

createTexTests - criar arquivo tex com quadros de respostas e questões de múltipla-escolha de forma aleatória

Este software cria um arquivo **tex** (e o correspondente **pdf**) com questões de múltipla-escolha e/ou dissertativa, podendo acrescentar um quadro de respostas na primeira página de cada teste, agrupado por turmas de alunos. Além disso, o software pode criar um arquivo com o gabarito de cada aluno, caso escolha a opção de provas aleatórias. Assim, as provas podem ser individuais e únicas para cada aluno e serão geradas a partir de arquivos csv de turmas de alunos, incluindo o número de matrícula e nome do aluno. O código foi desenvolvido na linguagem de programação python.

O conteúdo completo deste documento pode ser baixado de

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

Diretórios necessários para rodar o programa:

- classes
- questions
- figs

Os arquivos *turmas.csv* de um subdiretório de *classes* serão lidos, onde constam os dados dos alunos. Por exemplo, considere o arquivo *./classes/2015q3/2015_BC0505_q3_A1.csv* contendo apenas 2 alunos da turma *2015_BC0505_q3_A1*, com o seguinte conteúdo:

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

As questões de múltipla-escolha seguem a seguinte formatação: questões classificadas como fáceis (QE::), médias (QM::) e difíceis (QH::). Por exemplo, veja o conteúdo do arquivo *./questions/p1/questions1.txt*. Existe também a possibilidade de criar questões dissertativas (QT::), sem alternativas.

Dentro de cada classe de questões fácil, média e difícil, de forma opcional, é possível criar assunto(s) da questão e também criar subclasses. Para as subclasses **o objetivo é definir variações de uma questão e no sorteio pegar apenas uma questão de cada subclasse**. Para isto, foi criado um outro delimitador, com apenas um caracter para definir cada subclasse, por exemplo, QE::a::, QE::b::, ..., QE::A::, QE::B::, ..., QE::0::, QE::1::, ..., QE::9::. O conteúdo de cada questão e de cada alternativa segue a formatação do **latex**, desconsiderando a sequência de caracteres de início de linha QE::, QM::, QH::, QT:: e A: e o caracteres "::". Veja um exemplo de arquivo de questões:

```
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

As alternativas de cada questão devem seguir dos caracteres "A:". **Para provas aleatórias, sempre a primeira alternativa é a correta.** No banco de questões em arquivos *./questions/pasta/arquivo.txt*, sempre usar o mesmo número de alternativas.

Incluir questões dissertativas (opcional)

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"
```

```
"AQUI"
escreva(X, Y)
}
}
\end{verbatim}
```

```
\drawLines{12}
```

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

```
\drawLines{15}
```

Para rodar o programa *createTexTests*

Instale

- linguagem **ipython**: <https://store.continuum.io/cshop/anaconda> (<https://store.continuum.io/cshop/anaconda>), versão 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

Fazer download da pasta que está em <https://github.com/fzampirolli/MCTest5> (<https://github.com/fzampirolli/MCTest5>) (canto inferior direito, Download ZIP), onde será salvo em seu disco **MCTest5-master.zip**. Descompacte.

Abrir um terminal e ir para a pasta,

```
cd *./MCTest5-master/
```

Edite/crie os arquivos de questões em *./questions/testp1/questions1.txt* e os arquivos de turmas de alunos em *./classes/2015q3test/2015_BC0505_q3_A2.csv*. Se necessário, criem ou renomeie subpastas em *questions* e *classes* e arquivos *txt* e *csv*.

Após a instalação, para rodar o programa *createTexTests.py* digite por exemplo:

```
ipython createTexTests.py config.txt
```

O arquivo **config.txt** deve ser alterado conforme a necessidade do usuário. Porém, não se deve alterar os nomes das variáveis antes do primeiro :: em cada linha. Veja um exemplo de abaixo:

```
numQE           :: 4           :: número de questões fáceis
numQM           :: 3           :: número de questões médias
numQH           :: 2           :: número de questões difíceis
numQT           :: 2           :: número de questões dissertativas
pasteQuestions  :: testp1      :: pasta contendo o banco de questões em args.
txt
pasteClasses    :: 2015q3test  :: pasta conteno as turmas em arquivos csv
randomTests     :: 1           :: =0, questões não aleatórias
MCTest sheets   :: 2           :: =0 somente quadros de respostas; =1 somente
```

```

questões; =2 cc
template      :: 1          :: =0, não salva arquivo com os gabaritos
*_GAB
duplexPrinting :: 1          :: =0, impressão em um lado da folha, cc,
frente e verso
maxQuestQuadro :: 20        :: número máximo de questões por quadro
maxQuadrosHoz  :: 4          :: número máximo de quadros na horizontal
headerByQuestion :: 1       :: =1 um cabeçalho por questão dissertativa

title          :: Universidade Federal do ABC
course          :: Processamento da Informação - BC0505
teachers        :: Denise Goya, Itana Stiubiener, Francisco Zampiolli, Luiz
Rozante, Monael Ribeiro
period          :: 3/2015
modality        :: Semipresencial
date            :: 22/10/2015
logo            :: ufabc.eps      :: logo tem que estar na pasta figs
language        :: portuguese    :: por enquanto, portuguese ou english

instructions1 ::
1. Pinte somente DENTRO DOS CÍRCULOS de cada questão. 2. Não rasure. 3. Cada
questão possui uma única resposta correta.
4. Somente serão consideradas as respostas na região "Parte 1" desta página para
as questões de múltipa-escolha.

instructions2 ::
1. Proibida a consulta de livros ou anotações. 2. Proibido o uso de
dispositivos eletrônicos. 3. Somente serão
consideradas as respostas na região "Parte 1 - Quadro(s) de Respostas" para as
questões de múltipla-escolha.

titPart1 :: Parte 1 - Quadro(s) de Respostas - Não utilize esta FOLHA como
rascunho!
titPart2 :: Parte 2 - Questões de Múltipa-Escolha
titPart3 :: Parte 3 - Questões Dissertativas

endTable ::
\begin{table}[h]
\centering
\textbf{Cálculo dos Conceitos} \\ \vspace{5mm}
\begin{tabular}{|c|c|c|c|} \hline
Questões 1-12 & Questão Dissertativa & Final\\ \hline
& & \\
& & \\
& & \\ \hline
\end{tabular}
\end{table}

```

Após rodar o programa *createTexTests.py*, serão criados, para cada turma, dois arquivos na pasta *tex/2015q3/testp1/*, por exemplo para a turma definida no arquivo *./classes/2015q3test/2015_BC0505_q3_A2.csv*:

- 2015_BC0505_q3_A1.tex

- 2015_BC0505_q3_A1.pdf

Se em **config.txt** a variável **randomTests** tiver valor 1, serão criados questões aleatórias e nesta pasta também será gerado o arquivo com o gabarito de cada aluno:

- 2015_BC0505_q3_A1__seuEmail@dominio.com_GAB

ATENÇÃO: toda vez que o programa *createTexTests.py* for executado será criado um arquivo **único** com o gabarito de cada aluno, apagando versões anteriores com o mesmo nome de turma, caso existam. É recomendável renomear o arquivo GAB com o seu email e fazer uma cópia na pasta **./corrections**, onde será feita a correção automática das provas. Este arquivo GAB poderá ser usado para correções automáticas usando o MCTest, como detalhado neste documento a seguir.

Para corrigir as provas usando o MCTest

Digitalizar os quadros de respostas de cada aluno em formato pdf, resolução 150dpi, agrupados em turmas de alunos, com o seguinte padrão de nome:

2015_BC0505_q3_A1__seuEmail@dominio.pdf

Enviar este arquivo e o arquivo abaixo com os gabaritos (gerado pelo programa *createTexTests.py*) por ftp:

2015_BC0505_q3_A1__seuEmail@dominio_GAB

Para enviar por ftp estes arquivos digite no *shell*:

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

Mude para a pasta onde serão feitos os **uploads** dos arquivos no servidor:

```
ftp> cd upload
```

A única restrição é usar um nome de arquivo particular, onde antes de "___" (dois *underlines*) define a turma e após deve existir um email, veja exemplo:

```
ftp> put 2015_BC0505_q3_A1__fzampirolli@gmail.com_GAB
ftp> put 2015_BC0505_q3_A1__fzampirolli@gmail.com.pdf
```

O programa MCTest irá enviar o processamento das páginas para este email. Se quiser fazer download do arquivo csv gerado, digite **dir** para verificar se já foi gerado no servidor vision e em seguida:

```
ftp> get 2015_BC0505_q3_A1__fzampirolli@gmail.com.pdf.csv
```

O programa roda a cada minuto a pasta ftp procurando arquivos pdf, que ainda não foram processados (ou seja, se não existir um arquivo com o mesmo nome, mas com extensão csv, contendo as correções).

Como existem muitos *scanners*, com diferentes qualidades e resoluções, é recomendável testar este processo com algumas provas antes de aplicar em uma turma com muitos alunos (estou à disposição para ajudar no que for preciso).

Prof. Francisco de Assis Zampirolli
Centro de Matemática Computação e Cognição
Universidade Federal do ABC
fzampirolli@ufabc.edu.br

Se o seu objetivo é apenas usar o programa *createTexTests.py* do *shell* para criar arquivos tex com um conjunto de provas a partir de arquivos csv de turmas e de arquivos txt de questões, ignore o restante deste documento.

Observação

Caso alguma alteração seja feita neste arquivo **createTexTexts.ipynb** e/ou não exista o arquivo *createTexTexts.py*, gere-o deste notebook, com *File->Download as-> Python (.py)*. Para isto terá que instalar o **anaconda** (<http://continuum.io/downloads> (<http://continuum.io/downloads>)).

In [1]:

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

# parte do código apresentado neste documento foi inspirado de https://code.google.com/p/

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
mypathClasses = mypath+'classes'+barra
mypathTex = mypath+'tex'+barra
listextQuestoes = ['*.txt']
listextTurmas = ['*.csv']

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

def getQuestao(i, todaslinhas):
    tam = len(todaslinhas)
    while i < tam and todaslinhas[i][:3] not in ['QT:', 'QE:', 'QM:', 'QH:']: # acha
        i += 1
```

```

#if i == tam: return(i, ' ') # não achou questão
tp = todaslinhas[i][:3]
q = []
q.append(todaslinhas[i])
i += 1
while i < tam and todaslinhas[i][:todaslinhas[i].find(':')] not in ['QT','QE']
    q.append(todaslinhas[i])
    i += 1
if i<=tam and tp == 'QT:': # questao do tipo texto
    return (i, ' '.join([x for x in q]))
if i<tam and tp in ['QE:', 'QM:', 'QH:'] and todaslinhas[i][:2] in ['QT', 'QE:',
    print 'ERRO: questão sem alternativas'
return (i, ' '.join([x for x in q]))

```

```

def getResposta(i, todaslinhas):
    tam = len(todaslinhas)
    while i < tam and todaslinhas[i][:2] not in ['A:']: # acha uma questão
        i += 1
    #if i == tam: return(i, ' ') # não achou questão
    q = []
    q.append(todaslinhas[i])
    i += 1
    while i < tam and todaslinhas[i][:todaslinhas[i].find(':')] not in ['QT','QE']
        q.append(todaslinhas[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()
argnum = 0
questnum = 0
questtotal = 0

```

```

for a in arquivos: # para cada arquivo de questões
    arq = open(a)
    todaslinhas = arq.readlines()

    tam = len(todaslinhas)
    i = 0
    while i<tam:
        i, q = getQuestao(i, todaslinhas)

```

```

        d = dict()

```

```

        d["t"] = ' '

```

```

        vet = q.split(':')

```

```

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]
    s = normalize('NFKD', vet[1].decode('utf-8')).encode('ASCII', 'ignore')
    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', 'ignore')
    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 todaslinhas[i][:todaslinhas[i].find(':')] in ['Q', 'E', 'M', 'H']:
        i, r = getResposta(i,todaslinhas)
        #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

return listao

```

```

def criaListaTipos(listao, tipo, numQ):
    questTipo = [y for y in listao if y['t'] == tipo and y['st']==''] # pega todas as questões de um tipo

    st = [(y['st'], y['n']) for y in listao if y['t'] == tipo and y['st']!=''] # subtipos
    if st:
        stSet = list(set([i[0] for i in st])) # retira elementos repetidos
        for i in stSet: # para cada subtipo, pego apenas UMA questão
            li = [(y['st'], y['n']) for y in listao if y['t'] == tipo and y['st']==i]
            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 "num de questões disponíveis %s: \t %-5d" % (tipo, len(questTipo))
print "\nERRO: número de questões solicitadas é incompatível com o num de q
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 embaralhadas; as provas sao listas de tuplas (q,a,n,arq)
        considerando que toda resposta correta esta na opcao A, posição 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 = criaListaTipos(listao,'QE',numQE) # tem que ficar aqui para
        questQM = criaListaTipos(listao,'QM',numQM) # uma questão aleatória c
        questQH = criaListaTipos(listao,'QH',numQH) # caso exista
        questQT = criaListaTipos(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])
            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 conteúdo
cSet = []
if c:
    for i in c:
        for j in i.split(' - '): # retira questão com mais de um conteúdo
            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 respostas
    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; alguns alunos não fizeram a prova" % questaoporprova

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 listextQuestoes:
        for file in np.sort(glob.glob(mypathQuestions+p+barra+ext)):
            fileQuestoes.append(file)
    return fileQuestoes

def readClassFiles(p):
    fileTurmas = []
    listdirTurmas = glob.os.listdir(mypathClasses)
    listdirTurmas.append('')
    for ext in listextTurmas:
        for file in np.sort(glob.glob(mypathClasses+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', 'ignore')
                alunos.append([fi,row[0],s])
        print "turma lida %-40s com %d alunos" % (fi,len(alunos))
        turmas.append(alunos)
    print ""
    return turmas

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

    if randomTests==0: #questões não aleatórias
        print "Atenção: foi escolhido a opção de gerar provas não aleatórias"
        print "neste caso, se o professor desejar utilizar a correção automática"
        print "usando o MCTest, será necessário fornecer um arquivo com o gabarito"
        print "ou considerar que sempre a primeira prova processada do pdf é um gabarito"

    else:

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

        for ff in sorted(set(files)):

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

            past = f[10:]
            filename = past[past.find(barra):]
            #print "Arquivo: %s" % filename

```

```

past = mypath'ex+barra'+past[:past.find(barra)]
try:
    os.stat(past)
except:
    os.mkdir(past)

past += barra+pasteQuestion
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:"
    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:"

```

```

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

```

header da página 1/2

```

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

```

def createTexTests(provas): *# salva em disco todos os testes em arquivos .tex*
preambulol = ""

```

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

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

\\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+pasteQuestion
    try:
        os.stat(past)
    except:
        os.mkdir(past)

    f = past+filename

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

    print "aquivo latex salvo com as provas de todos os alunos da(s) turma"
    arqprova = open(f, 'w')
    arqprova.write(preambulol1.decode('utf-8').encode("latin1"))

    if False:
        arqprova.write("\\begin{center}\\n\\n")
        arqprova.write("\\resizebox{!}{5mm}{Provas criadas por}\\\\vspace{1cm}")
        arqprova.write("\\resizebox{!}{5mm}{createTexTests.py}\\\\vspace{1cm}")
        arqprova.write("\\resizebox{!}{5mm}{para o arquivo/turma:}\\\\vspace{1cm}")
        arqprova.write("\\resizebox{!}{5mm}{"+ filename[1:-4].replace('_', ' ')+"}")
        arqprova.write("\\resizebox{!}{5mm}{Guarde com segurança o arquivo}")
        arqprova.write("\\resizebox{!}{5mm}{"+ filename[1:-4].replace('_', ' ')+"}")

```

```

argprova.write("\\resizebox{!}{5mm}{Este arquivo contem os gabari
argprova.write("\\resizebox{!}{5mm}{individuais de cada teste!!!}
argprova.write("\\resizebox{!}{7mm}{\\'E \\'unico toda vez que ge
argprova.write("\\end{center}\\n\\n")
argprova.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 múltipla 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

```

```

    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}{%s}" % (t[1],t[2]))
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['titPart1']+"}\\n")

arqprova.write("\\vspace{-5mm}\\noindent\\varhrulefill[0.5pt]{10cm}\\n")
arqprova.write("\\vspace{-3mm}\\noindent\\varhrulefill[0.5pt]{10cm}\\n")

#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 último
            numQuestStart = int(fimQuadro_ij[i][j] - numQuestoes + 1)
        else:
            numQuestStart = int(fimQuadro_ij[i][j] - numQuestoes + 1)

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

        arqprova.write("\\begin{tikzpicture}[font=\\tiny]
        arqprova.write("    \\foreach \\letter/\\position in {\\quadros}
        arqprova.write("        \\node[inner sep=3pt] at (\\position) {\\letter}
        arqprova.write("    }\\n")
        arqprova.write("    \\foreach \\line in {%s,...,%s}
        arqprova.write("        \\begin{scope}[xshift=0cm,yshift=\\heightof{\\letter}\\cm]
        arqprova.write("            \\foreach \\letter/\\position in {\\quadros}
        #arqprova.write("                \\node[draw,fill,gray] at (\\position) {\\letter}
        arqprova.write("            \\node at (-0.1,0) {\\quadros}
        arqprova.write("            \\node[fill=black!30,draw=black] at (\\position) {\\letter}
        arqprova.write("            \\node[fill=white,draw=black] at (\\position) {\\letter}
        arqprova.write("        }\\n")
        arqprova.write("        \\end{scope}\\n")
        arqprova.write("    }\\n")
        arqprova.write("\\end{tikzpicture}\\hspace{%s cm}" % (t[1]-t[2]))
        arqprova.write("\\n\\n")

```



```

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

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

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

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

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

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

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

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

    if (config['instructions2']!="\\n"):
        argprova.write("\\\\{\\scriptsize\\n\\n\\noindent\\n")
        argprova.write(config['instructions2'].decode('utf-8'))
        argprova.write("\\end{verbatim}\\n")

    if (config['titPart2']!="\\n"):
        argprova.write("\\begin{center}\\textbf{"+config['titPart2'].decode('utf-8')+"}\\n")

    argprova.write("{\\small\\n")
    argprova.write("\\begin{questions}\\n")
    argprova.write("\\itemsep0pt\\parskip0pt\\parsep0pt\\n")
    for q in t[3]: # questões
        if q[1]:
            qstr = q[0]
            #print ">>>",qstr
            argprova.write("\\question %s\\n" % qstr.decode('utf-8'))
            argprova.write("\\begin{choices}\\n") #oneparchoice
            argprova.write("\\itemsep0pt\\parskip0pt\\parsep0pt\\n")
            for r in q[1]: # respostas
                #print ">>",r
                argprova.write("\\choice %s\\n" % r.decode('utf-8'))
            argprova.write("\\end{choices}\\n")
        argprova.write("\\end{questions}\\n")
    argprova.write("}")
    argprova.write("\\n \\ \\ \\n \\newpage\\n")

if numQT>0:
    ##### questoes dissertativas - Parte 3 #####
    if headerByQuestion!=1: # =1, um cabeçalho por questão
        defineHeader(argprova,strTurma,t[1],t[2]) # cabeçalho
        argprova.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\\n")
for q in sorted(t[3]): # questões
    if q[1]==[]:
        if headerByQuestion==1: # um cabeçalho na página
            defineHeader(arqprova,strTurma,t[1],t[2])
            arqprova.write("\\n\\n\\vspace{4mm}")
            arqprova.write("\\noindent %s \\n\\n" % q[0].descricao)
            arqprova.write("\\n \\ \\ \\ \\n \\newpage\\n")
            #arqprova.write("\\question %s\\n" % q[0].descricao)
#arqprova.write("\\end{questions}\\n")
arqprova.write("}\\n")
```

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

```
def geraTex2PDF(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+pasteQuestion
        f = past+arq
        p = os.getcwd()
        os.chdir(p+past[1:])
        os.system('cd '+f[len(past)+1:])
        os.system('latex '+f[len(past)+1:]+' '+barra+arq)
        os.system('dvipdf '+f[len(past)+1:]+' '+barra+arq[:-4]+'.dvi')
        os.system('rm *.aux *.dvi *.aux *.log')
        os.chdir(p)
```

```
def getConfigLines(i, todaslinhas):
    tam = len(todaslinhas)
    while i < tam and todaslinhas[i]=='\\n' and len(todaslinhas[i].split('::'))<2:
        i += 1
    v = todaslinhas[i].split('::')
    s = []
    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(todaslinhas[i].split('::'))<2:
        ss = todaslinhas[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
    arq = open(file)
    todaslinhas = arq.readlines()
    tam = len(todaslinhas)
    i = 0
    config = dict()
    while i<tam:
        i, v, s = getConfigLines(i, todaslinhas)
        config[v] = s

```

```

def main():
    global turmas, gabaritos, randomTests, barra, MCTest_sheets, pasteQuestion, p
    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
            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
            pasteQuestion = config['pasteQuestions'] # pasta com o bd de questões
            pasteClasses = config['pasteClasses']
            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'])

```

```

            turmas = classesReadFiles(readClassFiles(pasteClasses))
            provas=[]
            gabaritos=[]
            listao = questionsReadFiles(readQuestionsFiles(pasteQuestion))
            provas, gabaritos = createTests(listao, turmas)
            createTexTests(provas)
            if template!=0:
                savesTemplates(gabaritos)
            geraTex2PDF(provas)

```

```

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

```

```

if __name__ == '__main__':
    main()

```

