

Relatório do Trabalho Prático de Linguagens de Programação

Análise de Inscrições no Ensino Superior

Este relatório descreve um programa que importa os dados de uma folha de Excel para uma base de dados e permite a consulta das queries e gráficos das mesmas através de um interface gráfico.

Pedro Faisco n.º 6055
Pedro Cacione n.º 9989



Índice

Introdução.....	2
Contextualização do Trabalho.....	2
Objectivos Gerais	2
Estrutura do Relatório	3
Teoria	4
Parte Experimental	6
Linguagem de Programação.....	6
Ambiente de Desenvolvimento	6
Sistema Operativo	7
Hardware	7
Sistema Experimental	7
Resultados Experimentais	8
Protocolo Experimental	8
Apresentação e Discussão dos Resultados Experimentais.....	8
Conclusões	12
Bibliografia.....	13
Anexos.....	14

Introdução

Contextualização do Trabalho

O trabalho desenvolvido para esta cadeira tem como objectivo uma melhor interpretação e análise das capacidades das linguagens de programação, neste caso, com especial foco em Python uma linguagens de programação dinâmica. Estas linguagens caracterizam-se por serem de nível elevado por assumirem em tempo de execução, comportamentos comuns a outras linguagens em tempo de compilação. A linguagem de programação Python é interpretada, tem objectivos gerais e enfatiza a legibilidade do código. Suporta a programação multiparadigmática: orientação por objectos; estilo imperativo e funcional.

O trabalho está focado na interpretação de um ficheiro xls para uma base de dados de modo a debitar as *queries* por nós definidas e respectivos gráficos

Objectivos Gerais

Os objectivos gerais deste trabalho são:

Usar um ficheiro em formato Excel 97, designado Inscritos_2010-2011.xls, e fazer a leitura com o módulo xlrd seguido da sua importação para uma base de dados sqlite3 utilizando o módulo sqlalchemy (object relational mapping) de modo a executar a leitura e escrita do ficheiro. De seguida utilizamos o módulo WxGade para realizar o interface gráfico do projecto. Estes objectivos são executados de modo a obter os resultados das estatísticas escritas num ficheiro em formato CSV que vai indicar:

- Os cursos que incluem o termo computadores e informática, em associação com o estabelecimento de ensino, unidade orgânica e anos lectivos em que funcionaram;
- A quantidade de alunos que nestes se inscreveram ao longo dos anos;
- A quantidade de cursos por nível de formação ao longo dos anos;
- A quantidade de alunos por nível de formação ao longo dos anos.

O programa foi desenvolvido usando um MVC (Model, View, Controller) responsável por fazer a gestão dos dados deste (Model) e proporcionar os meios para o utilizador interagir com o programa (View/Controller).

O programa permite a construção de gráficos, por meio do módulo matplotlib, para cada curso seleccionado e da evolução do número de alunos inscritos ao longo dos anos. Também representar a evolução do número total de alunos inscritos ao longo dos anos por nível de formação.

Os dados das estatísticas e dos gráficos provêm de consultas à base de dados.

Todos os aspectos do programa são comentados nos ficheiros fonte e realizados de modo a que serem processados pelo pydoc. Os comentários incluem informação sobre: os módulos; as classes; as funções e os argumentos das funções; as variáveis e todos os elementos de funcionamento da aplicação.

Estrutura do Relatório

O relatório está estruturado de modo a seguir os requisitos propostos pelo professor no enunciado do trabalho pratica.

Dedica uma página á capa onde expõe a identificação do trabalho.

Índice, elaborado com o propósito de facilitar a procura dentro do documento.

Uma introdução com 3 subsecções: Contextualização do trabalho, onde se enquadra o trabalho com os objectivos da cadeira; Objectivos, onde é feita a introdução aos objectivos propostos explicando os que foram realizados como os que não o foram; e ainda esta subsecção onde se faz a descrição da estrutura do actual relatório.

A teoria onde é explicado os aspectos teóricos considerados relevantes, tal com o funcionamento de algumas bibliotecas utilizadas no desenvolvimento deste programa.

A parte experimental onde se encontram as seguintes subsecções: Linguagem de programação; Ambiente de desenvolvimento; Sistema operativo; Hardware. Sistema Experimental onde está descrita a estrutura do código realizado e ainda os Resultados experimentais que ainda tem mais duas subsubsecções, uma que descrever o protocolo experimental usado para obter os resultados e outra que apresenta a discussão dos resultados experimentais.

No final são apresentadas as conclusões deste trabalho e se expõem as nossas perspectivas de melhoria, a Bibliografia com as referências bibliográficas e os Anexos

Teoria

No trabalho desenvolvido para esta cadeira de Linguagens de Programação foi usada a linguagem de programação Python, uma linguagem de programação dinâmica. Estas linguagens caracterizam-se por serem de nível elevado por assumirem em tempo de execução, comportamentos comuns a outras linguagens em tempo de compilação. A linguagem de programação Python é de nível elevado e interpretada, tem objectivos gerais e enfatiza a legibilidade do código. Suporta a programação multiparadigmática: orientação por objectos; estilo imperativo e funcional. Foi lançada por Guido van Rossum em 1991. Actualmente possui um modelo de desenvolvimento comunitário, aberto e gerenciado pela organização sem fins lucrativos Python Software Foundation. Apesar de várias partes da linguagem possuírem padrões e especificações formais, a linguagem como um todo não é formalmente especificada. O padrão *de facto* é a implementação CPython.

A linguagem foi projectada com a filosofia de enfatizar a importância do esforço do programador sobre o esforço computacional. Prioriza a legibilidade do código sobre a velocidade ou expressividade. Combina uma *sintaxe* concisa e clara com os recursos poderosos de sua biblioteca padrão e por módulos e *frameworks* desenvolvidos por terceiros.

O nome Python teve a sua origem no grupo humorístico britânico Monty Python, criador do programa *Monty Python's Flying Circus*, daí as citações que aparecem ao longo deste relatório.

O Python possui módulos para praticamente qualquer tarefa. Essa é uma das características da linguagem que o torna tão eficiente. Esta linguagem simplesmente move os dados por conjuntos de iteração e escreve esses dados correctamente ordenados numa folha de Excel. Os dados foram sendo lidos e gravados por meio da folha original sequencialmente pelo tempo, mas precisamos de ter os dados escritos por conjuntos de amostras que são compensados, tanto pelo número de repetições como pelo número de condições. Nós usamos o módulo XLRD para ler o arquivo do Excel, respectivamente.

Usamos o módulo sqlite3 para trabalhar com a base de dados SQLite. SQLite é uma biblioteca C que fornece uma base de dados com base em disco, leve que não requer um processo de servidor separado e permite o acesso a base de dados utilizando uma variante diferente do padrão da linguagem de consulta SQL.

Neste trabalho usamos a Object Relational Mapper (ORM, O / RM, e mapping O / R) que é uma técnica de programação para conversão de dados entre os sistemas de tipo incompatível na base de dados relacionais e linguagens de programação orientada a objectos, criando, com efeito, uma "base de dados de objecto virtual" que é usada a partir de dentro da linguagem de programação Python. Há dois pacotes gratuitos e comerciais disponíveis que realizam mapeamento objecto-relacional, embora alguns programadores optem por criar as suas próprias ferramentas ORM. Nós usamos o SQLAlchemy, uma ferramenta de Python SQL que fornece um conjunto de ferramentas que nos permite estabelecer uma ligação à base de dados trabalhando esses mesmos dados como Objectos. SQLAlchemy é mais famoso por seu Object Relational Mapper (ORM), um componente opcional que fornece o data mapper pattern, onde as classes podem ser mapeadas para uma base de dados, permitindo que o modelo dos objectos e o esquema da base de dados seja desenvolvido de uma forma limpa e bem estruturada desde o início.

O Model-view-controller (MVC) é um modelo de desenvolvimento de Software, actualmente considerado uma "arquitetura padrão". O modelo isola a parte "lógica" da aplicação da parte do interface do utilizador (Inserir e exibir dados), permitindo desenvolver, editar e testar separadamente cada parte. O problema é que o próprio MVC, como todos os conceitos padrão, por si só, não resolve todos os problemas e algumas vezes pode acabar por criar outros problemas. O uso ideal do MVC com cuidado e uma série de padrões de projecto podem nos garantir uma aplicação muito bem escrita e de fácil manutenção. Aliar esses padrões com a simplicidade e elegância do Python não é um equilíbrio tão fácil

de se alcançar, mas que é gratificante. No nosso caso também não foi possível utilizar o MVC “puro” pois o controller fica muito complexo logo tivemos que o dividir em várias partes mediante a actividade.

Inicialmente para a parte do Interface foi utilizado o PyQt que é um *wrapper* da linguagem Python para a biblioteca Qt, que é a base do KDE (ambiente desktop para Linux). Existe uma biblioteca complementar, PyKDE, que actua sobre elementos específicos do KDE, como por exemplo interacção com o *kicker* e a barra de tarefas. Suporta as plataformas Unix, Linux, Windows, Mac OS/X. No entanto acabamos por fazer o interface com o wxGlade que é um designer gráfico escrito em Python usando um kit de ferramentas GUI wxPython, que ajuda a criar interfaces wxWidgets / wxPython. No momento em que ele pode gerar Python, C ++, Perl, Lisp e XRC código (recursos wxWidgets 'XML'). Baseado no modelo Glade, o famoso GTK + / GNOME construtor de GUI, com que partes wxGlade a filosofia e os look & feel (mas não uma linha de código). Não é um IDE cheio de recursos, mas simplesmente um "designer": o código gerado não faz nada além de exibir os widgets criados.

Na execução dos gráficos utilizamos o módulo matplotlib (MPL) que é voltado para a geração de gráficos bi-dimensionais de vários tipos, preparado para a utilização tanto de forma interactiva quanto em *scripts*, aplicações web ou integrado a interfaces gráficas (GUIs) de vários tipos.

No final o resultado das *queries* é apresentado num ficheiro CSV, um ficheiro como o próprio nome indica, um ficheiro sem formatação em que os valores estão separados por vírgulas, delimitados por aspas e, em que, cada linha tem um registo (Artigo, cliente ou fornecedor). Pode-se criar facilmente um ficheiro CSV a partir de qualquer ficheiro de Folha de Cálculo, sendo os mais conhecidos o Microsoft Excel e o OpenOffice Calc.

Para gerar a documentação utilizamos o pydoc, um módulo que gera documentação (em formato *user friendly*) sobre módulos Python, a partir dos *docstrings* que estão presentes nestes. A documentação pode ser apresentada em mais do que uma forma. Uma delas, é o formato *man page*. Para ver a documentação de um módulo em formato *man page*, podemos invocar o pydoc através de um *shell* do sistema operativo.

Parte Experimental

Linguagem de Programação

Utilizamos o Python por ser requisito estabelecido pelo enunciado para a elaboração do trabalho, a versão escolhida foi a 2.7.3. É uma linguagem alto nível, interpretada e interactiva. Os conceitos fundamentais da linguagem são simples de entender. A sintaxe da linguagem é clara e fácil de aprender; o código produzido é normalmente curto e legível. Python é expressivo, com abstracções de alto nível. Na grande maioria dos casos, um programa em Python será muito mais curto que seu correspondente escrito em outra linguagem. Isto também faz com o ciclo de desenvolvimento seja rápido e apresente potencial de defeitos reduzido - menos código, menos oportunidade para errar. Existe suporte para uma diversidade grande de bibliotecas externas. Ou seja, pode-se fazer em Python qualquer tipo de programa, mesmo que utilize gráficos, funções matemáticas complexas, ou uma determinada base de dados SQL. É possível escrever extensões a Python em C e C++ quando é necessário desempenho máximo, ou quando for desejável fazer interface com alguma ferramenta que possua biblioteca apenas nestas linguagens. Python permite que o programa execute inalterado em múltiplas plataformas; em outras palavras, a sua aplicação feita para Linux normalmente funcionará sem problemas em Windows e em outros sistemas onde existir um interpretador Python. O Python é livre: além do interpretador ser distribuído como software livre (e portanto, gratuitamente), pode ser usado para criar qualquer tipo de software -- proprietário ou livre. O projecto e implementação da linguagem são discutidos aberta e diariamente em uma lista de correio electrónico, e qualquer um é bem-vindo para propor alterações por meio de um processo simples e pouco burocrático.

Ambiente de Desenvolvimento

Adoptamos vários ambientes de desenvolvimento nomeadamente o “Sublime Text 2” por ser ágil e fácil utilizar e também “WxGlade” que é acessível e fácil de gerar código Python.

Sistema Operativo

Neste trabalho foi utilizado o Windows 8 Pro 64-bit, Oracle VM VirtualBox Manager a correr Ubuntu 64-bit e o Ubuntu 64-bit (nativo).

Hardware

Intel(R) Core(TM)2 Duo CPU T9400 @ 2.53Ghz

4,00GB

Intel(R) Core(TM)2 Duo CPU T7250 @ 2.0Ghz

3,00GB

Sistema Experimental

O código está dividido em três partes cada uma cumpre uma função dentro dos requisitos estabelecidos o que não quer dizer que tenham sido elaborados por esta ordem:

A primeira parte refere-se à criação dos ficheiros “Model” com o objectivo de organizar a informação referente aos locais de leitura e escrita fornecida pelo ficheiro xls (Inscritos_2010-2011 (formato Excel xls).xls) onde o model_Xls.py grava os índices das colunas o Model_BD.py cria as tabelas usando o módulo sqlalchemy. No ficheiro Model_plot.py temos duas classes: a do Model_Plot() e a Controller_Plot() por não haver necessidade de separar o código em ficheiros diferentes devido à extensão. E por fim temos o ficheiro Model_csv.py onde também foram inseridas duas classes Model_stat() e Controller_stat() pelas mesmas razões anteriormente referidas e o objectivo deste ficheiro é a criação de um ficheiro csv com as estatísticas pedidas pelo enunciado.

Na segunda parte foram criados os ficheiros “Controller”. Um ficheiro (Controller_Xls.py) com o objectivo de fazer a recolha da Informação existente num ficheiro xls (Inscritos_2010-2011 (formato Excel xls).xls). Esta informação é inserida numa base de dados usando para isso o módulo sqlalchemy através do ficheiro Controller_DB.py.

Na terceira parte foi criado o ficheiro View.py com o objectivo de criar o interface gráfico usando para isso o módulo WxGlade. A ligação entre todos os módulos é feita com o ficheiro Controller_MAIN.py, este ficheiro vai ligar a parte do interface gráfico às funções dos outros “Controller” para uma execução mais rápida dos gráficos e dos ficheiros csv foram adicionadas threads bem como no MainLoop do programa.

Resultados Experimentais

Protocolo Experimental

Foram realizados vários testes com o propósito de averiguar falhas e não foram encontradas nenhuma.

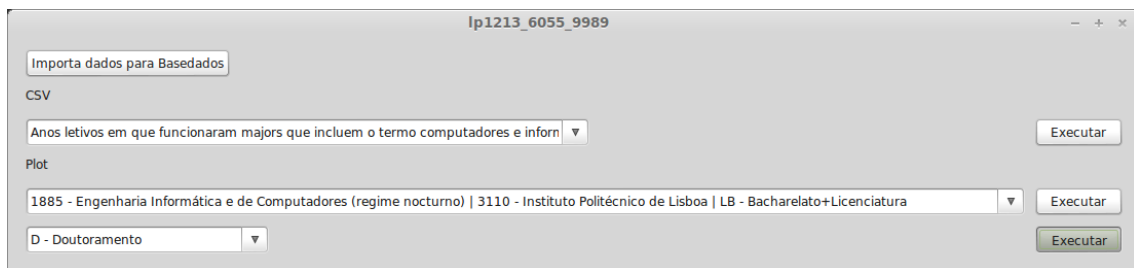
Apresentação e Discussão dos Resultados Experimentais

Os resultados obtidos são exactamente os pedidos no enunciado tanto os estatísticos como os gráficos. De acordo com o pedido mostramos alguns exemplos dos resultados obtidos.

Base de Dados

A Base de dados leva 4.29226207733 seconds a carregar.

Interface



The screenshot shows a software window titled "Ip1213_6055_9989". Inside the window, there is a button labeled "Importa dados para Basedados". Below this, under the heading "CSV", there is a text input field containing "Anos letivos em que funcionaram majors que incluem o termo computadores e inform" and a dropdown arrow, with an "Executar" button to its right. Under the heading "Plot", there is a text input field containing "1885 - Engenharia Informática e de Computadores (regime nocturno) | 3110 - Instituto Politécnico de Lisboa | LB - Bacharelato+Licenciatura" and a dropdown arrow, with an "Executar" button to its right. At the bottom, there is a text input field containing "D - Doutoramento" and a dropdown arrow, with an "Executar" button to its right.

Estatísticas:

Todos os cursos com as palavras “Computadores” e ”Informática”

A	B	C
1283 - Engenharia Informática e de Computadores (Preparatórios)	0100 - Universidade dos Açores	0130 - Universidade dos Açores - Ponta Del
9865 - Ciências de Engenharia - Eng Informática e de Computadores (Preparatórios)	0100 - Universidade dos Açores	0130 - Universidade dos Açores - Ponta Del
8084 - Engenharia Informática e de Computadores (Preparatórios)	0100 - Universidade dos Açores	0130 - Universidade dos Açores - Ponta Del
0294 - Engenharia Informática e de Computadores	0800 - Universidade Técnica de Lisboa	0807 - Instituto Superior Técnico
9037 - Ciências de Engenharia - Engenharia Informática e de Computadores	0800 - Universidade Técnica de Lisboa	0807 - Instituto Superior Técnico
9121 - Engenharia Informática e de Computadores	0800 - Universidade Técnica de Lisboa	0807 - Instituto Superior Técnico
4207 - Engenharia Informática e de Computadores	0800 - Universidade Técnica de Lisboa	0807 - Instituto Superior Técnico
9427 - Engenharia Informática e de Computadores	0800 - Universidade Técnica de Lisboa	0807 - Instituto Superior Técnico
5301 - Engenharia Informática e de Computadores	0800 - Universidade Técnica de Lisboa	0807 - Instituto Superior Técnico
5142 - Engenharia Informática e de Computadores	0800 - Universidade Técnica de Lisboa	0807 - Instituto Superior Técnico
0294 - Engenharia Informática e de Computadores	0800 - Universidade Técnica de Lisboa	0808 - Instituto Superior Técnico (Instalação
9037 - Ciências de Engenharia - Engenharia Informática e de Computadores	0800 - Universidade Técnica de Lisboa	0808 - Instituto Superior Técnico (Instalação
9121 - Engenharia Informática e de Computadores	0800 - Universidade Técnica de Lisboa	0808 - Instituto Superior Técnico (Instalação
9427 - Engenharia Informática e de Computadores	0800 - Universidade Técnica de Lisboa	0808 - Instituto Superior Técnico (Instalação
1455 - Engenharia Informática e de Computadores	3110 - Instituto Politécnico de Lisboa	3118 - Instituto Superior de Engenharia de L
1885 - Engenharia Informática e de Computadores (regime nocturno)	3110 - Instituto Politécnico de Lisboa	3118 - Instituto Superior de Engenharia de L
9121 - Engenharia Informática e de Computadores	3110 - Instituto Politécnico de Lisboa	3118 - Instituto Superior de Engenharia de L
9427 - Engenharia Informática e de Computadores	3110 - Instituto Politécnico de Lisboa	3118 - Instituto Superior de Engenharia de L
8441 - Segurança Informática em Redes de Computadores (regime pós-laboral)	3130 - Instituto Politécnico do Porto	3138 - Escola Superior de Tecnologia e Ges
9121 - Engenharia Informática e de Computadores	4306 - Instituto Superior de Estudos Interculturais e Transdisciplinares - Almada	

0.25483690021 seconds

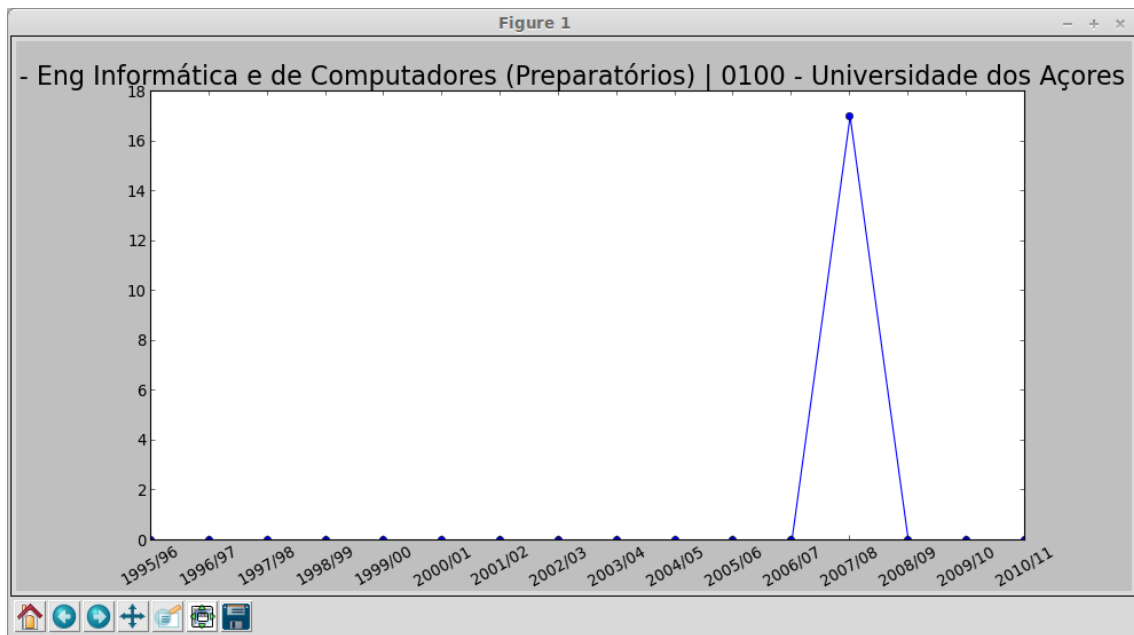
Todos os níveis de inscritos em cursos com as palavras “computadores” e ”informatica” em todos os anos.

A	B	C	D	E	F	G	H	I	J	K	L	M	N
D - Doutoramento	1995/96	0											
D - Doutoramento	1996/97	0											
D - Doutoramento	1997/98	17											
D - Doutoramento	1998/99	18											
D - Doutoramento	1999/00	20											
D - Doutoramento	2000/01	21											
D - Doutoramento	2001/02	22											
D - Doutoramento	2002/03	36											
D - Doutoramento	2003/04	48											
D - Doutoramento	2004/05	55											
D - Doutoramento	2005/06	65											
D - Doutoramento	2006/07	83											
D - Doutoramento	2007/08	71											
D - Doutoramento	2008/09	69											
D - Doutoramento	2009/10	61											
D - Doutoramento	2010/11	49											
D3 - Doutoramento - 3º ciclo	1995/96	0											
D3 - Doutoramento - 3º ciclo	1996/97	0											
D3 - Doutoramento - 3º ciclo	1997/98	0											
D3 - Doutoramento - 3º ciclo	1998/99	0											
D3 - Doutoramento - 3º ciclo	1999/00	0											
D3 - Doutoramento - 3º ciclo	2000/01	0											
D3 - Doutoramento - 3º ciclo	2001/02	0											
D3 - Doutoramento - 3º ciclo	2002/03	0											
D3 - Doutoramento - 3º ciclo	2003/04	0											
D3 - Doutoramento - 3º ciclo	2004/05	0											
D3 - Doutoramento - 3º ciclo	2005/06	0											
D3 - Doutoramento - 3º ciclo	2006/07	0											
D3 - Doutoramento - 3º ciclo	2007/08	20											
D3 - Doutoramento - 3º ciclo	2008/09	46											
D3 - Doutoramento - 3º ciclo	2009/10	64											
D3 - Doutoramento - 3º ciclo	2010/11	87											
L - Licenciatura	1995/96	1172											
L - Licenciatura	1996/97	1182											

0.3014578456 seconds

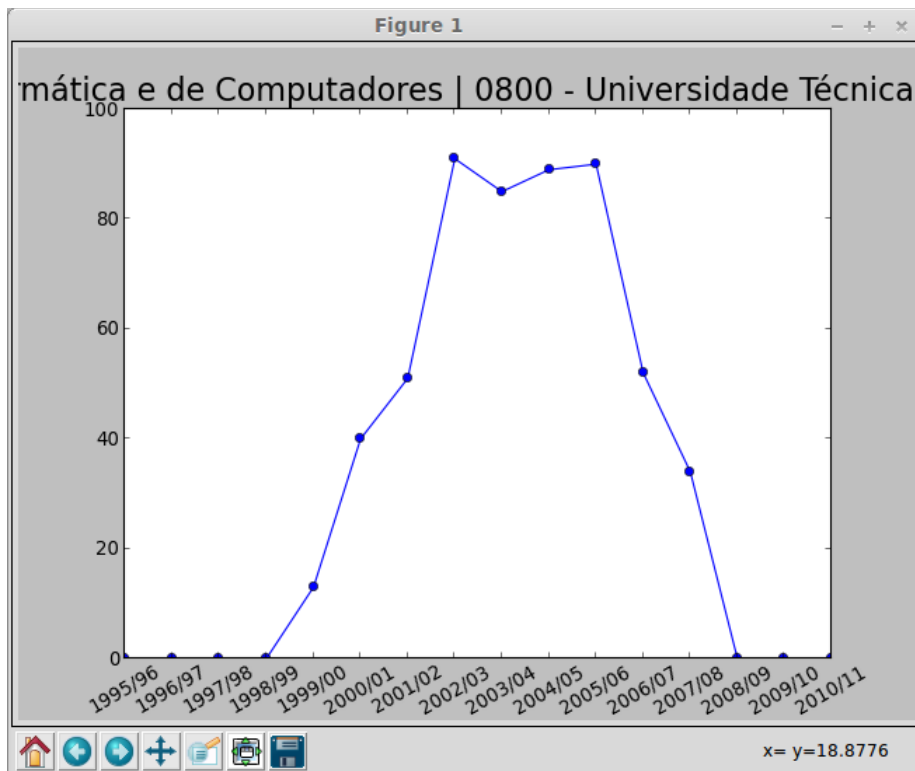
Gráficos

Inscritos no curso de Eng. Informática ao longo dos anos na Universidade dos Açores



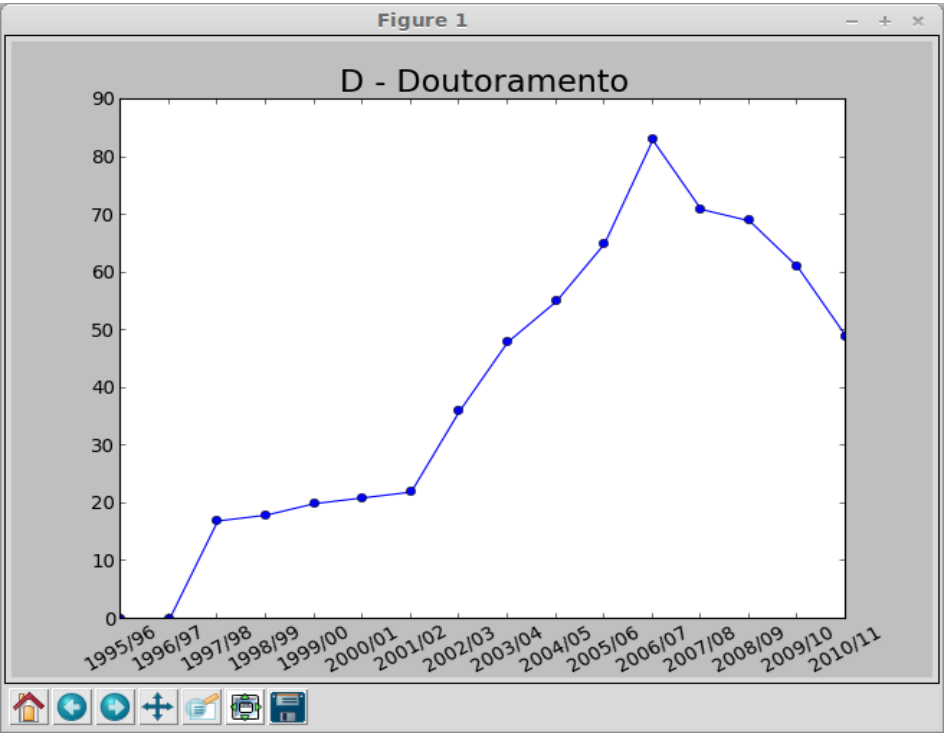
0.49226207733 seconds

Inscritos em todos os cursos com os nomes “Informática” e “Computadores” na Universidade Técnica de Lisboa.



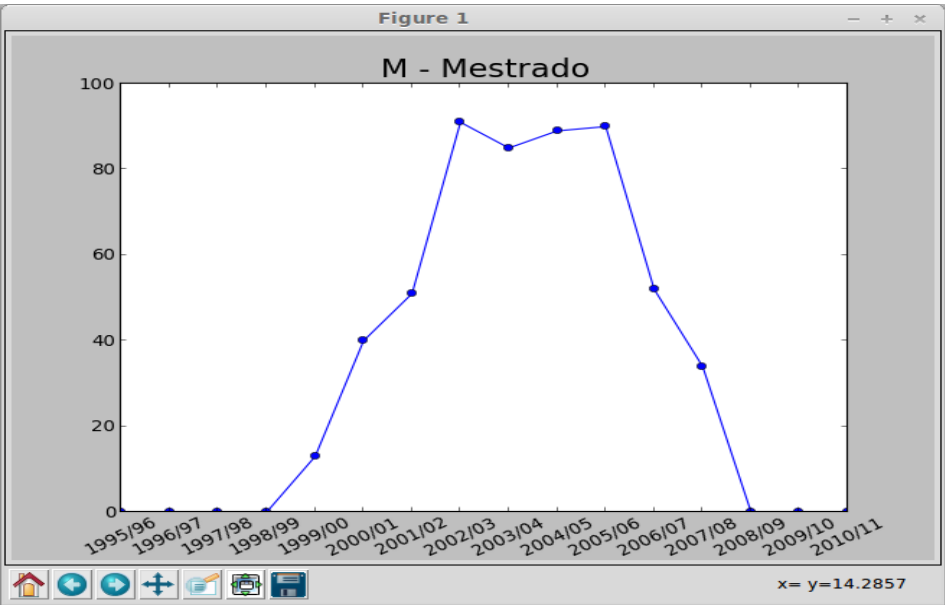
0.75286230021 seconds

Inscritos nos Doutoramentos com as palavras “Informatica” e “Computadores”



0.57141201201 seconds

Inscritos em todos os Mestrado com as palavras “Informatica” e “Computadores”



0.68740124780 seconds

Conclusões

Neste trabalho abordamos uma possível forma de importação de um ficheiro xls para uma base de dados de modo a debitar as *querys* definidas pelo enunciado do trabalho prático e respectivos gráficos. Após a análise e realização do mesmo concluímos que há várias formas de realizar os objectivos propostos, não definindo porem um método melhor que o outro pois não houve tempo para sair muito da linha traçada pelo professor no enunciado do trabalho pratico.

Os resultados foram alcançados de acordo com o pedido no enunciado, com precisão e num tempo considerado rápido (instantâneo) o que mostra a qualidade de um código bem estruturado, constatando-se uma resposta imediata mesmo com o uso de hardware mais antiquado, para isso contribuíram também o acréscimo de threads.

Bibliografia

<http://docs.python.org>

<http://www.yasinuludag.com>

<http://www.sqlalchemy.org>

<http://matplotlib.org>

<http://www.riverbankcomputing.co.uk>

<http://www.imdb.com/title/tt0063929/quotes>

<http://warp.povusers.org/programming/mvc.html>

<http://css.dzone.com/articles/wxpython-and-sqlalchemy>

Downey, Allen B. , Think Python - How to Think Like a Computer Scientist, O'Reilly, 2007

Summerfield, Mark, Rapid GUI Programming with Python and Qt, Prentice Hall, 2007

Anexos

Controller_Xls.py

```
#!/usr/bin/env python
#-*- coding: utf-8 -*-

from Model_Xls import Model_Xls
from xlrd import open_workbook, empty_cell

class Controller_Xls(Model_Xls):

    """
        Controller Xls Reads info of Major from Xls file
    """

    def open_Book(self):
        """
            Opens the book from xls file
        """
        wb = open_workbook( "../Inscritos_2010-2011 (formato Excel xls).xls", 'rb' )
        s = wb.sheet_by_index( 30 )
        return s

    def read_xls(self):
        """
            Read data from cells and return a list with information
        """
        res_list=[]
        for row_index in range(4, self.sheet.nrows):

            cur = self.sheet.cell_value(row_index,3)
            self.l_major_data.append(cur)
            self.l_major_data += self.get_Major_data(row_index)

            if self.check_comp(row_index):
                self.l_years_major = self.get_year_data(row_index)
                self.l_major_data.append(self.l_years_major)
                res_list.append( self.l_major_data)
                print self.l_major_data

            self.l_major_data=[]
            self.l_years_major=[]

        return res_list

    def get_Major_data(self, row_index):
        """
            Get the a list with information of major name, university, faculty and degree
        """
        for col_index in range(0,3):
            if self.sheet.cell(row_index,col_index).value != empty_cell.value:
                if col_index==0:
                    self.uni=self.sheet.cell_value(row_index,0)
                    self.fac=''
```

```

        pass

        if col_index==1:

            self.fac=self.sheet.cell_value(row_index,1)

            pass

        if col_index==2:

            self.niv=self.sheet.cell_value(row_index,2)

l=[self.uni, self.fac, self.niv]

return l

def check_comp(self, row_index):

    """

        Check if the Major contains the words "Computadores" and "Informática"

        retrun boolean if exist or not

    """

    return self.l_major_data[0].find(u'Computadores') > 0 and self.l_major_data[0].find(u'Informática') > 0

def get_year_data(self, row_index):

    """

        Get the number of students per year of each Major

        retrun a list with all years

    """

    lista_Years=[]

    for c in (self.Dic_Year.keys()):

        value =self.sheet.cell_value(row_index, c)

        if isinstance(value, float):

            Years=(self.Dic_Year[c], value)

            lista_Years.append(Years)

        else:

            zero = 0.0

            Years=(self.Dic_Year[c], zero)

            lista_Years.append(Years)

    return lista_Years

```


Controller_DB.py

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

from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy import create_engine
from sqlalchemy import Float, String, Integer, distinct, func
from sqlalchemy.orm import query, sessionmaker
from Model_DB import Major, Year, Model_DB
import os

"""
    Module Controlling database related issues
"""

class Controller_DB(Model_DB):
    """
        Class Controller_DB Being part of MVC architecture has the
        Methods to control functions related with Database
    """
    def create_db(self):
        """
            Call Module_DB to create the Database file with the tabels
        """
        os.system('python Model_DB.py')

    def insertBD_Major(self, name_University, name_School, name_Major, degree, l_Years):
        """
            Insert tuples at database
        """
        s = self.create_session()
        c = Major(name_Major = name_Major, name_School = name_School, degree = degree, name_University=
name_University)
        for a in l_Years:
            c.Year.extend([Year(Year_c=a[0], number_students=a[1])])
            s.add(c)
            s.commit()
        try:
            s.commit()
        except:
            print 'ERROR_Commit'

    def create_session(self):
        """
            Create a session with sqlalchemy to connect with database to
            query or insert data in Database
        """
        Session = sessionmaker(bind = self.engine)
        s = Session()
        return s

#####
```

```

#
#
#
#####

def query_majors_info(self):
    """
    Do query to database returning info about Major

    """
    s = self.create_session()
    l = s.query(Major).all()
    return l

def query_major(self):
    """
    Query database returning Major Name, university and degree

    """
    s = self.create_session()
    l = s.query(Major.name_Major, Major.name_University, Major.degree_Major).all()
    res = []
    for i in l:
        s = i[0] + ' | ' + i[1] + ' | ' + i[2]
        res.append(s)

    return res

def query_degree(self):
    """
    Query return a list of all degrees without repeting

    """
    s = self.create_session()
    l = s.query(distinct(Major.degree_Major)).all()
    res = []
    for i in l:
        res.append(i[0])

    return res

def query_major_funcionamento(self):
    """
    Years when the courses were working

    """
    s = self.create_session()
    q_c = self.query_majors_info()
    res=[]
    for c in q_c:
        major_info=[]
        major_info.append(c.name_Major.encode('utf-8'))
        major_info.append(c.name_University.encode('utf-8'))
        major_info.append(c.name_School.encode('utf-8'))
        major_info.append(c.degree_Major.encode('utf-8'))

        x = s.query(Year).filter(Year.id_Major == c.id).all()

```

```

        for a in x:
            if a.number_students > 0:
                major_info.append(a.Year)

        res.append(major_info)

    return res

def query_students_degrees(self, degree = None):
    """
        query Number of students per year on each Degree
    """
    s = self.create_session()

    cont = s.query(Major.degree_Major, Year.Year, func.sum(Year.number_students)).join(Year).filter(Major.id ==
Year.id_Major).group_by(Major.degree_Major, Year.Year).all()

    res=[]
    for i in cont:
        res.append((i[0].encode('utf-8'),i[1], i[2]))

    return res

def query_students_major(self):
    """
        query number of students per year per major
    """
    s = self.create_session()

    cont = s.query(Major.name_Major, Year.Year, func.sum(Year.number_students)).join(Year).filter(Major.id ==
Year.id_Major).group_by(Major.id, Year.Year).all()

    res=[]
    for i in cont:
        res.append((i[0].encode('utf-8'),i[1], i[2]))

    return res

def query_major_degrees(self):
    """
        Query number of majors per degree
    """
    s = self.create_session()

    cont = s.query(Major.degree_Major, Year.Year, func.count(Major.id)).join(Year).filter(Major.id ==
Year.id_Major).group_by(Year.Year, Major.name_Major).all()

    res=[]
    for i in cont:
        res.append((i[0].encode('utf-8'),i[1], i[2]))

    return res

def query_students_major_plot(self, id_Major):
    """
        query number of students per year at major with id_Major
    """
    s = self.create_session()

    cont = s.query(Major.name_Major, Year.Year, func.sum(Year.number_students)).join(Year).filter(Major.id ==
id_Major).group_by(Major.id, Year.Year).all()

    res=[]
    for i in cont:
        res.append((i[0].encode('utf-8'),i[1], i[2]))

    return res

```

Controller_MAIN.py

```
#!/usr/bin/env python
#-*- coding: utf-8 -*-

from Controller_Xls import Controller_Xls
from Controller_DB import Controller_DB
from Model_csv import Controller_stat
from Model_plot import Controller_Plot

import sys

from View import MainFrame

import wx

"""
    Control the main of the of application and launch the view
"""

class Controller_Main(object):
    """
        Class containing the main functions of app
    """

    def make_DB(self, event):
        """
            Create Database when the buttun is clicked
        """

        c_D = Controller_DB()
        c_D.create_db()

        c_xls = Controller_Xls()
        lista = c_xls.read_xls()

        for l in lista:
            c_D.insertBD_Major(name_University=l[1], name_School=l[2], name_Major=l[0], degree=l[3],
l_Years=l[4])

    def make_csv(self, event, cmbbox):
        """
            Receicves data from main window to make csv file
            and create it
        """

        C_D = Controller_DB()
        query = cmbbox.GetCurrentSelection()

        if query == 0:
            l = C_D.query_major_funcionamento()
            C_S = Controller_stat('q1.csv')
            C_S.write_to_csv(l)

        elif query == 1:
            l = C_D.query_students_degrees()
            C_S = Controller_stat('q2.csv')
            C_S.write_to_csv(l)

        elif query == 2:
            l = C_D.query_students_major()
            C_S = Controller_stat('q3.csv')
            C_S.write_to_csv(l)

        elif query == 3:
            l = C_D.query_major_degree()
            C_S = Controller_stat('q4.csv')
```

```

C_S.write_to_csv(1)

def plot_major(self, event, cmbx ):
    """
        Plot the info about the major selected on combobox
    """
    id_major = cmbx.GetCurrentSelection()
    name = cmbx.GetStringSelection()

    id_major = id_major + 1

    C_D = Controller_DB()
    lista = C_D.query_students_major_plot(id_major)

    x=[]
    y=[]
    for i in lista:
        x.append(i[1])
        y.append(i[2])

    C_p = Controller_Plot(name ,x, y)
    C_p.plot_data()
    pass

def plot_degree(self, event, cmbx ):
    """
        Plot the info about the degree selected on combobox
    """
    sel = cmbx.GetStringSelection()
    if sel != '':
        C_D = Controller_DB()
        lista = C_D.query_students_degrees()

        x=[]
        y=[]

        for i in lista:
            if i[0].find(sel.encode('utf-8')) >= 0:
                x.append(i[1])
                y.append(i[2])

        C_p = Controller_Plot(sel,x, y)
        C_p.plot_data()
        pass

if __name__ == "__main__":
    class AppView(wx.App):
        def OnInit(self):

            wx.InitAllImageHandlers()

            frame_1 = MainFrame(None, -1, "")

```

```

        Cm = Controller_Main()

        frame_1.btnMakeDB.Bind(wx.EVT_BUTTON, Cm.make_DB, frame_1.btnMakeDB)

        frame_1.btnCSV.Bind(wx.EVT_BUTTON, lambda event, cmb = frame_1.cmbCSV: Cm.make_csv(event, cmb),
frame_1.btnCSV)

        frame_1.btnPlotMajor.Bind(wx.EVT_BUTTON, lambda event, cmb = frame_1.cmbMajor:
Cm.plot_major(event, cmb), frame_1.btnPlotMajor)

        frame_1.btnPlotDegree.Bind(wx.EVT_BUTTON, lambda event, cmb = frame_1.cmbDegree:
Cm.plot_degree(event, cmb), frame_1.btnPlotDegree)

        self.SetTopWindow(frame_1)

        frame_1.Show()

        return 1

# end of class AppView

lp1213_6055_9989 = AppView(0)
lp1213_6055_9989.MainLoop()

```

Model_cvs.py

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

import csv

"""
    Deals with csv creation
"""

class Model_stat(object):

    """
        Stores information to make csv files
    """

    def __init__(self, file_name):

        """
            Constructor of class Model_stat
        """

        self.file_name = file_name
        self.csvfile = open(self.file_name, 'wb')
        print 'opening file...'

class Controller_stat(Model_stat):

    """
        Class controller of csv
    """

    def write_to_csv(self, l):

        """
            wirte statistics to Csv File
        """

        writer = csv.writer(self.csvfile, delimiter=',')

        for i in l:
            writer.writerow( i )
            print i

        self.csvfile.close()
```

Model_DB.py

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

from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy import create_engine
from sqlalchemy import Column, Float, String, ForeignKey, Integer
from sqlalchemy.orm import relationship, backref

class Model_DB(object):

    """

    Class Model_DB

    """

    def __init__(self, name = 'sqlite:///basedados.db'):

        """

        Constructor of class Model_DB when an instance of this class is created

        create a Base and an engine to connect to Database

        """

        self.engine = create_engine(name, echo = True)
        self.Base = declarative_base(bind = self.engine)

m = Model_DB()
Base = m.Base

class Major(Base):

    """

    Class of table Major to sqlalchemy with info about

    Majors

    """

    __tablename__ = 'Major'

    id = Column(Integer, primary_key = True)
    name_Major = Column('name_Major', String)
    name_School = Column('name_School', String)
    name_University = Column('name_University', String)
    degree_Major = Column('degree_Major', String)

    def __init__(self, name_Major, name_School, degree, name_University):

        """

        Constructor of class tabel Major

        """

        self.name_Major = name_Major
        self.name_School = name_School
        self.name_University = name_University
        self.degree_Major = degree

class Year(Base):

    """

    Class of table Year to sqlalchemy containing info about

    each major of each year

    """
```



```

__tablename__ = 'Year'

id = Column(Integer, primary_key=True)

Year = Column('Year', String)

number_students = Column('number_students', Float)

major = relationship("Major", backref=backref("Year"))

id_Major = Column(Integer, ForeignKey('Major.id'))


def __init__(self, number_students, Year_c ):
    """
        Constructor of class table Year
    """
    self.Year = Year_c
    self.number_students = number_students


if __name__ == '__main__':
    m.Base.metadata.create_all()

```

Model_plot.py

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

import pylab
import matplotlib.ticker

"""

    Module To plot

    include Model and Controller to plot

"""

class Model_Plot():

    """

        Model Plot has data to create the graph

    """

    def __init__(self, title, x_list, y_list):

        """

            Constructor of class Model Plot stores the x and y data and title

        """

        self.x_list = x_list

        self.y_list = y_list

        self.title = title

class Controller_Plot(Model_Plot):

    """

        Class to create graph

    """

    def plot_data(self):

        """

            Plot the data stored on model plot

        """

        pylab.xticks(range(len(self.x_list)), self.x_list, rotation=30)

        pylab.plot(self.y_list, '-o')

        pylab.title(self.title, fontsize=20)

        pylab.xlabel('Years', fontsize=20)

        pylab.show()
```

Model_Xls.py

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

"""
    Model Xls stores data related to xls file reading
"""

class Model_Xls(object):

    """
        ModelXls class that stores information about xls file reading
    """

    def __init__(self, file_name = "./Inscritos_2010-2011 (formato Excel xls).xls"):

        """
            constructor of class model xls initializes the dictionary with columns index of year
            and data about xls file
        """

        self.COL_YEAR_95_96 = 7
        self.COL_YEAR_96_97 = 10
        self.COL_YEAR_97_98 = 13
        self.COL_YEAR_98_99 = 16
        self.COL_YEAR_99_00 = 19
        self.COL_YEAR_00_01 = 22
        self.COL_YEAR_01_02 = 25
        self.COL_YEAR_02_03 = 28
        self.COL_YEAR_03_04 = 31
        self.COL_YEAR_04_05 = 34
        self.COL_YEAR_05_06 = 37
        self.COL_YEAR_06_07 = 40
        self.COL_YEAR_07_08 = 43
        self.COL_YEAR_08_09 = 47
        self.COL_YEAR_09_10 = 50
        self.COL_YEAR_10_11 = 53

        self.Dic_Year={self.COL_YEAR_95_96:'1995/96',
                        self.COL_YEAR_96_97:'1996/97',
                        self.COL_YEAR_97_98:'1997/98',
                        self.COL_YEAR_98_99:'1998/99',
                        self.COL_YEAR_99_00:'1999/00',
                        self.COL_YEAR_00_01:'2000/01',
                        self.COL_YEAR_01_02:'2001/02',
                        self.COL_YEAR_02_03:'2002/03',
                        self.COL_YEAR_03_04:'2003/04',
                        self.COL_YEAR_04_05:'2004/05',
                        self.COL_YEAR_05_06:'2005/06',
                        self.COL_YEAR_06_07:'2006/07',
                        self.COL_YEAR_07_08:'2007/08',
                        self.COL_YEAR_08_09:'2008/09',
                        self.COL_YEAR_09_10:'2009/10',
                        self.COL_YEAR_10_11:'2010/11'

                        }

        self.file_name = file_name

        self.sheet = 30

        self.sheet = self.open_Book()

        self.l_major_data = []

        self.l_years_major = []

        self.uni, self.fac, self.niv = '', '', ''
```

View.py

```
#!/usr/bin/env python

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

# generated by wxGlade 0.6.5 on Thu Dec 6 23:30:41 2012

from Controller_DB import Controller_DB

import wx

# begin wxGlade: extracode

# end wxGlade

"""
Module View generated automatically by wx glade
with the GUI
"""

class MainFrame(wx.Frame):

    def __init__(self, *args, **kwargs):
        # begin wxGlade: MainFrame.__init__

        cdb = Controller_DB()

        self.lst_cmb_CSV = ['Anos letivos em que funcionaram majors que incluem o termo computadores e informática',
                            'quantidade de estudante que nestes se inscreveram ao longo dos Years',
                            'quantidade de majors por nível de formação ao longo dos Years',
                            'quantidade de students por nível de formação ao longo dos Years']

        try:

            self.lst_cmb_Major = cdb.query_major()

            self.lst_cmb_Degree = cdb.query_degree()

        except:

            self.lst_cmb_Major = []

            self.lst_cmb_Degree = []

            print 'No such table!'

        kwargs["style"] = wx.DEFAULT_FRAME_STYLE

        wx.Frame.__init__(self, *args, **kwargs)

        self.btnMakeDB = wx.Button(self, -1, "Importa dados para Basedados")

        self.CSV = wx.StaticText(self, -1, "CSV")

        self.cmbCSV = wx.ComboBox(self, -1, choices=self.lst_cmb_CSV, style=wx.CB_READONLY)

        self.btnCSV = wx.Button(self, -1, "Executar")

        self.Plot = wx.StaticText(self, -1, "Plot")

        self.cmbMajor = wx.ComboBox(self, -1, choices=self.lst_cmb_Major, style=wx.CB_READONLY)

        self.btnPlotMajor = wx.Button(self, -1, "Executar")

        self.cmbDegree = wx.ComboBox(self, -1, choices=self.lst_cmb_Degree, style=wx.CB_READONLY)

        self.btnPlotDegree = wx.Button(self, -1, "Executar")

        self.__set_properties()

        self.__do_layout()

        # end wxGlade

    def __set_properties(self):

        # begin wxGlade: MainFrame.__set_properties

        self.SetTitle("lp1213_6055_9989")

        # end wxGlade
```

```

def __do_layout(self):
    # begin wxGlade: MainFrame.__do_layout

    sizer_1 = wx.BoxSizer(wx.VERTICAL)

    grid_sizer_1 = wx.FlexGridSizer(7, 2, 10, 10)

    grid_sizer_1.Add(self.btnMakeDB, 0, 0, 0)

    grid_sizer_1.Add((20, 20), 0, 0, 0)

    grid_sizer_1.Add(self.CSV, 0, 0, 0)

    grid_sizer_1.Add((20, 20), 0, 0, 0)

    grid_sizer_1.Add(self.cmbCSV, 0, 0, 0)

    grid_sizer_1.Add(self.btnCSV, 0, 0, 0)

    grid_sizer_1.Add(self.Plot, 0, 0, 0)

    grid_sizer_1.Add((20, 20), 0, 0, 0)

    grid_sizer_1.Add(self.cmbMajor, 0, 0, 0)

    grid_sizer_1.Add(self.btnPlotMajor, 0, 0, 0)

    grid_sizer_1.Add(self.cmbDegree, 0, 0, 0)

    grid_sizer_1.Add(self.btnPlotDegree, 0, 0, 0)

    sizer_1.Add(grid_sizer_1, 1, wx.ALL | wx.EXPAND, 15)

    self.SetSizer(sizer_1)

    sizer_1.Fit(self)

    self.Layout()

    # end wxGlade

# end of class MainFrame

# end of class MyMenuBar

```