Instituto Politécnico de Beja

Análise de Colocações no Ensino Superior

Eduardo de Sousa Fernandes
nº12927

Damien Fialho
nº11243

2 de Dezembro de 2013

I. Introdução

Este trabalho consiste na realização de uma aplicação que processe e apresente estatísticas de candidaturas de alunos no ensino superior do ano lectivo 2013-2014 na primeira fase de candidatura. Esta aplicação deverá conter uma interface gráfica que permita a escolher qual estatística a ser apresentada para análise.

Os objetivos pretendidos para este trabalho são:

- **1ª Fase**
  - Leitura de um ficheiro em formato **XLS**
  - Criação e preenchimento de uma base de dados
- **2ª Fase**
  - Leitura da base de dados criada na primeira fase
  - Seleção dos dados
  - Cálculo das estatísticas
  - Apresentação de gráficos das estatísticas
- **3ªFase**
  - Criação de um ficheiro **CSV** com os resultados das estatísticas
  - Criação de uma interface gráfica

A **primeira** parte do trabalho consiste na leitura de um ficheiro em formato XLS e na criação de uma base de dados com os dados nesse ficheiro. Para a elaboração desta primeira parte, resolvemos utilizar o sqlite3 que serve para a criação e manutenção de uma base de dados.

Na **segunda** parte acedemos aos dados da base de dados a fim de calcular estatísticas de candidaturas de alunos e outros, e com os resultados criar gráficos. As estatísticas são calculadas a partir dos dados que foram inseridos na base de dados na primeira fase e os gráficos são apresentados através do Matplot.

Por **último** realizamos a interface gráfica que irá apresentar os gráficos ao utilizador a fim de permitir a análise das estatísticas. A interface gráfica é apresentada utilizando o pacote wxPython.

II.Teoria

Começou-se por criar o código num ficheiro Python (**trabalho.py**), mais tarde, para prevenir a repetição de código, foram criadas funções onde essas funções foram colocadas num ficheiro designado por **functions.py** para que possam ser acedidas globalmente em todo o programa.

Assim, a aplicação foi separada em vários ficheiros de Python para permitir uma maior acessibilidade ao longo do programa.

Esses ficheiros de Python são:

- **trabalho.py**
- **classes.py**
- **functions.py**
- **mainmenu_design.py**

Para a realização da aplicação também foi necessário a utilização de bibliotecas, que foram as seguintes :

- **csv** (**C**omma **S**eperated **V**alues) , é uma biblioteca utilizada para a leitura e escrita de ficheiros Excel ou Base de Dados.
- **xlrd**, é uma biblioteca que permite a leitura de ficheiros Excel, também permite formatar a informação recolhida desses mesmos.
- **sqlite3**, é uma biblioteca que permite a criação e a leitura de base de dados.
- **wx**, é uma biblioteca que permite a criação de uma interface grafica (**GUI**).
- **matplotlib.pyplot e numpy**, são bibliotecas que permitem a criação de graficos.

Assim, com essas bibliotecas e separação do programa em varios ficheiros, foi-nos permitido a criação facil desse mesmo.

III. Parte Experimental

3.1 Realização Experimental

A linguagem de programação Python é de nível elevado, tem objetivos gerais e enfatiza a legibilidade do código. Suporta a programação multiparadigmática: orientação por objetos; estilo imperativo e funcional onde apresenta um sistema de tipo dinâmico completo e gestão de memória automática.

As linguagens de programação dinâmicas podem incluir :

- a modificação do sistema de tipos;
- a extensão de objetos e definições;
- a extensão dos programas por adição de novo código;

Tudo isto em tempo de execução.

O ambiente de desenvolvimento utilizado foi **Geany**, que é um editor de texto que utiliza o pacote **GTK2** e serve para desenvolvimento de aplicações em diversas linguagens de programação.

Este trabalho foi elaborado numa máquina virtual fornecida pelo docente da disciplina de **Linguagens de Programação** que contém o sistema operativo **Debian**. Este sistema operativo é constituído por uma quantidade enorme de pacotes que contêm softwares pré-compilados e distribuídos em formatos que possibilitam uma fácil instalação.

3.2 Sistema Experimental

1ª fase

- Leitura do ficheiro Excel

O programa começa por abrir um ficheiro em formato Excel, em seguida seleciona a folha onde se encontram os dados que vão ser utilizados para a realização deste trabalho. Neste caso os dados são apresentados a partir da 4ª linha, sendo por isso implementado um método que começa a ler os dados a partir dessa mesma linha. Dentro deste método temos um outro método que lê e armazena os dados para dentro de um array.

- Criação da base de dados

Após a leitura do ficheiro Excel o programa está pronto para criar a base de dados utilizando o sqlite3. Esta base de dados é composta por uma tabela onde serão armazenados os dados que foram lidos do ficheiro em formato Excel.

2ª fase

- **Classes**

As classes dos distritos e das instituições servem para armazenar dados e processar os mesmos.

- **Functions - get_data / get_dist_data**

No ficheiro das funções existe uma função get_data que utiliza o sqlite3 para ler de forma seletiva os dados armazenados na base de dados, guarda-os num array e devolve esse mesmo array.

Existem também os métodos get_dist_data que carrega os dados do ficheiro em formato Excel referente aos distritos para dentro de um array e devolve-o.

- **Functions - implementDistrict / isInList**

Funções implementDistrict e isInList, a primeira atribui distritos às universidades ao verificar através do segundo se existe algum distrito no nome da universidade existente na seleção de distritos criada a partir do ficheiro Excel. Caso não seja encontrado o distrito é atribuido o nome “Unkown”.

- **Functions - get_inst / get_dist / get_dist_per / get_total_per**

De seguida são apresentadas as funções que selecionam os dados que serão utilizados, que neste caso são os distritos ou instituições, efetuando posteriormente os cálculos das estatísticas.

- **Functions - create_graph**

Agora que as estatísticas foram calculadas nas funções anteriores, o programa utiliza o pacote Matplot que vai fazer a leitura das estatísticas e em seguida apresentá-las em forma de gráfico.

3ª Fase

- Class Menu - __init__

O método __init__ é o construtor da class Menu que vai criar a interface gráfica que vamos utilizar. Neste caso vai existir uma combobox, que apresenta as estatísticas existentes para análise, e um botão, que seleciona a estatística escolhida pelo utilizador para apresentar o gráfico da mesma.

- Class Menu - button1Click

Para apresentar os gráficos a partir da nossa interface temos de verificar qual a estatística selecionada na combobox e chamar o gráfico da função desejada.

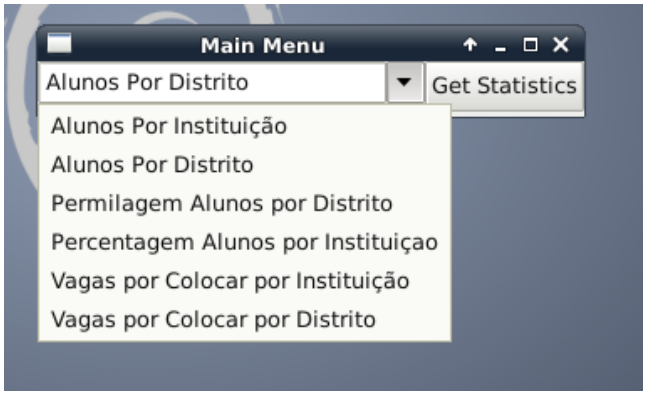
3.3 Resultados Experimentais

Os resultados obtidos na elaboração deste trabalho foram sempre testados em duas máquinas máquinas diferentes. Neste caso foram utilizadas as máquinas dos programadores.

Como se pode verificar na imagem seguinte foi utilizada a função print para verificar a estatística dos alunos colocados por distrito. Neste caso escolhemos apresentar esta estatística apenas por uma questão de espaço, uma vez que as outras estatísticas contêm muitos mais dados.

```
[Açores->415.0, Algarve->827.0, Aveiro->1722.0, Beira Interior->1004.0, Coimbra-
>4431.0, Évora->785.0, Lisboa->12200.0, Minho->2331.0, Porto->6791.0, Trás-os-Mo
ntes e Alto Douro->1057.0, Madeira->453.0, Beja->159.0, Cávado e do Ave->406.0,
Bragança->417.0, Castelo Branco->376.0, Guarda->183.0, Leiria->1150.0, Portalegr
e->166.0, Santarém->397.0, Setúbal->502.0, Viana do Castelo->502.0, Viseu->528.0
, Tomar->103.0, Estoril->436.0, Unknown->74.0]
```

Nesta situação foi testado o aspeto da interface gráfica do programa que permite escolher qual a estatística que queremos analisar graficamente.



Conclusão

Conseguimos concluir as tarefas propostas no enunciado to trabalho de forma simples e eficaz. Tentámos organizar o código da melhor forma possível separando-o em diversas classes e funções utilizadas ao longo do trabalho.

Na realização deste trabalho tivemos alguns problemas iniciais com o encoding mas que acabaram por ser resolvidos sem grandes demoras.

Com este trabalho ficámos a conhecer melhor a linguagem de programação Python que, como todas as linguagens de programação, tem os seus aspetos positivos e negativos. Um dos aspectos que achámos mais favorável a esta linguagem foi a sua simplicidade em relação às linhas de código.

É importante ter-se um conhecimento mais geral dos tipos de linguagens de programação que existem e este trabalho ajudou-nos a expandir esse nosso conhecimento.

Bibliografia

Para a elaboração deste trabalho foram consultados os ficheiros em formato PDF fornecidos pelo docente na página moodle da disciplina de Linguagens de Programação.

Anexos

```
# -*- coding: utf-8 -*-
# autor: 12927 11243
# data: 23 de Outubro de 2013

import wx
import xlrd
import sqlite3

from classes import *
from functions import *
from xlrd import open_workbook
from mainmenu_design import *

"""
#############
## STARTUP ##
#################################################################################
## Setups the files and variables. It also opens the Excel files to be used.
"""

ficheiro = open_workbook('cna131fresultados.xls')
District_Database = open_workbook('district-database.xls')

District_Data = []
RawData = []
FinalData = []
Count = 0
CountInsert = 0

"""
###################
## EXCEL READING ##
#################################################################
## Reads the Excel and prepares it to be saved on a database.
"""

for s in ficheiro.sheets():
        for row in range(s.nrows):
                if Count > 2:
                        for col in range(s.ncols):

                                if CountInsert > (s.ncols - 1):
                                        FinalData.append(RawData)
                                        RawData = []
                                        CountInsert = 0

                                RawData.append(s.cell_value(row,col))
                                CountInsert += 1

                Count+=1
"""
##############
## DATABASE ##
#############################################################
## Creates and connects to the database for saving the data.
"""
Connection = sqlite3.connect('cna131fresultados.db')
Command = Connection.cursor()

# Criaçao da TABELA
Command.execute('DROP TABLE IF EXISTS cna131')
Command.execute('CREATE TABLE cna131 (COD_INST text, COD_CUR text, NOME_INST text, NOME_
Connection.commit()

for indx in range(len(FinalData) - 1):
        Command.execute('INSERT INTO cna131 VALUES(?,?,?,?,?,?,?,?,?)',FinalData[indx])

Connection.commit()
```

```
"""
####################
## Save Statistics ##
################################################
## Saves the statistics on a excel format file.
"""

save_statistics()

"""
###############
## MAIN MENU ##
##############################
## Starts up the MainMenu GUI.
"""

app = wx.PySimpleApp(0)
wx.InitAllImageHandlers()
MainMenu = Menu(None, -1, "")
app.SetTopWindow(MainMenu)
MainMenu.Show()
app.MainLoop()
```

```
# -*- coding: utf-8 -*-
# autor: 12927 11243
# data: 23 de Outubro de 2013

import xlrd
import sqlite3
import matplotlib.pyplot as plt
import numpy as np
import csv

from classes import *
from xlrd import open_workbook

"""
#################
# The Functions #
#######################################
## Global file with all the functions.
#######################################
"""

Connection = sqlite3.connect('cna131fresultados.db')
Command = Connection.cursor()

def get_data(data, where, offset, filt):

        """
        Reads the data from the database
        Param :
                data - array
                where - string
                offset - string
                filt - string
        Returns :
                MySqlData - list
        """

        MySqlData = []
        TEMPO = []

        if type(data) != str or type(Command) != sqlite3.Cursor:
                return None

        Command.execute('SELECT * FROM cna131')
        TempData = Command.fetchall()
```

```
        for tmp in TempData:

                OffCmnd = tmp[offset].encode('utf-8')
                Command.execute("SELECT " + data + " from cna131 WHERE " + where + "='" + OffCmnd + "'")

                if filt != '' :
                        OffCmnd = OffCmnd.split(filt)[0]

                if not OffCmnd in TEMPO:
                        MySqlData.append(Instituicao(OffCmnd,Command.fetchall()))
                        TEMPO.append(OffCmnd)

        return MySqlData

def get_dist_data():
        """
        Read the XLS file containing districts
        Returns :
                List - District_Data
        """

        District_Data = []
        District_Database = open_workbook('district-database.xls')

        for s in District_Database.sheets():
                for row in range(s.nrows):
                        for col in range(s.ncols):
                                District_Data.append(District(s.cell_value(row,col).encode('utf-8')))


        District_Data.append(District('Unknown'))
        return District_Data

def implementDistrict(data,DistData):

        """
        Set the districts to the insts
        Param :
                data - list
                DistData - list
        Returns :
                TEMPDATA - list
        """

        TEMPDATA = data

        for tmp in TEMPDATA:

                IN = isInList(tmp.ID,DistData)
                tmp.Dist = District(DistData[IN].ID,IN)

        return TEMPDATA

def isInList(stri,data):
        """
        Check if its in the list. If its not, return the last value on the "data" (AKA Unkown)
        Params :
                stri - string
                data - list
        Returns :
                indx - int
        """
        for indx in range(len(data)) :
                if stri.find(data[indx].ID) != -1:
                        return indx

        return indx
```

```
def get_inst(data):
        """
        Get the Institution
        Param :
                data - string
        Return :
                ALUN_DIST - list
        """

        #0 = ID, 1 = List
        Inst_Data = get_data(data,'COD_INST',0,'')

        ALUN_DIST = []

        for tmp in Inst_Data:
                Count = 0

                for Inst in tmp.Data:
                        Count += Inst[0]

                ALUN_DIST.append(Instituicao(tmp.ID,Count))

        return ALUN_DIST

def get_dist(data):

        """
        Get the District
        Param :
                data - string
        Return :
                Districts - list
        """

        Districts = get_dist_data()
        #0 = ID, 1 = List
        Data = implementDistrict(get_data(data,'NOME_INST',2,''),Districts)

        for temp in Data:
                Count = 0

                for inst in temp.Data:
                        Count += inst[0]

                Indx = temp.Dist.INDEX
                Districts[Indx].Count += Count


        return Districts

# Permilagem on dist
def get_dist_per(data):
        """
        Get the District
        Param :
                data - string
        Return :
                Dt - list
        """

        Dt = get_dist(data)

        for tmp in Dt:
                tmp.Count /= 1000

        return Dt
```

```
def get_total_perc():
        """
        Percent on Inst
        Return :
                Alunos - list
        """

        Alunos = get_inst('COLOC')
        TOTAL = 0.0

        for tmp in Alunos:
                TOTAL += tmp.Data

        for tmp in Alunos:
                tmp.Data /= TOTAL
                tmp.Data *= 100
                tmp.Data = round(tmp.Data,4)

        return Alunos

# Create the graphics
def create_graph(data, title, y_title,x_title,isDist):
        """
        Create the graphics
        Params :
                data - list
                title - string
                y_title - string
                x_title - string
                isDist - boolean
        """

        Values = []
        IDS = []

        fig = plt.figure()
        ax = plt.subplot(111)
        width = 20

        for tmp in data:
                if isDist :
                        Values.append(tmp.Count)
                        IDS.append(tmp.ID.decode('utf-8'))
                else:
                        Values.append(tmp.Data)
                        IDS.append(int(tmp.ID))

        Size = np.arange(len(IDS)) * width
        ax.bar(Size, Values, width = width)
        ax.set_xticks(Size + width/2)
        ax.set_xticklabels(IDS, rotation=90)
        ax.set_xlabel(x_title)
        ax.set_ylabel(y_title)
        ax.set_title(title)

        plt.grid(True)
        plt.show()
```

```
def save_statistics():
        """
        Save the statistics on a csv file
        """

        with open('statistics.csv','wb')as csvfile:
                writer = csv.writer(csvfile)

                # 1º Estatistica
                writer.writerow("==== Alunos Por Instituição ====")

                for t in get_inst('COLOC'):
                        writer.writerow(str(t))

                # 2º Estatistica
                writer.writerow("==== Alunos Por Distrito ====")

                for t in get_dist('COLOC'):
                        writer.writerow(str(t))

                # 3º Estatistica
                writer.writerow("==== Permilagem Alunos por Distrito ====")

                for t in get_dist_per('COLOC'):
                        writer.writerow(str(t))

                # 4º Estatistica
                writer.writerow("==== Percentagem Alunos por Instituiçao ====")

                for t in get_total_perc():
                        writer.writerow(str(t))

                # 5º Estatistica
                writer.writerow("==== Vagas por Colocar por Instituição ====")

                for t in get_inst('VAGA_SOBR'):
                        writer.writerow(str(t))

                # 6º Estatistica
                writer.writerow("==== Vagas por Colocar por Distrito ====")

                for t in get_dist('VAGA_SOBR'):
                        writer.writerow(str(t))
```

```
# -*- coding: utf-8 -*-
# autor: 12927 11243
# data: 23 de Outubro de 2013

class District(object):

        """
        ######################
        # The District Class #
        ######################
        ## Stores the districts
        ############################################################
        """

        def __init__(self, ID=None, INDEX=0):
                self.ID = ID
                self.Count = 0.0
                self.INDEX = INDEX

        def __str__(self):
                return str(self.ID) + "->" + str(self.Count)

        def __repr__(self):
                return str(self.ID) + "->" + str(self.Count)

class Instituicao(object):

        """
        ######################
        # The Instituicao Class #
        ######################
        ## Stores the Instituicao
        ############################################################
        """

        def __init__(self, ID=None, Data=None):
                self.ID = ID
                self.Data = Data
                self.Dist = None

        def __str__(self):
                return str(self.ID) + "->" + str(self.Data)

        def __repr__(self):
                return str(self.ID) + "->" + str(self.Data)
```

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# generated by wxGlade 0.6.5 on Sat Nov 30 15:57:01 2013

import wx
from functions import *

class Menu(wx.Frame):

        """
        ####################
        #The Main Menu Class#
        ####################
        ## Creates a GUI
        ## Param : -wx.Frame = Frame to attach the mainmenu to
        ######################################################
        """


        def __init__(self, *args, **kwds):

                """
                Starts up the menu (setup)
                """

                kwds["style"] = wx.DEFAULT_FRAME_STYLE
                wx.Frame.__init__(self, *args, **kwds)

                self.combo1 = wx.ComboBox(self, -1, choices=[u"Alunos Por Instituição", "Alunos Por Distrito", "Permil
Instituiçao", u"Vagas por Colocar por Instituição", "Vagas por Colocar por Distrito"], style=wx.CB_READONLY)
                self.combo1.SetToolTip(wx.ToolTip("Select a Statistics"))

                self.button1 = wx.Button(self, -1, "Get Statistics")
                self.button1.Bind(wx.EVT_BUTTON, self.button1Click, self.button1)

                self.__set_properties()
                self.__do_layout()
        # end wxGlade

        def button1Click(self, event):

                """ On button click, do this """

                Selected = self.combo1.GetSelection()
                Value = self.combo1.GetValue()

                if Selected == 0 :
                        create_graph(get_inst('COLOC'),'Grafico - ' + Value,'Alunos','ID',False)
                elif Selected == 1 :
                        create_graph(get_dist('COLOC'),'Grafico - ' + Value,'Alunos','Distritos',True)
                elif Selected == 2 :
                        create_graph(get_dist_per('COLOC'),'Grafico - ' + Value,'Alunos (Permilagem)','Distritos',True
                elif Selected == 3 :
                        create_graph(get_total_perc(),'Grafico - ' + Value,'Alunos (%)','ID',False)
                elif Selected == 4 :
                        create_graph(get_inst('VAGA_SOBR'),'Grafico - ' + Value,'Vagas','ID',False)
                else :
                        create_graph(get_dist('VAGA_SOBR'),'Grafico - ' + Value,'Vagas','ID',True)

        def __set_properties(self):

                """
                Set the Properties
                """

                self.SetTitle("Main Menu")
                self.combo1.SetSelection(0)


def __do_layout(self):

        """
        Set the Layout
        """

        Sizer = wx.BoxSizer(wx.HORIZONTAL)
        Sizer.Add(self.combo1, 0, 0, 0)
        Sizer.Add(self.button1, 0, 0, 0)
        self.SetSizer(Sizer)
        Sizer.Fit(self)
        self.Layout()
```