IPBeja Escola Superior de Tecnologia e Gestão

Escola Superior de Tecnologia e Gestão
Engenharia Informática

Linguagens de Programação
Python

Análise de Colocações no Ensino Superior

Docente: Jasnau Caeiro
Discentes: Damien Fialho, 11243
Eduardo Fernandes, 12927

Índice

Introdução

O trabalho destina-se a realizar uma aplicação com interface gráfica para o utilizador que processe e apresente estatísticas de candidaturas de alunos em cursos do superior na sua primeira fase.

Estas estatísticas são:

- o número de alunos colocados por instituição;
- o número de alunos colocados por distrito;
- a permilagem de alunos colocados por distrito;
- a percentagem de alunos colocados por instituição em relação a todos os alunos colocados;
- o número de vagas por colocar por instituição;
- o número de vagas por colocar por distrito.

Para a realização deste trabalho, tivemos de tomar decisões importantes acerca dos métodos a serem utilizados.

A **primeira** parte do trabalho implica a leitura de um ficheiro em formato Excel e em seguida guardar os dados em uma base de dados. 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**.

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.

```
#############
## STARTUP ##
#############

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

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

###################
## EXCEL READING ##
###################

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
```

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.

```
###############
## DATABASE ##
###############

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_CURS
float)')
Connection.commit()

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

Connection.commit()
```

2ª Fase

Nesta fase optámos por criar dois ficheiros Python e um ficheiro Excel, a fim de facilitar a leitura do código do nosso programa e torná-lo mais eficiente. Um dos ficheiros Python contém a criação de duas classes, uma para os distritos e outra para as instituições, enquanto o outro ficheiro é constituído por todas as funções utilizadas a partir desta fase do trabalho. Por último, o ficheiro Excel contém a seleção de distritos utilizados para este trabalho.

Classes

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

```
class District(object):
        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):
        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) + "->" + str(self.Dist)

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

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.

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

        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
```

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.

```
def get_dist_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
```

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”.

```
def implementDistrict(data,DistData):
        TEMPDATA = data

        for tmp in TEMPDATA:

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

        return TEMPDATA

def isInList(stri,data):

        for indx in range(len(data)) :
                if stri.find(data[indx].ID) != -1:
                        return indx

        return indx
```

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.

```
def get_inst(data):

        #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):

        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

def get_dist_per(data):
        Dt = get_dist(data)

        for tmp in Dt:
                tmp.Count /= 1000

        return Dt

def get_total_perc():
        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
```

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.

```
def create_graph(data, title, y_title,x_title,isDist):

        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()
```

3ª Fase

Esta é a última fase do trabalho onde criamos uma interface gráfica que dá a possibilidade ao utilizador de escolher qual a estatística a ser apresentada gráficamente pelo programa. Para a criação desta interface optámos por utilizar o pacote **wxPython**, escolhemos este pacote pois consideramos que este tem uma interface mais "user friendly".

O código utilizado nesta última fase foi separado do ficheiro Python principal com a finalidade de tornar a leitura do código mais simples e organizada.

Class Menu - __init__

O método seguinte é o método principal 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.

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

                self.combo1 = wx.ComboBox(self, -1, choices=[u"Alunos Por Instituição",
Distrito", u"Percentagem Alunos por Instituiçao", u"Vagas por Colocar por Instituição",
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
```

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.

```
def button1Click(self, event):
        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)
```

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 consultámos os ficheiros em formato .pdf fornecidos pelo docente Jasnau Caeiro na página moodle da disciplina de Linguagens de Programação (LP).