Universidade do Estado do Rio de Janeiro
Instituto Politécnico
Departamento de Modelagem Computacional

Relatório do Trabalho Computacional I

Calculadora Financeira

Matemática Discreta I

Equipe
Gabriel Calheias
Katarine Melo
Vitor Saraiva

Nova Friburgo - RJ
Junho de 2019

Resumo

Para a compra de bens de consumo com um valor relativamente alto, ampla gama da população recorre ao uso de instituições financeiras, que realizam financiamento a curto, médio ou longo prazo e para o calculo desse financiamento utiliza-se uma calculadora financeira, que é o tema desse relatório.

Neste relatório procura-se apresentar a construção do projeto de calculadora financeira citado anteriormente. A calculadora foi produzida utilizando a linguagem de programação Python 2.7 e também foram utilizados os cálculos dos sistemas de amortização SAC (Sistema de Amortização Constante) e Price (Sistema Francês de Amortização) para o cálculo dos valores de prestação, juros, etc.

Palavras-chave: Calculadora Financeira, Python, Matemática Discreta, SAC, Price.

Abstract

For the purchase of consumer goods with a relatively high value, a wide range of the population uses the use of financial institutions, which carry out short-, medium- or long-term financing and to calculate this financing, a financial calculator is used. the subject of this report.

This report seeks to present the construction of the financial calculator project mentioned above. The calculator was produced using the programming language Python 2.7 and also the calculations of the systems of amortization SAC (System of Constant Amortization) and Price (French System of Amortization) were used to calculate the values of installment, interest, etc.

Keywords: Financial Calculator, Python, Discrete Mathematics, SAC, Price.

ÍNDICE

1 Introdução.....5
2 Desenvolvimento.....6
3 Fundamentação Teórica.....9
4 Considerações Finais.....17
Referências Bibliográficas.....18

1 Introdução

Neste relatório, são apresentados e descritos os resultados do trabalho computacional da disciplina de matemática discreta 1, ministrada pelo professor Francisco Duarte.

A proposta do trabalho é fazer uma calculadora financeira na linguagem de programação Python 2.7 ou Python 3 com uma interface gráfica em que o usuário entre com o valor, o número de parcelas e a porcentagem de juros e o programa gere uma tabela de amortização SAC ou Price, dependendo da opção escolhida.

Este documento foi estruturado como segue. Nesta seção, foram os pontos básicos do trabalho, A Seção 2 contém os métodos utilizados para a confecção do programa e Na Seção 3, está a explicação de como funciona o código do programa e as funcionalidades do programa.

2 Desenvolvimento

Na confecção do programa foi utilizada a linguagem de programação Python 2.7 juntamente com as bibliotecas nativas do python chamadas Tkinter e SQLite 3. A biblioteca Tkinter foi usada para a confecção da interface gráfica e a biblioteca SQLite3 foi usada na criação de um banco de dados para o programa. Outros comandos nativos de python também foram utilizados tais como:

round(number, digits): esse comando arrendonda o parâmetro "number" em tantas casas decimais definidas no comando no lugar de "digits".

Python

for variável in lista: O laço for é uma estrutura de repetição que nos permite percorrer os itens de uma coleção e, para cada um deles, executar o bloco de código declarado no loop.

if condição: O laço if é uma estrutura de condição que permite avaliar uma expressão e, de acordo com seu resultado, executar uma determinada ação.

E vários outros comandos nativos de python e entre esses os mais utilizados foram os citados anteriormente e declaração e funções e variáveis.

Ex.: Definição de Funções

```
def nome_da_função(retorno_da_função):
    comandos
```

Ex.: Definição de Variáveis

```
v = 9.5  #Variável do tipo float (real)
i = 10   #Variável do tipo int (inteiro)
nome = "Francisco" #Variável do tipo string (conjunto de caracteres)
```

Os tipos de variáveis em python são: int, float, bool, string e complex.

Os utilizados no programa foram int, float e string.

Biblioteca Tkinter

Programas escritos usando a Tkinter são portáveis livremente entre Linux, Unix, Windows e Mac, além da garantia de que qualquer um poderá executar o programa sem precisar instalar bibliotecas extras no Windows.

Os comandos utilizados da biblioteca tkinter foram: Label, Entry, Button, ScrollBar, ListBox, Frame, entre outros.

Frame: é o quadro do programa, onde podemos organizar e dispor os outros widgets.

Label: é um widget que mostra texto ou imagens na interface gráfica, o usuário pode somente visualizar os textos e imagens da label não pode interagir com eles.

Button: diferente do frame e do label o button foi feito para a interação do usuário, no caso ele seria pressionado pelo usuário e ele executaria uma ação ou comando. Como o label ele também mostra um texto.

Entry: apresenta ao usuário um campo de texto de uma única linha para que o usuário possa digitar uma variável tipo string.

Listbox: mostra itens em forma de texto em uma única linha.

Scrollbar: ajuda o usuário a ver outras partes de outros widgets, como os que são muito grandes para aparecerem na tela.

Widget.grid(row='valor1',column='valor2'): posiciona os widgets criados nas respectivas linhas (row) e colunas (column) indicadas pelos valores 'valor 1' e 'valor 2'.

Tabela SAC - Sistema de Amortização Constante

1. Calcular a amortização, que será igual ao saldo devedor (SD) do período "0" – em outras palavras, o valor que foi financiado – dividido pelo número de parcelas:

$$Amort = \frac{SD_0}{N}$$

2. Calcular os juros em termos monetários, que será a taxa de juros multiplicada pelo saldo devedor no período anterior:

$$J = SD_{n-1} * i$$

3. Calcular a parcela (ou prestação), que será simplesmente a soma dos juros com a amortização:

$$Parcela = J_n + Amort_n$$

Tabela Price – Sistema Francês de Amortização

1. Calcular a parcela (ou prestação), que será simplesmente a soma dos juros com a amortização:

$$Parcela = PMT = P * \frac{(1+i)^n * i)}{(1+i)^n - 1)}$$

2. Calcular os juros em termos monetários, que será a taxa de juros multiplicada pelo saldo devedor no período anterior:

$$J = SD_{n-1} * i$$

3. Por fim, calcular a amortização, que será igual ao valor da prestação menos os juros do período:

$$Amort = PMT - J_n$$

2 Fundamentação Teórica

Nesta seção será explicado o programa e como ele funciona.

```
 9 ### Importações ###
10 from Tkinter import *
11 import sqlite3
12 import tkMessageBox
```

Nas linhas de 9 a 12 do programa foram chamadas as bibliotecas que serão utilizadas no código.

```
14 ### Globais ###
15 conn = sqlite3.connect("cadastros.db")
16 cursor = conn.cursor()
17
18 def criarTabela():
19     cursor.execute("""
20         CREATE TABLE IF NOT EXISTS cadastros (
21             N INTEGER NOT NULL PRIMARY KEY,
22             nome TEXT NOT NULL,
23             valor REAL NOT NULL,
24             parcelas INTEGER NOT NULL,
25             juros REAL NOT NULL
26         );
27     """)
28
29 criarTabela()
30 r=2
```

Nas linhas de 14 a 30 foram feitas as declarações globais: que serão usadas durante o código várias vezes. Linhas 15 e 16 foram chamados comandos da biblioteca SQLite3 para a criação de um banco de dados, Linhas 18 a 27 utilizando a mesma biblioteca foi criada um tabela no arquivo "cadastros.db" com os parâmetros citados e na linha 30 foi definida uma variável r com valor 2 que será utilizada nos comandos round, que serão utilizados várias no decorrer do código.

```
32 #### Definições da Aplicação Principal ###
33 principal = Tk()
34 principal.title("Calculadora Financeira - Principal")
35 principal.resizable(FALSE, FALSE)
36 principal.configure(background='white')
37 principal['borderwidth'] = 10
38 principal['relief'] = 'solid'
```

Nas linhas de 32 a 38 foram feitas as configurações da janela do programa. Na linha 35 é a configuração para a janela não poder ser aumentada nem na horizontal nem na vertical.

```
41 def limpar():
42     etN.delete(0, 'end')
43     etNome.delete(0, 'end')
44     etValor.delete(0, 'end')
45     etParcelas.delete(0, 'end')
46     etJuros.delete(0, 'end')
```

Nas linhas de 41 a 46 foi feita uma função para limpar os dados que forem digitados nos campos de texto Entry.

```
48 def adicionar_cadastro():
49     N = etN.get()
50     nome = etNome.get()
51     valor = etValor.get()
52     parcelas = etParcelas.get()
53     juros = etJuros.get()
54     cursor.execute("""
55         INSERT INTO cadastros (N,nome ,valor, parcelas,
56         juros) VALUES (?, ?, ?, ?, ?)""",
57         (N, nome, valor, parcelas, juros))
58     conn.commit()
59     lstCadastros.insert(END, (N, nome, valor, parcelas, juros))
```

Nas linhas de 48 a 59 foi feita uma função para salvar no banco de dados (linhas 54 a 58) e no widget Listbox (linha 59) os dados digitados nas caixas de texto Entry.

```
61 def carregarvalores():
62     etNome.delete(0, 'end')
63     etValor.delete(0, 'end')
64     etParcelas.delete(0, 'end')
65     etJuros.delete(0, 'end')
66     N_cadastro = etN.get()
67     for row in cursor.execute("""SELECT * FROM cadastros WHERE N=?""",
68     (N_cadastro,)):
69         etNome.insert(10, row[1])
70         etValor.insert(10, row[2])
71         etParcelas.insert(10, row[3])
72         etJuros.insert(10, row[4])
```

```
74 def deletecadastro():
75     newwin = Toplevel(principal)
76     newwin.resizable(FALSE, FALSE)
77     newwin.configure(background='white')
78     newwin.title("Calculadora Financeira - Deletar")
79     newwin['borderwidth'] = 10
80     newwin['relief'] = 'solid'
81     ### Widgets - Deletar Cadastro ###
82     lblDeletarCadastro = Label(newwin, text="Deletar Simulação",
83                                font="Arial 12 bold",background='white')
84     lblNDeletar = Label(newwin, text="Nº da Simulação: ",font="Arial 10 bold",
85                         background='white')
86     etNDeletar = Entry(newwin, width=30,fg='blue',background='lightgrey')
87     def deletar_cadastro():
88         N_cadastro = etNDeletar.get()
89         cursor.execute("""
90             DELETE FROM cadastros WHERE N=?""", (N_cadastro,))
91         conn.commit()
92         lstCadastros.delete(0, END)
93         lista = cursor.execute("""
94             SELECT * FROM cadastros;
95             """)
96         for i in lista:
97             lstCadastros.insert(END, i)
98     btnDel = Button(newwin, text="Deletar",font="Arial 10 bold",
99                     command=deletar_cadastro)
100    ### Posicionamento de Widgets - Deletar Cadastro ###
101    lblDeletarCadastro.grid(row=0, column=0,columnspan=2)
102    lblNDeletar.grid(row=1, column=0)
103    etNDeletar.grid(row=1, column=1)
104    btnDel.grid(row=2, column=1)
```

Nas linhas 61 a 72 foi feita uma função para carregar os valores salvos no banco de dados (Linhas 67 a 72 um laço de repetição for varre o banco de dados e coloca os dados da linha com N igual ao digitado nas respectivas caixas de texto Entry).Nas linhas 74 a 104 foi feita uma função que remove o cadastro salvo do banco de dados (linhas 87 a 95) e do widget Listbox (linhas 96 e 87).Na linha 75 foi usado o comando Toplevel para criar uma nova janela na frente da janela principal do programa, nas linhas de 76 a 80 foram configuradas as características dessa nova janela e nas linhas 100 a 104 foram posicionados os widgets criados na nova janela.

O botão btnDel tem o papel de quando clicado, ele chama a função deletar_cadastro definida na linha 87, em forma de comando como marcado em amarelo.

```
106 def alterarvalores():
107     newwin = Toplevel(principal)
108     newwin.resizable(FALSE, FALSE)
109     newwin.configure(background='white')
110     newwin.title("Calculadora Financeira - Alterar")
111     newwin['borderwidth'] = 10
112     newwin['relief'] = 'solid'
113     ### Widgets - Alterar Valores ###
114     lblMudarValores = Label(newwin, text="Alterar Simulação",
115                             font="Arial 12 bold",background='white')
116     lblNMudar = Label(newwin, text="Nº da Simulação: ",font="Arial 10 bold",
117                       background='white')
118     lblNovoNome = Label(newwin, text="Novo Nome:",font="Arial 10 bold",
119                         background='white')
120     lblNovaValor = Label(newwin, text="Novo Valor: ",font="Arial 10 bold",
121                          background='white')
122     lblNovaParcela = Label(newwin, text="Novo Nº de Parcelas: ",
123                            font="Arial 10 bold",background='white')
124     lblNovoJuros = Label(newwin, text="Novo Juros(% a.m.):",
125                          font="Arial 10 bold",background='white')
126     etNMudar = Entry(newwin,fg='blue',background='lightgrey',width = 30)
127     etNovoNome = Entry(newwin,fg='blue',background='lightgrey',width = 30)
128     etNovoValor = Entry(newwin,fg='blue',background='lightgrey',width = 30)
129     etNovaParcela = Entry(newwin,fg='blue',background='lightgrey',width = 30)
130     etNovoJuros = Entry(newwin,fg='blue',background='lightgrey',width = 30)
131     def mudar_valores():
132         N_cadastro = etNMudar.get()
133         novo_nome = etNovoNome.get()
134         novo_valor = etNovoValor.get()
135         nova_parcela = etNovaParcela.get()
136         novo_juros = etNovoJuros.get()
137         if (novo_valor=="" or nova_parcela=="" or novo_juros=="" or novo_nome==""):
138             tkMessageBox.showerror("Erro!", "Algum Espaço Não Foi Preenchido")
139         else:
140             cursor.execute("""
141                 UPDATE cadastros SET nome = ? WHERE N = ?""",
142                 (novo_nome, N_cadastro))
143             cursor.execute("""
144                 UPDATE cadastros SET valor = ? WHERE N = ?""",
145                 (novo_valor, N_cadastro))
146             cursor.execute("""
147                 UPDATE cadastros SET parcelas = ? WHERE N = ?""",
148                 (nova_parcela, N_cadastro))
149             cursor.execute("""
150                 UPDATE cadastros SET juros = ? WHERE N = ?""",
151                 (novo_juros, N_cadastro))
152             conn.commit()
153             lstCadastros.delete(0, END)
154             lista = cursor.execute("""
155                 SELECT * FROM cadastros;
156                 """)
157             for i in lista:
158                 lstCadastros.insert(END, i)
159     btnMudarValor = Button(newwin, text="Alterar Simulação",
160                            command=mudar_valores,font="Arial 10 bold")
```

Nas linhas 106 a 160 foi criada uma função que igual a anterior cria uma nova janela na frente da principal para alterar os valores salvos.

```
161     ### Posicionamento de Widgets - Alterar Valores ###
162     lblMudarValores.grid(row=0, column=0,columnspan=2)
163     lblNMudar.grid(row=1, column=0)
164     etNMudar.grid(row=1, column=1)
165     lblNovoNome.grid(row=2, column=0)
166     etNovoNome.grid(row=2, column=1)
167     lblNovaValor.grid(row=3, column=0)
168     etNovoValor.grid(row=3, column=1)
169     lblNovaParcela.grid(row=4, column=0)
170     etNovaParcela.grid(row=4, column=1)
171     lblNovoJuros.grid(row=5, column=0)
172     etNovoJuros.grid(row=5, column=1)
173     btnMudarValor.grid(row=6, column=1)
```

Nas linhas de 161 a 173 foram posicionados os widgets da função alterarvalores.

Nas linhas 176 e 248 do código fonte do programa foram criadas também mais 2 funções que fazem os cálculos para gerar as tabelas SAC e Price. Essas funções fazem os cálculos e para gerar a janela com a tabela foi utilizado um laço de repetição for com Label mostrado abaixo:

```
for c in range(0, n):
    j=i*sd
    pmt=a+j
    sd=sd-a
    if sd<=0:
        sd=0
    tk.Label(frame,text=str(c+1),font="Arial 10 bold",fg="blue",
             background='darkgrey',borderwidth="1",relief="solid").grid(row=c+4,column=0)
    tk.Label(frame,text="R$ "+str(round(pmt,r)),font="Arial 10",
             fg="darkgreen",borderwidth="1",relief="solid").grid(row=c+4,column=1)
    tk.Label(frame,text="R$ "+str(round(a,r)),font="Arial 10",
             fg="darkgreen",borderwidth="1",relief="solid").grid(row=c+4,column=2)
    tk.Label(frame,text="R$ "+str(round(j,r)),font="Arial 10",
             fg="darkgreen",borderwidth="1",relief="solid").grid(row=c+4,column=3)
    tk.Label(frame,text="R$ "+str(round(sd,r)),font="Arial 10",
             fg="red",borderwidth="1",relief="solid").grid(row=c+4,column=4)
```

Esse mostrado foi o utilizado para a tabela SAC mas o utilizado para a tabela Price é similar, somente trocando a forma de fazer os cálculos de amortização, juros e parcelas.

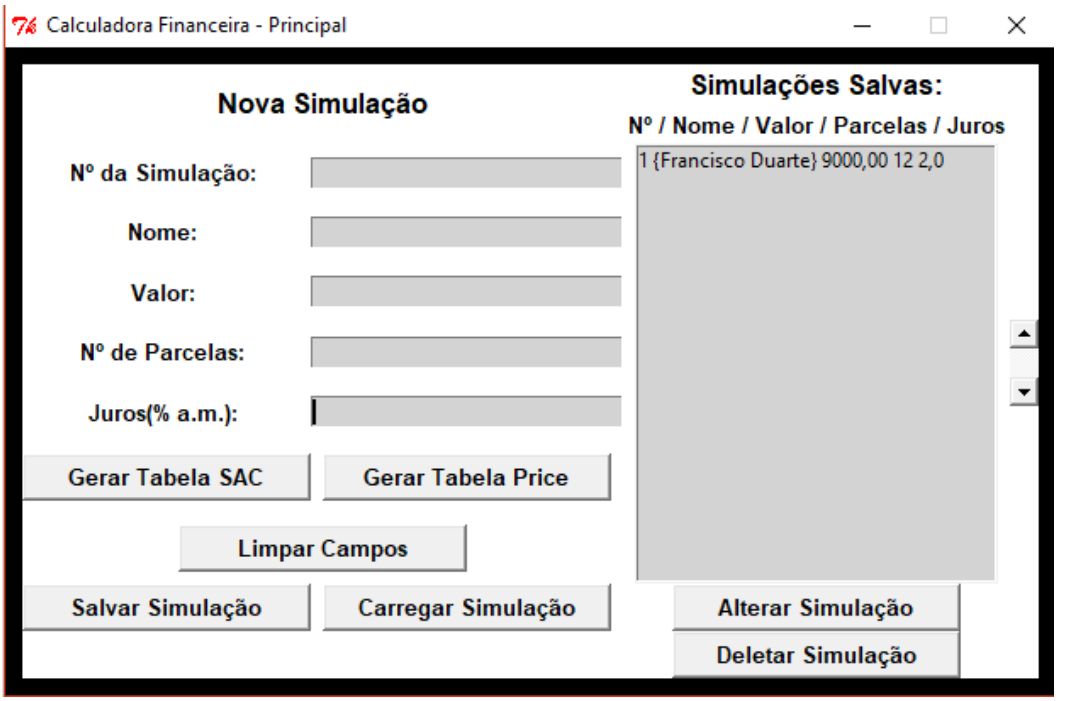
Para coletar os dados que o usuário digita foi utilizado o seguinte comando:

```
v = float(etValor.get().replace(',', '.'))
i = float(etJuros.get().replace(',', '.'))
n = int(etParcelas.get())
nome = etNome.get()
```

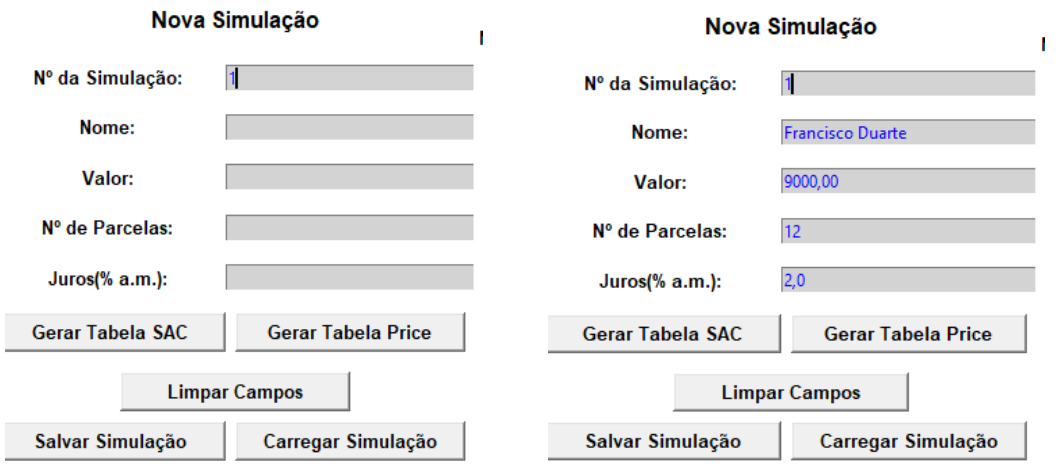
float para converter o valor digitado de texto para real e int para converter para inteiro. O comando que coleta os valores é .get() e o comando .replace(',', '.') substitui vírgula por ponto pois o código representa os números com casas decimais utilizando ponto para separar o número de suas casas decimais no lugar da vírgula.

Guia de Utilização

Aqui será explicado o passo a passo de utilização do programa, abaixo sera mostrada uma imagem da janela principal do programa:

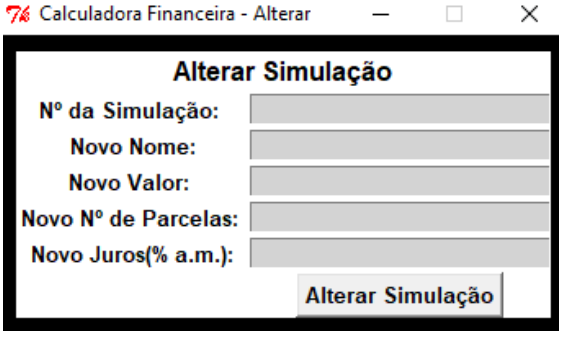


A parte em cinza escuro maior é o Listbox e as menores Entrys, os textos que não estão em retângulos são os Labels e os dentro de retângulos cinza mais claros são Buttons. O programa funciona da seguinte forma, o usuário digita nos Entrys os dados correspondentes e se ele quiser salvar a simulação clica no botão “Salvar Simulação” e ela sera salva no banco de dados e no Listbox para consultas futuras como mostrado no Listbox da imagem acima.



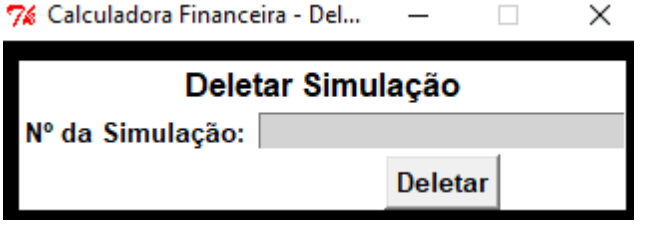
Se em "Nº de Simulação" for digitado o número da simulação salva e for clicado em "Carregar Simulação", os valores salvos irão preencher os Entrys como mostrado acima.

Ao se clicar no botão alterar simulação abriria a seguinte janela:



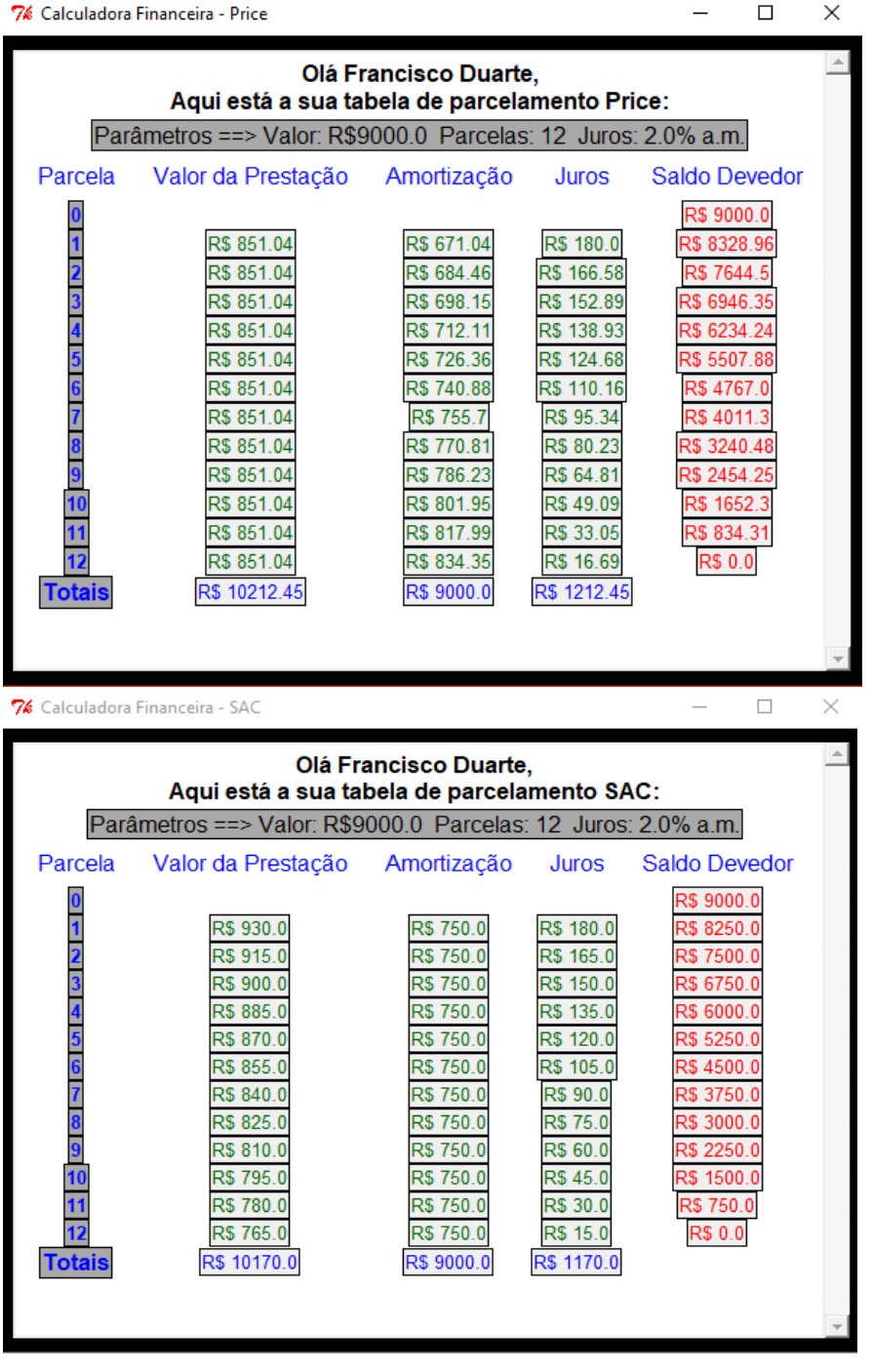
Nessa janela o usuário digitaria os novos valores a serem salvos na simulação a qual foi digitada o número.

Ao se clicar no botão deletar simulação abriria a seguinte janela:



Nessa janela só necessário digitar o número da simulação a ser deletada e clicar no botão.

Ao se clicar nos botões Gerar Tabela SAC ou Gerar Tabela Price seria aberta uma nova janela com a tabela escolhida, segue as tabelas geradas dos dois sistemas de amortização para os valores de 9000 reais 2% de juros e 12 parcelas:



3 Considerações Finais

Na confecção do trabalho foram utilizados três fatores que geraram novos conhecimentos que seria: Linguagem de programação Python e confecção de interface gráfica em Python.

Python é uma linguagem de programação de alto nível e interpretada, que é bem útil para resolução de vários problemas. Sua estrutura é bem simples, exatamente por ser uma linguagem de alto nível e interpretada, ela é bem intuitiva e fácil de se entender.

Tkinter, uma biblioteca nativa do Python e a utilizada para a confecção da interface gráfica, é um conhecimento muito bom para se ter, principalmente por se tratar de interface gráfica. Durante a confecção do programa também foram achadas outras formas de se fazer uma interface gráfica para um programa em Python, uma delas foi utilizando a linguagem Kivy que é bem útil pois pode também ser utilizada para a confecção de uma interface gráfica para um aplicativo de smartphone.

Referências Bibliográficas

https://www.w3schools.com/python/ref_func_round.asp
https://docs.python.org/2/library/functions.html
https://www.devmedia.com.br/for-python-estrutura-de-repeticao-for/38513
https://www.devmedia.com.br/python-estrutura-condicional-if-else/38217
https://pt.wikibooks.org/wiki/Python/Conceitos_b%C3%A1sicos/Tipos_e_operadores
https://tkdocs.com/tutorial/widgets.html
https://www.wrprates.com/o-que-e-tabela-sac-sistema-de-amortizacao-constante/
https://www.wrprates.com/o-que-e-tabela-price-sistema-frances/