
UNIVERSIDADE ESTADUAL DA PARAÍBA
CAMPUS I - CAMPINA GRANDE
PRÓ REITORIA DE PÓS-GRADUAÇÃO E PESQUISA
PROGRAMA DE PÓS-GRADUAÇÃO EM MATEMÁTICA
MESTRADO PROFISSIONAL EM MATEMÁTICA EM REDE NACIONAL

FLÁVIO RÉGIS ALVES JÚNIOR

# CALCULADORAS PYTHON PARA AUXÍLIO NO ENSINO DE MATEMÁTICA NO ENSINO BÁSICO

CAMPINA GRANDE
2026

**FLÁVIO RÉGIS ALVES JÚNIOR**

# CALCULADORAS PYTHON PARA AUXÍLIO DO ENSINO DE MATEMÁTICA NO ENSINO BÁSICO

Trabalho de Conclusão de Curso apresentado ao Corpo Docente do Programa de Pós-Graduação em Matetmática - CCT - UEPB, na modalidade Mestrado Profissional como requisito parcial à obtenção do título de Mestre..

**Área de concentração**: Ensino de Matemática

**Orientador**: Prof. Dr. Israel Buriti Galvão

**CAMPINA GRANDE**
**2026**

## SUMÁRIO

Página

# 1 PRODUTO EDUCACIONAL

Conforme defendido nessa dissertação, a utilização de recursos tecnológicos configura-se como uma estratégia eficaz para o ensino de conteúdos matemáticos de forma contextualizada, promovendo a articulação entre teoria e prática no cotidiano escolar. Também foi defendido a viabilidade da inserção da linguagem de programação Python no ensino básico, tendo em vista tratar-se de uma ferramenta versátil e multifuncional, capaz de potencializar a aprendizagem e estimular o pensamento lógico-matemático dos estudantes.

Com o propósito de atender às demandas contemporâneas por um ensino de Matemática mais interativo e alinhado ao contexto tecnológico, propõe-se, como Produto Educacional, o desenvolvimento de um conjunto de calculadoras elaboradas em linguagem Python. Tal recurso visa não apenas dinamizar o processo de ensino e aprendizagem da Matemática, tornando-o mais atrativo e significativo, mas também contribuir para o desenvolvimento de competências relacionadas ao raciocínio lógico, à resolução de problemas e à iniciação ao pensamento computacional.

O presente produto foi desenvolvido na plataforma Google Colab, a qual se configura como um ambiente computacional disponibilizado pelo Google que possibilita a implementação e execução de algoritmos na linguagem de programação Python. No processo de construção, foram empregados tipos de dados primitivos, tais como string, numéricos e booleanos, bem como tipos de dados estruturados, como listas. Ademais, exploraram-se de maneira sistemática recursos fundamentais da linguagem, notadamente as estruturas de decisão e os laços de repetição, os quais desempenharam papel central na formulação e operacionalização das expressões e fórmulas matemáticas propostas.

O produto resultante consistiu em um conjunto de calculadoras digitais voltadas ao ensino de conteúdos matemáticos, contemplando tópicos como equações quadráticas, sistemas lineares, juros simples, juros compostos, progressões aritméticas, entre outros. Essas ferramentas, ao aliarem a linguagem de programação Python a conceitos matemáticos, configuram-se como instrumentos pedagógicos que podem favorecer o processo de ensino-aprendizagem da Matemática na educação básica, uma vez que permitem ao estudante a realização de cálculos de maneira interativa, dinâmica e contextualizada, promovendo maior engajamento e compreensão conceitual.

# 2 Calculadoras Python

Aqui serão fornecidas calculadoras em linguagem de programação Python elaboradas na ferramenta gratuita Google Colab. Cada calculadora é precedida de um breve resumo para o qual as mesmas se prestam a comprovar os conteúdos.

## 2.1 Equações Quadráticas

**Definição 2.1.** Uma **equação do 2º grau** (quadrática) é toda equação na forma:

$$ax^2 + bx + c = 0 \quad a \neq 0,$$

onde:

- $a, b, c$ são coeficientes reais;
- $x$ é a incógnita.

### 2.1.1 Fórmula de Bhaskara

A solução geral é dada por:

$$x = \frac{-b \pm \Delta}{2a}, \quad \text{com} \quad \Delta = b^2 - 4ac.$$

Neste caso, tem-se:

- Se $\Delta > 0$: Duas raízes reais distintas.
- Se $\Delta = 0$: Uma raiz real dupla (raízes repetidas).
- Se $\Delta < 0$: Duas raízes complexas conjugadas.

### 2.1.2 Análise do Discriminante Delta

O discriminante determina a natureza das raízes:

$$\Delta = b^2 - 4ac.$$

- $\Delta > 0$: Parábola corta o eixo $x$ em 2 pontos.
- $\Delta = 0$: Parábola tangencia o eixo $x$.
- $\Delta < 0$: Parábola não intersecta o eixo $x$.

### 2.1.3 Relações entre Coeficientes e Raízes (Girard)

Para as raízes $x_1$ e $x_2$:

$$x_1 + x_2 = -\frac{b}{a}, \quad x_1 \cdot x_2 = \frac{c}{a}.$$

### 2.1.4 Forma Fatorada

Se $x_1$ e $x_2$ são raízes:

$$ax^2 + bx + c = ax - x_1x - x_2.$$

### 2.1.5 Vértice da Parábola

As coordenadas do vértice $V$ da função $fx = ax^2 + bx + c$:

$$V\ -\frac{b}{2a}, -\frac{\Delta}{4a}\ .$$

Execute os códigos a seguir.

### 2.1.6 Calculadora Python

```
import numpy as np
import matplotlib.pyplot as plt
from cmath import sqrt

def equacao_segundo_grau(a, b, c):
    """Resolve a equação ax² + bx + c = 0 e retorna raízes, delta e
    ↪ vértice."""
    delta = b ** 2 - 4 * a * c
    raiz_delta = sqrt(delta) if delta >= 0 else sqrt(-delta) * 1j
    x1 = (-b + raiz_delta) / (2*a)
    x2 = (-b - raiz_delta) / (2*a)
    vertice_x = -b / (2*a)
    vertice_y = -delta / (4*a)
    return x1, x2, delta, (vertice_x, vertice_y)

def mostrar_resultados(a, b, c):
    x1, x2, delta, vertice = equacao_segundo_grau(a, b, c)

    print(f"\nEquação: {a}x² + {b}x + {c} = 0")
    print(f"Delta () = {delta}")
```

```
if delta > 0:
    print("Duas raízes reais distintas:")
elif delta == 0:
    print("Uma raiz real dupla:")
else:
    print("Duas raízes complexas conjugadas:")

print(f"x = {x1:.2f}, x = {x2:.2f}")
print(f"Vértice da parábola: ({vertice[0]:.2f}, {vertice[1]:.2f})")

# Configuração do gráfico
plt.figure(figsize=(8, 6))
x = np.linspace(vertice[0] - 3, vertice[0] + 3, 400)
y = a*x**2 + b*x + c

# Plot da parábola
plt.plot(x, y, label=f"y = {a}x² + {b}x + {c}", linewidth=2)

# Destacar eixos x e y
plt.axhline(0, color='black', linewidth=1.5, linestyle='-',
 ↪  label='Eixo x (y=0)')  # Eixo x
plt.axvline(0, color='black', linewidth=1.5, linestyle='-',
 ↪  label='Eixo y (x=0)')  # Eixo y

# Vértice e raízes
plt.scatter([vertice[0]], [vertice[1]], color='red', s=100,
 ↪  label='Vértice')
if delta >= 0:
    plt.scatter([x1.real, x2.real], [0, 0], color='green', s=100,
     ↪  label='Raízes')

# Configurações visuais
plt.xlabel("Eixo x", fontsize=12)
plt.ylabel("Eixo y", fontsize=12)
plt.title("Gráfico da Parábola", fontsize=14)
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend(fontsize=10)
plt.xlim(min(x) - 1, max(x) + 1)  # Ajuste dos limites para melhor
 ↪  visualização
```

```
    plt.ylim(min(y) - 2, max(y) + 2)
    plt.show()


# Interação com o usuário para uso da Calculadora

# Entrando com os coeficientes:

a = int(input("Escreva o termo a ( 0) da equação: "))
b = int(input("Escreva o termo b da equação: "))
c = int(input("Escreva o termo c da equação: "))
print("Resultados: ")

mostrar_resultados(a, b, c)
```

Para acessar e usar esta calculadora acesse o link abaixo:

Link para o Google Colab

## 2.2 Sistemas Lineares

**Definição 2.2.** Uma **equação linear** (primeiro grau) é toda equação na forma:

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b$$

$a_1, a_2, \ldots, a_n$ são coeficientes reais;

$x_1, x_2, \ldots, x_n$ são as incógnitas;

$b$ é o termo independente.

### 2.2.1 Sistemas de equações lineares

É um conjunto de $m$ equações lineares com $n$ incógnitas, conforme o representado adiante:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 + \cdots + a_{mn}x_n &= b_m \end{aligned} \tag{2.1}$$

### 2.2.2 Solução de um sistema de equações lineares

Os valores das variáveis que satisfazem simultaneamente a todas as equações de um sistema constituem sua solução. Esses valores são chamados de raízes do sistema de equações lineares. Para encontrar a solução de um sistema de equações lineares, aplicam-se os seguintes métodos:

**Método da substituição**

Consiste em isolar uma das incógnitas em uma das equações e, então, substituir a expressão encontrada na outra equação.

Considere os $a$, $b$, $c$ e $d$ números reais e o sistema linear $2 \times 2$:

$$ax + by = e \tag{2.2}$$

$$cx + dy = f \tag{2.3}$$

Isolando $y$ na equação 2.3 têm-se:

$$y = \frac{f - cx}{d}.$$

Substituindo em 2.2 resulta numa equação que contém apenas a incógnita $x$ tornando possível descobrir o seu valor.

$$ax + b\,\frac{f - cx}{d} = e.$$

Para sistemas lineares com 3 incógnitas repetimos o procedimento mais uma vez até restar apenas uma incógnita a descobrir o valor.

**Método por Eliminação ou Adição**

No método de eliminação, também conhecido como método de eliminação de Gauss ou método de escalonamento, o objetivo é transformar o sistema em sua forma mais simples (escalonada), de modo que seja possível resolver as incógnitas por substituição regressiva.

Considere os sistemas de equações lineares:

$$\begin{aligned} a_{11}x + a_{12}y &= b_1 \\ a_{21}x + a_{22}y &= b_2 \end{aligned} \tag{2.4}$$

ou

$$\begin{aligned} a_{11}x + a_{12}y + a_{13}z &= b_1 \\ a_{21}x + a_{22}y + a_{23}z &= b_2 \\ a_{31}x + a_{32}y + a_{33}z &= b_3 \end{aligned} \tag{2.5}$$

Resolver um sistema linear significa encontrar todos os valores que satisfazem as equações do sistema. Para isso, usamos algumas operações simples: podemos trocar duas linhas de lugar, multiplicar uma linha por um número diferente de zero ou somar/subtrair uma linha da outra. Essas operações servem para deixar a matriz do sistema em uma forma mais simples, chamada triangular, o que facilita bastante a resolução passo a passo das variáveis.

**Método de Cramer**

Os sistemas de equações lineares

$$a_1x + b_1y = c_1$$
$$a_2x + b_2y = c_2$$

e

$$a_1x + b_1y + c_1z = d_1$$
$$a_2x + b_2y + c_2z = d_2$$
$$a_3x + b_3y + c_3z = d_3$$

possuem as respectivas matrizes dos coeficientes:

$$\begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix} \quad \text{e} \quad \begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix}$$

Tais matrizes possuem os seguintes determinantes:

$$D_2 = a_1b_2 - a_2b_1; \tag{2.6}$$
$$D_3 = a_1b_2c_3 + b_1c_2a_3 + c_1a_2b_3 \tag{2.7}$$
$$- a_3b_2c_1 - b_3c_2a_1 - c_3a_2b_1; \tag{2.8}$$

Pelo método de Cramer, temos:

$$x = \frac{D_x}{D}, \quad y = \frac{D_y}{D}, \quad z = \frac{D_z}{D},$$

em que os determinantes $D_x$, $D_y$ e $D_z$ são obtidos substituindo-se, respectivamente, as colunas correspondentes às variáveis $x$, $y$ e $z$ pela coluna dos termos independentes (resultados), ou seja, $c_1, c_2$ para o sistema $2 \times 2$, e $d_1, d_2, d_3$ para o sistema $3 \times 3$.

### 2.2.3 Condições de Solução

Seja o sistema de equações lineares:

$$a_1x + b_1y = c_1$$
$$a_2x + b_2y = c_2$$

Quanto ao conjunto solução, existem as seguintes possibilidades, estabelecidas pelas relações abaixo descritas:

**Sistema possível e determinado (solução única)**

$$\frac{a_1}{a_2} \neq \frac{b_1}{b_2}$$

**Sistema possível e com infinitas soluções**

$$\frac{a_1}{a_2} = \frac{b_1}{b_2} = \frac{c_1}{c_2}$$

**Sistema impossível (sem solução)**

$$\frac{a_1}{a_2} = \frac{b_1}{b_2} \neq \frac{c_1}{c_2}$$

Execute os códigos a seguir.

### 2.2.4 Calculadora Python Sistemas Lineares 2x2

```
print("=== Calculadora de Sistemas Lineares 2x2 ===")

# Entrada dos coeficientes
a1 = float(input("Digite a1: "))
b1 = float(input("Digite b1: "))
c1 = float(input("Digite c1: "))

a2 = float(input("Digite a2: "))
b2 = float(input("Digite b2: "))
c2 = float(input("Digite c2: "))

# Cálculo do determinante
det = a1 * b2 - a2 * b1

if det == 0:
    print("O sistema não tem solução única (possivelmente impossível ou
    ↪ indeterminado).")
```

```
else:
    # Regra de Cramer
    x = (c1 * b2 - c2 * b1) / det
    y = (a1 * c2 - a2 * c1) / det

    print("\nSolução:")
    print(f"x = {x}")
    print(f"y = {y}")
```

Execute os códigos a seguir.

### 2.2.5 Calculadora Python Sistemas Lineares 3x3

```
    print("=== Resolver Sistema Linear 3x3 ===")

# Entrada dos coeficientes
a1 = float(input("Digite a1: "))
b1 = float(input("Digite b1: "))
c1 = float(input("Digite c1: "))
d1 = float(input("Digite d1: "))

a2 = float(input("Digite a2: "))
b2 = float(input("Digite b2: "))
c2 = float(input("Digite c2: "))
d2 = float(input("Digite d2: "))

a3 = float(input("Digite a3: "))
b3 = float(input("Digite b3: "))
c3 = float(input("Digite c3: "))
d3 = float(input("Digite d3: "))

# Determinante principal (D)
D = (a1 * (b2 * c3 - b3 * c2)
   - b1 * (a2 * c3 - a3 * c2)
   + c1 * (a2 * b3 - a3 * b2))

if D == 0:
    print("\nO sistema não tem solução única (determinante = 0).")
else:
    # Dx
```

```
Dx = (d1 * (b2 * c3 - b3 * c2)
    - b1 * (d2 * c3 - d3 * c2)
    + c1 * (d2 * b3 - d3 * b2))

# Dy
Dy = (a1 * (d2 * c3 - d3 * c2)
    - d1 * (a2 * c3 - a3 * c2)
    + c1 * (a2 * d3 - a3 * d2))

# Dz
Dz = (a1 * (b2 * d3 - b3 * d2)
    - b1 * (a2 * d3 - a3 * d2)
    + d1 * (a2 * b3 - a3 * b2))

x = Dx / D
y = Dy / D
z = Dz / D

print("\nSolução do sistema:")
print(f"x = {x}")
print(f"y = {y}")
print(f"z = {z}")
```

Para acessar e usar esta calculadora acesse o link abaixo:

Link para o Google Colab

## 2.3 Progressão Aritmética

**Definição 2.3. Uma progressão aritmética (PA)** é uma sequência na qual a diferença entre cada termo e o termo anterior é constante. Essa diferença constante é chamada razão da progressão e representada pela letra $r$.

### 2.3.1 Fórmula do Termo Geral

o Termo geral da PA pode ser obtido pela expressão:

$$a_n = a_1 + n - 1r$$

onde:

- $a_n$ é o termo que se deseja encontrar;

- $a_1$ é o primeiro termo da sequência;
- $n$ é a posição do termo;
- $r$ é a razão

### 2.3.2 Soma de todos os termos da PA

Soma os $n$ termos de uma PA de razão $n$ :

$$S_n = \frac{a_1 + a_n n}{2}$$

Onde $S_n$ é a soma de todos os termos.
Execute os códigos a seguir.

### 2.3.3 Calculadora Python Progressão Aritmética

```
    def termo_geral_PA(a1, r, n):
    """
    Calcula o n-ésimo termo de uma progressão aritmética.
    Fórmula: an = a1 + (n - 1) * r
    """
    an = a1 + (n - 1) * r
    return an

def soma_PA(a1, r, n):
    """
    Calcula a soma dos n primeiros termos de uma PA.
    Fórmula: Sn = n/2 * (2a1 + (n - 1) * r)
    """
    sn = (n / 2) * (2 * a1 + (n - 1) * r)
    return sn

def n_primeiros_termos(a1, r, n):
    """
    Retorna uma lista com os n primeiros termos da PA.
    """
    n_termos = []
    for i in range(1, n + 1):
        n_termos.append(termo_geral_PA(a1, r, i))
    return n_termos
```

```
# Entrada do usuário
a1 = int(input("Informe o primeiro termo (a1): "))
r = int(input("Informe a razão (r): "))
n = int(input("Informe o número de termos (n): "))

# Cálculos
an = termo_geral_PA(a1, r, n)
sn = soma_PA(a1, r, n)
lista = n_primeiros_termos(a1, r, n)

# Saída
print(f"O {n}º termo da PA é: {an}")
print(f"A soma dos {n} primeiros termos da PA é: {sn}")
print(f"A lista dos {n} primeiros termos da PA é: {lista}")
```

Para acessar e usar esta calculadora acesse o link abaixo:

Link para o Google Colab

## 2.4 Progressão Geométrica

**Definição 2.4. Uma progressão geométrica (PG)** é uma sequência numérica onde cada termo, a partir do segundo, é igual ao produto do termo anterior por uma constante, chamada de razão $q$.

### 2.4.1 Tipos de progressões geométricas

- crescente: $q > 1$, $a_1 > 0$ ou $0 < q < 1$, $a_1 < -1$;
- Decrescente: $0 < q < 1$, $a_1 > 0$ ou $q > 0$, $a_1 < 0$;
- Oscilante: $q < 0$ ou $-1 < q < 0$;
- Constante: $q = 1$.

### 2.4.2 Fórmula do Termo Geral

Um termo geral de uma PG pode ser obtido pela expressão:

$$a_n = a_1 q^{n-1}$$

onde:

- $a_n$ é o termo que se deseja encontrar;

- $a_1$ é o primeiro termo da sequência;
- $n$ é a posição do termo;
- $q$ é a razão da PG.

### 2.4.3 Soma de todos os termos de uma PG

Progressão Geométrica finita de razão $q$:

$$S_n = a_1 \frac{q^n - 1}{q - 1}$$

onde $S_n$ é a soma de todos os termos de uma PG finita.
Progressão Geométrica infinita de razão $q$, $-1 < q < 1$:

$$S = \frac{a_1}{1 - q}$$

Execute os códigos a seguir.

### 2.4.4 Calculadora Python Progressão Geométrica

```
    def termo_geral_pg(a1, q, n):
    """Calcula o n-ésimo termo da PG."""
    an = a1 * q ** (n - 1)
    return an



def soma_n_termos_pg(a1, q, n):
    """Calcula a soma dos n primeiros termos da PG."""
    if q == 1:
        return a1 * n
    else:
        return a1 * (1 - q ** n) / (1 - q)

def soma_infinita_pg(a1, q):
    """Calcula a soma infinita da PG, se convergente."""
    if abs(q) < 1:
        return a1 / (1 - q)
    else:
        return None
def n_primeiros_termos(a1, q, n):
    """
```

```
    Retorna uma lista com os n primeiros termos da PG.
    """
    n_termos = []
    for i in range(1, n + 1):
        n_termos.append(termo_geral_pg(a1, q, i))
    return n_termos

# Entrada do usuário
a1 = float(input("Digite o primeiro termo (a1): "))
q = float(input("Digite a razão (q): "))
n = int(input("Digite o número de termos (n): "))

# Cálculos
an = termo_geral_pg(a1, q, n)
sn = soma_n_termos_pg(a1, q, n)
lista = n_primeiros_termos(a1, q, n)
soma_inf = soma_infinita_pg(a1, q)


print(f"\nTermo geral (a{n}) = {an}")
print(f"Soma dos {n} primeiros termos = {sn}")
print(f"A lista dos {n} primeiros termos da PG é: {lista}")
if soma_inf is not None:
    print(f"Soma infinita da PG = {soma_inf}")
else:
    print("A soma infinita da PG não converge (|q|  1)")
```

Para acessar e usar esta calculadora acesse o link abaixo:

Link para o Google Colab

## 2.5 EQUAÇÕES EXPONENCIAIS

**Definição 2.5. Uma equação é dita exponencial** quando a incógnita aparece no expoente de uma ou mais potências..

Exemplo:

$$2^x = 32$$

### 2.5.1 Forma Geral:

$$a^{fx} = c$$

onde:

- $a$ e $c$ números reais positivos, $a \neq 1$;
- x variável real;
- f(x), função exponencial.

### 2.5.2 Propriedades exponenciais

Sejam $a$, $b$, $m$ e $n$ reais, $b \neq 0$, são válidas as propriedades seguintes:

- $a^m.a^n = a^{m+n}$;
- $\frac{a^m}{a^n} = a^{m-n}$, $a \neq 0$;
- $a.b^n = a^n.b^n$;
- $\frac{a}{b}^n = \frac{a^n}{b^n}$, $b \neq 0$;
- $a^{mn} = a^{mn}$;
- $a^{-n} = \frac{1}{a^{-n}}, a \neq 0$;
- $a^0 = 1$.

Execute os códigos a seguir.

### 2.5.3 Calculadora Exponencial

```
    def equacao_exponencial(x, a, m, n):
    """
    Retorna o valor da expressão a^(m*x + n)
    """
    return a ** (m * x + n)

# Entrada de dados
a = float(input("Digite a base a (a > 0 e a  1): "))
m = float(input("Digite o coeficiente m: "))
n = float(input("Digite o coeficiente n: "))
b = float(input("Digite o valor de b (resultado da equação): "))
```

```
# Verificações
if a <= 0 or a == 1:
    print("Erro: a deve ser maior que 0 e diferente de 1.")
elif b <= 0:
    print("Erro: b deve ser maior que 0.")
else:
    # Parâmetros da busca
    x_inicial = -100
    x_final = 100
    passo = 0.0001
    tolerancia = 0.0001

    x = x_inicial
    encontrado = False

    while x <= x_final:
        resultado = equacao_exponencial(x, a, m, n)
        if abs(resultado - b) < tolerancia:
            print(f"Solução aproximada encontrada: x  {round(x, 5)}")
            encontrado = True
            break
        x += passo

    if not encontrado:
        print("Nenhuma solução real aproximada encontrada no intervalo.")
```

Para acessar e usar esta calculadora acesse o link abaixo:

Link para o Google Colab

## 2.6 LOGARITMOS

**Definição 2.6. Logaritmo** é o expoente de uma potência em determinada base cujo resultado é um valor certo.

Exemplo:

$$\log_2 4 = 2$$

Pois $2^2 = 4$

### 2.6.1 Forma Geral:

Se

$$a^x = b$$

então

$$\log_a b = x$$

onde:

- $b$ o número gerador do logaritmo;
- $a$ é a base;
- $x$ é o logaritmo de b.

### 2.6.2 Propriedades dos logaritmos

- Logaritmo do produto,
$$\log_b x \cdot y = \log_b x + \log_b y;$$
- Logaritmo do quociente,
$$\log_b \frac{x}{y} = \log_b x - \log_b y;$$
- Logaritmo da potência,
$$\log_b x^a = a \cdot \log_b x;$$
- Logaritmo da raiz,
$$\log_b \sqrt[n]{x} = \log_b x^{\frac{1}{n}} = \frac{1}{n} \cdot \log_b x;$$
- Mudança de base,
$$\log_b x = \frac{\log_k x}{\log_k} b;$$
- Logaritmo de 1, para qualquer $b > 0$ e $b \neq 1$,
$$\log_b 1 = 0;$$
- Logaritmo da própria base,
$$\log_b b = 1.$$

Execute os códigos a seguir.

### 2.6.3 Calculadora Logaritmo

```
    # Cálculo básico de logaritmo sem funções prontas
# log_a(b) = x  ->  a^x = b

base = float(input("Digite a base (a): "))
valor = float(input("Digite o valor (b): "))

# Verificações
if base <= 0 or base == 1 or valor <= 0:
    print("A base deve ser positiva e diferente de 1, e o valor deve ser
     ↪ positivo.")
else:
    # Começamos com um chute inicial para x
    x = 0.0
    passo = 0.0001  # Quanto menor, mais preciso (mas mais lento)

    # Vamos aumentar x até a^x ultrapassar o valor
    while base ** x < valor:
        x += passo

    print(f"log_{base}({valor})  {x:.2f}")
```

Para acessar e usar esta calculadora acesse o link abaixo:

Link para o Google Colab

## 2.7 TRIGONOMETRIA

### 2.7.1 RELAÇÕES TRIGONOMÉTRICAS

Seja um triângulo retângulo ABC, de lados medindo a, b, e c e ângulos $\hat{A}$, $\hat{B}$ e $\hat{C}$ onde, Hipotenusa é o lado oposto ao ângulo reto, Cateto oposto é o lado oposto ao ângulo de interesse e Cateto adjacente lado que vizinho ao ângulo excluindo a hipotenusa. São válidas as seguintes relações:

**Seno**

$$\text{sen} = \frac{\text{cateto oposto}}{\text{hipotenusa}}$$

**Cosseno**

$$\cos = \frac{\text{cateto adjacente}}{\text{hipotenusa}}$$

**Tangente**

$$\text{tg} = \frac{\text{cateto oposto}}{\text{cateto adjacente}}$$

Execute os códigos a seguir.

### 2.7.2 Calculadora Relações Trigonométricas

```
import math

# Entrada dos comprimentos dos lados
cateto_oposto = float(input("Digite o comprimento do cateto oposto: "))
cateto_adjacente = float(input("Digite o comprimento do cateto adjacente:
↪  "))
hipotenusa = float(input("Digite o comprimento da hipotenusa: "))

# Verificação se é triângulo retângulo (sem math.isclose)
if hipotenusa**2 == cateto_oposto**2 + cateto_adjacente**2:
    # Cálculos
    seno = cateto_oposto / hipotenusa
    cosseno = cateto_adjacente / hipotenusa
    tangente = cateto_oposto / cateto_adjacente

    # Saída
    print(f"Seno: {seno:.2f}")
    print(f"Cosseno: {cosseno:.2f}")
    print(f"Tangente: {tangente:.2f}")
    print("É um triângulo retângulo.")
else:
    print("Não é um triângulo retângulo.")
```

Para acessar e usar esta calculadora acesse o link abaixo:

Link para o Google Colab

## 2.8 JUROS SIMPLES

**Definição 2.7. Juros simples** é um tipo de capitalização onde os juros são calculados apenas sobre o **capital inicial (C)**, sem incidência sobre os juros já acumulados.

### 2.8.1 Fórmula Calcular Juros Simples

O juros simples pode ser obtido pela expressão:

$$J = \frac{C.i.t}{100}$$

onde:

- $J$ é o valor do juros que se deseja encontrar;
- $C$ é o valor do capital de incidência da taxa;
- $i$ é a taxa de juros aplicada;
- $t$ é o período, o tempo da aplicação.

-

### 2.8.2 Fórmula para Calcular o Montante

Soma o valor do juros com valor do Capital:

$$M = J + C$$

Onde:

- - $M$ é valor do Montante;
- $J$ é o juros;
- $C$ é o valor do Capital.

Execute os códigos a seguir.

### 2.8.3 Calculadora Juros Simples

```
print("=== Calculadora de Juros Simples ===")

print("\nObs.: Utilize ponto (.)")
print("no lugar das vírgulas para indicar os centavos\n")
```

```
# Entrada de dados
capital = float(input("Digite o capital (R$): "))
taxa = float(input("Digite a taxa de juros (%): "))
tempo = float(input("Digite o tempo (meses, anos, etc.): "))

# Cálculo dos juros e montante
juros = (capital * tempo * taxa) / 100
montante = capital + juros

# Saída formatada
print(f"\nResultado:")
print(f"Juros: R$ {juros:.2f}")
print(f"Montante: R$ {montante:.2f}")
```

Para acessar e usar esta calculadora acesse o link abaixo:

Link para o Google Colab

## 2.9 JUROS COMPOSTOS

**Definição 2.8.** Os juros são calculados sobre o capital inicial e também sobre os juros acumulados ao longo do tempo, juros sobre juros.

### 2.9.1 Fórmula Calcular o valor Montante do juros composto

$$M = C.1 + i^t$$

onde:

- $M$ é o valor total que se deseja encontrar;
- $C$ é o valor do capital de incidência da taxa;
- $i$ é a taxa de juros aplicada;
- $t$ é o período, o tempo da aplicação.

### 2.9.2 Fórmula para Calcular o valor do Juros

$$J = M - C$$

Onde:

- $M$ é valor do Montante;
- $J$ é o juros;
- $C$ é o valor do Capital.

Execute os códigos a seguir.

### 2.9.3 Calculadora Juros Compostos

```
    print("=== Calculadora de Juros Compostos ===")

# Entrada de dados
capital = float(input("Digite o capital inicial (R$): "))
taxa = float(input("Digite a taxa de juros (%): "))
tempo = float(input("Digite o tempo (meses, anos, etc.): "))

# Conversão da taxa para decimal
i = taxa / 100

# Cálculo do montante e juros compostos
montante = capital * (1 + i) ** tempo
juros = montante - capital

# Saída dos resultados
print(f"\nResultado:")
print(f"Montante: R$ {montante:.2f}")
print(f"Juros: R$ {juros:.2f}")
```

Para acessar e usar esta calculadora acesse o link abaixo:

Link para o Google Colab

## 2.10 ESTATÍSTICAS, MEDIDAS DE TENDÊNCIA CENTRAL

### 2.10.1 Média Aritmética

É a soma de todos os valores dividida pelo número de elementos.

$$M_x = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

onde:

- $M_x$ é o valor da média aritmética;
- $x_1, x_2, \ldots, x_n$ são os valores;
- $n$ é total de elementos.

### 2.10.2 Média Aritmética Ponderada

A média ponderada é calculada pela soma dos produtos de cada valor pelo seu respectivo peso, dividida pela soma total dos pesos.

$$M_{xp} = \frac{x_1\dot{p}_1 + x_2\dot{p}_2 + \cdots + x_n\dot{p}n}{p_1 + p_2 + \cdots + p_n}$$

onde:

- $M_{xp}$ é o valor da média aritmética ponderada;
- $x_1, x_2, \ldots, x_n$ são os valores;
- $p_1, p_2 \ldots p_n$ são os pesos;

### 2.10.3 Mediana

É o valor central de um conjunto ordenado quando a quantidade de elementos é ímpar, ou a média aritmética dos dois valores centrais quando a quantidade de elementos é par.

### 2.10.4 Moda

É o valor que mais aparece em um conjunto de dados.
Execute os códigos a seguir.

### 2.10.5 Calculadora Estatística

```
    # Entrada de dados
valores = input("Digite os números separados por espaço: ")
lista = list(map(float, valores.split()))
```

```
n = len(lista)

# Média
soma = 0
for num in lista:
    soma += num
media = soma / n

# Mediana
lista_ordenada = sorted(lista) # Função "sorted" usada para ordenar
 ↪  listas
if n % 2 == 1: # o símbolo "%" indica resto da divisão
    mediana = lista_ordenada[n // 2] # o símbolo "//" indica a parte
     ↪  inteira da divisão

else:
    meio1 = lista_ordenada[n // 2 - 1]
    meio2 = lista_ordenada[n // 2]
    mediana = (meio1 + meio2) / 2

# Moda
frequencias = {}
for num in lista:
    if num in frequencias:
        frequencias[num] += 1
    else:
        frequencias[num] = 1

# Encontrando a maior frequência
max_freq = 0
for n in frequencias.values():
  if n > max_freq:
    max_freq = n

modas = [num for num, freq in frequencias.items() if freq == max_freq]

if len(modas) == 1:
    moda = modas[0]
else:
```

```
    moda = "Sem moda (ou multimodal)"

# Resultados
print("\nRESULTADOS:")
print(f"Média: {media}")
print(f"Mediana: {mediana}")
print(f"Moda: {moda}")
```

Para acessar e usar esta calculadora acesse o link abaixo:

Link para o Google Colab

## 2.11 PROBABILIDADE

**Definição 2.9.** Chamamos **Probabilidade** a medida percentual de acontecer um evento determinado.

### 2.11.1 Fórmula:

$$PE = \frac{\text{casos favoráveis}}{\text{casos possíveis}}$$

com $0 \leq PE \leq 1$,

- 0, evento impossível;
- 1, evento certo.

### 2.11.2 Espaço amostral (S)

Conjunto de todos os casos possíveis.

### 2.11.3 Evento (E)

Subconjunto do espaço amostral, casos prováveis.

### 2.11.4 Propriedades:

- Probabilidade complementar,

$$PE_c = 1 - PE$$

- Probabilidade da União ( eventos mutuamente exclusivos ),

$$PA \cup B = PA + PB$$

- Probabilidade da interseção ( eventos independentes),

$$P A \cap B = P A . P B$$

- Probabilidade condicional, cálculo depende de um evento já ocorrido,

$$P A \bigcup B = \frac{P A \cap B}{P B}$$

Execute os códigos a seguir.

### 2.11.5 Calculadora Probabilidade

```
    # Cálculo de probabilidades: incondicional, condicional e união

print("=== Cálculo de Probabilidade ===")

# Entrada de dados
total_casos = int(input("Digite o número total de casos possíveis: "))
casos_A = int(input("Digite o número de casos favoráveis ao evento A: "))
casos_B = int(input("Digite o número de casos favoráveis ao evento B: "))
casos_AeB = int(input("Digite o número de casos favoráveis a A e B (A
 ↪  B): "))

# Verificações
if total_casos <= 0:
    print("O número total de casos possíveis deve ser maior que zero.")
elif not (0 <= casos_A <= total_casos and 0 <= casos_B <= total_casos and
 ↪  0 <= casos_AeB <= total_casos):
    print("Os valores informados são inválidos.")
else:
    # Probabilidades incondicionais
    p_A = casos_A / total_casos
    p_B = casos_B / total_casos

    # Probabilidade de A e B
    p_AeB = casos_AeB / total_casos

    # Probabilidade da união
    p_AuB = p_A + p_B - p_AeB
```

```
# Probabilidades condicionais
if p_B > 0:
    p_A_dado_B = p_AeB / p_B
else:
    p_A_dado_B = None

if p_A > 0:
    p_B_dado_A = p_AeB / p_A
else:
    p_B_dado_A = None

# Exibição
print("\n--- Resultados ---")
print(f"P(A) = {p_A:.2f} ou {p_A*100:.2f}%")
print(f"P(B) = {p_B:.2f} ou {p_B*100:.2f}%")
print(f"P(A  B) = {p_AeB:.2f} ou {p_AeB*100:.2f}%")
print(f"P(A  B) = {p_AuB:.2f} ou {p_AuB*100:.2f}%")

if p_A_dado_B is not None:
    print(f"P(A | B) = {p_A_dado_B:.2f} ou {p_A_dado_B*100:.2f}%")
else:
    print("P(A | B) não pode ser calculada (P(B) = 0).")

if p_B_dado_A is not None:
    print(f"P(B | A) = {p_B_dado_A:.2f} ou {p_B_dado_A*100:.2f}%")
else:
    print("P(B | A) não pode ser calculada (P(A) = 0).")
```

Para acessar e usar esta calculadora acesse o link abaixo:

Link para o Google Colab

# 3 PROPOSTAS DE USO

Considerando o atual contexto educacional, fortemente influenciado pelo avanço das tecnologias digitais, a utilização de calculadoras desenvolvidas em Python apresenta-se como uma estratégia pedagógica inovadora no ensino de Matemática no nível básico. Tal recurso contribui para tornar o processo de aprendizagem mais participativo e dinâmico, ao possibilitar que os estudantes, além de adquirirem conhecimentos matemáticos, desenvolvam também noções fundamentais de programação e algoritmos.

As calculadoras em Python permitem que o discente compreenda a lógica subjacente aos códigos, estabeleça relações entre operações matemáticas e sua representação computacional, bem como explore conceitos de forma aplicada ao universo digital. Desse modo, a aprendizagem ultrapassa a mera resolução de cálculos, favorecendo a formação de habilidades cognitivas associadas ao pensamento computacional.

Do ponto de vista docente, trata-se de um recurso com grande potencial de apoio pedagógico, visto que pode ser empregado na correção de atividades, na elaboração de avaliações e na execução de cálculos de maior complexidade. Assim, constitui-se em uma ferramenta capaz de potencializar tanto a prática de ensino quanto a aprendizagem discente, integrando o ensino de Matemática ao contexto tecnológico contemporâneo.

# 4 CONSIDERAÇÕES FINAIS

Em conclusão, o produto educacional desenvolvido, fundamentado em ferramentas tecnológicas, mostra-se alinhado ao atual cenário marcado pela expansão das inteligências artificiais e pela crescente automação. Essa iniciativa revela-se promissora para o ensino de Matemática no Brasil, ao oferecer recursos capazes de potencializar o processo de ensino-aprendizagem por meio de abordagens mais dinâmicas, interativas e contextualizadas.

Todavia, a implementação efetiva desse tipo de recurso demanda a superação de desafios estruturais ainda presentes na realidade educacional brasileira, entre os quais se destacam as dificuldades de acesso a computadores e à internet em determinadas regiões. Superados tais obstáculos, os docentes poderão explorar de maneira mais ampla e significativa o potencial pedagógico dessas ferramentas, integrando-as às suas práticas didáticas e, assim, contribuindo para a ampliação das oportunidades de aprendizagem matemática e para o fortalecimento da formação dos estudantes da educação básica.

Visando ampliar o acesso a recursos tecnológicos voltados ao ensino de Matemática, apresenta-se, a seguir, o conjunto de endereços eletrônicos (URLs) das calculadoras desenvolvidas em linguagem Python.

# REFERÊNCIAS


[1] ALENCAR FILHO, Edgard de. **Teoria elementar dos números**. São Paulo: Nobel, 1981. exemplo de citação: (ALENCAR FILHO, 1981) exemplo de citação: Alencar Filho, 1981

[2] GOOGLE. **Google Colaboratory**. [S.l.]: Google, 2017. Disponível em: https://colab.research.google.com/. Acesso em: 27 ago. 2025.

[3] Equações quadráticas, Linguagem Python. https://colab.research.google.com/drive/1EpVMSzMAMxk3tf8adeQjJgpdBFhs1fwM?usp=sharing.

[4] Sistema de Equações Lineares, Linguagem Python. https://colab.research.google.com/drive/11Y3mZ1RcZYe7vVNniEGbo1B-5vmuEYXS?usp=sharing.

[5] Progressão Aritmética, Linguagem Python. https://colab.research.google.com/drive/1kSvjrDdfNjaEu8tI77msoj9E9fmnNeHI?usp=sharing.

[6] Progressão Geométrica, Linguagem Python. https://colab.research.google.com/drive/1kG0WKPtqPAQe_VsFFXdgelkQBa_jp6AA?usp=sharing.

[7] Equações Exponenciais, Linguagem Python. https://colab.research.google.com/drive/1k50Gc1TZPAMolXm9Ea3oV8if5CNeBbRY?usp=sharing.

[8] Logaritmos, Linguagem Python. https://colab.research.google.com/drive/1wj0CdWSUX1Xz6ZG4AZH7Qe6_MbbfRypL?usp=sharing.

[9] Relações Trigonométricas, Linguagem Python. https://colab.research.google.com/drive/1ndJYnnlXcpngIO4SfK10gimKGPPl4h8Z?usp=sharing.

[10] Juros Simples, Linguagem Python. https://colab.research.google.com/drive/1IaAnQX_KCYrDZPtRr2-wEsKyCRkp9LeH?usp=sharingg.

[11] Juros Compostos, Linguagem Python. https://colab.research.google.com/drive/1IYJwYBD7o-mwopkAj55zlkihZ8a7eU6W?usp=sharing.

[12] Estatística, Medidas de tendência central, Linguagem Python. https://colab.research.google.com/drive/1WMEJQlhbMbxRwrhv9A_cq5nHHcjZElVp?usp=sharing.

[13] Probabilidade, Linguagem Python. https://colab.research.google.com/drive/1nSUYHdYcJ7NK6Ts_j0p9kZMeEvk3g8d6?usp=sharing.