Programação com Interfaces Gráfica

Mario Benevides e Paulo Roma

Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brasil

Projeto 4 - Calcular Consumo de um Carro

Agenda

Aulas Passadas:

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

Nesta Aula: Projeto envolvendo estes conceitos

Projeto 4: Calcular Consumo de um Carro

Escreva um programa para implementar uma classe para controlar o consumo de um carro a partir de uma arquivo contendo um série de medidas de distância e quantidade de combustível consumida.

- Construa uma classe para ler uma série de leituras de hodômetro e galões consumidos após encher o tanque de gasolina de um veículo, e calcular o consumo entre cada paragem no posto de gasolina.

Projeto 4: Classes

Duas Classes

- MileageCalculator

  Lê, calcula e imprimi consumo de gasolina em km/litro e milhas/galões

- FillUp

  Manipula objetos do tipo FillUp, que são leituras do odômetro e de galões consumidos

Classe FillUp

```
class FillUp:
  ##
  # Odometer reading when the tank was filled.
  #
  odometer = 0.0

  ##
  # Gallons needed to fill the tank.
  #
  gallons = 0.0
```

Classe FillUp

- *__init__*
- *getOdometer* - retorna a leitura do odômetro
- *getGallons* - retorna a leitura de galões consumidos

Método *__init__* da Classe FillUp

```
##
  # Constructs a new FillUp object with the given data.
  # @param givenOdometer
  #   odometer reading
  # @param givenGallons
  #   number of gallons
  #
  def __init__(self, givenOdometer, givenGallons):
    self.odometer = givenOdometer
    self.gallons = givenGallons
```

Métodos *getOdometer* e *getGallons* da Classe FillUp

- Método getOdometer

```
##
  # Returns the odometer reading.
  # @return
  #   the odometer reading
  #
  def getOdometer(self):
    return self.odometer
```

- Método getGallons

```
  ##
  # Returns the number of gallons.
  # @return
  #   number of gallons
  #
  def getGallons(self):
    return self.gallons
```

Classe MileageCalculator

```
## Class for controlling the gasoline consumption of a car.
#
class MileageCalculator:

    ### fillup list, which aggregates two values.
    fillup = []

    ### debugging state.
    debug = False
```

Métodos da Classe MileageCalculator

- *__init__*
- *Reader* - lê no arquivo leituras de distância e gasolina
- *consumption* - calcula o consumo
- *__repr__*
- *__str__*

Método *__init__* da Classe MileageCalculator

```
##
#   Constructor.
#   Opens filename and calls Reader for inputting the mileage re
#   Raises an exception if filename does not exist.
#
#   @param filename mileage file name.
#
def __init__ (self, filename):
 try:
   f = open(filename, 'r')
 except IOError:
   print ('Cannot open file %s for reading' % filename)
   raise
 self.Reader(f)
```

Método *Reader* da Classe MileageCalculator

```
##
#   Reads a file with mileage  and gasoline  per line.
#   Creates FillUp object for each line and inserts in the fillup list.
#
#   @param f mileage file object.
#
def Reader (self, f):
 for line in f:
  tempwords = line.split(None)
  if len(tempwords) == 2:
   try:
    self.fillup.append(FillUp(float(tempwords[0]),float(tempwords[1])))
   except:
    print ('Invalid reading: %s\n' % tempwords)

 f.close()
```

Método *consumption* da Classe MileageCalculator

```
##
#   Calculates the consumption of the k-th entry of fillup list.
#
#   @param k fillup index.
#   @return (odometer[k]-odometer[k-1])/gallons[k].
#
def consumption ( self, k ):
 if k < 1 or k >= len(self.fillup): return None

 previous = self.fillup[k-1].getOdometer()
 current  = self.fillup[k].getOdometer()
 gallons  = self.fillup[k].getGallons()

 if MileageCalculator.debug:
  print("current %f\nprevious%f\ngallons %f\n"%(current,previous,gallons))

 return (current-previous)/gallons
```

Método *__repr__* da Classe MileageCalculator

```
##
#   Returns the consumption (in mi/gal) corresponding to each
#   entry of "fillup", by calling the method "consumption".
#
#   @return a string: a series of consumptions.
#
def __repr__(self):
  sb = "Miles per gallon\n"
  for i in range(1, len(self.fillup)):
     sb += "Consump. %d: %.3f\n" % (i,self.consumption(i))

  return sb
```

Método *__str__* da Classe MileageCalculator

```
##
#   Returns the consumption (in km/lt) corresponding to each entry
#    of "fillup", by calling the method "consumption".
#
#   1 gallon = 3.7854118 litres
#
#   1 mile   = 1.609344  kilometers
#
#   @return a string: a series of consumptions.
#
def __str__(self):
 sb = "Kilometers per litre\n"
 for i in range(1, len(self.fillup)):
  sb +="Consump.%d: %.3f\n"%(i,self.consumption(i)*1.609344/3.7854118)

 return sb
```

Main - Parte 1

```
##
#    Main method. Reads a series of pairs of mileage and number o
#    the average consumption: (current-previous)/gallons.
#
def main(argv=None):
    f = "mileage.txt"
    d = False
    if argv is None:
        argv = sys.argv

    if ( len(argv) > 2 ):
        f = argv[1]
        d = argv[2]=='True'
```

Main - Parte 2

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

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

O Arquivo

```
#!/usr/bin/env python
# coding: UTF-8
#
## @package MileageCalculator
#
#   Class for reading a series of odometer and gallons data
#   from filling up the gas tank of a vehicle, and calculating
#   the consumption between each gas station stop.
#
# @author Paulo Roma
# @date 24/08/2014
#

import sys
```

Classes FillUp + MileageCalculator + Main

Executando

```
$ more mileage.txt
91183 12.878
91538 11.007
91884 10.351
92164 9.644
92400 8.125
92812 12.629
123abc xyz

$ python MileageCalculator.py
Invalid reading: ['123abc', 'xyz']

Kilometers per litre
Consump. 1: 13.712
Consump. 2: 14.211
Consump. 3: 12.343
Consump. 4: 12.349
Consump. 5: 13.870
Consump. 6: 13.407
Consump. 7: 12.115

Miles per gallon
Consump. 1: 32.252
Consump. 2: 33.427
Consump. 3: 29.034
Consump. 4: 29.046
Consump. 5: 32.623
Consump. 6: 31.534
Consump. 7: 28.497
```