createTexTests - code to create tex and pdf files with tables of answers sheet and multiple-choice questions randomly

This software creates a **tex** file (and the corresponding **pdf**) with questions multiple-choice and / or dissertation, can add an answer sheet on the first page of each test, grouped by classes of students. In addition, the software can create a file with the template to each student, if you choose the option of random tests. Thus, the tests may be individual and unique to each student and will be generated from csv files of student groups, including the registration number and name of the student. The code was developed in the python programming language.

The full content of this document can be downloaded from

- http://vision.ufabc.edu.br/MCTest (http://vision.ufabc.edu.br/MCTest) or
- https://github.com/fzampirolli/MCTest5 (https://github.com/fzampirolli/MCTest5)

List folders needed to run the program:

- courses
- questions
- figs

The files *class.csv* of the subfolders of *courses* will be read, which contains the data of the students. For example, consider the file *./courses/course2016q2/2016_BC0505_q2_A2.csv* containing only 2 students of the class *2016_BC0505_q2_A1*, with the following:

```
11000123; Fulano Junior
11000111; Gustavo Neto
```

The multiple-choice questions follow the following format: issues classified as easy (QE::), medium (QM::) and hard (QH::). For example, see the file contents. *./questions/testp1/questions1.txt*. There is also the possibility of creating text questions (QT::) without alternatives.

Within each group of easy, medium and hard questions, optionally, you can create subject(s) of the question and also create subclasses. For subclasses **the goal is to define variations of a question and to take only a matter of each subclass**. For this, another delimiter is designed with just one character to define each subclass. For example, QE::a::, QE::b::, ..., QE::A::, QE::B::, ..., QE::0::, QE::1::, ..., QE::9::. The contents of each question and each alternative follows the formatting **latex**, disregarding the sequence of line start characters QE::, QM::, QH::, QT:: and A: and the characters "::". Here's an example questions file:

```
QE::Assunto 1::a:: pergunta fácil Q1a1 - com exemplo de fórmula em tex, com a
primeira variação da subclasse a:
$\sin A \cos B = \frac{1}{2}\left[ \sin(A-B)+\sin(A+B) \right]$
% esta linha é um comentário
A: resposta 1a1-a
A: resposta 1a1-b
```

```
A: resposta 1a1-c
A: resposta 1a1-d
A: resposta 1a1-e

QE::Assunto 1::a:: pergunta fácil Q1a2 - com segunda variações da subclasse a:
A: resposta 1a2-a
A: resposta 1a2-b
A: resposta 1a2-c
A: resposta 1a2-d
A: resposta 1a2-e

QE::Assunto 1::a:: pergunta fácil Q1a3 - com terceira variações da subclasse a:
A: resposta 1a3-a
A: resposta 1a3-b
A: resposta 1a3-c
A: resposta 1a3-d
A: resposta 1a3-e

QM::Assunto 2:: pergunta q2 classificada como média
A: resposta 2a - para provas aleatórias, sempre a primeira resposta é a correta
A: resposta 2b
A: resposta 2c
A: resposta 2d
A: resposta 2e

QH::Assunto 3::  pergunta cQ3
A: resposta 3a
A: resposta 3b
A: resposta 3c
A: resposta 3d
A: resposta 3e
```

The alternative of the each question should follow the characters "A:". **For random questions, the first alternative is the correct**. Inside subfolder of the questions folder, always use the same number of alternatives.

Include textual questions (optional)

```
QT:: O programa abaixo lê dois valores para as variáveis X e Y, efetua a troca
dos valores de forma que a variável X passe a ter o valor de Y, e que a variável
Y passe a ter o valor de X. Complete a(s) instrução(ões) "AQUI".
\begin{verbatim}
programa
{
    funcao inicio()
    {
        real X, Y, aux
        leia (X, Y)
        "AQUI"
        escreva(X, Y)
    }
```

```
}
\end{verbatim}

\drawLines{12}

QT:: Escreva um programa para inverter os elementos com conteúdos pares que
estão nas posições impares de um vetor de inteiros com X elementos (caso
existam), onde X é um inteiro definido pelo usuário.

\drawLines{15}
```

To run the program *createTexTests*

Install

- linguage **ipython**: https://store.continuum.io/cshop/anaconda
  (https://store.continuum.io/cshop/anaconda), version 64bits python 2.7
- latex:
  - mac: https://tug.org/mactex/mactex-download.html (https://tug.org/mactex/mactex-download.html)
  - linux: sudo apt-get install texlive texlive-latex-extra texlive-lang-portuguese
  - windows: http://miktex.org/download (http://miktex.org/download)

Download the folder that is https://github.com/fzampirolli/MCTest5 (https://github.com/fzampirolli/MCTest5)
(top right Download ZIP), which will be saved in your disk: **MCTest5-master.zip**; Unzip.

Open a terminal and go to the folder:

cd *./MCTest5-master/

Edit/create the questions files in *./questions/testp1/questions1.txt* and the students groups files
./courses/course2016q2/2016_BC0505_q2_A2.csv. If necessary, create or rename subfoders on *questions*
and *courses* and the files *txt* and *csv*.

After installation, to run the program *createTexTests.py*, for example:

ipython createTexTests.py config_en.txt

The **config.txt** file must be changed according to user needs. However, you should not change the variable
names before the first **::** in eacha row. See an example below:

```
numQE            :: 4          :: number of easy questions
numQM            :: 3          :: number of mean questions
numQH            :: 2          :: number of hard questions
numQT            :: 2          :: number of textual questions
folderCourse     :: course2016q2 :: folder containing the classes in csv files
folderQuestions  :: testp1      :: folder containing the database issues in txt
files
randomTests      :: 1          :: =0, questions not random
MCTest_sheets    :: 2          :: =0 only answer sheet; =1 only questions; =2 cc
template         :: 1          :: =0, unsaved file with the templates *_GAB
duplexPrinting   :: 1          :: =0, printing on one side of the sheet, cc, printing
```

```
double-sided
maxQuestQuadro :: 20        :: maximum number of questions per answer sheet
maxQuadrosHoz   :: 4        :: maximum number of questions in horizontally
headerByQuestion :: 1       :: =1 a header by textual question

title           :: Universidade Federal do ABC
course          :: Processamento da Informação - BC0505
teachers        :: Francisco de Assis Zampirolli
period          :: 2/2016
modality        :: Presencial
date            :: 22/06/2016
logo            :: ufabc.eps  :: logo in figs folder
language        :: english    :: for now, portuguese or english

instructions1 ::
1. Paint only INSIDE OF THE CIRCLES of each question.  2. No Rasure. 3. Each
question has only one correct answer.
4. It will be considered only the answers in "Part 1" area on this page to the
questions of multiple choice.

instructions2 ::
1. It is prohibited to search books and notes.  2. The use of electronic devices
is prohibited. 3. It will be
considered only the answers in "Part 1" area of the Multiple-Choice Questions
page.

instructions3 ::

titPart1 :: Parte 1 - Answer Sheet - Do not use this sheet as a draft!
titPart2 :: Parte 2 - Multiple-Choice Questions
titPart3 :: Parte 3 - Textual Questions

endTable ::
\begin{table}[h]
\centering
\textbf{Calculation of concepts} \\ \vspace{5mm}
\begin{tabular}{|c|c|c|c|} \hline
Questions 1-12 & Textual Questions & Final\\ \hline
            & &   \\
            & &   \\
            & &   \\ \hline
\end{tabular}
\end{table}
```

After runnign the *createTexTests.py* program, will be created for each course two files into *tex/course2016q2/testp1/*, for example to the course defined in the file *./courses/course2016q3/2016_BC0505_q2_A2.csv*:

- 2016_BC0505_q2_A1.tex
- 2016_BC0505_q2_A1.pdf

If **randomTests** variable into **config.txt** have value 1, random questions will be created and this folder will also be generated the file with template for each student:

- 2016_BC0505_q2_A1__yourEmail@domain.com_GAB

NOTE: every time that *createTexTests.py* is run, will create a only file with the template of the each student, erasing earlier versions with the same name. It is recommended to rename the GAB file with your email and make a copy in **./corrections** folder, where the automatic correction of the tests will be made. This GAB file can be used for automatic corrections using MCTest, as detailed in the following.

To Correct the tests using MCTest

Scan the answers sheet of each student in pdf format, resolutin 150dpi, grouped into classes of students, with the following name pattern:

2016_BC0505_q2_A1__yourEmail@domain.pdf

Send this file and the file below with the templates (generated by *createTexTexts.py* program) by ftp:

2016_BC0505_q2_A1__yourEmail@domain_GAB

To send by ftp these files enter in *shell*:

```
ftp vision.ufabc.edu.br
Name (vision.ufabc.edu.br:fz): anonymous
```

Switch to the folder where will be made the **uploads** of files on the server:

```
ftp> cd upload
```

The only restriction is to use a particular file name, where before "__" (two *underlines*) defines the class and after must be an email, see example:

```
ftp> put 2016_BC0505_q2_A1__fzampirolli@gmail.com_GAB
ftp> put 2016_BC0505_q2_A1__fzampirolli@gmail.com.pdf
```

The MCTest program will send the processing of pages to this email. If you want to download the csv files generated, write **dir** to check that has been generated on the server vision and then:

```
ftp> get 2016_BC0505_q2_A1__fzampirolli@gmail.com.pdf.csv
```

There is a program runs every minute on server, looking files pdf which have not yet been processed (ie, if there is not a file with same name but with csv extension, containing the corrections).

As there are many *scanners*, with different qualities and resolutions, it is recommended to test this process with some tests before applying in a class with many students (I am available to help with anything you need).

```
    Prof. Francisco de Assis Zampirolli
    Centro de Matemática Computação e Cognição
```

Universidade Federal do ABC

fzampirolli@ufabc.edu.br

If your goal is to just use the *createTexTests.py* in *shell* to create tex file with tests using csv files of classes and txt files of questions, skip the rest of the document.

Note

If any change is made in this file **createTexTexts.ipynb** and/or there is not file *createTexTexts.py*, create it this notebook with *File->Download as-> Python (.py)*. For this, will have to install the **anaconda** (http://continuum.io/downloads (http://continuum.io/downloads)).

In [1]:

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

# part of the code presented in this document was inspired https://code.google.co

import random, sys, os, os.path, glob, csv, socket, string, smtplib

import numpy as np
import matplotlib.pyplot as plt
from unicodedata import normalize # retirar acentuação

# variáveis globais
if os.name == 'nt': # Windows
    barra = '\\'
else:
    barra = '/'

mypath = '.'+barra
mypathQuestions = mypath+'questions'+barra
mypathCourses = mypath+'courses'+barra
mypathTex = mypath+'tex'+barra
listextQuestions = ['*.txt']
listextCourses = ['*.csv']


letras_1 = ['A','B','C','D','E','F','G','H','I','J', 'K','L', 'M','N','O','P','Q'

def getQuestion(i, AllLines):
    tam = len(AllLines)
    while i < tam and AllLines[i][:3] not in ['QT:','QE:','QM:','QH:']: # acha um
        i += 1
    #if i == tam: return(i,' ') # não achou questão
    tp = AllLines[i][:3]
    q = []
    q.append(AllLines[i])
    i += 1
    while i < tam and AllLines[i][:AllLines[i].find(':')] not in ['QT','QE','QM',
        q.append(AllLines[i])
        i += 1
    if i<=tam and tp == 'QT:': # questao do tipo texto
```

```python
        return (i,' '.join([x for x in q]))
    if i<tam and tp in ['QE:','QM:','QH:'] and AllLines[i][:2] in ['QT','QE:','QM
        print 'ERRO: questão sem alternativas'
    return (i,' '.join([x for x in q]))


def getAnswer(i, AllLines):
    tam = len(AllLines)
    while i < tam and AllLines[i][:2] not in ['A:']: # acha uma questão
        i += 1
    #if i == tam: return(i,' ') # não achou questão
    q = []
    q.append(AllLines[i])
    i += 1
    while i < tam and AllLines[i][:AllLines[i].find(':')] not in ['QT','QE','QM',
        q.append(AllLines[i])
        i += 1
    return (i,' '.join([x for x in q]))


def questionsReadFiles(arquivos):
    # estados possiveis: fora de alguma questao
    #                    dentro de uma questao - 'QT','QE','QM','QH' - pergunta
    #                    dentro de uma questao - A - respostas
    # as questões são dos tipos QT (somente texto), QE (fácil), QM (média) ou QH
    # podendo ter subtipos, por exemplo, se tiver 5 questões, QE:a:, será escolhi
    # aleatória, somente uma questão do subtipo "a".
    # As questões QT, contendo apenas textos, serão inseridas no final do tex.

    listao = []
    respostas = []
    d = dict()
    arqnum = 0
    questnum = 0
    questtotal = 0
    questions_file = 0

    for a in arquivos: # para cada arquivo de questões

        arq = open(a)
        AllLines = arq.readlines()

        tam = len(AllLines)
        i = 0
        while i<tam:
            i, q = getQuestion(i, AllLines)

            d = dict()

            d["t"] = ''
            vet = q.split('::')
            if len(vet)==2: #somente tipo
                d["t"] = vet[0]  # tipo QT, QE, QM ou QH
                d["q"] = vet[1].strip()
                d["c"] = ''
                d["st"] = ''
            elif len(vet)==3: # com conteúdo abordado da questão
                d["t"] = vet[0]
```

```python
                s = normalize('NFKD', vet[1].decode('utf-8')).encode('ASCII', 'ig
                d["c"] = s
                d["q"] = vet[2].strip()
                d["st"] = ''
            elif len(vet)==4: # subtipo da questão, um caracter qualquer
                d["t"] = vet[0]  # tipo QT, QE, QM ou QH
                s = normalize('NFKD', vet[1].decode('utf-8')).encode('ASCII', 'ig
                d["c"] = s
                d["st"] = vet[2]
                d["q"] = vet[3].strip()

            d["n"] = questnum
            d["arq"] = arqnum

            respostas = []
            if d["t"] != "QT":
                contRespostas = 0
                while i < tam and AllLines[i][:AllLines[i].find(':')] in ['A']:
                    i, r = getAnswer(i,AllLines)
                    #if i == tam: break # não achou questão
                    respostas.append(r[2:].strip())
                    contRespostas +=  1

                if contRespostas==0:
                    print 'ERRO: questão sem respostas'
                    sys.exit(-1)

            d["a"] = respostas

            listao.append(d)
            questnum += 1

        arq.close()
        arqnum += 1
        print "read the questions file: %-40s with %d questions" % (a,len(listao)
        questions_file = len(listao)

    print "\nTotal of questions without suptype:"
    print "Easy questions QE: %d" % (len([y for y in listao if y['t'] == 'QE' and
    print "Mean questions QM: %d" % (len([y for y in listao if y['t'] == 'QM' and
    print "Hard questions QH: %d" % (len([y for y in listao if y['t'] == 'QH' and
    print "Text questions QT: %d" % (len([y for y in listao if y['t'] == 'QT' and

    print "\nTotal of questions with suptype:"
    print "Easy questions QE: %d" % (len([y for y in listao if y['t'] == 'QE' and
    print "Mean questions QM: %d" % (len([y for y in listao if y['t'] == 'QM' and
    print "Hard questions QH: %d" % (len([y for y in listao if y['t'] == 'QH' and
    print "Text questions QT: %d" % (len([y for y in listao if y['t'] == 'QT' and

    return listao

def createListTypes(listao,tipo,numQ):
    questTipo = [y for y in listao if y['t'] == tipo and y['st']==''] # pega toda

    st =  [(y['st'],y['n']) for y in listao if y['t'] == tipo and y['st']!=''] #
    if st:
        stSet = list(set([i[0] for i in st])) # retira elementos repetidos
```

```python
        for i in stSet: # para cada subtipo, pego apenas UMA questão aleatoriamen
            li =  [(y['st'],y['n']) for y in listao if y['t'] == tipo and y['st']
            escolhoUM = random.sample(li,1)
            ques  = [y for y in listao if y['n'] == escolhoUM[0][1]]
            questTipo.append(ques[0])

    if numQ > len(questTipo):
        print "number of available questions %s: \t %-5d" % (tipo, len(questTipo)
        print "\nERRO: number of solicitous questions is incompatible with the nu
        sys.exit(-1)

    return questTipo


def createTests(listao, turmas):
    """ se a variável randomTests==0:
        significa que não serão geradas provas aleatórias, ou seja, esta função v
        as primeiras questões fáceis, médias e difíceis e não vai embaralhar as r
        caberá ao professor gerar um arquivo *_GAB, fornecendo as soluções de cad

        se a variável randomTests!=0:
        cada prova eh gerada aleatoriamente a partir da lista de tuplas com todas
        recebe como argumentos: uma listao de tuplas (q,a), o num de provas a ser
        quantas questoes cada prova deve ter.
        retorna as provas embaralhas; as provas sao listas de tuplas (q,a,n,arq)
        considerando que toda resposta correta esta na opcao A, posicão 0, esta f
        tambem gabaritos, com as posições onde ficam a opção A após o embaralhame
        questão e de cada prova

        gabaritos = [Turma, matricula, gab, conteudos]
        provas    = [Turma, matricula, nome, questoes])

        código adaptado de https://code.google.com/p/criaprova/downloads/list
        """

    provas = []
    questaoporprova = numQE + numQM + numQH + numQT
    gabaritos = []

    countTurma = 0
    for t in turmas: # para cada turma

        for n in t: # para cada aluno da turma
            questoes = []

            questQE = createListTypes(listao,'QE',numQE) # tem que ficar aqui par
            questQM = createListTypes(listao,'QM',numQM) # uma questão aleatória
            questQH = createListTypes(listao,'QH',numQH) # caso exista
            questQT = createListTypes(listao,'QT',numQT)

            if int(randomTests)!=0: #questões aleatórias
                quest = random.sample(questQE,numQE)
                quest.extend(random.sample(questQM,numQM))
                quest.extend(random.sample(questQH,numQH))
                quest.extend(random.sample(questQT,numQT))
            else: #questões sequenciais, com as primeiras questões fáceis, médias
                quest = questQE[:numQE]
                quest.extend(questQM[:numQM])
```

```python
        quest.extend(questQH[:numQM])
        quest.extend(questQT[:numQT])

    indexQuest = []
    for q in quest:
        indexQuest.append(listao.index(q))

    c =  [y['c'] for y in listao if y['c']!=''] # pega questões COM conte
    cSet = []
    if c:
        for i in c:
            for j in i.split(' - '): # retira questão com mais de um cont
                cSet.append(j)

    conteudo = [] # cria uma lista de conteúdos
    if len(cSet):
        cSet = sorted(list(set(cSet))) # retira conteúdos repetidos
        for i in cSet:
            conteudo.append([i,[]])

    sequencia = []
    gab = []
    contQuest = 0
    for q in quest:
        perg = q['q']
        contQuest += 1

        if q['c']: # questão tem conteúdo(s)
            ii=0
            for i in conteudo:
                #print ">>>",i[0],q['c']
                for j in q['c'].split(' - '):
                    if i[0] == j:
                        #print ">",i[0],j
                        conteudo[ii][1].extend([contQuest])
                ii += 1

        embaralhaResps = []
        g = []
        if q["t"] != 'QT': # não é uma questão dissertativa, embaralha re
            if int(randomTests)!=0: #questões aleatórias
                embaralhaResps = random.sample(q['a'],len(q['a']))
            else:
                embaralhaResps = q['a'][:len(q['a'])]

            g = embaralhaResps.index(q['a'][0])
            gab.append(g)

        sequencia.append((perg,embaralhaResps,q['n'],q['arq'],g))
        if q['t'] != 'QNI':
            pass

    questoes.extend(sequencia)

    if len(questoes) < questaoporprova:
        print "\n\n Prova %d com menos questoes do que o solicitado; algu
```

```python
            if int(randomTests)!=0: #questões aleatórias, então salva gabarito
                gabaritos.append([n[0], n[1], gab, indexQuest, conteudo])

            provas.append([n[0], n[1], n[2], questoes])

        countTurma = countTurma + 1

    return provas, gabaritos

def readQuestionsFiles(p):
    fileQuestoes = []
    listdirQuestoes = glob.os.listdir(mypathQuestions)
    listdirQuestoes.append('')
    for ext in listextQuestions:
        for file in np.sort(glob.glob(mypathQuestions+p+barra+ext)):
            fileQuestoes.append(file)
    return fileQuestoes

def readClassFiles(p):
    fileTurmas = []
    listdirTurmas = glob.os.listdir(mypathCourses)
    listdirTurmas.append('')
    for ext in listextCourses:
        for file in np.sort(glob.glob(mypathCourses+p+barra+ext)):
            fileTurmas.append(file)
    return fileTurmas

def classesReadFiles(files):
    print ""
    turmas = []
    for fi in files:
        alunos=[]
        with open(fi, 'rb') as f:
            reader = csv.reader(f, delimiter=';')
            for row in reader:
                #print ">>>>", row
                s = normalize('NFKD', row[1].decode('utf-8')).encode('ASCII', 'ig
                alunos.append([fi,row[0],s])
        print "read the class file: %-40s with %d students" % (fi,len(alunos))
        turmas.append(alunos)
    print ""
    return turmas

def savesTemplates(gabaritos): # salva em disco todos os gabaritos num arquivo cs
    print ""

    if randomTests==0: #questões não aleatórias
        print "Warning: You chose in config.txt the option to generate non-random
        print "In this case, if you want to use the automatic correction using MC
        print "you must provide a file with the template or consider that the fir
        print "in pdf file is a template."

    else:

        files = []
        for g in gabaritos:
            files.append(g[0])
```

```python
        for ff in sorted(set(files)):

            f = ff[:-4]+'__seuEmail@dominio.com_GAB'

            past = f[10:]
            filename = past[past.find(barra):]
            past = mypathTex+barra+past[:past.find(barra)]
            try:
                os.stat(past)
            except:
                os.mkdir(past)

            past += barra+folderQuestions
            try:
                os.stat(past)
            except:
                os.mkdir(past)

            f = past+filename

            print "aquivo salvo com os gabaritos da cada aluno da turma:",f

            #[n[0], n[1], gab, indexQuest, conteudo]
            with open(f, 'w') as csvfile:
                for gab in gabaritos:
                    if ff is gab[0]:
                        spamWriter = csv.writer(csvfile, delimiter=' ',quotechar=
                        pathFile = gab[0]
                        if os.name=='nt': #windows
                            pathFile = pathFile.replace(barra,'/')
                        s = ''.join([x for x in str(gab[2])] )
                        s = s.strip('[')
                        s = s.strip(']')
                        s = s.replace(' ','')
                        s = s.strip()

                        i = ''.join([x for x in str(gab[3])] )
                        i = i.strip('[')
                        i = i.strip(']')
                        i = i.replace(' ','')
                        i = i.strip()

                        t = ''.join([x for x in str(gab[4])] )

                        spamWriter.writerow([pathFile, ';',  gab[1],';', s, ';',

def defineHeader(arqprova,strTurma,idAluno,nomeAluno): # define o cabeçalho de ca
    global instrucoes

    if config['language'].replace('\n','')=='portuguese':
        turma = "\\textbf{Turma:} %s\n" %  strTurma
        idAluno = "\\textbf{Matrícula:} %s\n" %  str(idAluno)
        nomeAluno="\\textbf{Aluno:} %s\n" % nomeAluno
        strAluno = "\\noindent"+nomeAluno+"\\hfill"+idAluno+"\\hfill"+turma+"\\hs
        ass = "\\noindent\\textbf{Ass:}\\rule{11.5cm}{0.1pt}\\hfill\\hspace{1cm}\
        instrucoes = "Instru\\c c\\~oes: "
        course = "Disciplina:"
```

```python
        teachers = "Professor(es):"
        period = "Quadrimestre:"
        modality = "Modalidade:"
        date = "Data:"

    if config['language'].replace('\n','')=='english':
        turma = "\\textbf{Room:} %s\n" %  strTurma
        idAluno = "\\textbf{Registration:} %s\n" %  str(idAluno)
        nomeAluno="\\textbf{Student:} %s\n" % nomeAluno
        strAluno = "\\noindent"+nomeAluno+"\\hfill"+idAluno+"\\hfill"+turma+"\\hs
        ass = "\\noindent\\textbf{Sig.:}\\rule{11.5cm}{0.1pt}\\hfill\\hspace{1cm}
        instrucoes = "Instructions: "
        course = "Course:"
        teachers = "Teacher(s):"
        period = "Period:"
        modality = "Modality:"
        date = "Date:"

    arqprova.write("")
    if duplexPrinting!=0:
        arqprova.write("\\makeatletter\\renewcommand*\\cleardoublepage{\\ifodd\\c
        arqprova.write("\\makeatother\n")
        arqprova.write("\\cleardoublepage\n")

    # header da página 1/2
    arqprova.write("\\begin{table}[h]\\centering\n")
    arqprova.write("\\begin{tabular}{|p{16mm}|p{16cm}|}\n\hline")
    arqprova.write("\multirow{4}{*}{\\hspace{-2mm}\\includegraphics[width=2cm]{..
    arqprova.write("&\\vspace{-2mm}\\noindent\\large\\textbf{"+config['title'].de
    arqprova.write("&\\noindent\\textbf{"+course+"} "+config['course'].decode('ut
    arqprova.write("&\\noindent\\textbf{"+teachers+"} "+config['teachers'].decode
    arqprova.write("&\\noindent\\textbf{"+period+"} "+config['period']+"\\hfill")
    arqprova.write("\\textbf{"+modality+"} "+config['modality']+"\\hfill")
    arqprova.write("\\textbf{"+date+"} "+config['date']+"\\hspace{-8mm}\\\\\\\n\hli
    arqprova.write("\\end{tabular}\n")
    arqprova.write("\\end{table}\n")
    arqprova.write("\\vspace{-4mm}\\small{\n"+strAluno.decode('utf-8').encode("la
    arqprova.write("\n\\vspace{8mm}\n")
    arqprova.write(ass.decode('utf-8').encode("latin1"))
    arqprova.write("}")


def createTexTests(provas): # salva em disco todos os testes em arquivos .tex
    preambulo1 = """
        \documentclass[10pt,brazil,a4paper]{exam}
        \usepackage[latin1]{inputenc}
        \usepackage[portuguese]{babel}
        \usepackage[dvips]{graphicx}
        %\usepackage{multicol}
        %\usepackage{shadow}
        %\usepackage{pifont}
        %\usepackage{listings}
        %\usepackage{fancyvrb}

        \\newcommand*\\varhrulefill[1][0.4pt]{\\leavevmode\\leaders\\hrule height

        \\def\\drawLines#1{{\\color{cyan}\\foreach \\x in {1,...,#1}{\\par\\vspac
```

```
    \usepackage{enumitem}
    \usepackage{multirow}
    \usepackage{amsmath}
    \usepackage{changepage,ifthen}
    %\usepackage{boxedminipage}
    %\usepackage{theorem}
    \usepackage{verbatim}
    \usepackage{tabularx}
    %\usepackage{moreverb}
    \usepackage{times}
    %\usepackage{relsize}
    \usepackage{pst-barcode}
    \usepackage{tikz}
    \setlength{\\textwidth}{185mm}
    \setlength{\\oddsidemargin}{-0.5in}
    \setlength{\\evensidemargin}{0in}
    \setlength{\\columnsep}{8mm}
    \setlength{\\topmargin}{-28mm}
    \setlength{\\textheight}{265mm}
    \setlength{\\itemsep}{0in}
    \\begin{document}
    \\pagestyle{empty}
    %\lstset{language=python}
    """

files = []
for t in provas: # acha as turmas
    files.append(t[0])

for fff in sorted(set(files)): # para cada turma

    f = fff[:-4]+'.tex'

    past = f[10:]
    filename = past[past.find(barra):]
    past = mypathTex+barra+past[:past.find(barra)]
    try:
        os.stat(past)
    except:
        os.mkdir(past)

    past += barra+folderQuestions
    try:
        os.stat(past)
    except:
        os.mkdir(past)

    f = past+filename


    with open(f, 'w') as arqprova:

        print "latex file saved with the tests of all students of the class(e
        arqprova = open(f,'w')
        arqprova.write(preambulo1.decode('utf-8').encode("latin1"))

        if False:
```

```python
        arqprova.write("\\begin{center}\n\n")
        arqprova.write("\\resizebox{!}{5mm}{Provas criadas por}\\vspace{1
        arqprova.write("\\resizebox{!}{5mm}{createTexTests.py}\\vspace{1c
        arqprova.write("\\resizebox{!}{5mm}{para o arquivo/turma:}\\vspac
        arqprova.write("\\resizebox{!}{5mm}{"+ filename[1:-4].replace('_'
        arqprova.write("\\resizebox{!}{5mm}{Guarde com seguran\c ca o arq
        arqprova.write("\\resizebox{!}{5mm}{"+ filename[1:-4].replace('_'
        arqprova.write("\\resizebox{!}{5mm}{Este arquivo contem os gabari
        arqprova.write("\\resizebox{!}{5mm}{individuais de cada teste!!!}
        arqprova.write("\\resizebox{!}{7mm}{\\'E \\'unico toda vez que ge
        arqprova.write("\\end{center}\n\n")
        arqprova.write("\\newpage\\thispagestyle{empty}\\mbox{}\\newpage\


    for t in provas:
        if fff is t[0]: # se prova é da mesma turma, acrescente
            ff = t[0]
            ff = ff[:-4]
            strTurma = ff[len(ff)-ff[::-1].find(barra):]
            strTurma = strTurma.replace("_","$\\_$")


            ###### Padroes dos quadros de respostas ######

            numQuestoes = len(t[3])-numQT # somente questões de múltpla e
            numRespostas = len(t[3][0][1])

            if numQuestoes>0:

                let = letras_1[0:numRespostas]
                strResps = (',').join([(let[x]+'/'+ str(x+1)) for x in ra

                # questões por quadro
                numQuadros = numQuestoes/maxQuestQuadro
                numResto = numQuestoes % maxQuestQuadro
                if numResto:
                    numQuadros+=1
                if numQuadros==0:
                    numQuadros+=1

                maxQuadrosHoz = int(config['maxQuadrosHoz'])

                if numQuestoes/maxQuestQuadro < maxQuadrosHoz:
                    maxQuadrosHoz = int(numQuestoes/maxQuestQuadro)

                numQuadrosHoz = numQuadros
                if maxQuadrosHoz<numQuadros:
                    numQuadrosHoz = maxQuadrosHoz

                numQuestoesQuadro = maxQuestQuadro
                if numQuestoes < maxQuestQuadro:
                    numQuestoesQuadro = numQuestoes
                QL=1
                if maxQuadrosHoz:
                    QL = numQuadros/maxQuadrosHoz # quadros por linha
                QC = numQuadrosHoz                # quadros por coluna
                if QC==0:
                    QC=1
```

```python
                    QC-1
            fimQuadro_ij = np.zeros([QL,QC])
            contadorQuestoes = 0
            for j in range(QC):
                for i in range(QL):
                    contadorQuestoes += numQuestoesQuadro
                    fimQuadro_ij[i][j] = contadorQuestoes
            if numQuestoes > maxQuestQuadro:
                fimQuadro_ij[QL-1][QC-1] += numResto

            numQuestStart = numQuestEnd = 0

        if int(MCTest_sheets)!=1: # foi escolhido a opção de gerar sc

            ################# pagina de resposta - Parte 1 ##########

            defineHeader(arqprova,strTurma,t[1],t[2]) # cabeçalho da

            arqprova.write("\\begin{pspicture}(6,0in)\n")
            arqprova.write("\\psbarcode[scalex=1.6,scaley=0.35]{%s}{}
            arqprova.write("\\end{pspicture}\n")


            if (config['instructions1']!="\n"):
                arqprova.write("\\\\{\\scriptsize\n\n\\noindent\\text
                arqprova.write(config['instructions1'].decode('utf-8'
                arqprova.write("\\end{verbatim}\n")

            if (config['titPart1']!="\n"):
                arqprova.write("\\begin{center}\\textbf{"+config['tit

            arqprova.write("\\vspace{-5mm}\\noindent\\varhrulefill[0.
            arqprova.write("\\vspace{-3mm}\\noindent\\varhrulefill[0.

            #print numQuestoes,numQuadros,numQuestoesQuadro, numResto
            arqprova.write("\\begin{center}\n")

            for i in range(QL): # para cada linha de quadros
                for j in range(QC): # para cada coluna de quadros

                    if fimQuadro_ij[i][j] == numQuestoes: # para o úl
                        numQuestStart = int(fimQuadro_ij[i][j] - numQ
                    else:
                        numQuestStart = int(fimQuadro_ij[i][j] - numQ

                    numQuestEnd = int(fimQuadro_ij[i][j])
                    #print "quadro",i,j, numQuestStart, numQuestEnd

                    arqprova.write("\\begin{tikzpicture}[font=\\tiny]
                    arqprova.write("  \\foreach \\letter/\\position i
                    arqprova.write("    \\node[inner sep=3pt] at ({\\
                    arqprova.write("  }\n")
                    arqprova.write("  \\foreach \\line in {%s,...,%s}
                    arqprova.write("    \\begin{scope}[xshift=0cm,ys
                    arqprova.write("      \\foreach \\letter/\\posit
                    #arqprova.write("        \\node[draw,fill,gray
                    arqprova.write("        \\node at (-0.1,0) {\\
                    arqprova.write("        \\node[fill=black!30,d
```

```python
                    arqprova.write("                \\node[fill=white,draw
                    arqprova.write("        }\n")
                    arqprova.write("      \\end{scope}\n")
                    arqprova.write("  }\n")
                    arqprova.write("\\end{tikzpicture}\\hspace{%s cm}
                arqprova.write("\n\n")


            arqprova.write("\\end{center}\n")
            saltaLinhas = max(0,15-numQuestoesQuadro/2)

            #arqprova.write("\\vspace{%s cm}\\noindent\\hrulefill\n\n

            arqprova.write("\\vspace{1cm}\\noindent\\varhrulefill[0.4
            arqprova.write("\\vspace{-3mm}\\noindent\\varhrulefill[0.

            arqprova.write(config['endTable'].decode('utf-8').encode(

            arqprova.write("\\newpage")
            if duplexPrinting!=0:
                arqprova.write("\\thispagestyle{empty}\\mbox{}\n \\ \

        if int(MCTest_sheets)!=0: # foi escolhido a opção de gerar sc

            ################  pagina de questoes - Parte 2 ########


            if numQuestoes>0:
                defineHeader(arqprova,strTurma,t[1],t[2]) # cabeçalho
                arqprova.write("\n\\vspace{4mm}\n")

                if (config['instructions2']!="\n"):
                    arqprova.write("\\\\{\\scriptsize\n\n\\noindent\\
                    arqprova.write(config['instructions2'].decode('ut
                    arqprova.write("\\end{verbatim}\n")

                if (config['titPart2']!="\n"):
                    arqprova.write("\\begin{center}\\textbf{"+config[

                arqprova.write("{\\small\n")
                arqprova.write("\\begin{questions}\n")
                arqprova.write("\\itemsep0pt\\parskip0pt\\parsep0pt\n
                for q in t[3]: # questões
                    if q[1]:
                        qstr = q[0]
                        #print ">>>",qstr
                        arqprova.write("\\question %s\n" % qstr.decod
                        arqprova.write("\\begin{choices}\n") #onepard
                        arqprova.write("\\itemsep0pt\\parskip0pt\\par
                        for r in q[1]: # respostas
                            #print ">>",r
                            arqprova.write("\\choice %s\n" % r.decode
                        arqprova.write("\\end{choices}\n")
                arqprova.write("\\end{questions}\n")
                arqprova.write("}")
                arqprova.write("\n \ \ \\ \n \\newpage\n")

            if numOT>0:
```

```python
                                if numQr>0:
                                    ################## questoes dissertativas – Parte 3
                                    if headerByQuestion!=1: # =1, um cabeçalho por questã
                                        defineHeader(arqprova,strTurma,t[1],t[2]) # cabeç
                                        arqprova.write("\n\\vspace{4mm}\n")

                                    if config['titPart3']!="\n":
                                        arqprova.write("\\begin{center}\\textbf{"+config[

                                    arqprova.write("{\\small\n")
                                    #arqprova.write("\\begin{questions}\n")
                                    #arqprova.write("\\itemsep0pt\\parskip0pt\\parsep0pt\
                                    for q in sorted(t[3]): # questões
                                        if q[1]==[]:
                                            if headerByQuestion==1: # um cabeçalho na pág
                                                defineHeader(arqprova,strTurma,t[1],t[2])
                                                arqprova.write("\n\n\\vspace{4mm}")
                                            arqprova.write("\\noindent %s \n\n" % q[0].de
                                            arqprova.write("\n \ \ \\ \n \\newpage\n")
                                            #arqprova.write("\\question %s\n" % q[0].deco
                                    #arqprova.write("\\end{questions}\n")
                                    arqprova.write("}\n")


                    arqprova.write("\\end{document}")
                    arqprova.close() # final do arquivo

def createTex2PDF(provas):
    files = []
    for t in provas: # acha as turmas
        files.append(t[0])
    for fff in sorted(set(files)): # para cada turma
        f = fff[:-4]+'.tex'
        past = f[10:]
        arq = past[past.find(barra):]
        past = mypathTex+past[:past.find(barra)]
        past += barra+folderQuestions
        f = past+arq
        p = os.getcwd()
        os.chdir(p+past[1:])
        os.system('cd '+f[len(past)+1:])
        os.system('latex '+'.'+arq)
        if os.name == 'nt': # Windows
            os.system('dvips -P pdf '+'.'+arq[:-4]+'.dvi')
            os.system('ps2pdf '+'.'+arq[:-4]+'.ps')
            os.system('del *.aux *.dvi *.aux *.log *.ps')
        else:
            os.system('dvipdf '+'.'+arq[:-4]+'.dvi')
            os.system('rm *.aux *.dvi *.aux *.log')
        os.chdir(p)


def getConfigLines(i, AllLines):
    tam = len(AllLines)
    while i < tam and AllLines[i]=='\n' and len(AllLines[i].split('::'))<2: # ach
        i += 1
    v = AllLines[i].split('::')
    s = []
```

```python
    v0 = v[0]
    v0 = v0.replace(' ','')
    v0 = v0.replace('\t','')
    ss = v[1]
    ss = ss.lstrip()
    ss = ss.rstrip()
    ss = ss.replace('\t','')
    s.append(ss)
    i += 1
    while i < tam and len(AllLines[i].split('::'))<2:
        ss = AllLines[i]
        ss = ss.lstrip()
        ss = ss.rstrip()
        ss = ss.replace('\t','')
        s.append(ss)
        i += 1
    return (i,v0,'\n'.join([x for x in s]))

def getConfig(file):
    global config, folderQuestions, folderCourse, randomTests, barra, MCTest_shee
    global numQE, numQM, numQH, numQT, duplexPrinting, maxQuestQuadro, maxQuadros
    global template

    arq = open(file)
    AllLines = arq.readlines()
    tam = len(AllLines)
    i = 0
    config = dict()
    while i<tam:
        i, v, s = getConfigLines(i, AllLines)
        config[v] = s

    numQE = int(config['numQE']) # num. questoes fáceis
    numQM = int(config['numQM']) # num. questoes médias
    numQH = int(config['numQH']) # num. questoes difíceis
    numQT = int(config['numQT']) # num. questoes dissertativas
    folderQuestions = config['folderQuestions'] # pasta com o bd de questões
    folderCourse = config['folderCourse']
    randomTests = int(config['randomTests'])
    MCTest_sheets = int(config['MCTest_sheets'])
    duplexPrinting = int(config['duplexPrinting'])
    template = int(config['template'])
    maxQuestQuadro = int(config['maxQuestQuadro'])
    maxQuadrosHoz = int(config['maxQuadrosHoz'])
    headerByQuestion = int(config['headerByQuestion'])

def main():
    global turmas, gabaritos, randomTests, barra, MCTest_sheets, folderQuestions,
    global numQE, numQM, numQH, numQT, duplexPrinting, maxQuestQuadro, maxQuadros
    global config

    try:
        if len(sys.argv)==2:
            getConfig(sys.argv[1]) # ler as variáveis de configuração e layout

            turmas = classesReadFiles(readClassFiles(folderCourse))
            provas=[]
            gabaritos=[]
```

```
            gabaritos=[]
            listao = questionsReadFiles(readQuestionsFiles(folderQuestions))
            provas, gabaritos = createTests(listao, turmas)
            createTexTests(provas)
            if template!=0:
                savesTemplates(gabaritos)
            createTex2PDF(provas)

    except ValueError:
        print "Oops!  Erro in File:",sys.argv[1], "Try again..."


if __name__ == '__main__':
    main()
```

To test this code anaconda:

Read the settings file:

In [2]:

```
getConfig('config.txt')
```

Read the files of students classes:

In [3]:

```
turmas = classesReadFiles(readClassFiles(folderCourse))
```

```
read the class file: ./courses/course2016q2/2016_BC0505_q2_A2.csv wi
th 2 students
read the class file: ./courses/course2016q2/2016_BC0505_q2_A3.csv wi
th 3 students
```

Read the questions files:

```
In [4]:
```

```
provas=[]
gabaritos=[]
listao = questionsReadFiles(readQuestionsFiles(folderQuestions))
```

```
read the questions file: ./questions/testp1/questions1.txt        wi
th 8 questions
read the questions file: ./questions/testp1/questions2.txt        wi
th 9 questions
read the questions file: ./questions/testp1/questionsText.txt     wi
th 5 questions

Total of questions without suptype:
Easy questions QE: 4
Mean questions QM: 4
Hard questions QH: 5
Text questions QT: 1

Total of questions with suptype:
Easy questions QE: 4
Mean questions QM: 0
Hard questions QH: 0
Text questions QT: 4
```

Create the tex files with tests:

```
In [5]:
```

```
provas, gabaritos = createTests(listao, turmas)
createTexTests(provas)
```

```
latex file saved with the tests of all students of the class(es): ./
tex//course2016q2/testp1/2016_BC0505_q2_A2.tex
latex file saved with the tests of all students of the class(es): ./
tex//course2016q2/testp1/2016_BC0505_q2_A3.tex
```

Save the file with the templates:

```
In [6]:
```

```
if template!=0:
    savesTemplates(gabaritos)
```

```
aquivo salvo com os gabaritos da cada aluno da turma: ./tex//course2
016q2/testp1/2016_BC0505_q2_A2__seuEmail@dominio.com_GAB
aquivo salvo com os gabaritos da cada aluno da turma: ./tex//course2
016q2/testp1/2016_BC0505_q2_A3__seuEmail@dominio.com_GAB
```

Compile the files tex generating the files pdf:

```
In [7]:
```

```
createTex2PDF(provas)
```