UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE MATEMÁTICA

CURSO DE LICENCIATURA EM MATEMÁTICA

THIAGO TABOSA DE SOUZA

UMA BREVE INTRODUÇÃO À LINGUAGEM PYTHON PARA PROFESSORES DE MATEMÁTICA

CAUCAIA

2020

THIAGO TABOSA DE SOUZA

UMA BREVE INTRODUÇÃO À LINGUAGEM PYTHON PARA PROFESSORES DE MATEMÁTICA

Monografia apresentada ao curso de Licenciatura em Matemática da Universidade Federal do Ceará, como requisito para a obtenção do título de Graduado em Licenciatura em Matemática.

Coordenador: Prof. Jorge Carvalho Brandão
Orientador: Prof. Eliseu do Nascimento Silva

CAUCAIA
2020

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

T117b Tabosa de Souza, Thiago.
Uma breve introdução à linguagem Python para professores de matemática / Thiago Tabosa de Souza. – 2020.
33 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Ciências, Curso de Matemática, Fortaleza, 2020.
Orientação: Prof. Dr. Jorge Carvalho Brandão.
Coorientação: Prof. Me. Eliseu do Nascimento Silva.

1. Matemática. 2. Linguagem de programação. 3. Python. 4. Numpy. I. Título.

CDD 510

THIAGO TABOSA DE SOUZA

UMA BREVE INTRODUÇÃO À LINGUAGEM PYTHON PARA PROFESSORES DE MATEMÁTICA

Monografia apresentada ao curso de Licenciatura em Matemática da Universidade Federal do Ceará, como requisito para a obtenção do título de Graduado em Licenciatura em Matemática.

Aprovado em: ___/___/_____.

BANCA EXAMINADORA

_____

Prof. Jorge Carvalho Brandão (Orientador)
Universidade Federal do Ceará (UFC)

_____

Prof. Eliseu do Nascimento Silva
Universidade Federal do Ceará (UFC)

À Deus.

Aos meus pais e a minha querida esposa Lidiane de Almeida.

AGRADECIMENTOS

Aos professores Jorge Carvalho Brandão e Eliseu do Nascimento Silva, pela excelente orientação e que também estão participando da banca examinadora.

Aos colegas da turma, pelas reflexões, críticas e sugestões recebidas.

“Quem ousa ensinar nunca deve parar de aprender.”

(John Cotton Dana (texto traduzido)

RESUMO

A ideia desse trabalho é mostrar a simplicidade da linguagem de programação *Pyhton* para um professor de matemática que não saiba nenhuma das linguagens de programação. Falar sobre esse assunto para um professor que não possui prévio conhecimento em programação pode ser algo bastante aterrorizante. Dessa forma, surge a pretensão em demonstrar que programar em *Python* é mais simples do que se imagina e que essa linguagem pode ser utilizada para a resolução de vários problemas matemáticos. Durante essa dissertação, o computador será utilizado como ferramenta para resolver três problemas simples envolvendo assuntos comuns na rotina de trabalho de um professor de matemática, sendo eles: calcular as raízes de uma equação do 2º grau pela Fórmula de *Bhaskara*; calcular o fatorial de um número e calcular a matriz inversa de uma matriz. Esses problemas foram escolhidos para servir como apresentação inicial à alguns conceitos básicos de como a linguagem funciona, demonstrando em seguida um pouco a força da linguagem *Python* aplicada na matemática.

Palavras-chave: Matemática. Linguagem de Programação. *Python*. *Numpy*.

ABSTRACT

The idea of this work is to show the simplicity of the Pyhton programming language for a mathematics teacher who does not know any of the programming languages. Talking about it to a teacher who has no previous programming knowledge can be quite terrifying. Thus, the intention arises to demonstrate that programming in Python is simpler than imagined and that this language can be used to solve various mathematical problems. During this dissertation, the computer will be used as a tool to solve three simple problems involving common subjects in the work routine of a mathematics teacher, namely: calculating the roots of a 2nd degree equation using the Bhaskara Formula; calculate the factorial of a number and calculate the inverse matrix of a matrix. These problems were chosen to serve as an initial presentation to some basic concepts of how the language works, and then demonstrate a little the strength of the Python language applied in mathematics.

Keywords: Mathematics. Programming Language. Python. Numpy.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

BNCC	Base Nacional Comum Curricular

SUMÁRIO

1 INTRODUÇÃO

Há exatos 40 anos Seymour Papert lançava o seu livro chamado *Mindstorms: Children, Computers, and Powerful Idea*s. Essa obra é considerada uma verdadeira pedra fundamental para a inserção do computador no contexto educacional. Baseando-se nas ideias de construir o conhecimento por meio das teorias de Jean Piaget, Papert criou a linguagem de programação LOGO que serve de base para o ensino de programação para crianças no ensino básico. (PAPERT, 1980).

Desde então a importância em aprender uma linguagem de programação aumentou e vários países no mundo investem nesse caminho para a melhoria na qualidade da educação de uma maneira geral. Cita-se como exemplo o Reino Unido que investiu no ano de 2018 a quantia de cem milhões de libras (equivalente hoje a mais de setecentos milhões de reais) para aprimorar o ensino de programação em todas as escolas do país. (ALVES, 2018).

Além desse caso, podemos citar países como a Austrália, Alemanha e Estados Unidos que também estão se enveredando pelo mesmo caminho do Reino Unido (PIVA, 2016). A importância que esses países prestam a esse assunto não é algo descabido ou exagerado. Estudos mostram que a nível mundial, 92% (noventa e dois por cento) dos futuros empregos exigirão competências digitais e 45% (quarenta e cinco por cento) exigirão trabalhadores que possam configurar e trabalhar com confiança em sistemas e tecnologias digitais. (GOOGLE, [2018]). Isso demonstra que investir no ensino de linguagens de programação é, de fato, investir no próprio futuro.

No Brasil, a nossa BNCC (MEC, 1996) discorre sobre dez competências gerais da educação, destaca-se a competência número cinco:

> Compreender, utilizar e criar tecnologias digitais de informação e comunicação de forma crítica, significativa, reflexiva e ética nas diversas práticas sociais (incluindo as escolares) para se comunicar, acessar e disseminar informações, produzir conhecimentos, resolver problemas e exercer protagonismo e autoria na vida pessoal e coletiva.

A ideia supracitada comunga com o que chamamos de pensamento computacional. De acordo com Jeannette Wing (1999) diretora do Instituto de Ciência de Dados da Universidade de Columbia, o pensamento computacional é uma forma para seres humanos resolverem problemas; não é tentar fazer com que seres humanos pensem como computadores. Computadores são tediosos e enfadonhos; humanos são espertos e imaginativos. Nós humanos tornamos a computação empolgante. Equipados com aparelhos computacionais, usamos nossa inteligência para resolver problemas que não ousaríamos sequer tentar antes da era da

computação e construir sistemas com funcionalidades limitadas apenas pela nossa imaginação. (WING, 2016).

Diante dessa contextualização faz-se urgente a necessidade do professor de matemática utilizar linguagens de programação para mostrar aos seus alunos como ela é uma ferramenta extraordinária e como pode ser utilizada para a solução de inúmeros problemas envolvendo os conteúdos ensinados na sala de aula.

Infelizmente um grande obstáculo para essa ampla utilização parte do fato de que o próprio professor não conhece nenhuma linguagem de programação, todavia, usam com certa destreza e facilidade o computador nas suas atividades diárias, contudo, não possuem conhecimento profundo sobre o tema. Existe um censo comum em pensar que aprender a programar computadores seria algo complexo, moroso e que exigiria demasiado esforço para aprender. Durante esse trabalho, tentaremos desmistificar essa ideia e mostrar que a programação de computadores pode ser algo bastante parecido com alguns conceitos e ideias que são comumente disseminadas pelos educadores. Em alguns casos, bastaria apenas uma "tradução" da linguagem matemática para a linguagem computacional.

Uma pergunta se faz necessária nesse momento: no meio de tantas linguagens de programação existentes qual delas utilizaremos? Para responder essa pergunta é quase imediato pensarmos na linguagem LOGO criada por Papert. De fato, ela é deveras interessante, principalmente quando aplicada às crianças pequenas durante a educação básica. Entretanto, essa linguagem apresenta algumas restrições que reduzem o seu potencial limitando-o.

Inicialmente a linguagem LOGO era destinada para uso gráfico e visual com a utilização da tartaruga se movendo pela tela. Posteriormente o LOGO foi aperfeiçoado e passou a permitir a utilização de fórmulas, textos, etc. De qualquer modo, por ser bastante pragmático não iremos ver nenhum programador profissional utilizar a linguagem LOGO nos seus trabalhos cotidianos. No mundo real outras linguagens de programação se apresentam mais interessantes, pois, são mais robustas, completas e fornecem uma maior versatilidade de uso por parte do programador (PAPER, 1988).

Hoje em dia a linguagem *Python* é mais usada a nível global. Num *ranking* das vinte linguagens mais populares atualmente, o *Python* se encontra em segundo lugar empatado com o *Java* e perdendo apenas para o *Java script* (VENTURA, 2020).

Em termos de números, existem mais de duzentos e setenta mil projetos que envolvem essa linguagem. A abrangência do *Python* é enorme, por exemplo, existem bibliotecas que são utilizadas para a criação de jogos de inteligência artificial, *big data*, plotagem de gráficos, cálculo numérico, etc. Aliado a todo esse poder, o *Python* é também uma

das linguagens mais fáceis e amigáveis para se aprender (PYPI, 2020). Devido a junção dessas características, será essa a linguagem que utilizaremos.

No próximo capítulo aprenderemos uma noção geral do que seria exatamente um algoritmo. Esse conceito é importante pois toda e qualquer linguagem de programação que existe funciona através de algoritmos, com essa idealização feita partiremos para o *Python* propriamente dito. O nosso foco será em resolver três problemas matemáticos explicando algumas ideias e características da linguagem que existem por trás de cada um desses problemas, sendo eles: calcular as raízes de uma equação do 2º grau pela fórmula de *Bhaskara*, calcular o fatorial de um número e calcular a matriz inversa de uma matriz.

2 ALGORITMO

Antes de iniciarmos é importante salientar que a origem do algoritmo não se deu com o advento dos computadores, ele advém de tempos remotos. Um dos algoritmos mais antigos que se tem notícia é o chamado algoritmo de Euclides, encontrado em seu livro denominado “Os Elementos”. O matemático grego lançou essa obra no ano de 300 a.C. (trezentos antes de Cristo) (WIKIPEDIA, 2020). No caso, o algoritmo de Euclides é um método simples e eficiente para se calcular o máximo divisor comum de dois números distintos entre si (WIKIPEDIA, 2018).

De acordo com a enciclopédia *Britannica* (2016), a definição de um algoritmo seria: “procedimento sistemático que produz - em um número finito de etapas - a resposta a uma pergunta ou a solução de um problema.”

Uma associação bastante utilizada para adquirir melhor entendimento seria exatamente um algoritmo consistente na comparação com uma receita de bolo. Na receita existem os ingredientes e o modo de preparo, ou seja, o passo a passo da receita. Nessa comparação os ingredientes seriam os dados e as variáveis do programa e o modo de preparo seria o desenvolvimento do próprio algoritmo em si, ou seja, a sequência de passos a ser realizada pelo computador para atingir o resultado almejado.

Pela definição supracitada é fácil perceber que um professor de matemática usa vários algoritmos no seu dia a dia em sala de aula, o tempo todo. Vamos observar mais atentamente o caso de quando um professor ensina uma criança a fazer um cálculo somando números com dois ou mais dígitos. A título de ilustração, vamos somar 157 + 87. O professor irá fazer exatamente essa conta na lousa:

$$\begin{array}{cccc} & {\scriptstyle 1} & {\scriptstyle 1} & \\ & 1 & 5 & 7 \\ + & 8 & 7 & \\ \hline & 2 & 4 & 4 \end{array}$$

Nesse caso os dados de entrada do problema serão os dois números que devemos somar. O algoritmo começa a ser feito quando “armamos” a conta dispondo os dois números, um em cima do outro, onde o maior deles ficará na parte de cima e o menor na parte de baixo. Após a conta estar armada começaremos os cálculos, iremos agora somar os dígitos coluna a coluna. Primeiramente, somaremos 7 + 7 = 14.

Como não podemos colocar esse número de dois dígitos de uma só vez, devemos ensinar a regra do “vai um” para cima da próxima coluna de dígitos a serem somados. Temos

agora que somar a próxima coluna, ou seja, 1 + 5 + 8 = 14. Novamente, usaremos a ideia do “vai um” e, por fim, somaremos 1 + 1 = 2. Pronto, temos o resultado buscado, ou seja, a soma de 157 + 87 é igual a 244. É fácil observar que todo esse processo se enquadra perfeitamente na definição de algoritmo supracitada. Temos os dados a serem usados e a sequência a ser feita para a obtenção do resultado procurado.

O exemplo realizado é bastante simples, porém, ele consegue nos ajudar a perceber que o professor de matemática usa, muitas vezes sem perceber, diversos algoritmos. Que são usados para calcular raízes de equações, área de figuras, multiplicação de matrizes, probabilidades, etc. De modo genérico, o professor bastaria fazer uma “tradução” dos conceitos matemáticos plenamente dominados por ele para a linguagem de programação que ele desejar aprender.

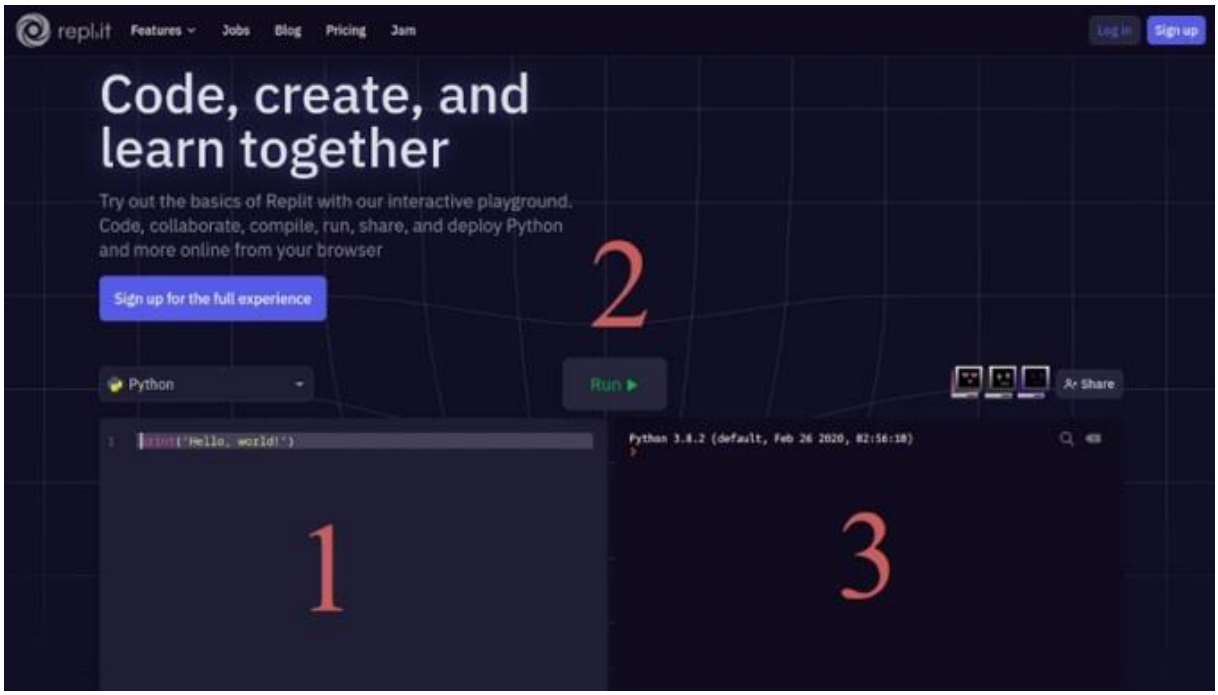
3 PYTHON

Agora que sabemos o que seria um algoritmo, vamos começar a programação em *Python* especificamente. Antes, devemos elucidar algumas dúvidas que provavelmente aparecerão durante essa caminhada. O material necessário que precisaremos ter em mãos para o acompanhamento do conteúdo consiste apenas num computador com acesso à *internet* e um navegador *web*. Não precisaremos instalar nenhum programa ou realizar nenhuma configuração adicional. Originalmente para poder programar em *Python* precisaríamos instalá-la. Entretanto, graças aos avanços da chamada *Web* 2.0 atualmente é possível desenvolver a criação e execução de programas a partir do próprio navegador de *internet* (WIKIPEDIA, 2020).

Aos que necessitam adotar uma utilização mais avançada ou maior performance durante a execução dos programas ainda será necessário a instalação da linguagem no computador. Contudo, para o uso comum essa necessidade não se verificará. A ideia seria iniciar os estudos de forma mais simples e caso a surja a necessidade pode-se buscar outras fontes de informações e livros para maior aprofundamento. A página da *internet* que precisaremos acessar para a criação dos programas será: https://repl.it/languages/python3

Quando abrimos a página percebemos que ela é dividida em três partes distintas.

Figura 1 - Página para criação dos programas



Fonte: Repl.it (2020)

A parte 1 é o espaço onde escreveremos o algoritmo em *Python*, também chamada área de edição. A parte 2 é o botão que iremos clicar para compilar o código. O computador é

uma máquina que funciona pela lógica binária, ou seja, ela apenas entende zeros e uns. Essa linguagem é denominada de linguagem de máquina. (WIKIVERSIDADE, 2011).

O compilador transformará todo o código que acabamos de escrever em linguagem de máquina. Finalmente, a parte 3 será a execução do programa, também chamada área de console. Para entendermos melhor todo o processo observe que a parte 1 já contém um algoritmo escrito. Ao clicarmos na parte 2 o código será compilado e o resultado da compilação será apresentado na parte 3. Observe que a parte 3 mudará e irá escrever na tela as palavras "*Hello, world*!".

Esse primeiro programa que acabamos de executar é o chamado "*Hello, world*!", muito conhecido no mundo da programação, pois, geralmente é o primeiro algoritmo a ser escrito e ensinado em vários livros e apostilas de programação em diversas linguagens distintas. As linguagens de programação permitem o que chamamos de entrada e saída de dados, a entrada seriam os dados iniciais que digitamos via teclado do computador.

O algoritmo recebe esses dados criando variáveis internas para manipulá-las, calcular o que foi estabelecido no algoritmo e, por fim, apresentar na tela o resultado calculado. Essa apresentação dos resultados obtidos na tela é o que chamamos de saída de dados. Esse primeiro programa mostra exatamente um tipo de saída de dados, no caso, o comando "*print*" é utilizado para imprimir na tela alguma coisa. Após o comando "*print*" o programa escreveu "("*Hello, world*!")". O comando "*print*" exige que se coloque os parênteses para delimitar o que será impresso na tela e o que está dentro das aspas duplas é considerado um texto comum. "*Hello, world*", que significa "Olá, mundo!" em português.

Outra observação que devemos ter sobre a linguagem *Python*, ela é o que chamamos de "*case sensitive*", ou seja, ela diferencia palavras escritas em maiúsculas de minúsculas. O comando acima que escrevemos, o "*print*", só pode ser digitado exatamente dessa forma. Se digitarmos, por exemplo, o comando "*Print*" ou "*PRINT*" a linguagem *Python* não irá reconhecer esses comandos e apresentará erro.

Outro detalhe importante para salientar diz respeito as declarações de variáveis em *Python*. Elas são feitas de forma simples e direta, sem a necessidade de definir o seu tipo. Vamos supor que queremos calcular a área da circunferência que utiliza o número pi (π). Uma maneira possível para a declaração desse valor seria apenas escrevendo: "pi = 3.14", somente. Atente que utilizamos o ponto no lugar da vírgula para separar a parte inteira da parte decimal do número. Esse padrão é o mesmo da língua inglesa, o idioma de origem da linguagem. Por esse motivo o ponto é utilizado em substituição da vírgula.

A matemática possui alguns conjuntos de números diferentes entre si. Temos o conjunto dos números inteiros, naturais, complexos, irracionais… No caso do *Python*, trabalharemos basicamente apenas com dois conjuntos: "*int*" e "*float*". Utilizaremos o "*int*" quando estivermos calculando com números inteiros e o "*float*" quando estivermos trabalhando com números em ponto flutuante, ou seja, com números reais não inteiros. A declaração da variável "pi" acima é uma forma implícita de declaração.

Outra forma possível seria declará-la explicitamente. Para isso, poderíamos escrever "pi = float(3.14)". Dessa forma estamos deixando claro que a variável "pi" é um número em ponto flutuante. Todo computador possui um espaço físico restrito, ou seja, uma quantidade limitada de memória internamente. Como consequência direta disso, seria impossível para *Python* ou para qualquer outra linguagem de programação armazenar ou calcular números infinitos. Sabemos, pela matemática, que os números irracionais como o valor de π possuem infinitas casas decimais depois da vírgula. No caso, o *Python* consegue trabalhar máximo com uma boa aproximação desse valor, ou seja, com várias casas decimais após a virgula, mas não com todas as suas infinitas casas. Assim, como dito anteriormente, as variáveis diferenciam letras maiúsculas de minúsculas, então se digitarmos a variável "Pi" e "pi" elas serão reconhecidas como diferentes pelo sistema.

Por fim, as variáveis não podem possuir espaços em branco no meio do nome, por exemplo, variáveis da seguinte forma "pi 1", "pi 2", média dos alunos" não serão reconhecidas pelo sistema. Para escrever variáveis com nomes maiores é possível utilizar o caractere sublinhado para a separação. Assim, as variáveis acima ficariam "pi_1" , "pi_2" e "media_dos_alunos".

Vale ressaltar que a linguagem possui o caractere "#" que é um indicativo de comentário no código, ou seja, o que vier após esse símbolo não será executado pelo compilador. Esses comentários são úteis para tornar o algoritmo mais legível e explicado para quem for ler o código posteriormente.

Outro detalhe é sobre a chamada indentação. Ela informa que em alguns comandos específicos temos que afastar o texto da sua margem esquerda inicial e escrever à direita dessa margem. Ela é utilizada na linguagem para indicar uma certa hierarquia na sequência de execução dos comandos. Quando estivermos falando do comando "*while*" iremos destacar um pouco mais esses detalhes sobre a indentação.

Após fazer algumas ponderações sobre a linguagem é possível partir para a escrita do primeiro programa.

3.1 CALCULAR AS RAÍZES DE UMA EQUAÇÃO DO 2º GRAU PELA FÓRMULA DE *BHASKARA*

Pela matemática, sabemos que uma equação do 2º grau possui o seguinte aspecto geral: $ax^2 + bx + c = 0$, onde a, b, c são os coeficientes da equação e $a \neq 0$. A fórmula de *Bhaskara* nos diz que devemos calcular o valor de Δ que é dado por (SANTOS, 2018):

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

Sejam x_1 e x_2 as duas raízes da equação. Elas podem ser calculadas como:

$$x_1 = \frac{-b+\sqrt{\Delta}}{2a} \text{ e } x_2 = \frac{-b-\sqrt{\Delta}}{2a}$$

Já que relembramos a fórmula, podemos então calcular esse seguinte exemplo simples: $x^2 + 2x - 24$

$$\Delta = b^2 - 4ac = 4 + 96 = 100$$

Agora vamos achar as raízes:

$$x_1 = \frac{-b+\sqrt{\Delta}}{2a} = \frac{-2+10}{2} = 4 \text{ e } x_2 = \frac{-b-\sqrt{\Delta}}{2a} = \frac{-2-10}{2} = -6$$

Já que relembramos um pouco da matemática, passaremos ao primeiro algoritmo em *Python*. Para isso, observe que precisamos criar as variáveis de entrada *a*, *b*, *c*. Para em seguida criar as variáveis chamadas delta, x_1 e x_2 e calcular os seus valores.

Observe que a função estudada possui tanto cálculos envolvendo potência como radiciação. Antes de explicar as duas operações é preciso mostrar as quatro operações mais simples de soma, subtração, multiplicação e divisão.

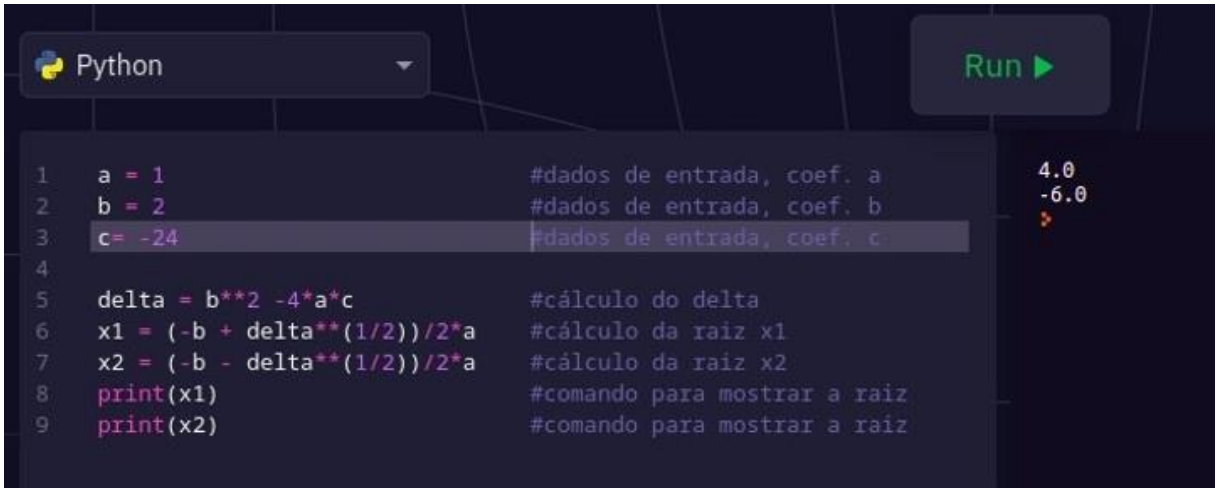
A linguagem possui o sinal de "+" para indicar a soma, o sinal de "-" para indicar a subtração, o sinal de "*" para a multiplicação e, por fim, o sinal de "/" para a divisão. O *Python* não possui os colchetes e chaves como nas expressões matemáticas, nesse caso elas viram apenas parênteses. Além disso a linguagem reconhece a ordem matemática das operações, então o *Python* resolverá es expressões numéricas na ordem correta, ou seja, as potências, depois as raízes, seguido da multiplicação e divisão e, por último, a adição e subtração.

Assim, expressões como $\{[(1 + 2.3).2 + 4] - 10\} + 5$ virariam"(((1 + 2*3)*2 + 4) – 10) +5" e a linguagem calcularia o valor correto do termo entre parênteses que seria 7 e não 9. A linguagem possui um caractere especial para indicar que o usuário deseja elevar um valor pelo outro usando o comando "**". Além disso, a mesma também funciona para calcular a radiciação, pois, sabemos que $\sqrt{2} = 2_2^1$. Observe logo abaixo algumas potências e raízes em matemática e como estas ficariam escritas em *Python*.

Matemática	*Python*
2^2	$2^{**}2$
2^{4+4}	$2^{**}2\ (4+4)$
$\sqrt[3]{(2^2+4)^2}$	$2^{**}2+4)\ **\ (2/3)$

Após realizar todas as ponderações e explicações, é chegada a hora de criar o primeiro programa para calcular as raízes de uma equação do 2º grau. Tomando como exemplo a função calculada matematicamente acima, a função $x^2 + 2x - 24$. Observe abaixo um exemplo de fácil compreensão explicando como o código seria.

Figura 2 - Primeira versão do programa para calcular a fórmula de Bhaskara

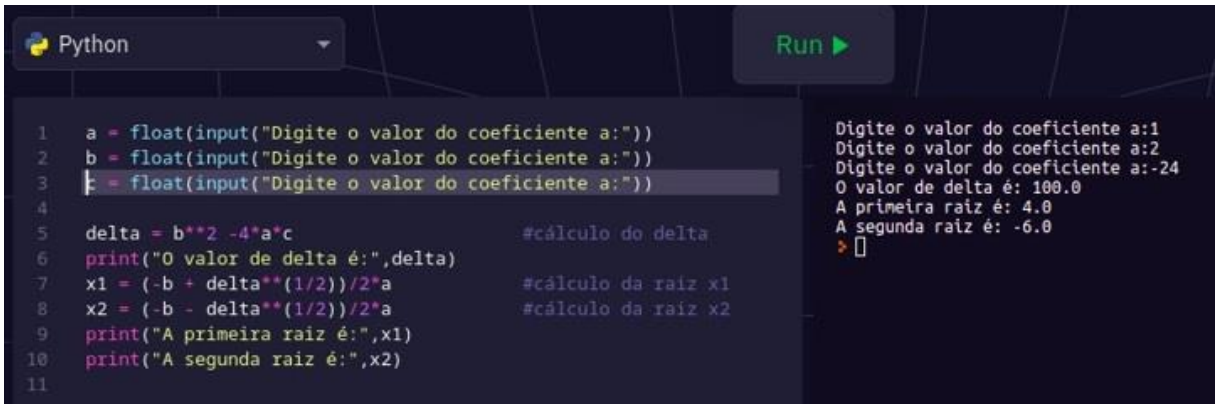


Fonte: Fonte: Repl.it (2020)

Atente que nas linhas 1, 2 e 3 são declaradas as variáveis a, b e c para receber os coeficientes 1, 2 e -24 da equação $x^2 + 2x - 24$. Na linha 5 calcula-se o valor do delta e nas linhas 6 e 7 são calculadas as duas raízes da equação. Por fim, nas linhas 8 e 9 o computador recebe a ordem para apresentar as duas raízes da equação pelo comando "*print*". Esse resultado apareceu logo ao lado na área do console, ondem aparecem os valores 4.0 e -6.0. O programa foi executado perfeitamente atingindo o seu objetivo de calcular as raízes da equação.

Após fazer o primeiro programa, é preciso aperfeiçoá-lo para que seja possível digitar os valores dos coeficientes a, b e c. Desse modo, o programa poderá calcular qualquer equação do 2º grau e não somente aquela especificada acima. Para isso, é necessário utilizar o comando chamado "*input*", que permite uma interação entre o usuário e o programa. Além do comando "*print*" que pode escrever uma mensagem na tela, o comando "*input*" também pode realizar a mesma função, apesar do seu foco ser recebimento de algo digitado para interagir com o usuário. Segue imagem do programa atualizado e melhorado:

Figura 3 - Segunda versão do programa para calcular a fórmula de Bhaskara



Fonte: Fonte: Repl.it (2020)

Observe a mensagem escrita entre “” dentro do comando “*input*” nas linhas 1, 2 e 3. Além disso, usa-se o comando “*float*()” antes do comando “*input*()” para indicar que a variável “a” será do tipo ponto flutuante, dessa forma é possível calcular as raízes de uma gama maior de funções e não apenas aquelas que possuem raízes inteiras.

3.2 CALCULAR O FATORIAL DE UM NÚMERO

Primeiramente vamos relembrar a parte matemática envolvendo o problema do cálculo fatorial. Calcular um fatorial só faz sentido quando trabalhado em cima dos números naturais. Sendo representado pelo símbolo “!” indicando que esse número natural será multiplicado por todos os seus antecessores até que chegue a 1. Segue abaixo os fatoriais de zero a 3 para melhor entendimento:

$$0! = 1$$

$$1! = 1$$

$$2! = 2.1 = 2$$

$$3! = 3.2.1 = 6$$

Agora que relembramos esse conceito, prosseguiremos à linguagem de programação propriamente dita. Antes, porém, apresentaremos um comando novo que será útil para o desenvolvimento do programa, trata-se do comando “*while*” que serve para criar estruturas de repetição. A estrutura desse comando gira em torno de pôr uma condição após o “*while*”, ou seja, uma sentença que pode ser verdadeira ou falsa. Se verdadeira, ela entrará dentro do “*while*” e executará os comandos que estão internos a ele.

A forma que o *Python* separa o que está dentro e fora do comando "*while*" é utilizando da endentação, comentada anteriormente. Segue apresentação do algoritmo.

Figura 4 - Primeira versão do programa para calcular fatorial

Fonte: Fonte: Repl.it (2020)

Na linha 1 mandamos o computador digitar uma frase ao usuário e receber o número que será digitado por ele. Chamamos essa variável de "n". Além dessa variável, criamos outras 2 variáveis. A variável "*temp*" que será utilizada apenas dentro do comando "*while*" e a variável "*fat*" que acumulará o resultado do cálculo do fatorial. Essas três variáveis inicialmente ficaram com o mesmo valor de "n". Na linha 4, observe o comando "*while*", após ele, aparece o comando "temp != 1 :".

Nessa condição estamos perguntando ao computador se a variável "*temp*" é diferente do número 1, essa afirmação pode ser verdadeira ou falsa. Supondo que o usuário digite um número maior que 1, por exemplo o número 3, a condição será verdadeira e a linguagem *Python* entrará para executar o que estiver dentro do comando "*while*".

Por fim, o comando "*while*" precisa do ":" para indicar o fim da condicional e início do comando interno do laço. Observe que as linhas 5 e 6 estão afastadas ao centro quando comparadas com as outras linhas. Isso é o que chamamos de endentação, ela serve para indicar que essas duas linhas específicas são de dentro do comando "*while*" e só podem ser executadas quando a condição for verdadeira. Quando ela for falsa, o programa pulará essas duas linha e irá para a linha de código seguinte.

Por isso o comando "*while*" é chamado estrutura de repetição. Enquanto a condição for verdadeira será executada e repetida. Deve-se dispor de cuidado extra com essas estruturas pois sendo condição sempre verdadeira, será repetida fazendo com que o programa trave e fique preso nessas infinitas repetições. Para evitar esse problema adicionamos a linha 5 do código. Essa linha garante que a variável "*temp*" irá diminuir até se tornar igual a 1, o que deixa a condicional falsa e, consequentemente, faz com que o programa não trave e saia do "*while*".

Nesse momento surge o seguinte questionamento: Se o usuário digitar o número zero ou um número menor ainda como o -2 ? O programa obviamente irá travar e se repetir indefinidamente dentro do comando "*while*" pois a condicional foi na possibilidade de o número digitado pelo usuário ser maior do que 1. Como o número é menor do que 1 e, pela linha 5, ele ficará menor ainda, sempre será diferente de 1 e nunca sairá de dentro do "*while*".

Esse problema precisa ser resolvido através de uma nova estrutura baseada no comando "*if*", que representa a estrutura condicional. Esse comando possui certas semelhanças com o comando "*while*" e também apresenta uma condição a ser testada internamente para a execução. Vamos ao código do fatorial incrementado pelo comando "*if*":

Figura 5 - Segunda versão do programa para calcular fatorial com o comando if

Fonte: Fonte: Repl.it (2020)

Na linha 2, logo após receber a variável "n" digitada pelo usuário, utilizamos o comando "*if*" para testar se o número é maior do que zero. Se assim for, a linguagem executará a mesma sequência de comandos realizada anteriormente. Caso contrário, o comando "*else*" será executado e mostrará uma mensagem ao usuário pedindo para que digite um número natural da próxima vez.

Para encerrar esse capítulo, segue abaixo a forma de fazer alguns comparativos em *Python*:

Matemática	*Python*
=	==
≠	! =
>	>
<	<
≥	>=
≤	<=

3.3 CALCULAR A MATRIZ INVERSA DE UMA MATRIZ

Para que uma matriz qualquer possa ter uma inversa, o seu determinante precisa ser diferente de zero. O segundo passo do cálculo, supondo que atenda a condicional do determinante, seria multiplicar a matriz procurada pela matriz que fornecida e igualar a matriz identidade. Observe um exemplo simples composto por matriz de ordem 2 para relembrar os conceitos.

Seja a matriz $A = \begin{pmatrix} 2 & 1 \\ 5 & 3 \end{pmatrix}$

Calculando o seu determinante, temos: $det(A) = 3.2 - 1.5 = 6 - 5 = 1$. Como o determinante é diferente de zero, é possível calcular a matriz inversa. Vamos resolver a seguinte expressão:

$$\begin{pmatrix} 2 & 1 \\ 5 & 3 \end{pmatrix} \cdot \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 2a + c & 2b + d \\ 5a + 3c & 5b + 3d \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

A partir de agora é preciso resolver os dois sistemas:

$$\begin{cases} 2a + c = 1 \\ 5a + 3x = 0 \end{cases} \text{ e } \begin{cases} 2b + d = 0 \\ 5b + 3d = 1 \end{cases}$$

Resolvendo os dois sistemas encontramos que $a = 3,\ b = -1,\ c = -5\ e\ d = 2$. Escrevendo a matriz, temos:

$$A^{-1} = \begin{pmatrix} 3 & -1 \\ -5 & 2 \end{pmatrix}$$

Observe que mesmo para uma matriz pequena e simples de ordem 2, calcular a inversa de uma matriz envolve a resolução de muitos cálculos.

Para resolver o problema na linguagem *Python*, poderíamos escrever algoritmos que refizessem todos os cálculos, todavia, isso não exploraria todas as possibilidades da linguagem. Um dos motivos do *Python* ser tão querido pelos programadores é que a linguagem possui uma grande comunidade de usuários por trás, que implementam várias bibliotecas de códigos e nos permitem reaproveitar funções e algoritmos. Dessa forma, não precisamos nos preocupar em sempre realizar todas as funções, reaproveitando o que algumas bibliotecas podem oferecer para facilitar o trabalho.

O comando utilizado para realizar essa tarefa é o "*import*". O *Python* possui várias bibliotecas para diversas funções. Nesse caso de matrizes, em particular, utilizaremos a

biblioteca chamada *Numpy*. Essa biblioteca permite trabalhar com várias funções envolvendo matrizes. Dentre eles, podemos destacar a função chamada, “numpy.matrix” para criar uma matriz, a função chamada “numpy.linalg.det” para calcular o determinante da matriz e a função “numpy.linalg.inv” para calcular a inversa da matriz.

Vamos escrever um algoritmo que pegue a matriz “a” acima, calcule o seu determinante e depois mostre a sua matriz inversa utilizando essas funções:

Figura 6 - Primeira versão do programa para calcular determinante e matriz inversa de matriz de ordem 2

Fonte: Repl.it (2020)

Apesar da complexidade envolvendo esses cálculos, em apenas 7 linhas é possível fazer um programa para executá-los, porque a biblioteca *Numpy* antecipou essas funções e estamos apenas reaproveitando-as para facilitar o trabalho.

Dentro do texto escrito nas aspas do comando “*print*” na linha 7 há um termo “\n” diferente. Esse termo não aparece junto do texto na área de console e ele serve para indicarmos ao programa para saltar uma linha antes de continuar a escrita. No caso, antes de escrever a matriz inversa ele salta uma linha e apresenta a matriz na linha de baixo.

Para modificar o problema para uma matriz de ordem maior, basta mudar a matriz “a” que os cálculos seriam os mesmos. Segue abaixo um exemplo de código apenas alterando a ordem e os elementos da matriz.

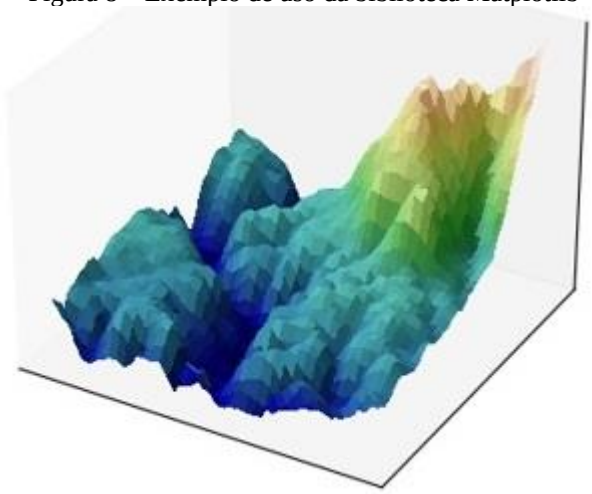
Figura 7 - Segunda versão do programa para calcular determinante e matriz inversa de matriz de ordem 3

Fonte: Repl.it (2020)

Além dessa incrível biblioteca, existe outra chamada *Sympy* que permite escrever e resolver expressões algébricas semelhantes ao programa *Geogebra* (GEOGEBRA, 2020). Outra biblioteca interessante seria a chamada *SciPy* que possibilita o cálculo de funções mais complexas como integrais e derivadas. Ela se torna uma espécie de extensão da biblioteca *Numpy* (SCIPY, 2020).

Por fim, existe a biblioteca chamada *Matplotlib* (2020) que permite a criação de gráficos complexos e bem feitos. Segue um exemplo dessa incrível engenharia utilizada pela *Matplotlib*:

Figura 8 – Exemplo de uso da biblioteca Matplotlib



Fonte: Matplotlib (2020)

Como a ideia desse trabalho é estabelecer uma breve introdução à linguagem, não exploraremos em detalhes todas as bibliotecas supracitadas apenas as apresentaremos para os usuários que desejarem se aprofundar e descobrir qual caminho seguir na busca dos seus conhecimentos.

4 CONCLUSÃO

A resolução dos três problemas propostos serviu como um alicerce para que fundamentássemos alguns conceitos básicos da linguagem e mostrar um caminho introdutório para o professor de matemática que desejar se aprofundar no mundo da linguagem *Python*, que demonstrou ser riquíssima e possuir funções inovadoras com a criação de novas bibliotecas, o que requer um esforço contínuo de usuários interessados em conhecer melhor a linguagem.

Acreditamos que, mesmo parecendo algo distante, o *Python* poderia ser melhor aproveitado dentro da escola, em especial pelo professor de matemática pela proximidade com conceitos, teorias e ideias matemáticas com a linguagem *Python*. Esse pequeno trabalho pode servir como um trampolim para o professor aprimorar seu conhecimento com a linguagem, fazer com que ele mergulhe nesse novo universo de conhecimento e trazer aos seus alunos o conhecimento extensivo que a *Python* possibilita.

REFERÊNCIAS

ALGORITMOS e Estruturas de Dados/O que é um Algoritmo?. **Wikipédia.** [S.l], 2020. Disponível em: https://pt.wikibooks.org/wiki/Algoritmos_e_Estruturas_de_Dados/O_que_%C3%A9_um_Algoritmo%3F. Acesso em: 26 out. 2020.

ALGORITMO. **Wikipédia.** [S.l], 2020. Disponível em: https://pt.wikipedia.org/wiki/Algoritmo. Acesso em: 18 out. 2020.

ALVES, Adriane. Como o Reino Unido pretende ensinar computação a todas as crianças. **Exame** [S.l], 2018. Disponível em: https://exame.com/tecnologia/como-o-reino-unido-pretende-ensinar-computacao-a-todas-as-criancas/. Acesso em: 08 nov. 2020.

ARK, Tom Vander. Stop Calculating And Start Teaching Computational Thinking. **FORBES**. [S.l], 2020. Disponível em: https://www.forbes.com/sites/tomvanderark/2020/06/29/stop-calculating-and-start-teaching-computational-thinking/?fbclid=IwAR0DsFmfgLG6ZbFZdENvkkcpQCu2Fe7m5ewO8WPYpITVqieQlKM7MyuXGA0&sh=1d0390c23786. Acesso em: 10 nov. 2020.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **ABNT NBR 10520**: apresentação de citações em documentos. Rio de Janeiro, 2002a.

_____. **ABNT NBR 12225**: títulos de lombada. Rio de Janeiro, 2004a.

_____. **ABNT NBR 14724**: trabalhos acadêmicos: apresentação. Rio de Janeiro, 2011a.

_____. **ABNT NBR 15287**: projeto de pesquisa: apresentação. Rio de Janeiro, 2011b.

_____. **ABNT NBR 6023**: referências: elaboração. Rio de Janeiro, 2002b.

_____. **ABNT NBR 6024**: numeração progressiva das seções de um documento. Rio de Janeiro, 2012a.

_____. **ABNT NBR 6027**: sumário. Rio de Janeiro, 2012b.

_____. **ABNT NBR 6028**: resumos. Rio de Janeiro, 2003.

_____. **ABNT NBR 6034:** índice. Rio de Janeiro, 2004b.

BNCC e programação de computadores: qual a relação?. **Instituto Ayrton Senna.** São Paulo, 2020. Disponível em: https://www.institutoayrtonsenna.org.br/pt-br/conteudos/estante-do-educador/BNCC-e-programacao-de-computadores-qual-a-relacao.html#:~:text=A%20programa%C3%A7%C3%A3o%20de%20computadores%20%C3%A9,plataformas%2C%20aplicativos%20e%20outras%20estruturas. Acesso em: 05 nov. 2020.

BRACKMANN, Christian Puhlmann. **Desenvolvimento do pensamento computacional através de atividades desplugadas na educação básica**. Rio Grande do Sul, 2017. 226 f. Disponível em: https://lume.ufrgs.br/bitstream/handle/10183/172208/001054290.pdf?sequence=1&isAllowed=y. Acesso em: 08 out. 2020.

_____. **Pensamento Computacional Brasil**. 2020. Disponível em: http://www.computacional.com.br/. Acesso em: 27 nov 2020.

COSTA, Rodrigo Alves; PEREIRA, **Rafaela Samara Oliveira. Ensino de programação na educação básica por meio do pensamento computacional: um relato de experiência**. Campina Grande – PB, 2019. Disponível em: https://www.brazilianjournals.com/index.php/BRJD/article/view/2715/2719. Acesso em: 26 nov. 2020.

EDUCAÇÃO é a base. **MEC** [S.n.t]. Disponível em: http://basenacionalcomum.mec.gov.br/. Acesso em: 15 nov. 2020

FERNANDES, Hugo Batista; SILVEIRA, Ismar Frango. **Pensamento computacional como auxílio a promoção da qualidade na EAD**. São Paulo, 2016. Disponível em: http://www.abed.org.br/congresso2016/trabalhos/205.pdf. Acesso em: Acesso em: 30 out. 2020.

GAROFALO, Débora. A linguagem de programação faz 50 anos, e ela precisa estar nas escolas. **Nova Escola**, 2017. Disponível em: https://novaescola.org.br/conteudo/9136/a-linguagem-de-programacao-faz-50-anos-e-ela-precisa-estar-nas-escolas. Acesso em: 12 nov. 2020.

_____. Como levar a programação para a sala de aula. **Nova Escola**, 2018. Disponível em: https://novaescola.org.br/conteudo/12303/como-levar-a-programacao-para-a-sala-de-aula. Acesso em: 06 nov. 2020.

_____. Por que ensinar programação na escola é importante. **UOL**, 2020. Disponível em: https://www.uol.com.br/ecoa/colunas/debora-garofalo/2020/09/16/por-que-ensinar-programacao-na-escola-e-importante.htm. Acesso em: 18 nov. 2020.

GEOGEBRA - Aplicativos Matemáticos. **Geogrebra**. [S.l], 2020. Disponível em: https://www.geogebra.org/. Acesso em: 31 out. 2020.

GOUVEIA, Rosimar. **Matriz inversa.** [S.l], 2019. Disponível em: https://www.todamateria.com.br/matriz-inversa/. Acesso em: 31 out. 2020.

IGOR. Phyton Tutorial. **DEVMEDIA.** [S.l], 2015. Disponível em: https://www.devmedia.com.br/python-tutorial/33274. Acesso em: 02 nov. 2020.

LINGUAGEM de Máquina. **Wikipédia**. [S.l], 2011. Disponível em: https://pt.wikiversity.org/wiki/Linguagem_de_M%C3%A1quina. Acesso em: 07 nov. 2020.

MARCONDES, Guilherme A. Barucke. **Matemática com Python**: um guia prático. São Paulo: Novatec, 2018.

MATPLOTLIB: Visualization with Python. **MATPLOTLIB**, 2012. Disponível em: https://matplotlib.org/. Acesso em: 05 out. 2020.

MINISTÉRIO DA EDUCAÇÃO, **Base nacional comum curricular:** educação é a base. Brasília, 1996. Disponível em: http://basenacionalcomum.mec.gov.br/images/BNCC_EI_EF_110518_versaofinal_site.pdf. Acesso em: 01 nov. 2020.

O futuro da sala de aula. **GOOGLE** [S.n.t]. Disponível em: http://services.google.com/fh/files/misc/future_of_the_classroom_br_pt_global_report.pdf?utm_source=web&utm_campaign=FY19-Q2-global-demandgen-website-other-futureoftheclassroom. Acesso em 12 nov. 2020.

PAPERT, Seymour. **Mindstorms: children, computers, and powerful ideas.** New York: Bacis Books, 1980

O Algoritmo de Euclides. **Wikipédia**. [S.l], 2020. Disponível em: https://pt.wikipedia.org/wiki/Algoritmo_de_Euclides. Acesso em: 14 nov. 2020.

OLIVEIRA, Raul Rodrigues de. **FATORIAL**. [S.l], 2020. Disponível em: https://brasilescola.uol.com.br/matematica/fatorial.htm. Acesso em: 01 nov. 2020.

OS Elementos. **Wikipédia**. [S.l], 2020. Disponível em: https://pt.wikipedia.org/wiki/Os_Elementos. Acesso em: 14 nov. 2020.

PIMENTA, Arruda Eucidio. **Ensino e aprendizagem na sociedade do entretenimento:** desafios para a formação docente. Educação, vol. 36, núm. 2, mayo-agosto, 2013, pp. 232-239 Pontifícia Universidade Católica do Rio Grande do Sul Porto Alegre, Brasil. Disponível em: https://www.redalyc.org/pdf/848/84827901011.pdf. Acesos em 10 out. 2020.

PIVA, Naiady. Inclusão da programação nos currículos escolares avança no exterior. **Gazeta do Povo.** [S.l], 2016. Disponível em: https://www.gazetadopovo.com.br/educacao/inclusao-da-programacao-nos-curriculos-escolares-avanca-no-exterior-9dlmbpvkpztvrp4vp1647l8fk/. Acesso em: 1 out. 2020.

PYTHON library for symbolic mathematics. **SYMPY**, 2020. Disponível em: https://www.sympy.org/en/index.html. Acesso em: 26 nov. 2020.

SAIBA como funciona um algoritmo e conheça os principais exemplos existentes no mercado. **Rock Content.** [S.l], 2019. Disponível em: https://rockcontent.com/br/blog/algoritmo/. Acesso em: 19 out. 2020.

SANTOS, Thamires. Fórmula de Bhaskara. **Educa Mais Brasil**. [S.l], 2018. Disponível em: https://www.educamaisbrasil.com.br/enem/matematica/formula-de-bhaskara. Acesso em: 28 nov. 2020.

SCIPY.org. **SCIPY**, 2020. Disponível em: https://scipy.org/. Acesso em: 09 nov. 2020.

SILVA, Jailson Cunha da. **Ensino de Programação para alunos do Ensino Básico: Um levantamento das pesquisas realizadas no Brasil.** Paraíba, 2017. Disponível em:

https://repositorio.ufpb.br/jspui/bitstream/123456789/3328/1/JCS14062017.pdf. Acesso em: 15 out. 2020.

SOUZA, Fernando. **Pensamento computacional e programação como ferramentas de aprendizagem.** São Paulo, 2019. Disponível em: https://institutoayrtonsenna.org.br/pt-br/meu-educador-meu-idolo/materialdeeducacao/pensamento-computacional-e-programacao-como-ferramentas-de-aprendizagem.html. Acesso em: 13 nov. 2020.

THE fundamental package for scientific computing with Python. **NUMPY**, 2019. Disponível em: https://numpy.org/. Acesso em: 02 nov. 2020.

VENTURA, Felipe. **Python e Java empatam em ranking de linguagens de programação**. Tecnoblog. [S.l], 2020. Disponível em: https://tecnoblog.net/327488/python-java-empatam-ranking-linguagens-programacao/. Acesso em: 23 nov. 2020.

WEB 2.0. **Wikipédia**. [S.l], 2020. Disponível em: https://pt.wikipedia.org/wiki/Web_2.0. Acesso em: 14 nov. 2020.

WING, Jeannette. **Pensamento computacional** – Um conjunto de atitudes e habilidades que todos, não só cientistas da computação, ficaram ansiosos para aprender e usar. Curitiba – PR, 2016. v.9, n.2. Disponível em: https://periodicos.utfpr.edu.br/rbect/article/view/4711. Acesso em: 07 nov. 2020.