**ATIVIDADE PRÁTICA   
LINGUAGEM DE PROGRAMAÇÃO**

**GUILHERME GIOVANNI VIEIRA LOURENÇO – 3799879  
PROF. WELLINGTON RODRIGO MONTEIRO**

JABOTICABAL  
2022

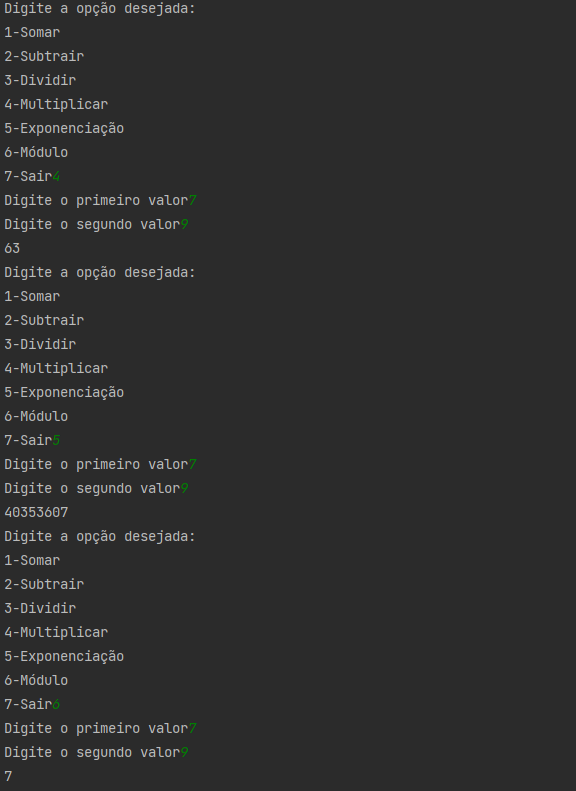
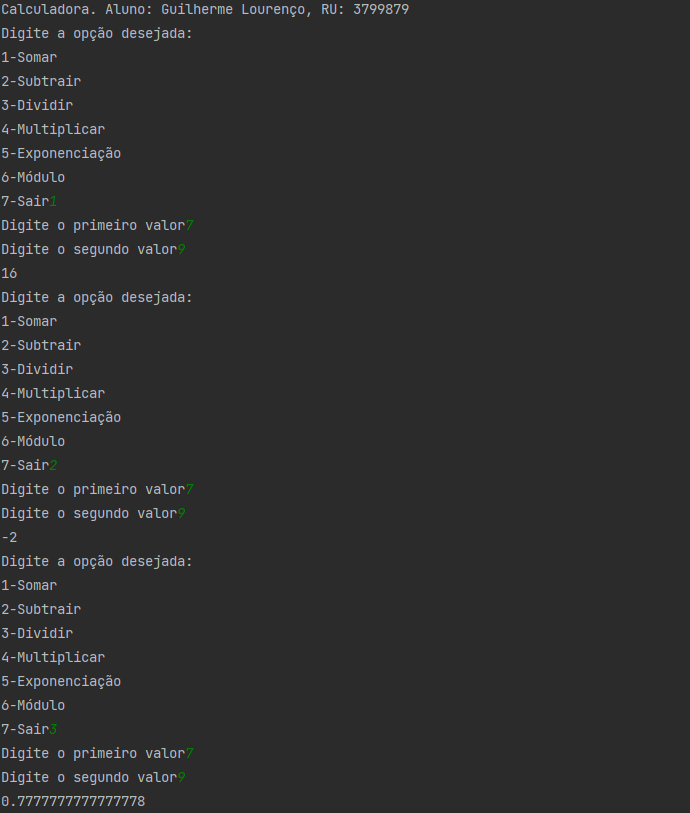
# EXERCÍCIOS A SEREM SOLUCIONADOS

1. Desenvolver a **classe calculadora** que faça qualquer operação matemática utilizando dois números inteiros, sendo os **dois últimos números de seu RU**. Caso o RU algum destes números sejam **zero**, **substituí-lo(s) pelo número 1**. Sendo as possíveis operações matemáticas: **soma(+), subtração(-), multiplicação(\*), divisão(/), exponenciação(^) e módulo(%)**. Além destas funcionalidades, o algoritmo deverá ter um **MENU** que possíbilite ao usuário escolher qual o tipo de operação que se deseja realizar e que possibilite ao usuário a digitar os dois números. *Apresentar todas as operações matemáticas da calculadora funcionando!*

**SOLUÇÃO:**

class calculadora: #criando a classe  
 #definindo funções de cada operação  
 def \_\_init\_\_(self, a, b) -> None: #método construtor  
 self.a = a  
 self.b = b  
 def soma(self):  
 return self.a + self.b  
 def subtracao(self):  
 return self.a - self.b  
 def divisao (self):  
 return self.a / self.b  
 def multiplicacao (self):  
 return self.a \* self.b  
 def exponenciacao (self):  
 return self.a \*\* self.b  
 def modulo (self):  
 return self.a % self.b  
 #fim da definição das funções  
 #inicio do programa principal  
print("Calculadora. Aluno: Guilherme Lourenço, RU: 3799879") #identificador pessoal  
while True: #loop o funcionamento do menu  
 try:  
 opcao = int(input("Digite a opção desejada:\n1-Somar\n2-Subtrair\n3-Dividir\n4-Multiplicar\n5-Exponenciação\n6-Módulo\n7-Sair"))  
 if opcao == 1:  
 a = int(input("Digite o primeiro valor"))  
 b = int(input("Digite o segundo valor"))  
 operacao = calculadora(a,b)  
 print(operacao.soma())  
 elif opcao == 2:  
 a = int(input("Digite o primeiro valor"))  
 b = int(input("Digite o segundo valor"))  
 operacao = calculadora(a, b)  
 print(operacao.subtracao())  
 elif opcao == 3:  
 a = int(input("Digite o primeiro valor"))  
 b = int(input("Digite o segundo valor"))  
 operacao = calculadora(a,b)  
 print(operacao.divisao())  
 elif opcao == 4:  
 a = int(input("Digite o primeiro valor"))  
 b = int(input("Digite o segundo valor"))  
 operacao = calculadora(a,b)  
 print(operacao.multiplicacao())  
 elif opcao == 5:  
 a = int(input("Digite o primeiro valor"))  
 b = int(input("Digite o segundo valor"))  
 operacao = calculadora(a, b)  
 print(operacao.exponenciacao())  
 elif opcao == 6:  
 a = int(input("Digite o primeiro valor"))  
 b = int(input("Digite o segundo valor"))  
 operacao = calculadora(a, b)  
 print(operacao.modulo())  
 elif opcao == 7:  
 print("Encerrando a calculadora...")  
 break #encerra o loop e fecha o programa  
 else:  
 print("Digite uma opção válida")  
 except ValueError: #tratamento de erro  
 print("Você digitou um valor inválido!\nTente novamente.")

**CONSOLE:**



II. Dada a equação linear **y = ax + bx – c**. Para os valores de **a, b e c** serão os **três últimos números de seu RU**. Caso, **algum número do RU seja igual a zero**, subistituí-lo(s) pelo **número 3**. Realizar o plot dos resultados, dado os valores de **x1 = 5; x2 = 7 e x3 = 9.** Para o plot você precisará utilizar a **biblioteca matplotlib** apresentada na Aula Prática 1; **colocar legenda no gráfico, alterar a cor da linha(s) de seu gráfico; nomear o eixo x, nomear o eixo y.**

*Exemplo:* Meu RU: 205 103 -> por ter zzero fica assim: **a =1; b=3; c=3**

*y = 1\*5 + 3\*5 – 3 y = 1\*9 + 9\*3 – 3*

*y = 5+ 15 - 3 = 17 y = 9 + 27 -3 = 33*

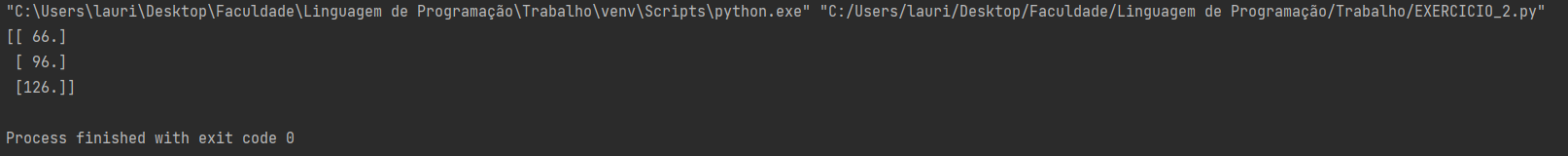
*y = 1\*7 + 7\*3 – 3*

*y = 7 + 21 - 3 = 25*

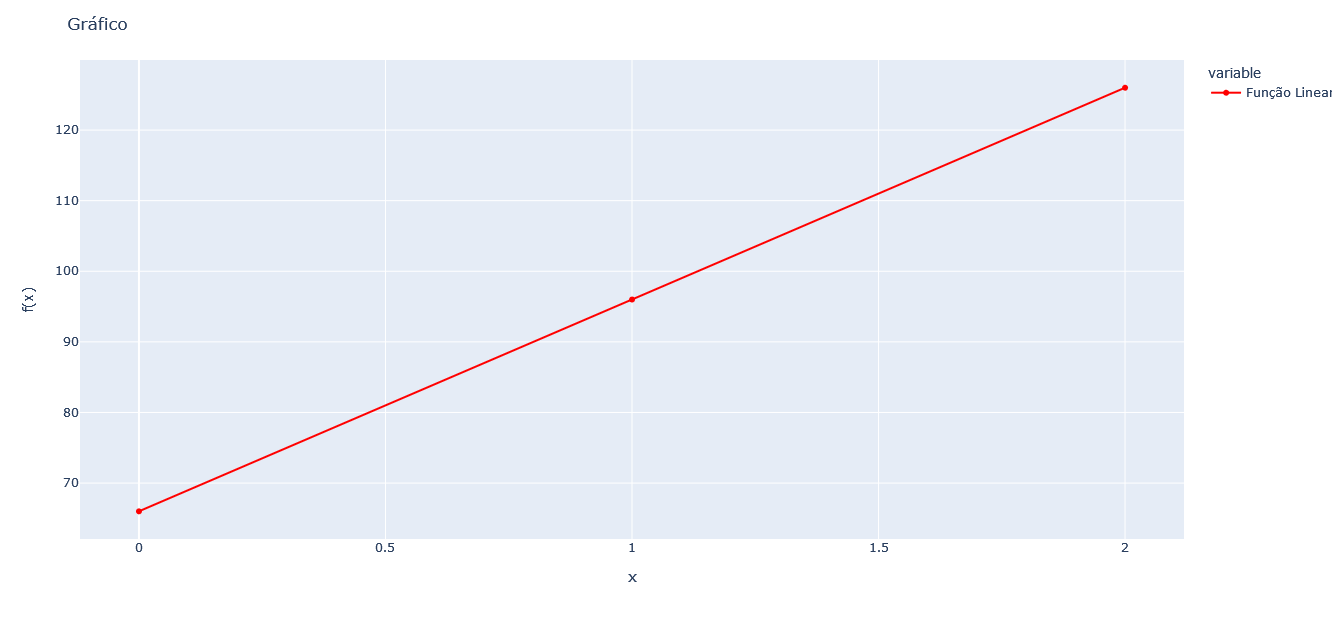
**SOLUÇÃO:**

#importando bibliotecas necessárias  
import plotly.express as px  
import numpy as np  
  
class equacao: #criando a classe  
 def \_\_init\_\_(self, a, b, c) -> None: #método construtor  
 self.a = a  
 self.b = b  
 self.c = c  
 def operacaox1(self):  
 x = 5  
 y = ((self.a \* x) + (self.b \* x)) - self.c  
 return y  
 def operacaox2(self):  
 x = 7  
 y = ((self.a \* x) + (self.b \* x)) - self.c  
 return y  
 def operacaox3(self):  
 x = 9  
 y = ((self.a \* x) + (self.b \* x)) - self.c  
 return y  
#programa prinicpal  
operacao = equacao(8,7,9) #criando variável a partir da classe "equacao" tendo como parâmetro os três últimos dígitos do meu RU  
#criando vetor de cada resultado das operações usando NumPy  
vetor\_x1 = np.array([operacao.operacaox1()])  
vetor\_x2 = np.array([operacao.operacaox2()])  
vetor\_x3 = np.array([operacao.operacaox3()])  
#criando arquivo de texto com os vetores usando NumPy  
np.savetxt('x1\_resultado.txt', vetor\_x1, fmt='%f', delimiter=';')  
np.savetxt('x2\_resultado.txt', vetor\_x2, fmt='%f', delimiter=';')  
np.savetxt('x3\_resultado.txt', vetor\_x3, fmt='%f', delimiter=';')  
#carregando arquivo de texto com os vetores usando NumPy  
array\_x1 = np.loadtxt('x1\_resultado.txt', dtype = np.float64, delimiter=';')  
array\_x2 = np.loadtxt('x2\_resultado.txt', dtype = np.float64, delimiter=';')  
array\_x3 = np.loadtxt('x3\_resultado.txt', dtype = np.float64, delimiter=';')  
  
array\_x123 = np.vstack([array\_x1, array\_x2, array\_x3]) #juntando os valores dos outros três arrays em uma única variável para plotar o gráfico  
print(array\_x123)  
fig = px.line(array\_x123) #plotando gráfico do tipo linha(line)  
fig.update\_layout(title = 'Gráfico', xaxis\_title = 'x', yaxis\_title = 'f(x)') #renomeando o gráfico, eixo x e eixo y  
fig.update\_traces(name = 'Função Linear', line = dict(color = "red"), mode = 'lines+markers') #legendando gráfico, alterando a cor e adicionando marcadores  
fig.show()

**CONSOLE:**

****

**GRÁFICO:**

****

III. **Realizar o upload do arquivo STORES.csv.** Renomear todas as colunas do arquivo STORES.csv, onde os respctivos nomes sejam compactados (Exemplo: **Daily\_Customer\_Count** foi renomeado para **Visitantes**). Após isto, para se analisar o desempenho das lojas de supermercado/mercado do arquivo STORES.csv encontre os valores **mínimo, máximo, médio e desvio padrão** das seguinte colunas: **"Items\_Available"; "Daily\_Customer\_Count"; e "Store\_Sales".**

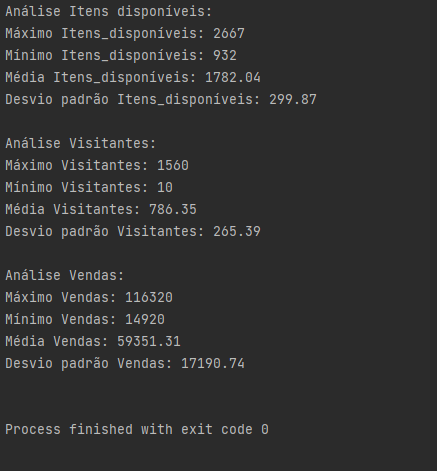
Algumas informações extras sobre a tabela do arquivo **STORES.csv**:

* *ID da loja: (Índice) ID da loja específica.*
* *Store ID: Área Física da loja em pátio.*
* *Store\_Area: Número de itens diferentes disponíveis na loja correspondente.*
* *DailyCustomerCount: Número de clientes que visitaram as lojas em média ao longo do mês.*
* *Store\_Sales: Vendas em (US$) que as lojas realizaram.*

**SOLUÇÃO:**

import pandas as pd #importando a biblioteca pandas  
  
stores = pd.read\_csv('Stores.csv', sep = ',') #lendo arquivo csv com pandas  
stores\_df = pd.DataFrame(stores) #criando um DataFrame  
stores\_df.rename(columns = {'Store ID' : 'ID\_Loja', 'Store\_Area' : 'Área', 'Items\_Available' : 'Itens\_disponíveis', 'Daily\_Customer\_Count' : 'Visitantes', 'Store\_Sales' : 'Vendas(US$)'}, inplace = True) #renomeando colunas  
#inicio das análises das colunas  
print("Análise Itens disponíveis:")  
print("Máximo Itens\_disponíveis: {}" .format(stores\_df['Itens\_disponíveis'].max()))  
print("Mínimo Itens\_disponíveis: {}" .format(stores\_df['Itens\_disponíveis'].min()))  
print("Média Itens\_disponíveis: {:.2f}" .format(stores\_df['Itens\_disponíveis'].mean()))  
print("Desvio padrão Itens\_disponíveis: {:.2f}\n" .format(stores\_df['Itens\_disponíveis'].std()))  
  
print("Análise Visitantes:")  
print("Máximo Visitantes: {}" .format(stores\_df['Visitantes'].max()))  
print("Mínimo Visitantes: {}" .format(stores\_df['Visitantes'].min()))  
print("Média Visitantes: {:.2f}" .format(stores\_df['Visitantes'].mean()))  
print("Desvio padrão Visitantes: {:.2f}\n" .format(stores\_df['Visitantes'].std()))  
  
print("Análise Vendas:")  
print("Máximo Vendas: {}" .format(stores\_df['Vendas(US$)'].max()))  
print("Mínimo Vendas: {}" .format(stores\_df['Vendas(US$)'].min()))  
print("Média Vendas: {:.2f}" .format(stores\_df['Vendas(US$)'].mean()))  
print("Desvio padrão Vendas: {:.2f}\n" .format(stores\_df['Vendas(US$)'].std()))  
#fim das análises das colunas

**CONSOLE:**

****