Programação com Interfaces Gráfica

Mario Benevides e Paulo Roma

Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brasil

Projeto 2 - Frações com Arquivos

Agenda

Aulas Passadas:

- Introdução a OO e Classes
- Classes
- Exceções
- Módulos
- Arquivos

Nesta Aula: Projeto envolvendo estes conceitos

Projeto 2: Fração com Arquivos

Escreva um programa para ler uma série de frações de um arquivo e imprimir a sua soma e produto.

Por exemplo:

Arquivo de Entrada:	**Saída do programa**:
2 3	Fracao 0: 2/3
5 7	Fracao 1: 5/7
1 4	Fracao 2: 1/4
8 3	Fracao 3: 8/3
2 4	Fracao 4: 1/2
1 7	Fracao 5: 1/7
3 8	Fracao 6: 3/8
	Soma: 893/168
	Produto: 5/588

Projeto 1: Objeto Fracoes - Métodos

Métodos Mágicos

- __init__
- __str__
- __repr__

Método

- Reader

O Arquivo

```
#!/usr/bin/env python
# coding: UTF-8
#
## @package _01e_fracoes
#
# Reads a file with a series of fractions, and prints their sum
# and product.
#
# @author Paulo Roma
# @since 25/09/2014

import sys

from _01a_fracao import Fracao
```

Classe Fracoes + Main

Classe Fracoes

```
# Process fractions on a given file.
class Fracoes:
    ##
    #   Constructor.
    #   Opens filename and calls Reader for inputting the fraction readings.
    #   Raises an exception if filename does not exist.
    #
    #   @param filename fraction file name.
    #
    def __init__(self, filename):
      ### lfracoes - a list of objects of type Fracao.
      self.lfracoes = []

      try:
         f = open(filename, 'r')
      except IOError:
         print ('Fracoes: Cannot open file %s for reading' % filename)
         raise
      self.Reader(f)
```

O Método *Reader*

```
##
# Reads a file with a numerator and denominator per line.
# Creates a Fracao object for each line and inserts it in lfracoes.
#
# @param f fraction file object.
#
def Reader (self, f):
    for line in f:
        temp = line.split(None)
        if len(temp) == 2:
            try:
              self.lfracoes.append(Fracao(int(temp[0]),int(temp[1])))
            except:
              print ('Fração Inválida: %s\n' % temp)
              continue
    f.close()
```

O Método *__str__*

```
##
    #   Returns the sum and product of all entries of "lfracoes".
    #
    #   @return a string: sum and product of all fractions.
    #
    def __str__(self):
        sb = ""
        f = Fracao(0,1)
        g = Fracao(1,1)
        for i in range(0, len(self.lfracoes)):
            f += self.lfracoes[i]
            g *= self.lfracoes[i]
        sb += "Soma: %s\nProduto: %s\n" % (f,g)

        return sb
```

O Método *__repr__*

```
##
    #   Returns each fraction in list "lfracoes".
    #
    #   @return a string: a series of fractions, one per line.
    #
    def __repr__(self):
        sb = ""
        for i in range(0, len(self.lfracoes)):
            sb += "Fracao %d: %s\n" % (i,self.lfracoes[i])

        return sb
```

Main

```
#   Reads a series of pairs  and prints the sum and product of all fractions.
#
def main(argv=None):
     f = "fracoes.txt"
     if argv is None:
        argv = sys.argv

     if ( len(argv) > 1 ):
          f = argv[1]

     try:
          m = Fracoes(f)
          print (repr(m))
          print (m)
     except IOError:
          sys.exit ( "File %s not found." % f )

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

Main - Executando

```
$ more fracoes.txt
5 7
8 10
12 23
2 9
4 8
10 33
11 99

$python fracoes.py
Fracao 0: 5/7
Fracao 1: 4/5
Fracao 2: 12/23
Fracao 3: 2/9
Fracao 4: 1/2
Fracao 5: 10/33
Fracao 6: 1/9

Soma: 56183/17710
Produto: 160/143451
```