ATIVIDADE PRÁTICA
LINGUAGEM DE PROGRAMAÇÃO

MOACIR DOMINGOS DA SILVA JUNIOR - RU: 3539252

Prof.ª ME MARIANE G. B. FERNANDES

DIADEMA
2022

EXERCÍCIOS A SEREM SOLUCIONADOS

I. Desenvolver a classe calculadora que faça qualquer operação matemática utilizando dois números inteiros, sendo os dois últimos número de seu **RU**. Caso o RU termine com zero, substituí-lo pelo número 1. Sendo as possíveis operações matemáticas: **soma, subtração, multiplicação, divisão, exponenciação e módulo**. Para isto, o algoritmo deverá ter um **MENU** que possibilite ao usuário escolher qual o tipo de operação que se deseja realizar. *Apresentar todas as operações matemáticas da calculadora funcionando!*

```
#RU: 3539252
class Calculadora():
  def __init__(self, a=5, b=2, opcao=None):
      self.a = a
      self.b = b
      self.opcao = opcao

  def start(self):

    print('''
      [1] Somar
      [2] Subtrair
      [3] Multiplicar
      [4] Dividir
      [5] Exponenciação
      [6] Módulo''')

    self.opcao = int(input())
    self.resultado()

  def somar(self):
    return self.a+self.b

  def subtrair(self):
    return self.a-self.b

  def multiplicacao(self):
    return self.a*self.b

  def dividir(self):
    return self.a/self.b
```

```
def exponenciacao(self):
    return self.a**self.b

 def modulo(self):
    return self.a%self.b

 def resultado(self):
    if self.opcao == 1:
      print(f"Soma = {self.somar()}")
    elif self.opcao == 2:
      print(f"Subtração = {self.subtrair()}")
    elif self.opcao == 3:
      print(f"Multiplicação = {self.multiplicacao()}")
    elif self.opcao == 4:
      print(f"Divisão = {self.dividir()}")
    elif self.opcao == 5:
      print(f"Exponenciação = {self.exponenciacao()}")
    elif self.opcao == 6:
      print(f"Módulo = {self.modulo()}")
    else:
      print("Opção inválida")

 def todos_resultado(self):
    print(f"Soma = {self.somar()}")
    print(f"Subtração = {self.subtrair()}")
    print(f"Multiplicação = {self.multiplicacao()}")
    print(f"Divisão = {self.dividir()}")
    print(f"Exponenciação = {self.exponenciacao()}")
    print(f"Módulo = {self.modulo()}")
```

IMAGEM DO EXERCÍCIO:

```
1 calculadora = Calculadora()
2 calculadora.start()
```

```
...
        [1] Somar
        [2] Subtrair
        [3] Multiplicar
        [4] Dividir
        [5] Exponenciação
        [6] Módulo
```

```
1 calculadora = Calculadora()
2 calculadora.start()
```

```
        [1] Somar
        [2] Subtrair
        [3] Multiplicar
        [4] Dividir
        [5] Exponenciação
        [6] Módulo
1
Soma = 7
```

```
1 calculadora = Calculadora()
2 calculadora.start()
```

```
        [1] Somar
        [2] Subtrair
        [3] Multiplicar
        [4] Dividir
        [5] Exponenciação
        [6] Módulo
4
Divisão = 2.5
```

```
1 calculadora = Calculadora()
2 calculadora.todos_resultado()
```

```
Soma = 7
Subtração = 3
Multiplicao = 10
Divicao = 2.5
Exponenciacao = 25
modulo = 1
```

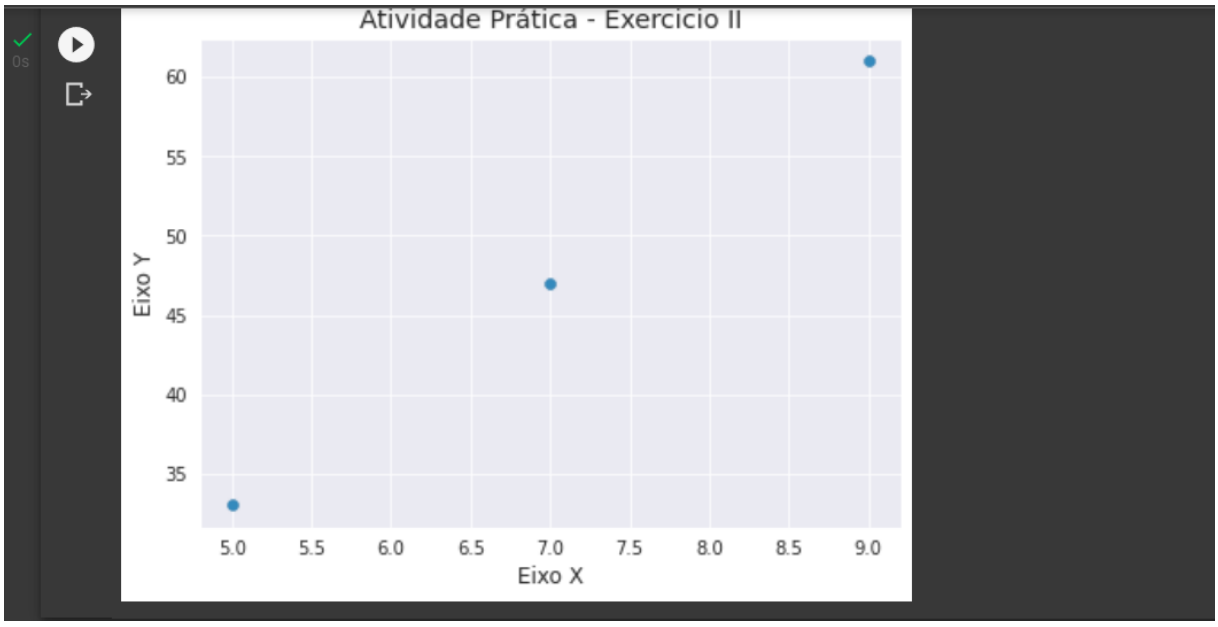
II. Encontre os valores para a variável y, onde y = ax + xb – c. Para os valores de a, b e c serão os três últimos números de seu RU. **Caso, algum número do RU seja igual a zero, subistituí-lo pelo número 3**. Realizar o plot dos resultados onde será **x = 5; x = 7 e x = 9**. Para o plot você precisará utilizar a **biblioteca matplotlib** apresentada em aula, colocar **legenda** no gráfico, **alterar a cor** dos gráficos (*linhas ou pontos*), **nomear o eixo x**, **nomear o eixo y**:

```
import numpy as np
# Meu RU: 3539252
a=2; b=5; c=2
x = 5
y1 = a*x+b*x-c

x = 7
y2 = a*x+b*x-c

x = 9
y3 = a*x+b*x-c

print(y1)
print(y2)
print(y3)
```

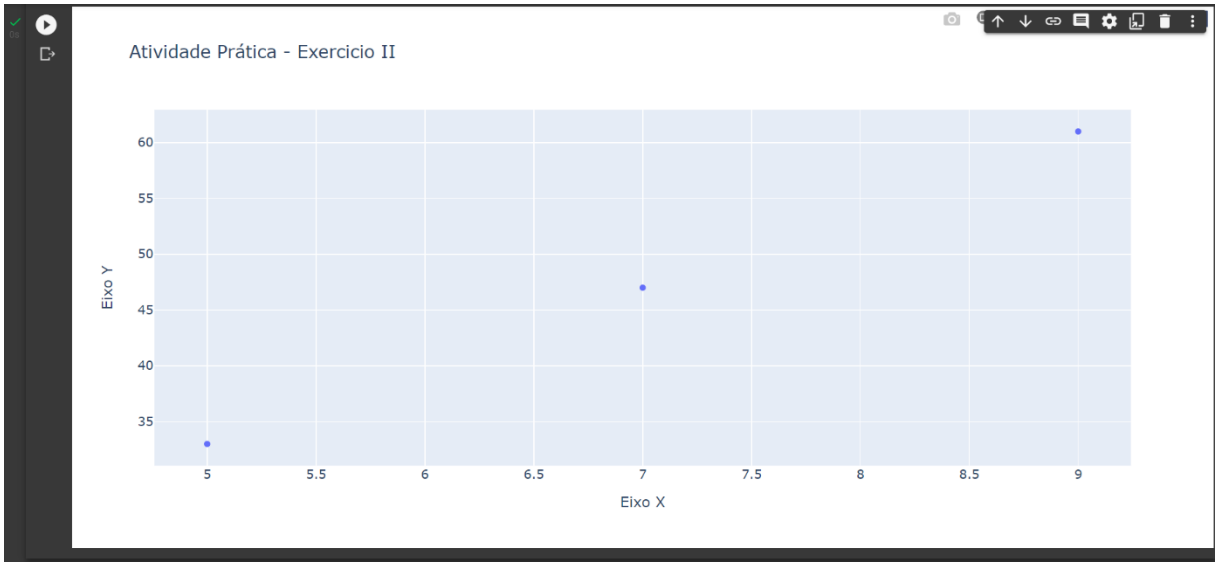
```python
import numpy as np
# Meu RU: 3539252
a=2; b=5; c=2
x = 5
y1 = a*x+b*x-c

x = 7
y2 = a*x+b*x-c

x = 9
y3 = a*x+b*x-c

print(y1)
print(y2)
print(y3)
```

```
33
47
61
```

```python
Eixo_X = [5, 7, 9]
Eixo_Y = [y1, y2, y3]
import matplotlib.pyplot as plt
import seaborn as sns


plt.style.use('ggplot')
plt.figure(figsize=(7,5))

sns.set_style('darkgrid')
sns.scatterplot(x=Eixo_X, y=Eixo_Y)


plt.scatter(x=Eixo_X, y=Eixo_Y)
plt.title('Atividade Prática - Exercicio II')
plt.xlabel('Eixo X')
plt.ylabel('Eixo Y')
```



```
import plotly.express as px

fig = px.scatter(x=Eixo_X, y=Eixo_Y,
 title='Atividade Prática - Exercicio II',
 labels={"x": "Eixo X", "y": "Eixo Y"})
fig.show()
```



III. Realizar o upload do arquivo STORES.csv. Renomear todas as colunas do arquivo STORES.csv, onde os respectivos 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 seguintes 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.*

```
#Conexão com Google Drive:
from google.colab import drive
drive.mount('/content/drive')
```

```
#Conexão com Google Drive:
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
!cp /content/drive/MyDrive/Atividade_Pratica_LP/*.* /content
arquivo = open('Stores.csv')
conteudo = arquivo.read()
import pandas as pd

pddf_csv = pd.read_csv('Stores.csv')
pddf_csv.head()
```

```
pddf_csv = pd.read_csv('Stores.csv')
pddf_csv.head()
```

	Store ID	Store_Area	Items_Available	Daily_Customer_Count	Store_Sales
0	1	1659	1961	530	66490
1	2	1461	1752	210	39820
2	3	1340	1609	720	54010
3	4	1451	1748	620	53730
4	5	1770	2111	450	46620

```
#Editando o título das colunas
pddf_csv.rename({'Store ID ':'Índice Loja','Store_Area':'Área da loja',
                 'Items_Available':'Itens disponíveis',
                 'Daily_Customer_Count':'Visitantes','Store_Sales':'Vendas'},
                 axis=1, inplace=True)
```

```
pddf_csv.head()
```

	Índice Loja	Área da loja	Itens disponíveis	Visitantes	Vendas
0	1	1659	1961	530	66490
1	2	1461	1752	210	39820
2	3	1340	1609	720	54010
3	4	1451	1748	620	53730
4	5	1770	2111	450	46620

```
pddf_csvColunas = [ 'Itens disponíveis', 'Visitantes','Vendas']

pddf_csvFiltrado = pddf_csv.filter(items=pddf_csvColunas)

pddf_csvFiltrado.head()
```

```
1 pddf_csvFiltrado.head()
```

	Itens disponíveis	Visitantes	Vendas
0	1961	530	66490
1	1752	210	39820
2	1609	720	54010
3	1748	620	53730
4	2111	450	46620

```
#valores mínimo
pddf_csvFiltrado.min()
```

```
1 #valores mínimo
2 pddf_csvFiltrado.min()
```

```
Itens disponíveis      932
Visitantes              10
Vendas                14920
dtype: int64
```

```
#valores  máximo
pddf_csvFiltrado.max()
```

```
Itens disponíveis      2667
Visitantes             1560
Vendas               116320
dtype: int64
```

```
#valores  médio
pddf_csvFiltrado.mean()
```

```
Itens disponíveis     1782.035714
Visitantes             786.350446
Vendas               59351.305804
dtype: float64
```

```
#desvio padrão
pddf_csvFiltrado.std()
```

```
Itens disponíveis      299.872053
Visitantes             265.389281
Vendas               17190.741895
dtype: float64
```