 + f =
Exception caught: unsupported operand type(s) for +: 'int' and 'Fracao'
f == h False
f=g=h
f == h True
f += 'a'
Exception caught: __iadd__
Fracao(2,0) =
Exception caught: Zero denominator
Fracao(5,3) / Fracao(0,4)
Exception caught: Zero denominator
```

O Arquivo

```
#!/usr/bin/env python
# coding: UTF-8
#
## @package _01a_fracao
#
# A very simple fraction class.
#
# @author Miguel Jonathan e Paulo Roma
# @since 16/09/2009
# @see http://docs.python.org/library/fractions.html
# @see http://docs.python.org/reference/datamodel.html#emulating-nu

import sys
```

Classes Fracao + Main