



UNIVERSIDADE FEDERAL
DE SERGIPE

Descomplicando matemática com Python

Igor Terriaga Santos

Agenda

1. Revisão da Linguagem Python
2. Polinômios
3. Matrizes
4. Funções
5. Equações e Inequações
6. Números Complexos
7. Gráficos

Objetivos

- Utilizar a linguagem de programação Python como ferramenta
- Conhecer algumas bibliotecas disponíveis
- Resolução de questões com Matrizes, equações, entre outras.
- Plotar gráficos de uma forma muito simples

Informações

Duração do mini-curso: 4h

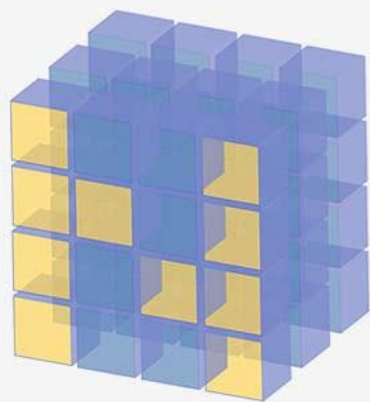
Coordenação: André Luis Meneses Silva

Ministrante: Igor Terriaga Santos

Público estimado: 20 pessoas

Bibliotecas que vamos conhecer

matplotlib



NumPy

Bibliotecas que vamos conhecer

NumPy – Computação Científica

NumPy é o pacote fundamental para computação científica com Python. Ele permite, entre outras coisas:

- Manipulação de matriz n-dimensional (uma matriz multidimensional rápida e eficiente que permite a vetorização de operações aritméticas), que é fundamental para o trabalho em **Ciência de dados**.
- Utilitários de álgebra linear e capacidade de gerar números aleatórios.

Bibliotecas que vamos conhecer

Sympy – Computação Simbólica

- SymPy é uma biblioteca Python para a matemática. Destina-se a tornar-se um sistema de álgebra computacional full-featured, mantendo o código tão simples quanto possível, a fim de ser compreensível e facilmente extensível. SymPy é inteiramente escrito em Python e não requer nenhuma biblioteca externa.

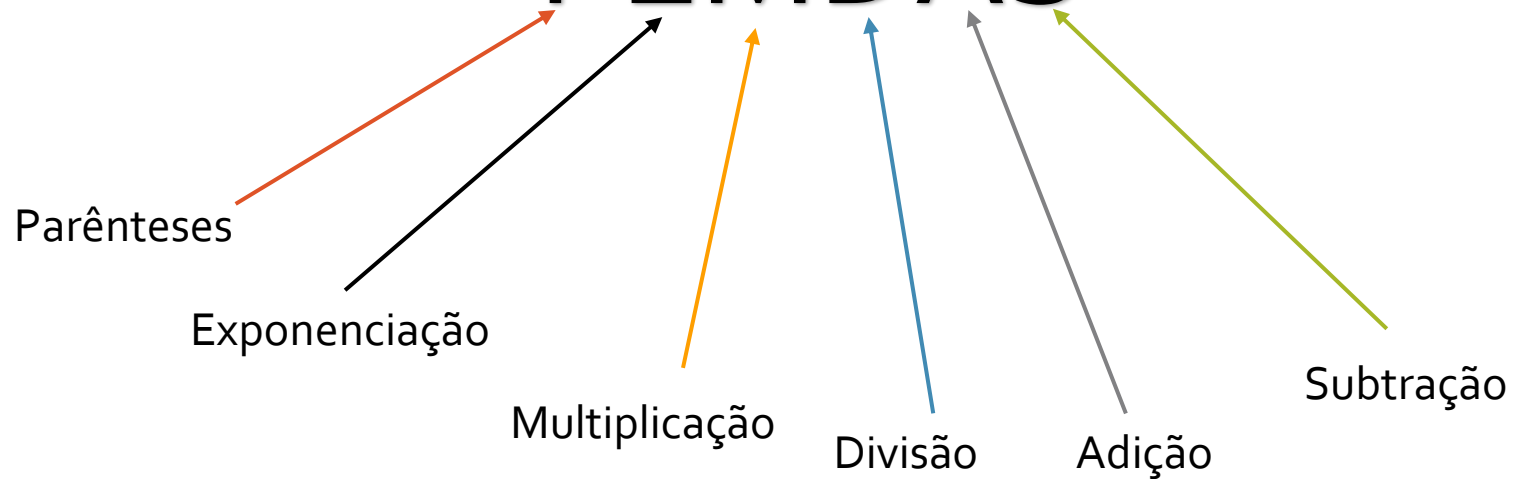
Bibliotecas que vamos conhecer

Matplotlib – Visualização de Dados

- Matplotlib é um módulo Python para visualização de dados. Matplotlib permite que você crie facilmente gráfico, histogramas e outras figuras profissionais. Usando Matplotlib você pode personalizar cada aspecto de uma figura. Quando usado no IPython, Matplotlib tem recursos interativos, como zoom e visão panorâmica. É possível também pode exportar gráficos para vetor comum e formatos gráficos: pdf, svg, jpg, png, bmp, gif, etc.

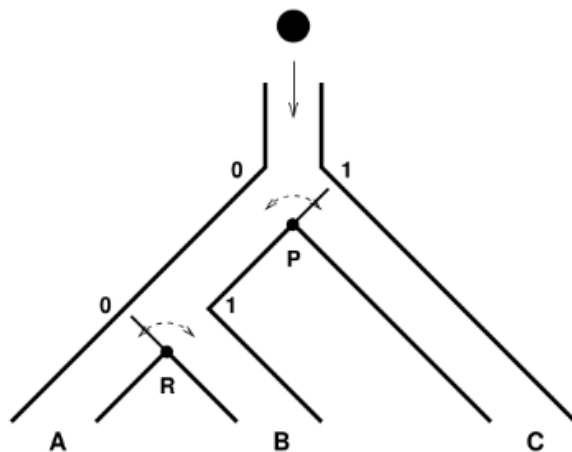
Para lembrar...

PEMDAS



Vamos a um problema simples.

Flíper é um tipo de jogo onde uma bolinha de metal cai por um labirinto de caminhos até chegar na parte de baixo do labirinto. A quantidade de pontos que o jogador ganha depende do caminho que a bolinha seguir. O jogador pode controlar o percurso da bolinha mudando a posição de algumas portinhas do labirinto. Cada portinha pode estar na posição 0, que significa virada para a esquerda, ou na posição 1 que quer dizer virada para a direita. Considere o flíper da figura abaixo, que tem duas portinhas. A portinha **P** está na posição 1 e a portinha **R**, na posição 0. Desse jeito, a bolinha vai cair pelo caminho B.



Você deve escrever um programa que, dadas as posições das portinhas **P** e **R**, neste flíper da figura, diga por qual dos três caminhos, A, B ou C, a bolinha vai cair!

Entrada

A entrada é composta por apenas uma linha contendo dois números **P** e **R**, indicando as posições das duas portinhas do flíper da figura.

Vamos a um problema simples.

Saída

A saída do seu programa deve ser também apenas uma linha, contendo uma letra maiúscula que indica o caminho por onde a bolinha vai cair: 'A', 'B' ou 'C'.

Restrições

- O número ***P*** pode ser 0 ou 1. O número ***R*** pode ser 0 ou 1.

Exemplos de Entrada	Exemplos de Saída
1 0	B
0 0	C



Polinômios

$$\begin{array}{r} \cancel{x^3} - 6x^2 - x + 12 \quad | \quad x - 2 \\ \underline{-\cancel{x^3} + 2x^2} \\ -4\cancel{x^2} - x + 12 \\ \phantom{-4\cancel{x^2}} + 4\cancel{x^2} - 8x \\ \phantom{-4\cancel{x^2} + 4\cancel{x^2}} \underline{-9\cancel{x} + 12} \\ \phantom{-4\cancel{x^2} + 4\cancel{x^2} - 9\cancel{x}} + 9\cancel{x} - 18 \\ \phantom{-4\cancel{x^2} + 4\cancel{x^2} - 9\cancel{x} + 9\cancel{x}} \underline{} \\ \phantom{-4\cancel{x^2} + 4\cancel{x^2} - 9\cancel{x} + 9\cancel{x}} -6 \end{array}$$

Operações com Polinômios

Podem ser feitas importando algumas funções da biblioteca `numpy`. Elas operam os polinômios definidos por vetores com os valores de seus coeficientes. Dessa forma, o polinômio $3x^2 + 4x - 2$ seria representado pela lista `[3, 4, -2]`.

- Adição: **`polyad(c1, c2)`**
- Subtração: **`polysub(c1, c2)`**
- Multiplicação: **`polymul(c1, c2)`**
- Divisão: **`polydiv(1c, c2)`**

Onde **`c1`** representa a lista com os coeficientes do polinômio 1, e **`c2`**, o vetor com os coeficientes de polinômio 2



Vamos ao quadro!



Matrizes

$$A = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix}$$

Sobre Matrizes

As matrizes são arranjos numéricos de grande utilidade na matemática e em diversas aplicações científicas. A linguagem de programação python possui específicas para matrizes na biblioteca numpy. Por meio delas, podem ser criadas matrizes de diferentes formatos e características.

Vale lembrar que Python considera a primeira linha de uma matriz a linha 0 e a primeira coluna a coluna 0. Dessa forma, o elemento da linha 4 e coluna 1 de uma matriz numérica seria endereçado por [3, 0]



Vamos ao quadro!

Sobre Matrizes

Usando a função *matrix* da biblioteca numpy, uma matriz de m linhas e n colunas pode ser criada, bastando elencar seus elementos:

```
# Cria matriz A com valores (int)
matriz_A = matrix([
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]])
```

Matriz transposta e inversa

A função *transpose* da biblioteca numpy permite gerar a transposta de qualquer matriz. Dessa forma, o commando *transpose(A)* gera a matriz transposta de A.

```
# Cria a matriz tranposta de A  
matriz_A_t = transpose(matriz_A)
```

Matriz transposta e inversa

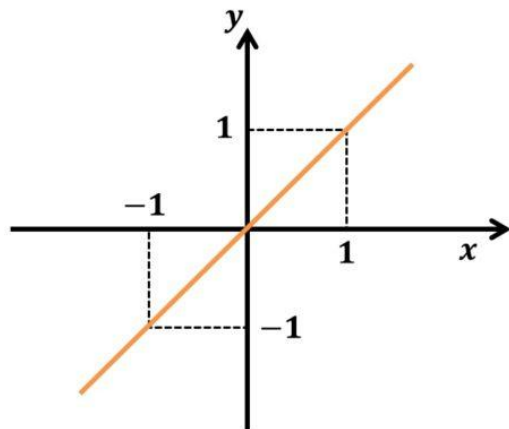
A operação *inv*, da função `linalg`, permite calcular a matriz inversa.

```
# Cria matriz inversa de A  
matriz_A_inv = linalg.inv(matriz_A)
```

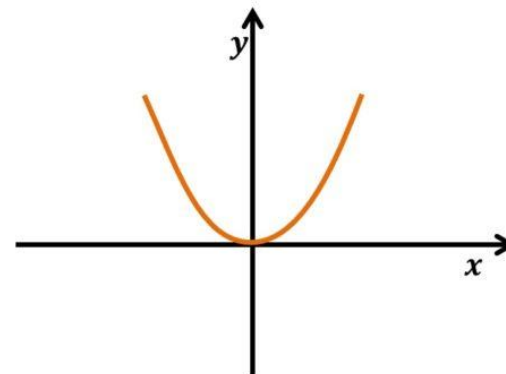


Funções

$$f(x) = x$$



$$f(x) = x^2$$



Sobre Funções

As funções matemáticas são facilmente tratadas na linguagem Python. Basta que no programa ela seja definida para depois fazer sua manipulação.

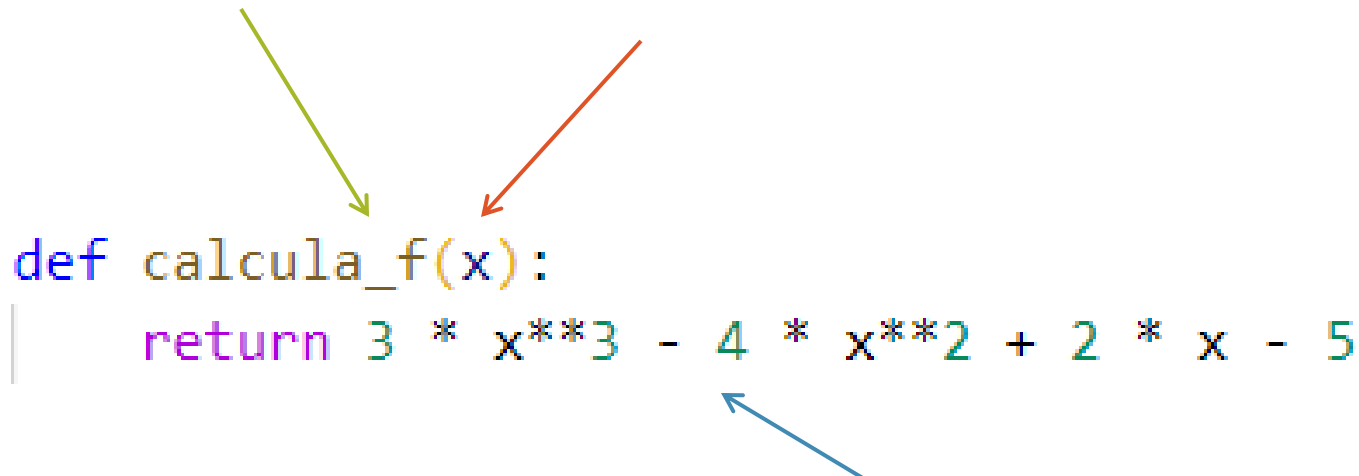
Uma função pode ser declarada usando o comando *def* e estabelecendo qual será o valor que retornará usando o comando *return*.

```
def calcula_f(x):  
    return 3 * x**3 - 4 * x**2 + 2 * x - 5
```

Sobre Funções

As funções definidas pode ser de quantas variáveis independentes se desejar. Por isso, basta que variáveis sejam separadas por vírgulas entre parênteses logo após escrever o nome da função.

Nome da função **Variável independente**



```
def calcula_f(x):  
    return 3 * x**3 - 4 * x**2 + 2 * x - 5
```

Retorno da minha função

Sobre Funções

A biblioteca *math* possui várias funções trigonométricas, tanto diretas quanto inversas, que podem ser usadas. Em todas elas, os ângulos são tratados em sua forma em radianos.

Porém essa biblioteca também possui funções para conversão de ângulos.

```
# Usa função seno. Argumento em graus  
angulo = radians(30)
```

```
# Usa função cosseno Argumento em graus  
y = cos(radians(120))
```




Equação e Inequação

$$3^2 - 5 \times 3 + 6 = 0 \Rightarrow$$

$$9 - 15 + 6 = 0 \Rightarrow$$

$$9 - 9 = 0 \Rightarrow$$

$$0 = 0$$

Sobre Equação e Inequações

Para solução de equações, o primeiro passo é declarar a função que representa a equação que se deseja resolver. Para isso, pode ser usado o comando *def*, e a solução buscada é igualando a função declarada a 0.

Dessa forma, na declaração da função, deve-se considerar sempre uma equação que a iguale a 0, fazendo os ajustes necessários.

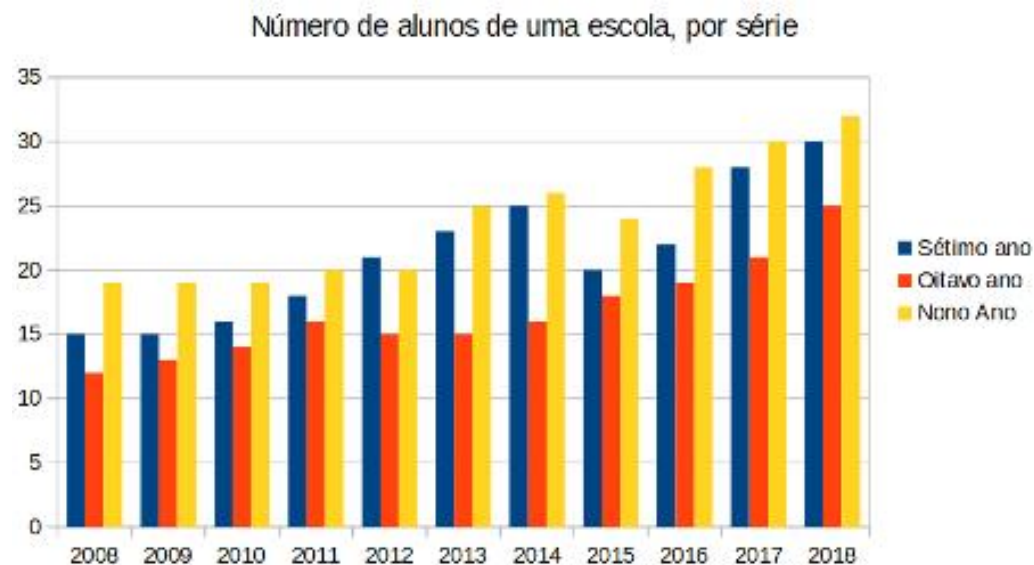
$$2x - 3 = 2 \rightarrow 2x - 3 - 2 \text{ or } 2x - 5$$

Sobre Equação e Inequações

A função usada na solução é a *solve* da biblioteca sympy. Porém, antes de executá-la, é preciso declarar variáveis independentes como símbolos, usando o comando `Symbol`, da mesma biblioteca.



Gráficos





Limites

Limites

Para o cálculo de limite, pode ser usada a função *limit* da biblioteca sympy. Os passos a serem seguidos são:

- Definir uma função
- Definir a variável independente como um símbolo (Symbol)
- Montar a operação limite (*limit*)
- Executar a operação (*doit*)

Limites

Para definir a variável independente como um símbolo, basta indicá-la entre aspas simples na função *Symbol*. Por exemplo, *Symbol('x')* indica que *x* é variável independente da função que será avaliada pela operação limite.

***Limit*(nome_função, variável_independente, valor)**

Limites

Para permitir os limites tendendo ao infinito ou $-\infty$, a biblioteca sympy possui a função S , da forma que:

- $S.Infinity$ indica infinito
- $-S.Infinity$ indica $-\infty$

Limites

$$\lim_{x \rightarrow 3} 1/x$$

$$\lim_{x \rightarrow +\infty} 1/x$$

$$\lim_{x \rightarrow 1} x^3 - 1/x - 1$$

$$\lim_{x \rightarrow \pi/6} \sin 2x / x$$

Material complementar

- Texto: Por que programar é o novo 'aprender inglês'
 - <https://www.nexojornal.com.br/expresso/2017/04/02/Por-que-programar-%C3%A9-o-novo-aprender-ingl%C3%AAs>
- Vídeo: Por que todos deveriam aprender a programar?
 - <https://www.youtube.com/watch?v=mHW1Hsqli6A>
- Vídeo: Curso em Vídeo: Curso Python #01 - Seja um Programador
 - <https://www.youtube.com/watch?v=S9uPNppGsGo>



Obrigado e até a próxima!



igsantos1996@gmail.com