Instituto Politécnico de Beja

Projecto de Análise de Inscrições no Ensino Superior desenvolvido em Python

Jorge Emanuel Candeias Coveiro, nº 10873
Carlos Eduardo Rosário, nº 12794

02 de Dezembro de 2012

1. Introdução

Este primeiro trabalho foi realizado no âmbito da unidade curricular de Linguagens de Programação do primeiro semestre do segundo ano do curso de Engenharia Informática da ESTIG, pelos alunos Jorge Coveiro e Carlos Rosário.

Contextualiza-se no prosseguimento da aprendizagem de duas linguagens de programação: Python e C#, tendo esta primeira parte sido desenvolvida na primeira linguagem referida com o objectivo analisar a variação de inscrições nas diversas áreas e níveis de estudo ao longo dos passados anos lectivos (de 1995 a 2011).

O relatório está repartido por uma breve introdução teórica, parte experimental e conclusão. Na parte teórica fazemos uma pequena abordagem da linguagem Python entre as diferentes linguagens de programação. Na parte experimental abordamos a realização do trabalho, os sistemas adoptados e os resultados práticos. Finalmente na conclusão fazemos uma retrospectiva de todo o trabalho e discutem-se melhorias do mesmo.

2. Teoria

Todas as linguagens de programação têm o mesmo poder computacional de uma *Máquina de Turing*, o que difere é a forma de como o código é escrito.

Estas são caracterizadas por uma forma (*síntaxe*) e significado (*semântica*). A síntaxe é usada para referir o estudo das regras que regem o comportamento de sistemas matemáticos como a lógica ou as linguagens de programação, enquanto a semântica estuda o significado para se expressar através de uma linguagem. A semântica pode ser *estática* ou *dinâmica*. Estática é o que acontece em tempo de compilação. Dinâmica é o que acontece em tempo de execução.

Em linguagens de programação começamos por abordar os diferentes *paradigmas computacionais*: imperativo (e.g. C), lógico ou declarativo (e.g. Prolog), funcionais (e.g. Haskell, Lisp) e object oriented. No entanto muitas suportam vários paradigmas e são assim chamadas de multiparadigmáticas (e.g. Java, Python).

A programação imperativa descreve a computação como acções, enunciados ou comandos que mudam o estado de um programa. Os paradigmas lógicos e funcionais situam-se muito perto das matemáticas enquanto as object oriented são muito parecidas às imperativas.

Numa linguagem lógica parece mais que se está a descreve um problema e as suas propriedades do que propriamente dizer ao computador como realizar os cálculos. Em vez de “tira um elemento da lista x”, diz-se “existe uma lista que é igual à lista x, mas tem menos um elemento”.

Nas linguagens funcionais abdica-se de tudo aquilo que forme efeitos colaterais, ou seja, altere a estrutura dos dados originais.

A linguagem Python faz parte da classe de linguagens de programação dinâmicas. É de um nível elevado, interpretada e permite uma leitura fácil do código. Suporta a programação multiparadigmática: orientada por objectos, imperativa e funcional. Tem também um sistema de gestão de memória automática e um biblioteca padrão extensa, facilitando a sua implementação.

3. Parte Experimental

3.1 Realização Experimental

Neste trabalho usamos a linguagem Python por não haver opção de escolha.
Utilizamos também os seguintes portáteis com o seu respectivo sistema operativo:

- ➔ Laptop Acer Intel i3-370M@2,4GHz c/ 4GB RAM + Ubuntu Linux (32bit)
- ➔ Laptop Acer AMD AthlonX2@1,2GHz c/ 3GB RAM + Linux Mint Cinnamon (64bit)

Para programar utilizamos vários ambientes de desenvolvimento e editores de texto entre os quais: Ninja-IDE, Sublime-Text-2, Geany, Spyder.

3.2 Sistema Experimental

O trabalho é constituido por 2 packages: o core e o gui e também um ficheiro main.py que é o cerne do programa que utiliza estes packages e o ficheiro Inscritos_2010-2011.xls.

O package core contém 3 módulos: o dbhandler.py, o xlshandler.py e o curso.py.

O módulo dbhandler.py contém a classe DbHandler que opera sobre a base de dados. O módulo xlshandler.py contém a classe XlsHandler que opera sobre o ficheiro Inscritos_2010-2011.xls. Por último o módulo curso.py que contém a classe Curso que é utilizada pela classe DbHandler para o uso posterior na criação dos gráficos.

Finalmente o package gui contém módulos para a interface gráfica do nosso projecto: matplotlibwidgetFile.py e PlotGUI.py

3.3 Resultados Experimentais

Os testes ao trabalho foram realizados em ambiente Linux através do Geany. Utilizamos o matplotlib para a representação dos dados em gráficos. Experimentalmente o tempo total de execução do nosso trabalho é de aproximadamente 23 segundos no portátil Intel i3 incluido escrita do ficheiro-resultado csv e consultas à base de dados.

O resultado das estatisticas foram escritos num ficheiro estatisticas.csv aonde podemos comprovar:

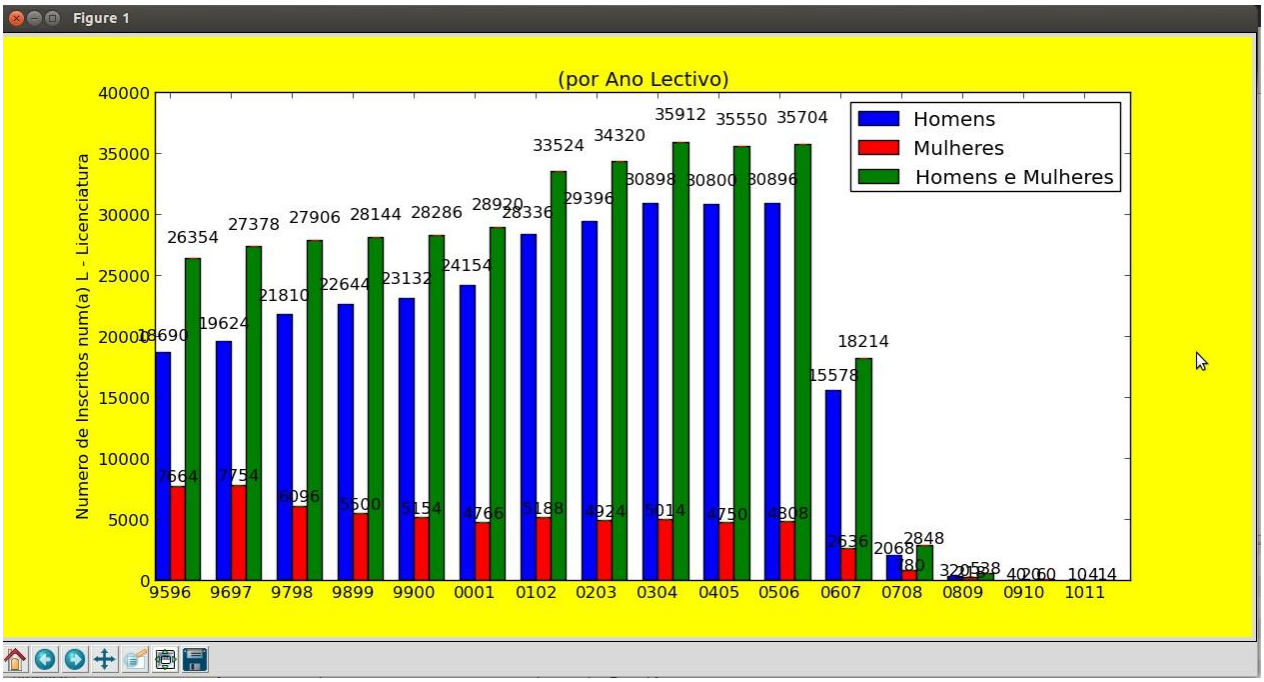
a) A presença de 928 cursos que incluem o termo computadores e informática.
b) Os totais de homens inscritos nestes cursos aumentam ao longo dos anos.
c) Os totais de mulheres inscritas nestes cursos diminui ao longo dos anos.
d) No geral há mais licenciaturas por número de cursos.
e) A quantidade de inscritos em Licenciaturas 1º Ciclo e Mestrado Integrado tem vindo a aumentar muito nos últimos anos.
f) Os bacharelatos diminuiram ao longo dos anos.

Em baixo, apresentamos o resultado dos nível de formação por número de cursos:

Nível de Formação	Número de Cursos
L - Licenciatura	162
P - Preparatórios de licenciatura	4
L1 - Licenciatura - 1º ciclo	234
PL - Preparatório de licenciatura 1ºciclo	2
M2 - Mestrado - 2º ciclo	90
D3 - Doutoramento - 3º ciclo	52
DE - Diploma de estudos superiores Especializados	20
MI - Mestrado Integrado	14
D - Doutoramento	42
C - Curso de Especialização Tecnológica	10
M - Mestrado	42
E – Especialização pós-licenciatura	24
B - Bacharelato	98
C0 - Curso de Especialização Tecnológica	26
LB - Bacharelato+Licenciatura	108

Exemplo de um gráfico gerado pelo matplotlib:

Neste gráfico verificamos a evolução do número de Inscritos em Licenciaturas ao longo dos anos, tanto para homens como para mulheres e ambos.



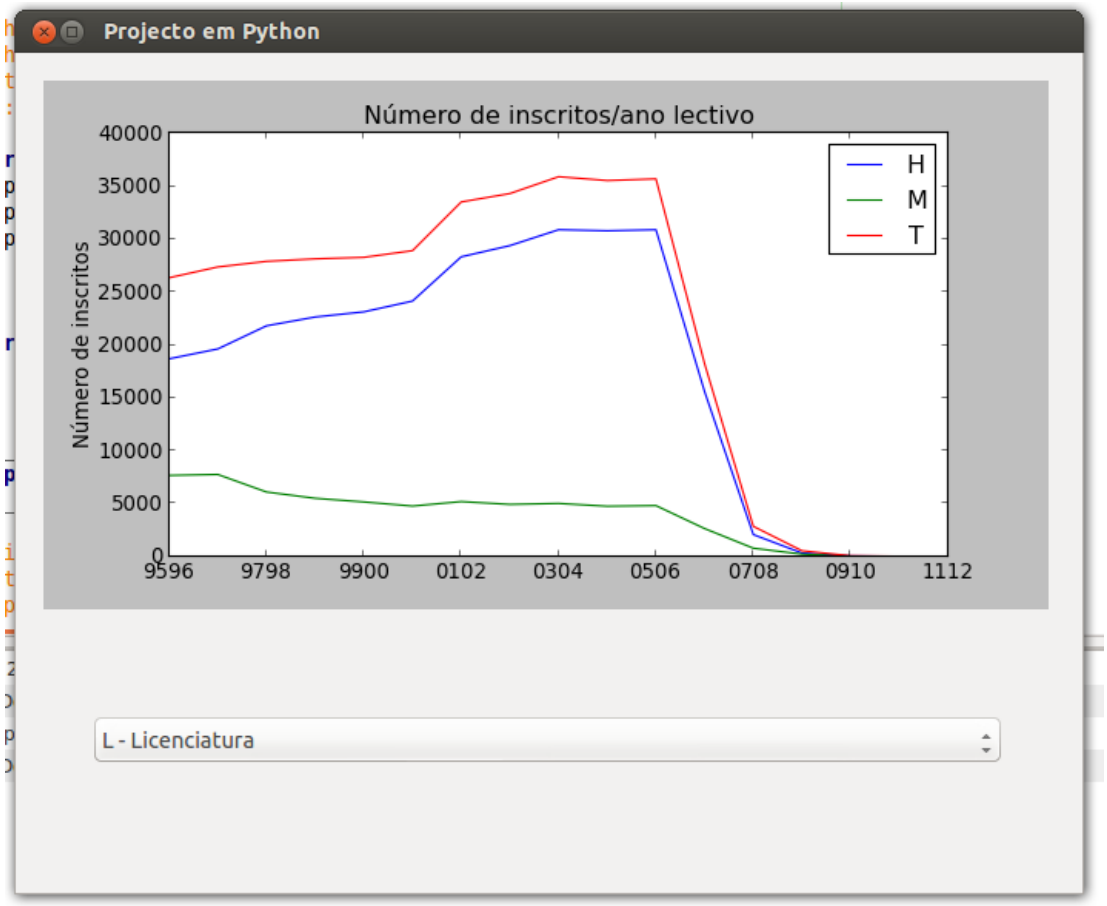


Imagem da GUI do programa

Conclusões

Com este trabalho comprovamos a flexibilidade e instrumentalidade da linguagem de programação Python. O resultado das estatisticas, escrita em ficheiros csv e leitura a partir da base de dados relevou-se importante como instrumento de análise e estudo desta mesma linguagem. Proporcionou-nos uma abordagem científica útil em relação às linguagens de programação.

Consideramos assim o nosso trabalho realizado com sucesso e em tempo útil.

Bibliografia

Google

Anexos

Ficheiro curso.py

```
# -*- coding: utf-8 -*-
'''
@author: 10873 Jorge Coveiro
@author: 12794 Carlos Rosário
@last updated: 3-12-2012
'''

class Curso:
	'''
	Classe que contem o número de alunos inscritos
	por ano  lectivo no curso 'nome', por sexo e o
	número total de alunos em todos os anos lectivos
	'''
	def __init__(self, nome = "", totalH = [], totalM = [], totalHM = [], totalAnos = 0):
		self.nome = nome
		self.totalH = totalH
		self.totalM = totalM
		self.totalHM = totalHM
```

Ficheiro dbhandler.py

```
# -*- coding: utf-8 -*-
'''
@author: 10873 Jorge Coveiro
@author: 12794 Carlos Rosário
@last updated: 3-12-2012
@obs: CLASSE PARA ESCRITA E LEITURA NA BASE DE DADOS
'''

import csv
import sqlite3
from xlshandler import XlsHandler
from curso import Curso
from nivelformacao import NivelFormacao
```

```
class DbHandler:
    '''
    Trata de todas as operações envolvendo
    a base de dados
    '''
    def __init__(self, db):
        self.ESTABELECIMENTO = 0
        self.UNIDADE_ORGANICA = 1
        self.NIVEL_FORMACAO = 2
        self.CURSO = 3
        self.HOMENS = 5
        self.MULHERES = 6
        self.HOMENS_MULHERES = 7
        self.table_name = ''
        self.db=db
        self.anos =
["9596","9697","9798","9899","9900","0001","0102","0203","0304","0405","0506","06
07","0708","0809","0910","1011"]

        self.conn=sqlite3.connect(self.db)
        pass

    def createTable(self, table_name):
        '''
        Cria uma tabela chamada table_name com os campos do ficheiro xls
        '''
        self.table_name = table_name
        cursor=self.conn.cursor()
        criatabela='''CREATE TABLE ''' + self.table_name + "(estabelecimento text,
unidade_organica text, nivel_formacao text, curso text, "

        for ano in self.anos:
            '''
            Escreve os argumentos da tabela
            "homens0001 number, mulheres0001 number, homens_mulheres0001 number"
            ....até 1011
            '''
            if ano=="1011":
                criatabela += "homens" + ano + " number, mulheres" + ano + " number,
homens_mulheres" + ano + ''' number)'''
            else:
                criatabela += "homens" + ano + " number, mulheres" + ano + " number,
homens_mulheres" + ano + " number, "

        cursor.execute(criatabela)
        self.conn.commit()
```

```
        cursor.close()

    def writeToTable(self, xls_f):
        '''
        Insere valores do ficheiro xls nos campos da tabela
        '''
        cursor=self.conn.cursor()
        xls = XlsHandler(xls_f)
        rows = xls.readRows()
        tmp = ''
        for row in rows:
            '''
            Faz uma insercao dos valores de cada linha do ficheiro xls
            para a base de dados
            '''
            homens=self.HOMENS
            mulheres=self.MULHERES
            homens_mulheres=self.HOMENS_MULHERES
            fim_linha=len(row)

            '''
            Insere os estabelecimentos e as
            unidades organicas em todas as linhas
            da base de dados
            '''
            if row[self.ESTABELECIMENTO]=='':
                row[self.ESTABELECIMENTO]=e
            else:
                e=row[self.ESTABELECIMENTO]
            '''
            tmp == e (serve para imprimir as unidades organicas relativas
            a um estabelecimento "e"), porque há estabelecimentos sem unidade orgânica
            '''
            if (row[self.UNIDADE_ORGANICA] == '') & (tmp == e):
                row[self.UNIDADE_ORGANICA]=uo
            else:
                uo=row[self.UNIDADE_ORGANICA]
            tmp=e
            if (row[self.NIVEL_FORMACAO] == ''):
                row[self.NIVEL_FORMACAO] = nf
            else:
                nf = row[self.NIVEL_FORMACAO]
            instruction='''INSERT INTO ''' + self.table_name + " VALUES(\"" +
str(row[self.ESTABELECIMENTO]) + "\", \"" + str(row[self.UNIDADE_ORGANICA]) +
"\", \"" + str(row[self.NIVEL_FORMACAO]) + "\", \"" + str(row[self.CURSO]) + "\", "
```

```
        while(homens_mulheres < fim_linha):
            '''
            Percorre as colunas  do ficheiro xls. Se o valor for "-"
            é trocado por '', e concatena os valores à string instruction
            '''
            if row[homens]=='-':
                row[homens]=0
            if row[mulheres]=='-':
                row[mulheres]=0
            if row[homens_mulheres]=='-':
                row[homens_mulheres]=0


            instruction += str(row[homens]) + ", " + str(row[mulheres]) + ", " +
str(row[homens_mulheres])

            if homens_mulheres == fim_linha-1:
                '''
                Se chegámos ao fim da linha do ficheiro xls
                então concatenamos à string instruction ")"
                para fecharmos a instrução.
                Se não continuamos a concatenar valores separados por ","
                '''
                instruction += ''')'''
                homens_mulheres+=homens+2
            else:
                instruction += ", "
                if row[homens_mulheres+1]=='':
                    '''
                    Aqui alteramos os indices
                    Nota: Há uma coluna vazia no ficheiro xls por isso
                    vamos ter de adicionar 2 em vez de 1 ao indice homens
                    '''
                    homens=homens_mulheres+2
                    mulheres=homens+1
                    homens_mulheres=mulheres+1
                    pass
                else:
                    homens=homens_mulheres+1
                    mulheres=homens+1
                    homens_mulheres=mulheres+1

        '''
        Finalmente executamos a instrucao por cada linha do xls
        '''
        cursor.execute(instruction)
```

```
            pass
        self.conn.commit()
        cursor.close()
        pass

    def dbToCsv(self, col_name, args):
        '''
        Escreve os dados args da coluna col_name num
        ficheiro .csv
        '''
        cursor = self.conn.cursor()
        cursor2 = self.conn.cursor()
        csv_writer = csv.writer(open("results/estatisticas.csv", "w"))
        nome_colunas_topo = ["Estabelecimento", "Unidade Orgânica", "Nivel de
Formação", "Curso"]
        nome_colunas_total = ['', '', '', '']
        nome_colunas_nf = ["Nível de Formação", '', '', '', "Número de Cursos"]
        nome_colunas_inscritos_nf = ["Nível de Formação", '', '', '']

        #Ciclo que completa o array com os titulos das colunas das diferentes
        #secções do ficheiro csv
        for i in range(len(self.anos)):
            #Acrescenta o ano lectivo ao nome das tabelas no inicio do ficheiro
            nome_colunas_topo += ["Homens " + str(self.anos[i])]
            nome_colunas_topo += ["Mulheres " + str(self.anos[i])]
            nome_colunas_topo += ["Homens e Mulheres " + str(self.anos[i])]

            #Acrescenta o ano lectivo a nome_colunas_total para
            #apresentar por cima dos totais dos alunos inscritos ao longo
            #desses anos lectivos
            nome_colunas_total += ["Homens " + str(self.anos[i])]
            nome_colunas_total += ["Mulheres " + str(self.anos[i])]
            nome_colunas_total += ["Homens e Mulheres " + str(self.anos[i])]

            nome_colunas_inscritos_nf += ["Homens " + str(self.anos[i])]
            nome_colunas_inscritos_nf += ["Mulheres " + str(self.anos[i])]
            nome_colunas_inscritos_nf += ["Homens e Mulheres " + str(self.anos[i])]

        csv_writer.writerow(nome_colunas_topo)
        #Procura na tabela table_name todos os resultados que
        #tenham table_name igual a args[0] e args[1], e escreve
        #os resultados no ficheiro estatisticas.csv
        cursor.execute("SELECT * FROM " + self.table_name + " WHERE " + col_name + "
LIKE " + "'%" + args[0] + "%'" + " OR " + col_name + " LIKE " + "'%" + args[1] +
"%'")
        for el in cursor:
```

```
            csv_writer.writerow([s.encode("utf-8") if isinstance(s,unicode) else s for s in el])
        csv_writer.writerow([])

        csv_writer.writerow(nome_colunas_total)

        #a_escrever vai conter a soma de todos os alunos(homens, mulheres, e ambos)
        #em cada ano lectivo dos cursos seleccionados na linha "157"
        a_escrever = ['Total','','','']
        coluna = ['homens', 'mulheres', 'homens_mulheres']
        for ano in self.anos:
            for i in range (3):
                cursor.execute("SELECT SUM(" + coluna[i] + str(ano) + ") FROM " +
self.table_name + " WHERE " + col_name + " LIKE '%" + args[0] + "%' OR " +
col_name + " LIKE '%" + args[1] + "%'")
                a_escrever.append(cursor.fetchone()[0])
        csv_writer.writerow(a_escrever)
        csv_writer.writerow([])

        csv_writer.writerow(nome_colunas_nf)
        #Escreve a quantidade de cursos por nível de formação
        cursor.execute("SELECT DISTINCT nivel_formacao FROM " + self.table_name + "
WHERE " + col_name + " LIKE '%" + args[0] + "%' OR " + col_name + " LIKE '%" +
args[1] + "%'")
        for nivel_formacao in cursor:
            a_escrever = [s.encode("utf-8") if isinstance(s, unicode) else s for s in
nivel_formacao]
            a_escrever += ['','','']

            #cursor2 contem o numero de cursos por nivel de formacao == nivel_formacao
            cursor2.execute("SELECT count(*) FROM " + self.table_name + " WHERE
(nivel_formacao == '" + nivel_formacao[0].encode("utf-8") + "') AND ((curso LIKE '%" +
args[0] + "%') OR (curso LIKE '%" + args[1] + "%'))")
            tmp = cursor2.fetchone()[0]
            a_escrever.append(tmp)
            csv_writer.writerow(a_escrever)
        csv_writer.writerow([])

        csv_writer.writerow(nome_colunas_inscritos_nf)
        #Escreve a quantidade de alunos por nível de formação ao longo dos anos
        cursor.execute("SELECT DISTINCT nivel_formacao FROM " + self.table_name + "
WHERE " + col_name + " LIKE '%" + args[0] + "%' OR " + col_name + " LIKE '%" +
args[1] + "%'")
        for nivel_formacao in cursor:
            a_escrever = [s.encode("utf-8") if isinstance(s, unicode) else s for s in
nivel_formacao]
            for n in range (3):
```

```
            a_escrever.append('')

        for ano in self.anos:
            for i in range(3):
                cursor2.execute("SELECT SUM(" + coluna[i] + ano + ") FROM " +
self.table_name + " WHERE (nivel_formacao == '" + nivel_formacao[0].encode("utf-8") +
"') AND ((" + col_name + " LIKE '%" + args[0] + "%') OR (" + col_name + " LIKE '%"
+ args[1] + "%'))")
                a_escrever.append(cursor2.fetchone()[0])
            csv_writer.writerow(a_escrever)

        del csv_writer

    def getInscritosCurso(self, col_name, args):
        '''
        Retorna uma lista com o nr de inscritos por ano lectivo
        nos diversos cursos
        '''
        cursor = self.conn.cursor()
        totalH = [0 for i in range(16)]
        totalM = [0 for i in range(16)]
        totalHM = [0 for i in range(16)]
        cursos = [] #lista de cursos

        cursor.execute("SELECT * FROM " + self.table_name + " WHERE " + col_name + "
LIKE " + "'%" + args[0] + "%'" + " OR " + col_name + " LIKE " + "'%" + args[1] +
"%'")

        for row in cursor:
            escreve_h = True
            escreve_m = False
            escreve_hm = False
            curso = Curso()
            curso.totalH = []
            curso.totalM = []
            curso.totalHM = []

            for i in range(len(row)):
                if (i == 3):
                    curso.nome = row[i]
                elif (i > 3):
                    if (escreve_h == True):
                        curso.totalH.append(int(row[i]))
                        escreve_h = False
                        escreve_m = True
                    elif (escreve_m == True):
```

```
                curso.totalM.append(int(row[i]))
                escreve_m = False
                escreve_hm = True
            elif (escreve_hm == True):
                curso.totalHM.append(int(row[i]))
                escreve_hm = False
                escreve_h = True
        i += 1

    cursos.append(curso)

  return cursos

def getInscritosNF(self):
  '''
  Retorna uma lista com o nr de inscritos por ano lectivo
  nos diversos niveis de formacao
  '''
  cursor = self.conn.cursor()
  cursor2 = self.conn.cursor()
  coluna = ['homens', 'mulheres', 'homens_mulheres']
  totalH = [0 for i in range(16)]
  totalM = [0 for i in range(16)]
  totalHM = [0 for i in range(16)]
  niveis_formacao = [] #lista de niveis de formacao

  cursor.execute("SELECT DISTINCT nivel_formacao FROM inscritos WHERE curso
LIKE '%Computadores%' OR curso LIKE '%Informática%'")
  for nf in cursor:
    nivel_formacao = NivelFormacao()
    nivel_formacao.nome = ""
    nivel_formacao.totalH = []
    nivel_formacao.totalM = []
    nivel_formacao.totalHM = []

    nivel_formacao.nome = nf[0]
    for ano in self.anos:
      for i in range(3):
        cursor2.execute("SELECT SUM(" + coluna[i] + ano + ") FROM inscritos
WHERE (nivel_formacao == '" + nf[0].encode("utf-8") + "') AND ((curso LIKE
'%Computadores%') OR (curso LIKE '%Informática%'))")
        if(i == 0):
          nivel_formacao.totalH.append(int(cursor2.fetchone()[0]))
        elif(i == 1):
          nivel_formacao.totalM.append(int(cursor2.fetchone()[0]))
        elif(i == 2):
```

```
                nivel_formacao.totalHM.append(int(cursor2.fetchone()[0]))
            pass
        niveis_formacao.append(nivel_formacao)
    return niveis_formacao

def closeConnection(self):
    '''
    Fecha a conexao à base de dados
    '''
    self.conn.close()
```

Ficheiro main.py

```
# -*- coding: utf-8 -*-
'''
@author: 10873 Jorge Coveiro
@author: 12794 Carlos Rosário
@last updated: 08-12-2012
@obs: PROGRAMA PARA A ANALISE DE DADOS DE UM .XLS
'''
import sys
sys.path.append('core')
sys.path.append('dataplot')
sys.path.append('gui')
from dbhandler import DbHandler
from curso import Curso
from PlotGUI import *
```

```
import random


try:
    _fromUtf8 = QtCore.QString.fromUtf8
except AttributeError:
    _fromUtf8 = lambda s: s
"""
def imprime(texto):
        texto = texto.toUtf8()
        print texto
"""
class GUIForm(QtGui.QDialog):

    def __init__(self, cursos, niveis_formacao, parent=None):
        self.cursos = cursos
        self.niveis_formacao = niveis_formacao

        QtGui.QWidget.__init__(self,parent)
        self.ui = Ui_Dialog()
        self.ui.setupUi(self, cursos, niveis_formacao)

        #Envia o texto do item da comboBox seleccionado
        self.ui.graph_comboBox.activated[int].connect(self.PlotFunc)

    def PlotFunc(self, val):
        if ((val < len(self.cursos) + 2) & (val > 1)): #2 sao os dois primeiros items da comboBox
            self.ui.widget.canvas.ax1.clear()
            self.ui.widget.canvas.ax2.clear()
            self.ui.widget.canvas.ax3.clear()

            self.ui.widget.canvas.ax1.set_ylabel(u'Número de inscritos')
            self.ui.widget.canvas.ax1.set_title(u'Número de inscritos/ano lectivo')
            self.ui.widget.canvas.ax1.plot(cursos[val-2].totalH)
            self.ui.widget.canvas.ax2.plot(cursos[val-2].totalM)
            self.ui.widget.canvas.ax3.plot(cursos[val-2].totalHM)
            self.ui.widget.canvas.ax1.legend('HMT')
            self.ui.widget.canvas.ax1.set_xticklabels( ('9596', '9798', '9900',
                                        '0102', '0304', '0506',
                                                '0708', '0910', '1112') )

        elif(val > len(self.cursos) + 2):
            self.ui.widget.canvas.ax1.clear()
            self.ui.widget.canvas.ax2.clear()
            self.ui.widget.canvas.ax3.clear()
```

```
            self.ui.widget.canvas.ax1.set_ylabel(u'Número de inscritos')
            self.ui.widget.canvas.ax1.set_title(u'Número de inscritos/ano lectivo')
            self.ui.widget.canvas.ax1.plot(niveis_formacao[val-len(self.cursos)-3].totalH)
            self.ui.widget.canvas.ax2.plot(niveis_formacao[val-len(self.cursos)-3].totalM)
            self.ui.widget.canvas.ax3.plot(niveis_formacao[val-len(self.cursos)-3].totalHM)
            self.ui.widget.canvas.ax1.legend('HMT')
            self.ui.widget.canvas.ax1.set_xticklabels( ('9596', '9798', '9900',
                                        '0102', '0304', '0506',
                                                '0708', '0910', '1112') )

        self.ui.widget.canvas.draw()


if __name__ == "__main__":
    db = DbHandler("database/Inscritos.sqlite3")
    db.createTable("inscritos")
    db.writeToTable("Inscritos_2010-2011.xls")
    db.dbToCsv("curso",["Computadores","Informática"])

    cursos = db.getInscritosCurso("curso", ["Computadores","Informática"])
    niveis_formacao = db.getInscritosNF()

    db.closeConnection()

    #Parte gráfica do programa
    app = QtGui.QApplication(sys.argv)
    myapp = GUIForm(cursos, niveis_formacao)
    myapp.show()
    sys.exit(app.exec_())
```

Ficheiro matplotlibwidgetFile.py

```
# -*- coding: utf-8 -*-
'''
@author: 10873 Jorge Coveiro
@author: 12794 Carlos Rosário
@obs: Esta porção do código foi adquirida na internet,
através de uma busca com o objectivo de incorporar um gráfico
realizado atravez do matplotlib, numa gui criada atravez de Qt4 Designer,
usando PyQt4
'''
from PyQt4 import QtGui
from matplotlib.backends.backend_qt4agg import FigureCanvasQTAgg as FigureCanvas

from matplotlib.figure import Figure

class MplCanvas(FigureCanvas):
```

```
    '''
    Cria uma tela para o gráfico ser desenhado.
    Esta tela vai ser usada pelo matplotlibWidget
    '''
    def __init__(self):
        self.fig = Figure()
        self.ax1 = self.fig.add_subplot(111)
        self.ax2 = self.fig.add_subplot(111)
        self.ax3 = self.fig.add_subplot(111)

        self.ax1.set_ylabel(u'Número de inscritos')
        self.ax1.set_title(u'Número de inscritos/ano lectivo')

        FigureCanvas.__init__(self, self.fig)
        FigureCanvas.setSizePolicy(self,
QtGui.QSizePolicy.Expanding,QtGui.QSizePolicy.Expanding)
        FigureCanvas.updateGeometry(self)


class matplotlibWidget(QtGui.QWidget):

    def __init__(self, parent = None):
        QtGui.QWidget.__init__(self, parent)
        self.canvas = MplCanvas()
        self.vbl = QtGui.QVBoxLayout()
        self.vbl.addWidget(self.canvas)
        self.setLayout(self.vbl)
```

Ficheiro nivelformacao.py

```
# -*- coding: utf-8 -*-

class NivelFormacao:
        '''
        Classe que contem o número de alunos inscritos
        por ano         lectivo no nivel de formação 'nome', por
        sexo e o número total de alunos em todos os anos lectivos
        '''
        def __init__(self, nome = "", totalH = [], totalM = [], totalHM = [], totalAnos = 0):
                self.nome = nome
                self.totalH = totalH
                self.totalM = totalM
                self.totalHM = totalHM
```

Ficheiro PlotGUI.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'PlotGUI_Final.ui'
#
# Created: Sat Dec  8 16:04:04 2012
#      by: PyQt4 UI code generator 4.9.3
#
# WARNING! All changes made in this file will be lost!
'''
Funções que foram modificadas:
setupUi : + 2 argumentos (cursos e niveis_formacao)
                    + 2 ciclos para preenchimento da combo box
'''
from PyQt4 import QtCore, QtGui
```

```
try:
    _fromUtf8 = QtCore.QString.fromUtf8
except AttributeError:
    _fromUtf8 = lambda s: s

class Ui_Dialog(object):
    def setupUi(self, Dialog, cursos, niveis_formacao):
        self.Dialog = Dialog
        Dialog.setObjectName(_fromUtf8("Dialog"))
        Dialog.resize(693, 545)
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap(_fromUtf8("icon/estig.ico")), QtGui.QIcon.Normal,
QtGui.QIcon.Off)
        Dialog.setWindowIcon(icon)
        self.widget = matplotlibWidget(self.Dialog) #Widget que vai conter os gráficos
        self.widget.setGeometry(QtCore.QRect(9, 9, 671, 361))
        self.widget.setObjectName(_fromUtf8("widget"))
        self.graph_comboBox = QtGui.QComboBox(Dialog)
        self.graph_comboBox.setGeometry(QtCore.QRect(50, 430, 591, 31))
        self.graph_comboBox.setObjectName(_fromUtf8("graph_comboBox"))
        self.graph_comboBox.addItem(_fromUtf8("")) # item_index == 0, "Escolha um
gráfico"
        self.graph_comboBox.addItem(_fromUtf8("")) # item_index == 1, "- - Por curso - -"
        self.graph_comboBox.setItemText(1, _fromUtf8(" - - Por curso - -"))

        #Adiciona items à combo box, correspondentes aos cursos existentes
        #com Computadores ou Informática no nome
        item_index = 2;
        for curso in cursos:
            self.graph_comboBox.addItem(_fromUtf8(""))
            self.graph_comboBox.setItemText(item_index, _fromUtf8(curso.nome))
            item_index += 1

        #Adiciona items à combo box, correspondentes aos cursos existentes
        #com Computadores ou Informática no nome
        self.graph_comboBox.addItem(_fromUtf8("")) # item_index == item_index, "- - Por
nivel de formação - -"
        self.graph_comboBox.setItemText(item_index, _fromUtf8(" - - Por nivel de formação -
-"))
        for nf in niveis_formacao:
            item_index += 1
            self.graph_comboBox.addItem(_fromUtf8(""))
            self.graph_comboBox.setItemText(item_index, _fromUtf8(nf.nome))

        self.retranslateUi(Dialog)
        QtCore.QMetaObject.connectSlotsByName(Dialog)
```

```
    def retranslateUi(self, Dialog):
        Dialog.setWindowTitle(QtGui.QApplication.translate("Dialog", "Projecto em Python", None, QtGui.QApplication.UnicodeUTF8))
        self.graph_comboBox.setItemText(0, QtGui.QApplication.translate("Dialog", "Escolha o gráfico", None, QtGui.QApplication.UnicodeUTF8))

from matplotlibwidgetFile import matplotlibWidget
```

Ficheiro xlshandler.py

```
#-*- coding: utf-8 -*-
'''
@author: 10873 Jorge Coveiro
@author: 12794 Carlos Rosário
@last updated: 15-11-2012
@obs: CLASSE PARA A LEITURA DE UM FICHEIRO .XLS
'''
import xlrd

class XlsHandler:
        '''
        Trata das operações com
        ficheiros xls
        '''
```

```
        def __init__(self, xls_f, starting_xls_row = 4, sheet_number = 30, irrelevant_nr_rows
= 9):
                self.starting_xls_row=starting_xls_row
                self.sheet_number=sheet_number
                self.irelevant_rows=irrelevant_nr_rows
                self.book=xlrd.open_workbook(xls_f)
                self.sheet=self.book.sheet_by_index(self.sheet_number)
                self.rows=[]
                self.readRows()
                pass

        def readRows(self):
                '''
                Devolve um alista rows com as
                linhas relevantes do ficheiro xls
                '''
                for rown in range(self.starting_xls_row, self.sheet.nrows -
self.irelevant_rows):
                        self.rows.append([s.encode("utf-8") if isinstance(s,unicode) else s for s
in self.sheet.row_values(rown)])
                        pass
                return self.rows
```
