



**UNIVERSIDADE FEDERAL DO PARANÁ**  
**BACHARELADO EM MATEMÁTICA**

Lucas Henrique de Castro Fonseca  
Paulo Ribeiro Junior  
Priscilla Pereira de Souza

**GERADOR DE LISTAS E PROVAS**

**CURITIBA**  
**2018**

**LUCAS HENRIQUE DE CASTRO FONSECA (GRR20185646)**

PAULO RIBEIRO JUNIOR (GRR20185647)  
PRISCILLA PEREIRA DE SOUZA (GRR20185642)

## GERADOR DE PROVAS E LISTAS

Relatório apresentado à disciplina  
Fundamentos de Programação de  
Computadores do Curso de Graduação em  
Bacharelado em Matemática da  
Universidade Federal do Paraná.  
Orientador: Prof. Jackson Antônio do Prado  
Lima.

Curitiba, 26 de novembro de 2018.


### SUMÁRIO

1.Introdução.....	3
2.Objetivo.....	4
3.Desenvolvimento.....	5

3.1.Banco de dados.....	5
3.2.Funções.....	5
3.3 Latex.....	6
3.4 Leitura dos dados.....	8
4.Conclusão.....	1

0

## 1. INTRODUÇÃO

Este relatório exterioriza informações concernentes ao trabalho efetuado em trios da disciplina de Fundamentos de Programação de Computadores, a respeito da produção de provas e listas de Cálculo I, que fornece ao  uma plataforma célere e prática de elaboração de exames e listas de exercícios.

## **2.Objetivo**

O intuito do presente trabalho é possibilitar ao docente gerar uma lista e/ou uma prova de Cálculo I, cujos conteúdos são: limites, continuidade e primórdios de derivadas. Ademais, o programa, criado na linguagem Python 3.2, proporciona a opção de que o documento gerado venha com as respectivas resoluções, conforme a finalidade visada pelo professor.

### **3.Desenvolvimento**

#### **3.1.Banco de dados**

O nosso banco de dados é uma lista, cujas entradas são outras listas; e essas são dadas da seguinte maneira:

$[x], ["Pergunta \text{ no formato latex}"] ["Resolução \text{ no formato latex}"]$

onde  $x$  é um dado elemento do conjunto  $\{1,2,3\}$  e seus elementos denotam o nível de complexidade dos exercícios.

A escala de dificuldade possui três categorias: trivial, média e difícil, que se relacionam a 1, 2 e 3, respetivamente.

Para  $x = 1$ , por exemplo, o nível de complicação é baixo e, para o usuário, tal valor corresponderá à variável N1.

O banco de dados está apresentado como a lista `quesitoResponsumque`.

### 3.2 Funções

O usuário é apresentado a duas opções, a primeira, de gerar o PDF de uma prova e a segunda, de engendrar o PDF duma lista de exercícios com a sua resolução no final. Separamos, pois, em duas etapas, a segmentação das questões e a geração do arquivo PDF.

Para isso, foram criadas duas funções. A primeira delas, é a função `prova(BancoDeManco,quantN1,quantN2,quantN3,Insti,Dies)`, que inicialmente, por uma série de `while` verifica se o número de questões restante de cada respectiva dificuldade, é maior do que zero, e enquanto isso acontecer é chamada a biblioteca `random` para ser usada a função `randint` que escolhe um número aleatório entre 0 e `len(BancoDeManco)-1` e associa esse número à variável `lajota`. Na nossa lista `BancoDeManco` na posição `[lajota][0]`, como visto na seção 3.1, acessamos o valor do nível de dificuldade da questão, verificamos se esse é o valor desejado para o respectivo `while` e também verificamos se o elemento na posição `[lajota][1]` já não está inserido na matriz `provinha`, pois não queremos uma prova com enunciados iguais. Satisfeitas as condições, adicionamos a entrada `[lajota][1]` da lista `BancoDeManco` na lista `provinha`, nela se encontram os enunciados das questões. O “tamanho do passo” de cada `while` é a quantidade de questões que o usuário pediu de cada nível, e o laço de repetição irá parar somente quando na lista `provinha` já estiver inserida com todos os enunciados para a prova.

```

8 def prova(BancoDeManco,quantN1,quantN2,quantN3,Insti,Dies):
9     provinha = []
10    #Pega a quantidade de questoes que voce pediu
11    while (quantN1 > 0):
12        lajota = random.randint(0, len(BancoDeManco)-1)
13        if (BancoDeManco[lajota][0] == [1] and not(BancoDeManco[lajota][1] in provinha)):
14            print(BancoDeManco[lajota][1])
15            provinha.append(BancoDeManco[lajota][1])
16            quantN1 -= 1
17    while quantN2 > 0:
18        lajota = random.randint(0, len(BancoDeManco)-1)
19        if (BancoDeManco[lajota][0] == [2] and not(BancoDeManco[lajota][1] in provinha)):
20            provinha.append(BancoDeManco[lajota][1])
21            quantN2 -= 1
22    while quantN3 > 0:
23        lajota = random.randint(0, len(BancoDeManco)-1)
24        if (BancoDeManco[lajota][0] == [3] and not(BancoDeManco[lajota][1] in provinha)):
25            provinha.append(BancoDeManco[lajota][1])
26            quantN3 -= 1

```

Figura 1

A segunda função, é a `lista(BancoDeManco)` que, em suma, possui a mesma funcionalidade da primeira, mas com a diferença de adicionar na lista `provinha` as questões com as suas resoluções, encontradas nas posições `[lajota][2]` da lista `BancoDeManco`.

```

61 def lista (BancoDeManco):
62     pergo = []
63     respo = []
64     #Pega a quantidade de questoes que voce pediu com as respostas no final
65     while quantN1 > 0:
66         lajota = random.randint(0, len(BancoDeManco)-1)
67         if (BancoDeManco[lajota][0] == 1 and not(BancoDeManco[lajota][1] in provinha) ):
68             pergo.append(BancoDeManco[lajota][1])
69             respo.append(BancoDeManco[lajota][2])
70             quantN1 -= 1
71     while quantN2 > 0:
72         lajota = random.randint(0, len(BancoDeManco)-1)
73         if (BancoDeManco[lajota][0] == 2 and not(BancoDeManco[lajota][1] in provinha) ):
74             pergo.append(BancoDeManco[lajota][1])
75             respo.append(BancoDeManco[lajota][2])
76             quantN2 -= 1
77     while quantN3 > 0:
78         lajota = random.randint(0, len(BancoDeManco)-1)
79         if (BancoDeManco[lajota][0] == 3 and not(BancoDeManco[lajota][1] in provinha) ):
80             pergo.append(BancoDeManco[lajota][1])
81             respo.append(BancoDeManco[lajota][2])
82             quantN3 -= 1
83     pergo.append("\n Respostas \n")
84     pergo.append(respo)
85     for mico in pergo:
86         print(mico)

```

Figura 2

### 3.3 Latex

Para gerar um arquivo no formato *Latex*, usamos o modelo `avalição.tex`, como visto na Figura 3. No nosso arquivo, onde se encontra `%(question1)s`, será colocado no lugar o elemento na posição `[1][0]` da lista `provinha`. E isso é feito, sucessivamente até a `%(question10)s`. Essa mudança, é feita usando o dicionário `replace_with` (Figura 4).

```

\begin{document}

\nomeUniversidade{%(university)s}
\logoUniversidade{fig/ufpr}
\escalaLogoUniversidade{0.25}
\nomeCurso{Departamento de Inform\atica - DInf\Avalia\c{c}\~ao Individual 3}
\nomeProfessor{Jackson Antonio do Prado Lima}
\nomeDisciplina{%(class)s}
\dataDaProva{%(date)s}
\siglaRegistroAcademico{GRR}

\info

\begin{multicols*}{2}
\begin{questions}
\question
%(question1)s
\question
%(question2)s
\question
%(question3)s
\question
%(question4)s
\question
%(question5)s
\question
%(question6)s
\question
%(question7)s
\question
%(question8)s
\question
%(question9)s
\question
%(question10)s
\end{questions}
\end{multicols*}
\end{document}

```

Figura 3

```

27 content = r'''\documentclass{article}
28 \usepackage{utf8}{inputenc}
29 \usepackage[T1]{fontenc}
30 \begin{document}
31 ... P \& B
32 \textbf{\huge %(school)s \\\}
33 \vspace{1cm}
34 \textbf{\Large %(title)s \\\}
35 ...
36 \end{document}
37 '''
38 # Usamos um modelo
39 with open("avaliacao.tex", "r", encoding="utf-8") as f:
40     content = f.read()
41 # trocamos no latex os valores
42 replace_with = {'university': Insti, 'class': "Cálculo 1 ",
43                 'date': Dies, 'question1': provinha[0][0],
44                 'question2': (provinha[1][0]),
45                 'question3': provinha[2][0],
46                 'question4': (provinha[3][0]),
47                 'question5': (provinha[4][0]),
48                 'question6': (provinha[5][0]),
49                 'question7': (provinha[6][0]),
50                 'question8': (provinha[7][0]),
51                 'question9': (provinha[8][0]),
52                 'question10': provinha[9][0]}
53
54 print(content % replace_with)

```

Figura 4



```

27 content = r'''\documentclass{article}
28 \usepackage[utf8]{inputenc}
29 \usepackage[T1]{fontenc}
30 \begin{document}
31 ... P \& B
32 \textbf{\huge %(school)s \\\
33 \vspace{1cm}
34 \textbf{\Large %(title)s \\\
35 ...
36 \end{document}
37 '''
38 # Usamos um modelo
39 with open("avaliacao.tex", "r") as f:
40     content = f.read()
41 # trocamos no latex os valores
42 replace_with = {'university': Insti, 'class': "Cálculo 1 ",
43                 'date': Dias, 'question1': provinha[0], 'question2': provinha[1], 'question3': provinha[2], 'question4': provinha[3], 'question5': provinha[4]}
44 print(content % replace_with)
45 # Criamos um arquivo novo com o resultado da mudança
46 with open('cover.tex', 'w') as f:
47     f.write(content % replace_with)
48 # compilamos o arquivo novo (preservando o modelo original)ss
49 cmd = ['pdflatex', '-interaction', 'nonstopmode', 'cover.tex']
50 proc = subprocess.Popen(cmd)
51 proc.communicate()
52 retcode = proc.returncode
53 if not retcode == 0:
54     os.unlink('cover.pdf')
55     raise ValueError('Error {} executing command: {}'.format(retcode, ' '.join(cmd)))
56 os.unlink('cover.tex')
57 os.unlink('cover.log')

```

Figura 5

### 3.4 Leitura dos dados

Inicialmente, todas as variáveis exceto a `cracudoBarbudo` possuem o valor zero, e os prints na Figura 6 são feitos para melhorar a visualização do usuário na tela de comando. Para o utente, são dadas duas opções: gerar o pdf da prova somente com os seus enunciados ou gerar esta com as suas resoluções. Enquanto o preceptor não digitar alguma das opções válidas, será exibida a seguinte mensagem : “Opção inválida! Digite a opção novamente.”

```

121 cracudoBarbudo = 666
122 N1 = 0
123 N2 = 0
124 N3 = 0
125 quantN1 = 0
126 quantN2 = 0
127 quantN3 = 0
128 print("\n"*1000)
129 print("""
130 LE PARDONNÉ LA PROVA :
131
132
133 THE WHOLE EDITION
134
135 """)
136 print("-"*20)
137 print("\n Olá, \n Das opções abaixo, quais você deseja para gerar um pdf? \n -Prova \n -Lista \n")
138 print("-"*20)
139 cracudoBarbudo = input("\n")
140 #menu para selecionar função
141 while (cracudoBarbudo != "Prova") and (cracudoBarbudo != "Lista") and (cracudoBarbudo != "porrada na cara"):
142     print("\n Opção inválida! Digite a opção novamente. \n")
143     cracudoBarbudo = input("")

```

Figura 6

Caso a opção desejada for “Prova”, entraremos no `if` da Figura 7, onde ocorrerá a definição da quantidade de questões de cada nível de dificuldade, para a criação de uma prova com 10 questões.

Com todos os parâmetros definidos, chamamos a função `prova(saiaDeFilo,quantN1,quantN2,quantN3,Insti,frangoCaipira)` e o PDF é gerado. Caso a opção digitada seja “Lista”, chamamos a função `list(saiadeFilo)`.

```

146
147 if (cracudoBarbudo == "Prova"):
148     total=10
149     print("Lembrando que é para ter {} questões.".format(total))
150     print(type(saiaDeFilo))
151     for frango in saiaDeFilo:
152
153
154         if frango[0] == [1]:
155             N1 +=1
156         elif frango[0] == [2]:
157             N2 +=1
158         elif frango[0] == [3]:
159             N3 +=1
160     while(total!=0):
161         quantN1 = int(input("Quantas gg izi guidorrizi?\n (No maximo {} questões desse nivel\n 0/10) ".format(N1)))
162         while quantN1 > N1 or quantN1 < 0:
163             quantN1 = int(input(" \n Caralho mano, não é difícil escolher um numero inteiro entre 0 e {} ! \n".format(N1)))
164         quantN2 = int(input("Quantas não tão retardada ?\n (No maximo {} questões desse nivel\n {}/10) ".format(N2,quantN1)))
165         while quantN2 > N2 or quantN2 < 0:
166             quantN2 = int(input(" \n Caralho mano, não é difícil escolher um numero inteiro entre 0 e {} ! \n".format(N2)))
167         quantN3 = int(input("Quantas pra fazer a galera do truco chorar?\n (No maximo {} questões desse nivel\n {}/10) ".format(N3,quantN1+quantN2)))
168         while quantN3 > N3 or quantN3 < 0:
169             quantN3 = int(input(" \n Caralho mano, não é difícil escolher um numero inteiro entre 0 e {} ! \n".format(N3)))
170         if (total == (quantN1+quantN2+quantN3)):
171             total = 0
172         else:
173             print("Poxa meu jovem, eu falei que é pra ter {}, isso não é nem {}, nem {}, {} entendeu ?".format(total,total-1,total+1,total))
174     Insti = input("\nQualé o nome daquela joça em que você da aula mesmo?\n")
175     frangoCaipira = input("\nQue dia que é pra ser essa prova?\n")
176     prova(saiaDeFilo,quantN1,quantN2,quantN3,Insti,frangoCaipira)

```

Figura 7

## 4.Conclusão

Inicialmente, a ideia de criar um código para fazer o que nós nos propomos a realizar nesse trabalho nem passou pela nossa cabeça. Era somente uma mera

brincadeira de estudar para as provas, debater os assuntos e efetuar as resoluções de quantos exercícios conseguíssemos para, mais tarde, disponibilizá-las na internet a fim de auxiliar a outros estudantes de Cálculo. Porém, com o decorrer do semestre, chegou o momento de começar o tão badalado trabalho de programação. O que fazer? Bom, honestamente, de pronto ficamos pensando em tudo e ao mesmo tempo em nada. Mas então veio aquela ideia... por que não utilizar um projeto já em andamento para o trabalho? Acabou que todos concordaram de imediato com a ideia, afinal, não parecia algo chato e também poderíamos empregar um banco de questões que já possuíamos no papel.

Laconicamente, acabou que a construção desse trabalho ofertou alguns momentos de cefaleia, como era de se esperar e, ainda que, no final, não tenhamos conseguido exibir de maneira adequada o PDF ao usuário, a proposta do trabalho possibilitou uma experiência bastante enriquecedora, com a procura por conhecimentos que não tínhamos previamente, tal como *Latex* e dicionários, por exemplo, contudo, a ideia de que estávamos adquirindo ferramentas úteis e que nosso programa poderia ajudar outras pessoas, sempre fomentou muitíssimo a continuidade de sua elaboração.

Por fim, esperamos que nosso trabalho e os ideais sob os quais ele foi arquitetado sirvam de inspiração e sejam proveitosos a outros que, como nós, amam divulgar, estudar, debater e ajudar outros a aprender ciência.