Universidade Federal de Mato Grosso
Instituto de Ciências Exatas e da Terra
Departamento de Matemática

Uma abordagem da criptografia com tecnologia para o aprimoramento da aprendizagem de funções polinomiais

Douglas Vinicius Antunes Rodrigues

Produto Educacional

Orientadora: Profa. Dra. Anna Lígia Oenning Soares

Cuiabá - MT

2024

Sumário

Resumo

Este produto educacional propõe uma sequênia didática para o aprimoramento do conhecimento sobre funções matemáticas utilizando a criptografia como ferramenta pedagógica e a linguagem de programação Python para automatizar o processo. A proposta, direcionada para estudantes do ensino médio, é baseada na BNCC e no Pensamento Computacional, explorando a codificação e decodificação de mensagens utilizando funções afim e quadrática, demonstrando a aplicação prática desses conceitos matemáticos e desenvolvendo habilidades de resolução de problemas.

Palavras chave: Ensino de matemática; Pensamento computacional; Função polinomial; Automação; Criptografia.

Introdução

A integração de tecnologias digitais no currículo escolar é um processo gradual e contínuo. O Artigo 1º da lei Nº 14.533, de 11 de janeiro de 2023 estabelece a Política Nacional de Educação Digital (PNED), que visa potencializar o acesso da população brasileira a recursos, ferramentas e práticas digitais. A PNED integra programas, projetos e ações de diferentes entes federados que abrangem inovação e tecnologia na educação, com apoio técnico ou financeiro do governo federal, e apresenta quatro eixos estruturantes: Inclusão Digital, Educação Digital Escolar, Capacitação e Especialização Digital e Pesquisa e Desenvolvimento em Tecnologias da Informação e Comunicação (TICs).

Metodologias ativas e projetos interdisciplinares têm sido adotados, envolvendo a integração de tecnologias em diferentes disciplinas. Na área da matemática, pesquisadores como José Armando Valente, que destaca o uso de tecnologias digitais e a aprendizagem colaborativa, e Seymour Papert, com sua teoria do construcionismo, incentivam a construção ativa do conhecimento e trazem possibilidades que visam criar um ambiente de aprendizagem mais dinâmico e centrado no estudante.

A Base Nacional Comum Curricular (BNCC) traz o Pensamento Computacional como uma habilidade essencial para os estudantes.

> No Ensino Médio, na área de Matemática e suas Tecnologias, os estudantes devem utilizar conceitos, procedimentos e estratégias não apenas para resolver problemas, mas também para formulá-los, descrever dados, selecionar modelos matemáticos e desenvolver o pensamento computacional, por meio da utilização de diferentes recursos da área. (BRASIL, 2018, p. 470)

Nessa etapa de ensino da educação básica todo conteúdo do componente curricular é aprimorado, visando o alcance de estágios que proporcionam uma visão da matemática integrada à realidade do estudante. Ainda, no mesmo documento:

> (...) a BNCC propõe que os estudantes utilizem tecnologias, como calculadoras e planilhas eletrônicas, desde os anos iniciais do Ensino Fundamental. Tal valorização possibilita que, ao chegarem aos anos finais, eles possam ser estimulados a desenvolver o pensamento computacional, por meio da interpretação e da elaboração de fluxogramas e algoritmos. (BRASIL, 2018, p. 518)

Uma forma de representação de algoritmos é a linguagem de programação. Aliás, a BNCC em seu texto traz, no componente curricular de matemática, as seguintes habilidades:

> (EM13MAT315) Investigar e registrar, por meio de um fluxograma, quando possível, um algoritmo que resolve um problema.
>
> (EM13MAT405) Utilizar os conceitos básicos de uma linguagem de programação na implementação de algoritmos escritos em linguagem corrente e/ou matemática.
>
> (EF06MA04) Construir algoritmo em linguagem natural e representá-lo por fluxograma que indique a resolução de um problema simples (por exemplo, se um número natural qualquer é par). (BRASIL, 2018)

Essas habilidades enfatizam o uso do algoritmo como ferramenta de aprendizado, o que mostra que a BNCC entra em conjunto com o Centro de Inovação para a Educação Brasileira (CIEB) para prover um currículo buscando inovações para chegar no objetivo dentro da área da matemática, sendo ele, "(...) possibilitar que os estudantes construam uma visão mais integrada da Matemática, ainda na perspectiva de sua aplicação à realidade" (BRASIL, 2018).

Por fim, destacando a habilidade EM13MAT405, o uso de uma linguagem de programação é uma maneira de desenvolver o Pensamento Computacional. Daí, conclui-se que, é viável utilizar uma linguagem atrelada com um conteúdo para gerar uma discussão construtiva que busca estar alinhada com o objetivo central da base curricular.

A metodologia proposta se baseia na construção de um sistema de criptografia simplificado, em que os coeficientes das funções polinomiais atuam como chaves criptográficas. A mensagem a ser codificada é inicialmente convertida em uma sequência numérica, utilizando uma tabela de associação entre caracteres e números. Em seguida, cada número da sequência é transformado pela função polinomial, gerando a mensagem codificada. Para decodificar a mensagem, o processo inverso é realizado, utilizando a função inversa ou resolvendo uma equação. Em seguida, a linguagem de programação Python é inserida para automatizar esse processo.

A aplicação prática da criptografia por meio de funções polinomiais pede oferecer aos estudantes a oportunidade de vivenciar, na prática, a conexão entre a matemática e a segurança da informação. Através da manipulação de funções, da resolução de equações e da análise de gráficos, os alunos visualizam os mecanismos por trás da criptografia, desenvolvendo, ao mesmo tempo, habilidades de raciocínio lógico, abstração e resolução de problemas.

Este produto educacional derivado da dissertação intitulada "Uma abordagem da criptografia com tecnologia para o aprimoramento da aprendizagem de funções polinomiais" (RODRIGUES, 2024), apresenta uma sequência didática utilizando a criptografia como ferramenta pedagógica e a linguagem de programação Python para automatização do processo, visando o aprimoramento do conhecimento matemático acerca de funções afim e quadrática.

Criptografia

Derivada de dois vocábulos gregos: kryptos (oculto) e graphien (escrever), a criptografia é uma técnica que consiste em ocultar o significado de uma mensagem. Se há necessidade de ocultar o que está nessa mensagem, então há um sujeito que não pode saber o conteúdo dela (FIARRESGA, 2010, p. 3).

No que se refere ao processo de criptografar um conteúdo, o mecanismo básico

é chamado de cifra (SHOUP e BONEH, 2023, p.4). O conteúdo a ser criptografado passa por transformações por meio das cifras, que por sua vez é dividido em dois tipos: cifras de transposição e cifras de substituição. Enquanto na cifra de transposição cada letra conserva a sua identidade, mas muda de posição dentro da mensagem; na cifra de substituição, cada letra conserva a sua posição, mas é substituída por uma outra letra ou símbolo (KAHN, 1968, p.xiii).

A maioria das cifras utiliza uma chave, que determina aspectos como a disposição das letras em um alfabeto cifrado ou o padrão de embaralhamento em uma transposição. Essa chave pode ser uma palavra, frase ou número, e é denominada palavra-chave, frase-chave ou número-chave, conforme apropriado (KAHN, 1968, p. xv). Pode-se entender que uma chave é uma regra pela qual o conteúdo irá passar para haver a transformação em um conteúdo criptografado. Essa regra precisa ser de conhecimento do destinatário para que possa realizar o processo de reversão da transformação e recuperar o conteúdo original.

Definiremos a chave criptográfica como uma sequência de $n+2$ termos, para todo $n \geq 1$, de tal forma que os primeiros $n+1$ números representam os coeficientes da função polinomial de grau n, e o último dígito m representa a quantidade de caracteres de cada letra a ser decodificada (modelo na Tabela 1). A Tabela 2, apresenta a função injetiva que associa cada $p(x_i)$ a um caractere, em que $x_i, a_i \in \mathbb{Z}$ e $i \in \mathbb{N}$.

Tabela 1: Estrutura de composição da chave criptográfica

Chave	Função
$(a_1,\ a_0,\ m)$	$p(x) = a_1x + a_0$
$(a_2,\ a_1,\ a_0,\ m)$	$p(x) = a_2x^2 + a_1x + a_0$
$(a_3,\ a_2,\ a_1,\ a_0,\ m)$	$p(x) = a_3x^3 + a_2x^2 + a_1x + a_0$
$\vdots$	$\vdots$
$(a_n,\ a_{n-1}, ...,\ a_1,\ a_0,\ m)$	$p(x) = a_nx^n + a_{n-1}x^{n-1} + ... + a_1x + a_0$

Fonte: PONTES et al. (2022).

Tabela 2: Base de uma Tabela de Cifras

$p(x)$	Caractere	$p(x)$	Caractere
$p(x_1)$	A	$p(x_{10})$	J
$p(x_2)$	B	$p(x_{11})$	K
$p(x_3)$	C	$p(x_{12})$	L
$p(x_4)$	D	$p(x_{13})$	M
$p(x_5)$	E	$p(x_{14})$	N
$p(x_6)$	F	$p(x_{15})$	O
$p(x_7)$	G	$p(x_{16})$	P
$p(x_8)$	H	$\vdots$	$\vdots$
$p(x_9)$	I	$p(x_r)$	

Fonte: Próprio autor.

Com base na regra definida para a chave e da tabela de conversão, podemos utilizar o algoritmo abaixo para codificar e decodificar uma mensagem:

1. Defina a função polinomial de grau n, $p(x) = a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0$, injetiva no intervalo $[x_1, x_r]$ para codificar cada caractere da mensagem;

2. Determine o parâmetro m. O valor de m deve ser o valor máximo de caracteres de $p(x)$ no intervalo $[x_1, x_r]$. Dessa forma, considere c_1 a quantidade de caracteres de $p(x_1)$ e c_r a quantidade de caracteres de $p(x_r)$, então o parâmetro m é definido por

$$m = max\{c_1, c_r\}$$

3. Escolha a mensagem para ser codificada. Suponhamos a frase "SEJA BEM-VINDO";

4. Calcule as cifras da mensagem. Para a mensagem "SEJA BEM-VINDO", calcule $p(x_{19}), p(x_5), ..., p(x_{15})$ e suponha que,

$$\begin{aligned} p(x_{19}) &= q_1 \\ p(x_5) &= q_2 \\ \vdots \quad &\quad \vdots \\ p(x_{15}) &= q_{14} \end{aligned}$$

em que, a quantidade de caracteres de q_j é igual a m, para $j = 1, ..., 14$, por exemplo, se $m = 2$ e $p(x_{19}) = 8$, então $q_1 = 08$.

Assim o vetor $[q_1 q_2 ... q_{14}]$ é a mensagem codificada;

5. Decodifique a mensagem. Este processo é realizado resolvendo as equações

$$p(x) = q_j, \quad \text{para} \quad 1 \leq j \leq 14$$

em que $x_1 \leq x \leq x_r$.

A respeito do segundo passo do algoritmo descrito acima, calcular o parâmetro m é necessário para o momento de decodificação. Quando obtemos uma mensagem já codificada, precisamos saber quantos algarismos serão selecionados por vez para decodifica-la, ou seja, identificar quantos algarismos representam um caractere da mensagem.

Criptografia com Função Afim

Nesta seção apresentamos o processo de criptografia de mensagens utilizando função afim e utilizamos a Tabela 3 como referência da regra que associa a imagem da função com os caracteres.

Tabela 3: Tabela de Cifras

$p(x)$	Caractere	$p(x)$	Caractere	$p(x)$	Caractere
$p(1)$	A	$p(11)$	K	$p(21)$	U
$p(2)$	B	$p(12)$	L	$p(22)$	V
$p(3)$	C	$p(13)$	M	$p(23)$	X
$p(4)$	D	$p(14)$	N	$p(24)$	W
$p(5)$	E	$p(15)$	O	$p(25)$	Y
$p(6)$	F	$p(16)$	P	$p(26)$	Z
$p(7)$	G	$p(17)$	Q	$p(27)$	-
$p(8)$	H	$p(18)$	R	$p(28)$	
$p(9)$	I	$p(19)$	S	$p(29)$	Õ
$p(10)$	J	$p(20)$	T	$p(30)$	Ç

Fonte: Próprio autor.

Com base na tabela de conversão, realizamos os passos do algoritmo:

1. A função injetiva $p(x) = x - 2$ foi definida para compor a chave;

2. Determinando o valor do parâmetro m. Temos que $p(1) = -1$ e $p(30) = 28$, uma vez que $c_1 = c_2 = 2$, temos que $m = max\{c_1, c_2\} = 2$. Desta forma, a chave criptográfica é formada por $(1, -2, 2)$;

3. Mensagem escolhida: "SEJA BEM-VINDO";

4. Codificando a mensagem calculando os valores de cada cifra, como segue, $p(19) = 17$, $p(5) = 03$, $p(10) = 08$, $p(1) = -1$, $p(28) = 26$, $p(2) = 00$, $p(5) = 03$, $p(13) = 11$, $p(27) = 25$, $p(22) = 20$, $p(9) = 07$, $p(14) = 12$, $p(4) = 02$ e $p(15) = 13$, assim, a mensagem codificada é dada pelo vetor [170308-12600031125200712021 3]

Com a mensagem codificada, realizamos o passo 5 da seguinte forma:

- Sabendo que $m = 2$, identificamos as duplas de algarismos que representam uma letra da mensagem, [17, 03, 08, -1, 26, 00, 03, 11, 25, 20, 07, 12, 02, 13];

- Utilizamos a função, conhecida por meio das informações trazidas pela chave de criptografia, $p(x) = x - 2$ para fazermos os cálculos:

  1º Letra: $17 \Rightarrow\ x - 2 = 17\ \Rightarrow\ x - 2 - 17 = 0\ \Rightarrow\ x - 19 = 0 \Rightarrow\ x = 19$.
  Logo, 19 é correspondente da letra S.

  2º Letra: $03 \Rightarrow\ x - 2 = 3\ \Rightarrow\ x - 2 - 3 = 0\ \Rightarrow\ x - 5 = 0\ \Rightarrow\ x = 5$.
  Logo, 5 é correspondente da letra E.

  [...]

  4º Letra: -1 $\Rightarrow\ x - 2 = -1\ \Rightarrow\ x - 2 + 1 = 0\ \Rightarrow\ x - 1 = 0\ \Rightarrow\ x = 1$.
  Logo, 1 é correspondente da letra A.

  [...]

  14º Letra: $13 \Rightarrow\ x - 2 = 13\ \Rightarrow\ x - 2 - 13 = 0\ \Rightarrow\ x - 15 = 0\ \Rightarrow x = 15$.
  Logo, 15 é correspondente da letra O.

- Após todos os cálculos, os quais rementem o processo de decodificação, é obtido a mensagem original "SEJA BEM-VINDO".

Criptografia com Função Quadrática

Nesta seção abordamos o processo de criptografia, como na seção anterior, utilizando função quadrática.

1. A função injetiva $p(x) = x^2 + x - 1$ foi definida para compor a chave;

2. Determinando o valor do parâmetro m. Temos que $p(1) = 1$ e $p(30) = 929$, uma vez que $c_1 = 1$ e $c_2 = 3$, temos que $m = max\{c_1, c_2\} = 3$. Dessa forma, a chave criptográfica é formada por $(1, 1, -1, 3)$;

3. Mensagem escolhida: "SEJA BEM-VINDO";

4. Codificando a mensagem calculando os valores de cada cifra, como segue, $p(19) = 379$, $p(5) = 029$, $p(10) = 109$, $p(1) = 001$, $p(28) = 811$, $p(2) = 005$, $p(5) = 029$, $p(13) = 181$, $p(27) = 755$, $p(22) = 505$, $p(9) = 089$, $p(14) = 209$, $p(4) = 019$, $p(15) = 239$, assim, a mensagem codificada é dada pelo vetor

$$[379029109001811005029181755505089209019239]$$

Com a mensagem codificada, realizamos o passo 5:

- Sabendo que $m = 3$, identificamos as triplas de algarismos que representam uma letra da mensagem, [379, 029, 109, 001, 811, 005, 029, 181, 755, 505, 089, 209, 019, 239];

- Utilizando a função, conhecida por meio das informações trazidas pela chave de criptografia, $p(x) = x^2 + x - 1$, fazer os cálculos:

  1º Letra: $379 \Rightarrow\ x^2 + x - 1 = 379\ \Rightarrow\ x^2 + x - 380 = 0\ \Rightarrow\ x^{'} = 19$ e $x^{''} = -20$.
  Logo, escolhemos o número 19 que é correspondente da letra S.

  2º Letra: $029 \Rightarrow\ x^2 + x - 1 = 29\ \Rightarrow\ x^2 + x - 30 = 0\ \Rightarrow\ x^{'} = 5$ e $x^{''} = -6$.
  Logo, escolhemos o número 5 que é correspondente da letra E.

  [...]

  8º Letra: $755 \Rightarrow\ x^2 + x - 1 = 755\ \Rightarrow\ x^2 + x - 756 = 0\ \Rightarrow x^{'} = 27$ e $x^{''} = -28$.
  Logo, escolhemos o número 27 que é correspondente ao símbolo $-$.

  [...]

14º Letra: $239 \Rightarrow\ x^2 + x - 1 = 239\ \Rightarrow\ x^2 + x - 240 = 0\ \Rightarrow x^{'} = 15$ e $x^{''} = -16$. Logo, escolhemos o número 15 que é correspondente da letra O.

- Após os cálculos, é obtido a mensagem decodificada: "SEJA BEM-VINDO".

Observe que para cada caractere decodificado há dois possíveis valores para x, que são as raízes da equação do segundo grau, entretanto, há somente um valor de x que pertence ao intervalo $[1, 30]$, onde a função quadrática é injetora. Portanto, não há dubiedade na decodificação, tornando sempre possível a obtenção da mensagem original.

Automatizando o Processo Criptográfico

De acordo com Lambert (2019, p. 4), um algoritmo consiste em um número finito de instruções bem definidas que descrevem, passo a passo, um processo a ser seguido para resolver um problema específico ou executar uma tarefa. Informalmente, é como uma receita de bolo. Os passos descrevem desde a quantidade de ovos que devem ser usados no início da receita, até o tempo que a massa deve ficar no forno antes de tirá-lo.

Analisando e colocando os passos que utilizamos para codificar e decodificar uma mensagem, pontuaremo-os em uma lista como um algoritmo:

- Escolha as cifras que serão utilizadas. Consequentemente, é definido o domínio da função;
- Defina uma função injetiva, afim ou quadrática, que servirá como chave do processo;
- Escolha a mensagem;
- Calcule os valores das imagens que formarão a mensagem codificada;
- Ao receber a mensagem codificada, identifique qual a função e o agrupamento para o cálculo;
- Calcule as raízes das funções;
- Use a tabela de cifras para decifrar a mensagem.

O algoritmo descrito acima em liguagem de programação Python segue os seguintes passos:

- Adicione, ao código, a tabela de cifras;
- Solicite ao usuário os coeficientes da função e a mensagem;
- Separe cada caractere da mensagem;
- Realize o processo de codificação;
- Mostre a mensagem codificada na tela;
- Para decodificar: Solicite ao usuário a mensagem codificada;
- Realize o processo de decodificação;
- Mostre a mensagem decodificada na tela.

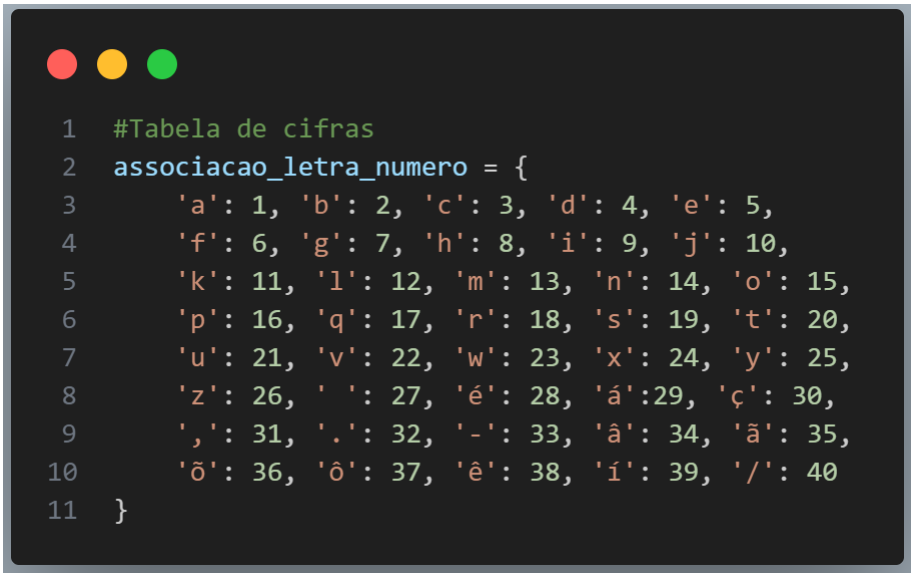
Após construirmos um passo a passo para orientação, é necessário detalhar como queremos realizar cada etapa. Nas próximas subseções, apresentaremos cada etapa do algoritmo. Ressaltamos que não utilizaremos o cálculo do parâmetro m no código, pois incluí-lo aumentaria a complexidade. Neste momento, queremos simplificar o máximo possível para alcançar um nível mais didático.

Codificando

Função Afim

Assim como na Tabela 3, definiremos quais são as cifras, ou seja, quais associações serão utilizadas durante o processo criptográfico. Para isso, criaremos uma estrutura que mapeie cada caractere escolhido a um número correspondente. Isso será útil para converter os caracteres em números durante o processo de codificação e os números em caracteres no processo de decodificação.

A Figura 1 apresenta como fica a tabela de cifras em código Python. Os caracteres escolhidos para compor a tabela são as chaves do dicionário, enquanto os números associados a cada caractere são os valores.

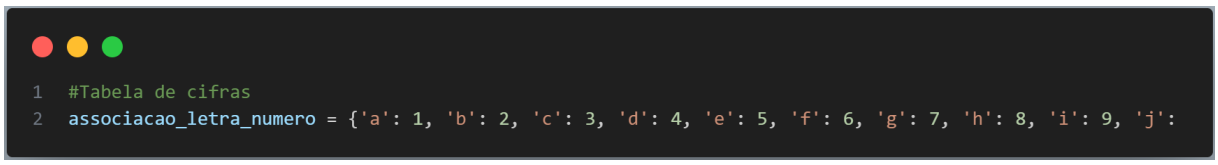


```
1  #Tabela de cifras
2  associacao_letra_numero = {
3      'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5,
4      'f': 6, 'g': 7, 'h': 8, 'i': 9, 'j': 10,
5      'k': 11, 'l': 12, 'm': 13, 'n': 14, 'o': 15,
6      'p': 16, 'q': 17, 'r': 18, 's': 19, 't': 20,
7      'u': 21, 'v': 22, 'w': 23, 'x': 24, 'y': 25,
8      'z': 26, ' ': 27, 'é': 28, 'á':29, 'ç': 30,
9      ',': 31, '.': 32, '-': 33, 'â': 34, 'ã': 35,
10     'õ': 36, 'ô': 37, 'ê': 38, 'í': 39, '/': 40
11 }
```

Figura 1: Associação caracteres - números

Na linha 1 da imagem acima, há o texto `#Tabela de cifras`. O computador não irá considerar essa linha como código, devido ao caractere especial "`#`"adicionado antes da frase. Quando colocamos esse símbolo dessa maneira, chamados a linha de **comentário**. Sua função é nomear ou orientar sobre os blocos subsequentes.

Nas linhas 2 a 11, construímos o bloco com as cifras. Escolhemos o nome `associacao_letra_numero` para o dicionário e optamos por utilizar 40 associações contendo letras, acentos e caracteres especiais. Lembrando que podemos escolher, também, letras maiúsculas, números e outros tipos de símbolos.



```
1  #Tabela de cifras
2  associacao_letra_numero = {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5, 'f': 6, 'g': 7, 'h': 8, 'i': 9, 'j':
```

Figura 2: Opção base de escrita

A Figura 2, mostra uma forma de alocar os elementos no dicionário, porém quando o dicionário tem uma quantidade grande de elementos, é mais claro para visualização que seja organizado como na Figura 1.

Em seguida, precisamos saber qual será a função afim e a mensagem a ser codificada. Então, solicitaremos a entrada no código dos coeficientes da função e, em sequência, da mensagem.

Criaremos as seguintes variáveis: `texto`, que receberá a mensagem a ser codifi-

cada; e `a` e `b` que receberão os valores dos coeficientes da função.

```
1  #Entrada dos coeficientes da função
2  a = int(input('Digite o coeficiente angular da função: '))
3  b = int(input('Digite o coeficiente linear da função: '))
4
5  #Entrada do texto a ser codificado
6  texto = str(input('Digite o texto: ')).lower()
```

Figura 3: Entrada dos coeficientes da função afim e da mensagem

Repare o comando `.lower()` na Figura 3. Ele é uma ação que podemos adicionar em strings para que todo o texto fique com caracteres minúsculas. Utilizá-la também nos permite simplificar a tabela de cifras ao não necessitar adicionar letras maiúsculas.

Esses tipos de ações são chamadas de **método**, ou seja, a ação `.lower()` é um método de strings.

Outros métodos que podem ser utilizadas são `.upper()`, que converte todos os caracteres da string para maiúsculas, e `.capitalize()`, que converte o primeiro caractere da string para maíuscula e os demais para minúsculas.

Uma vez que o programa sabe quais são os coeficientes da função e a mensagem, seguimos ao próximo passo antes da codificação.

O computador precisa cifrar um caractere por vez, logo é necessário separar cada caractere da mensagem e adicioná-los em uma lista. É como se estivéssemos soletrando a mensagem.

```
1  #Separar cada caractere do texto
2  lista = []
3  for c in texto:
4      lista.append(c)
```

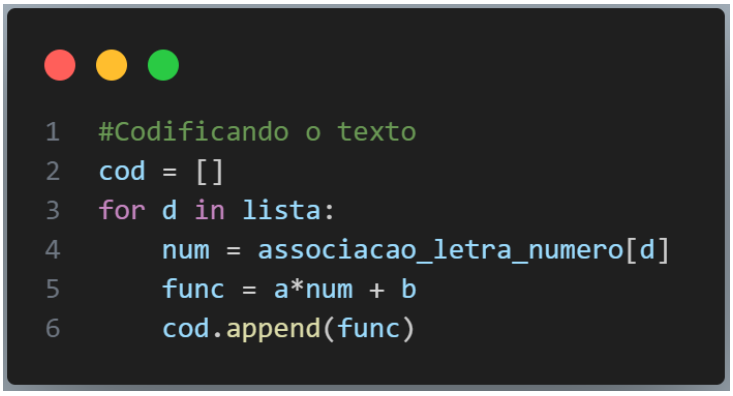
Figura 4: Separação dos caracteres da mensagem

Precisaremos do conceito de estruturas de repetição para adicionar caractere por caractere em uma lista. Na linha 2, da Figura 4, nomeamos a lista como `lista`. Nas

linhas 3 e 4 a estrutura de repetição *for* pode ser lida da seguinte forma: Para cada caractere na mensagem, adicione-os na lista. Então, a letra `c` na figura representa cada caractere digitado na variável `texto`. Em seguida, esse caractere `c` é adicionado à `lista` por meio do método `.append()`.

Agora que já separamos a mensagem, podemos codificá-la.

Criamos um loop *for* que irá, a cada iteração[1], pegar um elemento de `lista`, realizar a associação necessária por meio da tabela de cifras, adicionar em uma variável, calcular a imagem da função e adicionar a imagem em uma lista.



```
1  #Codificando o texto
2  cod = []
3  for d in lista:
4      num = associacao_letra_numero[d]
5      func = a*num + b
6      cod.append(func)
```

Figura 5: Bloco de codificação com função afim

Acompanhando pela Figura 5, iniciamos esse bloco criando uma nova lista chamada `cod`. Após, por meio da estrutura de repetição *for*, pegamos cada elemento de `lista` e calculamos a cifra. Na linha 4, adicionamos na variável `num` o elemento do dicionário associado ao caractere `d` da `lista`. Após, na linha 5, ciframos o valor `num` utilizando a expressão `a*num + b`, onde `a` e `b` foram definidos anteriormente, e atribuímos na variável `func`. O caractere cifrado é, então, adicionado na lista `cod` utilizando o método `.append()`.

Por fim, precisamos mostrar na tela como ficou a mensagem após a codificação..

Como na Figura 6, adicionamos o comando `print()` que mostrará no terminal, localizado no painel do VS Code, o que estiver entre parênteses. Nesse caso colocamos a lista `cod` entre parênteses, o que indica que será mostrado na tela o que estiver armazenado nessa lista.

[1]Iteração é a repetição de um processo ou uma sequência de instruções várias vezes

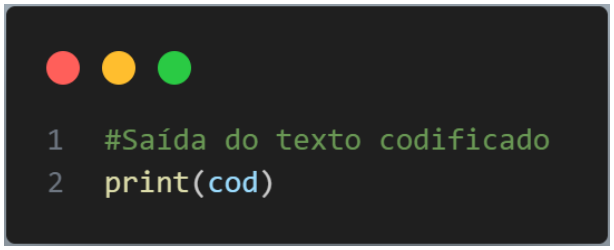


Figura 6: Saída da mensagem codificada

Com o código de codificação construído, podemos executá-lo clicando na opção Run Python File, localizada no canto superior direito da interface do VS Code (Figura 7).

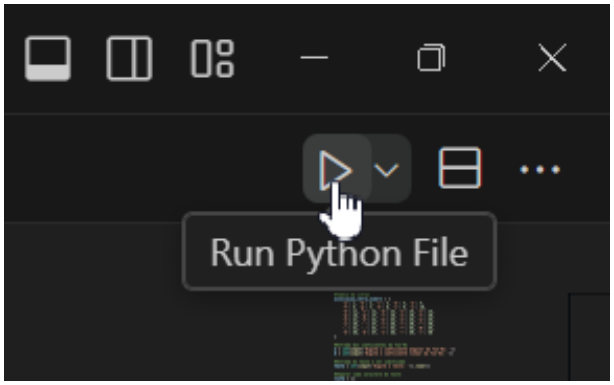


Figura 7: Executar o código

Ao executar, as entradas dos coeficiente da função e da mensagem são realizadas no terminal, assim como, a saída da mensagem codificada (Figura 8).

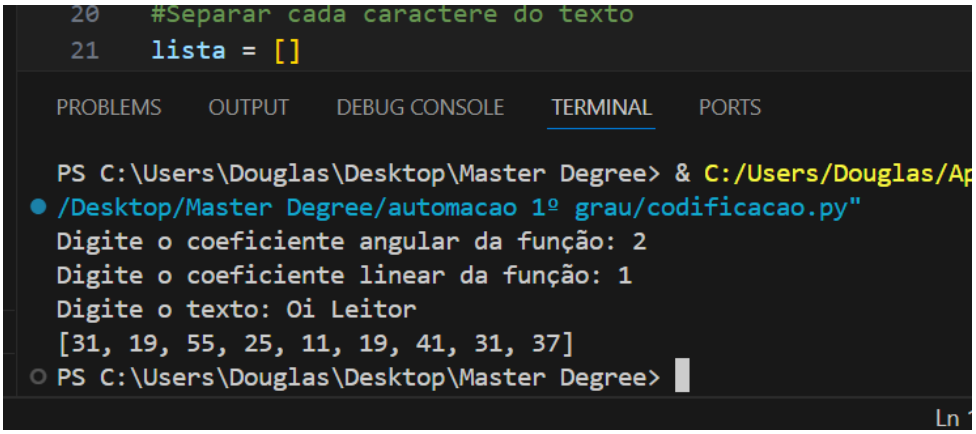


Figura 8: Entradas e saídas mostradas no terminal

Função Quadrática

Seguindo os passos do processo criptográfico com função afim, primeiro coloque no código a mesma tabela de cifras da Figura 1 (alteramos apenas o nome da variável para `correlacao_letra_numero`). Em seguida, criaremos as seguintes variáveis: `texto`, que receberá a mensagem a ser codificada; e `a`, `b` e `c` que receberão os valores dos coeficientes da função, conforme Figura 9.

```
#Entrada dos coeficientes da função
a = int(input('Digite o coeficiente a da função: '))
b = int(input('Digite o coeficiente b da função: '))
c = int(input('Digite o coeficiente c da função: '))

#Entrada do texto a ser codificado
texto = str(input('Digite o texto: ')).lower()
```

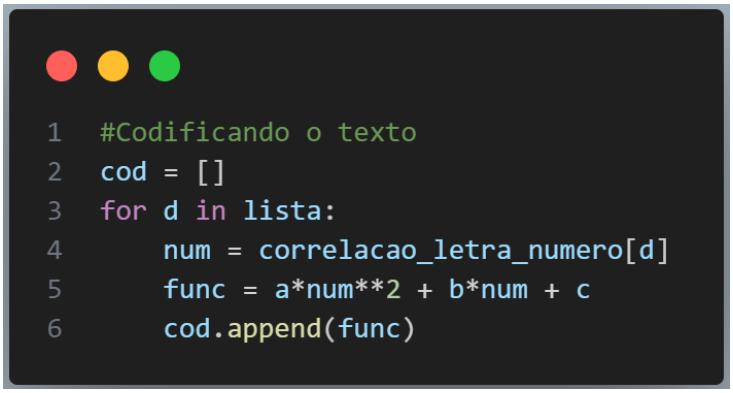
Figura 9: Entrada da mensagem e dos coeficientes da função quadrática

Utilizando os conceitos de lista e estruturas de repetição separamos capa caractere da mensagem atribuída na variável `texto` (Figura 10).

```
#Separar cada caractere do texto
lista = []
for t in texto:
    lista.append(t)
```

Figura 10: Separação dos caracteres da mensagem

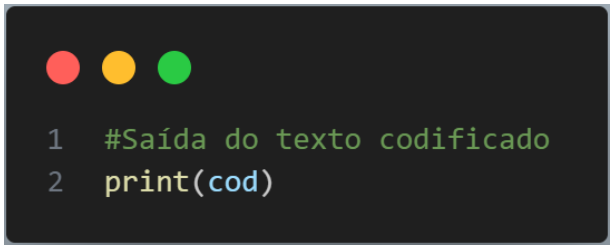
Após, usamos as estruturas de repetição e listas junto com dicionários, para cifrar cada caractere da mensagem.



```
1  #Codificando o texto
2  cod = []
3  for d in lista:
4      num = correlacao_letra_numero[d]
5      func = a*num**2 + b*num + c
6      cod.append(func)
```

Figura 11: Bloco de codificação com função quadrática

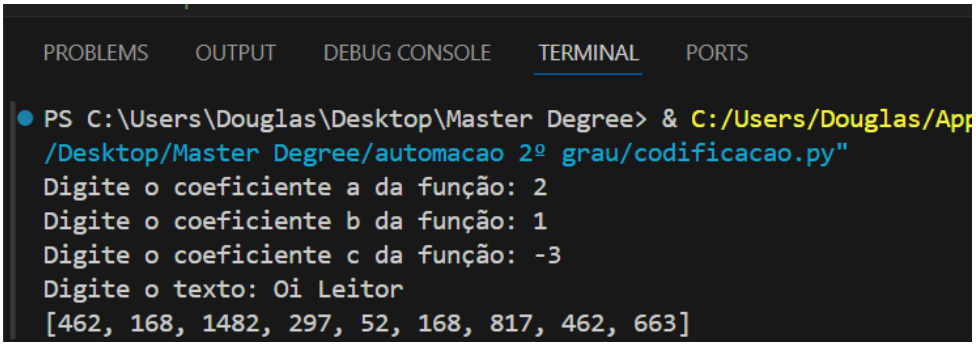
A Figura 11, segue a mesma construção da Figura 5. Na linha 4, adicionamos na variável `num` o elemento do dicionário associado ao caractere `d` da `lista`. Após, na linha 5, ciframos o valor `num` utilizando a expressão `a*num**2 +b*num + c` ($ax^2 + bx + c$), onde `a`, `b` e `c` foram definidos anteriormente, e atribuímos na variável `func`. O caractere cifrado é, então, adicionado na lista `cod` utilizando o método `.append()`.



```
1  #Saída do texto codificado
2  print(cod)
```

Figura 12: Saída da mensagem codificada

Enfim, apresentaremos a mensagem codificada, por meio do `print(cod)` (Figura 12).



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Douglas\Desktop\Master Degree> & C:/Users/Douglas/App
/Desktop/Master Degree/automacao 2º grau/codificacao.py"
Digite o coeficiente a da função: 2
Digite o coeficiente b da função: 1
Digite o coeficiente c da função: -3
Digite o texto: Oi Leitor
[462, 168, 1482, 297, 52, 168, 817, 462, 663]
```

Figura 13: Entradas e saídas mostradas no terminal

Quando o código é executado, clicando na opção mostrada na Figura 7, aparecerá as entradas e as saídas mostradas no exemplo da Figura 13.

Decodificando

Função Afim

Uma vez que construímos um código em Python para codificar uma mensagem usando função afim, iremos construir um para decodificar.

Iniciaremos adicionando uma tabela de cifras ao código. Essa tabela é semelhante a mostrada na Figura 1, porém têm seus elementos invertidos, ou seja, o que era chave agora é valor. A Figura 14 apresenta como ficará o dicionário.

```
#Tabela de cifras
associacao_numero_letra = {
    1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e',
    6: 'f', 7: 'g', 8: 'h', 9: 'i', 10: 'j',
    11: 'k', 12: 'l', 13: 'm', 14: 'n', 15: 'o',
    16: 'p', 17: 'q', 18: 'r', 19: 's', 20: 't',
    21: 'u', 22: 'v', 23: 'w', 24: 'x', 25: 'y',
    26: 'z', 27: ' ', 28: 'é', 29: 'á', 30: 'ç',
    31: ',', 32: '.', 33: '-', 34: 'â', 35: 'ã',
    36: 'õ', 37: 'ô', 38: 'ê', 39: 'í', 40: '/'
}
```

Figura 14: Associação números - caracteres

O próximo passo do algoritmo é receber as informações da mensagem e dos coeficientes da função.

Visto que a mensagem a ser decodificada está em formato numérico, criaremos uma lista e um loop *while* para receber cada um dos números que compõem a mensagem. Dentro do loop, é perguntado ao usuário se ele gostaria de adicionar um número, respondendo com “s”para sim ou “n”para não. Caso responda “n”, o loop é interrompido. Caso responda “s”, é pedido que ele insira o número que será adicionado na lista.

Em seguida, pedimos ao usuário que adicione os coeficientes da função.

```
1  lista = []
2  #Entrada dos valores codificados
3  while True:
4      pergunta = input('Gostaria de adicionar um número? (s/n)')
5
6      if pergunta == 'n':
7          break
8
9      x = int(input('Digite o código: '))
10     lista.append(x)
11
12 #Entrada dos coeficientes da função
13 a = int(input('Digite o coeficiente angular da função: '))
14 b = int(input('Digite o coeficiente linear da função: '))
```

Figura 15: Entrada da mensagem e dos coeficientes da função afim

Acompanhando a Figura 15, na linha 1, criamos uma lista que nomeamos de `lista`. Na linha 3 adicionamos uma estrutura de repetição escrevendo `while True`, indicando ao computador que todo o bloco deve ser repetido enquanto for verdade (`True`). Só irá parar de repetir caso a instrução `break`, dentro do bloco, seja executada. Na linha 4, inserimos uma variável `pergunta` a qual atribuímos a decisão de adicionar ou não um número na lista, por meio de um `input`. A estrutura de decisão *if*, linha 6, permite que o loop *while* seja interrompido caso a condição, `pergunta == 'n'`, seja atendida. A interrupção é realizada pela instrução `break`. Caso a variável `pergunta` receba "s", o loop não irá parar e ocorrerá, nas linhas 9 e 10, a adição de um dos números da mensagem codificada na `lista`. Após, o processo se repete.

As linhas 13 e 14, trazem duas variáveis, `a` e `b`, que receberão os coeficientes da função afim.

Outro detalhe essencial é no posicionamento da lista que criamos para armazenar a mensagem separada. Na Figura 15, nomeamos a variável que será uma lista como `lista`. Repare que coloca-se a lista fora do loop *while*, pois caso esteja dentro, tem-se um problema.

```python
while True:
    lista = []
    pergunta = input('Gostaria de adicionar um número? (s/n)')

    if pergunta == 'n':
        break

    x = int(input('Digite o código: '))
    lista.append(x)
```

Figura 16: Exemplo de posicionamento ruim

Na situação da Figura 16, sempre que ocorrer uma iteração, a lista irá ser recriada do zero, o que significa que os elementos anteriores são perdidos. Assim, colocando a lista fora do loop, garantimos que ela persista ao longo das iterações e mantenha seus elementos anteriores.

Até o momento, organizamos o código para receber as informações necessárias para decodificar a mensagem. Então, construiremos o bloco que irá realizar esse passo criando uma lista chamada `decod`, conforme Figura 17.

```python
#Decodificando o texto
decod = []
for c in lista:
    inver = (c - b)/a
    letra = associacao_numero_letra[inver]
    decod.append(letra)
```

Figura 17: Bloco de decodificação com função afim

Necessitaremos de uma estrutura de repetição *for* para decifrar a mensagem recebida. A linha 3 traz, `for c in lista` o que podemos ler como: para cada numero na lista.

Na linha 4, criamos a variável `inver` e atribuímos a ela a expressão `(c - b)/a`. Nesse momento estamos calculando a inversa da função afim, ou seja, $x = \dfrac{(y-b)}{a}$ para após buscar, por meio do dicionário na linha 5, a associação de `inver` com a letra correspondente. Então, na linha 6, adicionamos na lista `decod` a `letra` decifrada.

Ao final adicionamos o comando `print(decod)` (Figura 18) para mostrar na tela o resultado da decodificação.

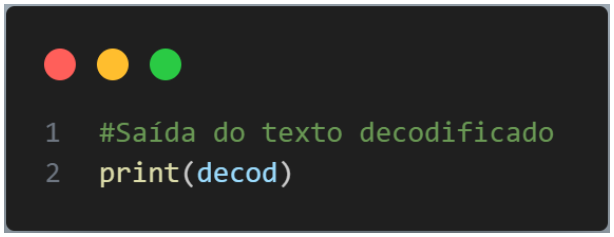


Figura 18: Saída da mensagem decodificada

Agora, podemos executar o código. Lembrando de inicia-lo em Run Python File no canto superior direito da tela (Figura 19).

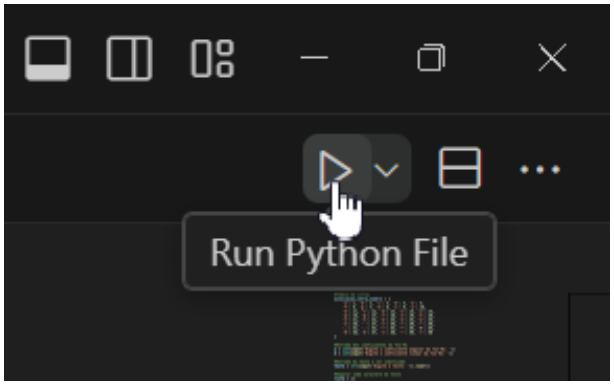


Figura 19: Executar o código

Ao executar, inicialmente é solicitado as entradas de cada número da mensagem codificada e, em seguida, das entradas dos coeficiente da função afim. Podemos acompanhar nas Figura 20 e 21.

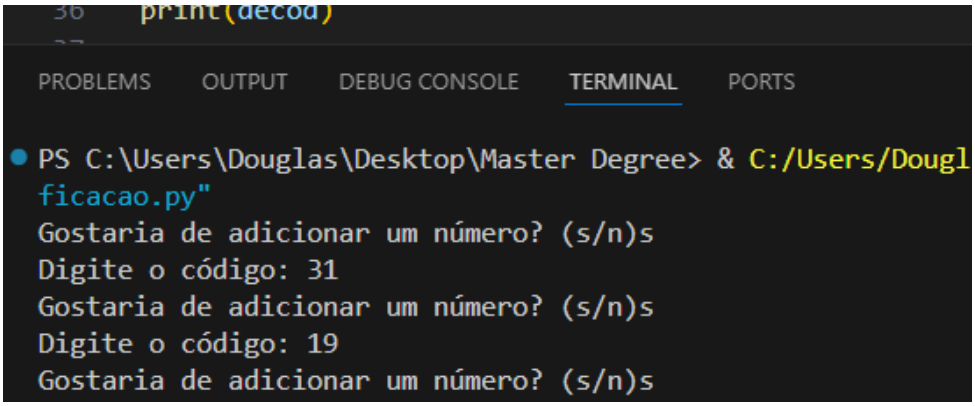


Figura 20: Entrada e saída de dados no terminal

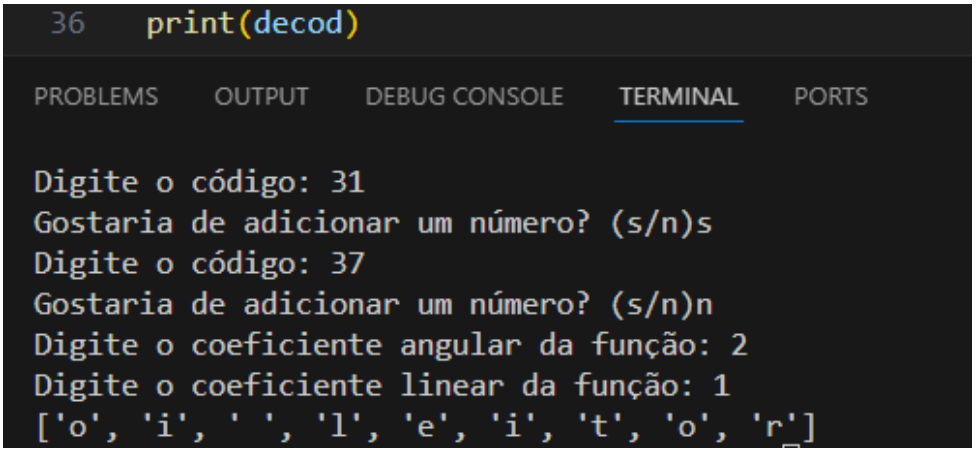


Figura 21: Entrada e saída de dados no terminal

Construímos, assim, os códigos que codificam e decodificam mensagens usando função afim como parte da chave criptográfica.

Função Quadrática

Uma vez que construímos um código em Python para codificar uma mensagem usando função quadrática, iremos construir um para decodificar.

Adicionamos uma tabela de cifras ao código. Será o mesmo dicionário apresentado na Figura 14 alterando apenas o nome da variável para `correlacao_numero_letra`. Precisamos, então, construir um bloco de código para receber cada caractere.

Utilizaremos os conceitos de: tipos de dados, entrada de dados, operadores relacionais, estrutura de decisão, estrutura de repetição e listas.

```python
#Entrada dos valores codificados
lista = []
while True:
    pergunta = input('Você gostaria de adicionar um número?(s/n)')

    if pergunta == 'n':
        break

    num = int(input('Adicione o número: '))
    lista.append(num)

#Entrada dos coeficientes da chave
a = int(input('Digite o coeficiente a da função: '))
b = int(input('Digite o coeficiente b da função: '))
c = int(input('Digite o coeficiente c da função: '))
```

Figura 22: Entrada da mensagem e dos coeficientes da função quadrática

A estrutura que construímos na Figura 22 é muito próxima da construída na Figura 15, com adição de mais uma variável para entrada do coeficiente `c` da função quadrática.

O próximo bloco é onde decifraremos cada caractere numérico da mensagem codificada.

```python
#Decodificando o texto
decod = []

for e in lista:
    c2 = (c - e)
    delta = b**2 - 4*a*c2
    raiz1 = (-b + (delta)**(1/2))/(2*a)
    raiz2 = (-b - (delta)**(1/2))/(2*a)

    if raiz1 in correlacao_numero_letra:
        num2 = correlacao_numero_letra[raiz1]
    elif raiz2 in correlacao_numero_letra:
        num2 = correlacao_numero_letra[raiz2]

    decod.append(num2)
```

Figura 23: Bloco de decodificação com função quadrática

Na linha 5 da Figura 23, a atribuição `c2 = (c - e)` se refere ao momento ocorrido na passagem,

$$x^2 + x - 1 = 1 \quad \Rightarrow \quad x^2 + x - 2 = 0,$$

repare que o coeficiente `c` da função, é o único que é alterado e os cálculos de decodificação são realizados com essa nova função.

Em seguida, linhas 6, 7 e 8, apresenta-se o método para o cálculo das raízes de funções quadráticas.

$$x = \frac{-b \pm \sqrt{\Delta}}{2a}$$

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

Nas linhas 10 a 13, está a escolha da raiz da função que irá compor a mensagem decodificada.

É nessa etapa que a mensagem é decodificada por meio de uma função.

```
1  #Saída do texto decodificado
2  print(decod)
```

Figura 24: Saída da mensagem decodificada

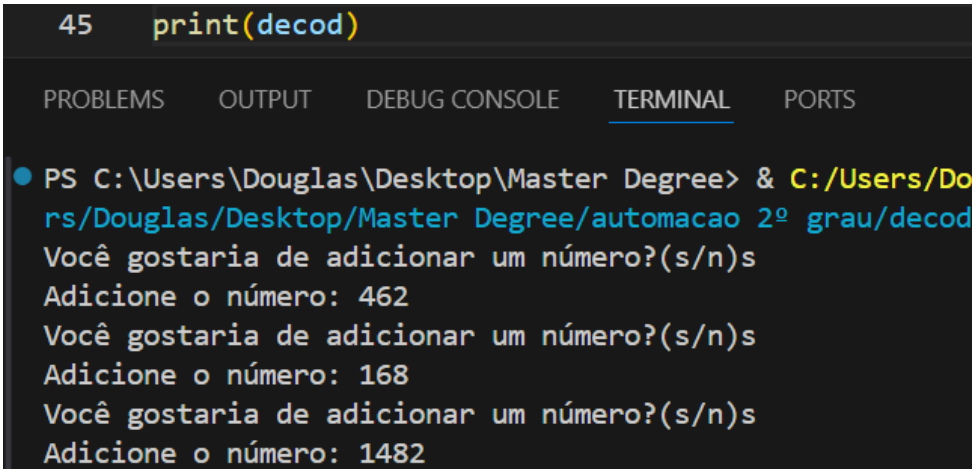


Figura 25: Entrada e saída de dados no terminal

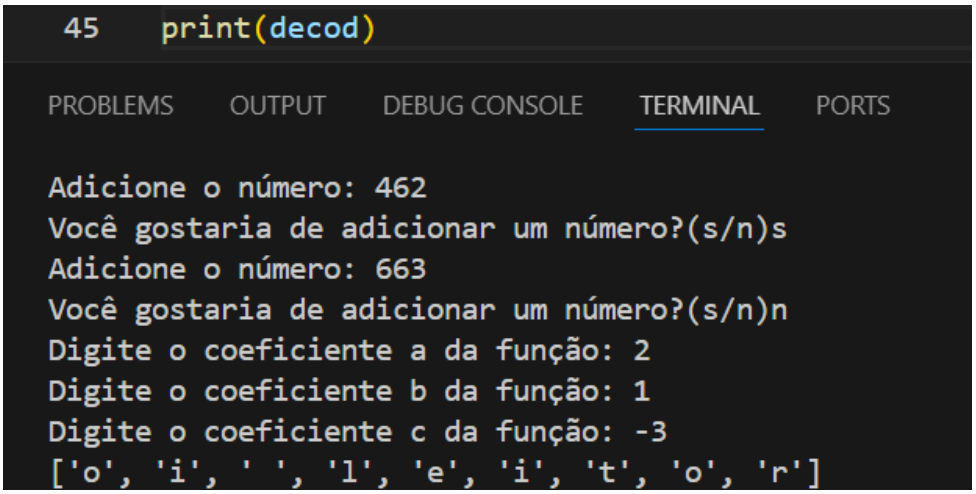


Figura 26: Entrada e saída de dados no terminal

Quando o código é executado, clicando na opção mostrada na Figura 7, aparecerá as entradas e as saídas mostradas no exemplo das Figuras 25 e 26.

Sequência Didática

Conteúdo

Funções afim e quadrática.

Habilidades (BNCC)

(EF09MA06) Compreender as funções como relações de dependência unívoca entre duas variáveis e suas representações numérica, algébrica e gráfica e utilizar esse conceito para analisar situações que envolvam relações funcionais entre duas variáveis.

(EM13MAT302) Construir modelos empregando as funções polinomiais de 1º ou 2º graus, para resolver problemas em contextos diversos, com ou sem apoio de tecnologias digitais.

(EM13MAT401) Converter representações algébricas de funções polinomiais de 1º grau em representações geométricas no plano cartesiano, distinguindo os casos nos quais o comportamento é proporcional, recorrendo ou não a softwares ou aplicativos de álgebra e geometria dinâmica.

(EM13MAT402) Converter representações algébricas de funções polinomiais de 2º grau em representações geométricas no plano cartesiano, distinguindo os casos nos quais uma variável for diretamente proporcional ao quadrado da outra, recorrendo ou não a softwares ou aplicativos de álgebra e geometria dinâmica, entre outros materiais.

(EM13MAT503) Investigar pontos de máximo ou de mínimo de funções quadráticas em contextos envolvendo superfícies, Matemática Financeira ou Cinemática, entre outros, com apoio de tecnologias digitais.

Objetivos

O objetivo principal é estimular a compreensão dos conceitos matemáticos por meio de atividades contextualizadas. Mediante a criptografia, os estudantes vivenciam a aplicação prática das funções, explorando relações de dependência entre variáveis, construindo modelos matemáticos e interpretando seus gráficos.

Anos: Ensino Médio.

Duração: Embora a sequência tenha cinco etapas, não estipulamos a quantidade de aulas, pois a construção dos conhecimentos pedidos em cada atividade podem exigir tempos diferentes para diferentes turmas.

Desenvolvimento

1ª etapa

Nessa etapa inicial, apresente aos alunos os problemas abaixo e peça que resolvam individualmente:

- Suponha que você está administrando uma empresa de entrega de alimentos e está analisando o desempenho de um dos seus entregadores. Você observa que, após 3 pedidos entregues, o número médio de pedidos entregues por hora por esse entrega-

dor é proporcional ao tempo de trabalho, além disso, você descobre que para cada hora trabalhada, o entregador entrega 2 pedidos. Considerando que das 8h às 8h40 foram entregues os 3 primeiros pedidos, qual o número de pedidos entregues até as 12h40?

- Considere uma função afim $f(x) = ax + b$. Se o coeficiente angular a é igual a 2 e o coeficiente linear b é igual a 3, determine o valor de $f(4)$.

- (UEL-PR) Um grupo de amigos alugou um ônibus com 40 lugares para uma excursão. Foi combinado com o dono do ônibus que cada participante pagaria R$60,00 pelo seu lugar e mais uma taxa de R$3,00 para cada lugar não ocupado. Qual o valor máximo que o dono do ônibus receberá?

- Considere uma função quadrática $f(x) = ax^2 + bx + c$. Se o coeficiente a é igual a -3, o coeficiente b é igual a 180 e o coeficiente c é igual a 0, determine o valor máximo da função.

Sondagem: essa primeira atividade serve como uma sondagem inicial. Ela é interessante porque põe os estudantes em contato com uma situação mais próxima da realidade em que precisam mostrar de que forma solucionariam os problemas apresentados.

Organização da turma: ao optar pelo trabalho individual, a intenção é fazer com que cada um acesse os conhecimentos que possui e busque solucionar a questão sozinho.

2ª etapa

O desenvolvimento do processo de criptografia utilizando funções envolve diversos conteúdos matemáticos, como funções afins e quadráticas, funções inversas, manipulação algébrica e resolução de sistemas de equações. Além disso, raciocínio lógico e habilidades de resolução de problemas são cruciais para seguir e implementar corretamente os passos da criptografia.

Traga os conceitos de função matemática utilizando os exemplos da 1ª etapa como base referencial para o assunto. Enfatize o conceito de **injetividade** de uma função, pois será uma propriedade essencial para a função que irá compor a chave criptográfica. Por fim, construa e analise os gráficos das funções vistas nos exemplos da 1ª etapa.

3ª etapa

Fazer uma breve apresentação sobre o conceito de criptografia. Após, apresentar o processo de codificar e decodificar uma mensagem. Uma opção aqui é realizar uma atividade em sala em que, em duplas, grupos interagem por meio do processo criptográfico.

Primeiramente, converse com a turma a respeito da escolha das correlações da tabela de cifras. Consequentemente, é determinado qual será o domínio em que a função estará. Em seguida, organize a turma em grupos. Por exemplo: em uma turma com 20 estudantes, poderiam-se formar quatro grupos com cinco integrantes cada. Dois desses grupos serão responsáveis por codificar e enviar uma mensagem, enquanto os outros dois grupos serão encarregados de decifrar a mensagem recebida.

Para ter-se um maior controle da atividade, algumas condição são necessárias como, por exemplo, um estudante escolher uma mensagem que contenha 8 palavras. Essa situação tornaria o processo muito extenso não sendo apropriado para uma atividade dentro do tempo de aula. Então, combine com os estudantes as seguintes condições:

- Jogar um dado de 6 faces duas vezes. O primeiro resultado é o valor do coeficiente a da função e o segundo o valor do coeficiente b. (de início estamos tratando da função afim)

- Jogar novamente um dado de 6 faces duas vezes. Tanto o primeiro quanto o segundo resultado se tratam do sinal dos coeficientes. Caso o resultado seja par, o sinal é positivo; se for ímpar, o sinal é negativo.

- Escolher uma frase com até 10 caracteres.

Até aqui, o primeiro grupo obterá a função polinomial e escolherá a mensagem. Na sequência, deve determinar o parâmetro m e cifrar a mensagem. Já o segundo grupo aguardará para receber, decifrar e, posteriormente, verificar com o primeiro grupo se a mensagem está correta.

Ao término da atividade reflita com os estudantes que encerraram a atividade quais as observações que fizeram sobre o procedimento.

Sondagem: Espera-se que o estudante compreenda bem os conceitos de domínio e imagem de uma função ao observar a mensagem original se tornando a mensagem codificada e, após, retornando à mensagem original por meio da decodificação.

Complemento: A criptografia pode ser apresentada com exemplos práticos, como

o *WhatsApp* e o *D4sign*[2]. O *WhatsApp* informa[3] aos usuários sobre a segurança das mensagens criptografadas. Já o *D4sign* garante a legitimidade das assinaturas eletrônicas.

No *D4sign*, o processo é o seguinte: o assinante recebe um e-mail com o documento a ser assinado. Este documento passa por uma função hash, transformando seu conteúdo em uma saída de tamanho fixo. A hash é então criptografada com a chave privada do assinante, criando a assinatura. Como a chave privada é exclusiva do assinante, apenas ele pode utilizá-la. Por fim, o documento assinado é enviado ao destinatário.

4ª etapa

Nesta etapa, selecione e distribua para os grupos responsáveis pela codificação, uma função quadrática que seja injetiva dentro de um domínio determinado (pode ser utilizada a mesma tabela de cifras já construída na etapa anterior). A partir da distribuição, os estudantes podem iniciar com o processo criptográfico. Diferente da etapa anterior, eles não precisarão jogar dados, pois a função já estará pronta. O que precisam fazer é escolher uma frase com até 10 caracteres. Não há alterações para os grupos que decodificarão a mensagem.

Assim como na etapa anterior, pontuem as observações lembrando-se sempre de relacionar com os conhecimentos de funções apresentadas desde o início da sequência didática.

Após desenvolverem toda a atividade manual, inclua o programa descrito no apêndice D da dissertação "Uma abordagem da criptografia com tecnologia para o aprimoramento da aprendizagem de funções polinomiais" (RODRIGUES, 2024), para trabalhar casos particulares. Nesse momento, peça para os estudantes realizarem o processo criptográfico de forma automatizada. A intenção é observar os casos em que a função quadrática não é injetiva no domínio e, então, ver e analisar os problemas que podem ocorrer.

Sondagem: Esse momento é uma ótima oportunidade para apresentar de que forma cada coeficiente da função quadrática influencia no formato do gráfico. Aproveite e peça para os estudantes observarem e fazerem o mesmo com a função afim, caso não tenham percebido na etapa anterior.

5ª etapa

[2]https://d4sign.com.br/
[3]Apêndice B

Este é o momento em que entramos em contato com a automação. Apresente aos estudantes a linguagem de programação Python fazendo alguns comandos simples e curtos, como adicionar um valor a uma variável ou realizar a soma de alguns números e mostrar na tela.

Orientações: O papel do estudante nessa etapa é fazer a "tradução"dos procedimentos feitos a mão para a linguagem de programação Python. A cada fase de cálculo realizado nas etapas anteriores, há uma maneira de se realizar computacionalmente e essa atividade é o foco nessa quinta etapa da sequência didática. É esperado que o estudante reconheça o que é necessário em cada fase de cálculo e utilize as ferramentas da linguagem de programação necessárias para que a automatização funcione.

Avaliação

A avaliação se dará de forma processual durante o decorrer das cinco etapas. Alguns pontos principais a serem desenvolvidos são:

1. Participação e Atitude: Observar o engajamento dos alunos nas atividades em grupo, a participação nas discussões, a colaboração e a criatividade na resolução de problemas; Avaliar a capacidade de os alunos formular perguntas relevantes, apresentar ideias e justificar suas respostas; Avaliar o interesse dos alunos pelos temas abordados, a curiosidade e a disposição para aprender.

2. Trabalho em Grupo: Avaliar a capacidade dos alunos de aplicar corretamente as funções afim e quadrática para codificar e decodificar mensagens, utilizando os coeficientes e sinais de forma precisa; Avaliar a capacidade dos alunos de analisar os resultados da criptografia, identificando a influência dos coeficientes nas funções e a relação entre domínio e imagem; Avaliar a colaboração e a comunicação entre os membros do grupo durante a realização da atividade.

3. Programação em Python: Avaliar a capacidade dos alunos de traduzir o processo de criptografia manual para o código Python, utilizando corretamente as estruturas de dados, variáveis, operadores e comandos de entrada e saída; Avaliar a capacidade de os alunos em testar o código, identificar e corrigir erros (debugging) e garantir que o programa funcione corretamente.

4. Teste Individual: Avaliar a compreensão dos alunos sobre os conceitos de função, domínio, imagem, coeficientes angular e linear, e a relação entre a representação algébrica, gráfica e tabular de funções; Avaliar a capacidade dos alunos de resolver proble-

mas que envolvam funções afim e quadrática em situações reais, como cálculo de custos, análise de gráficos e interpretação de dados.

Considerações Finais

Este produto educacional propôs automatizar o processo criptográfico e estudá-lo de forma a possibilitar o desenvolvimento de algumas funções matemáticas, com foco nos estudantes do ensino médio.

O ensino de programação nas escolas é incentivado por diversas organizações ao redor do mundo. Um exemplo é o projeto "Hora do Código"oferecido pela organização Code.org, que é uma introdução gratuita à ciência da computação com atividades e vídeos para alunos de todos os nveis de habilidade. A Base Nacional Comum Curricular (BNCC) também oportuniza, em seu documento, a possibilidade de aprimorar os conhecimentos em matemática por meio de ações tecnológicas digitais. A automação do processo criptográfico conversa diretamente com essas fontes e permite criar situações enfáticas que desenvolvem o pensamento computacional.

Embora o Python apresente estruturas de repetição ou decisão que precisam ser escritas ou lidas em inglês, o que pode representar um desafio adicional para estudantes que não dominam esse idioma, existem alternativas como PORTUGOL e o SCRATCH. No entanto, essas linguagens possuem menos recursos em comparação com o Python.

Por fim, a integração da programação com a matemática não apenas enriquece o currículo escolar, mas também prepara os estudantes para um futuro onde o pensamento computacional e a habilidade de resolver problemas complexos serão cada vez mais valorizados. As propostas de melhorias e alternativas mencionadas oferecem um caminho para futuras pesquisas e implementação prática no ambiente educacional.

Referências Bibliográficas

BRASIL (2018). Base nacional comum curricular. Disponível em `http://http://basenacionalcomum.mec.gov.br/images/BNCC_EI_EF_110518_versaofinal_site.pdf` Acesso em: 15/12/2023.

FIARRESGA, V. M. C. (2010). Criptografia e matemática. Dissertação de Mestrado, Universidade de Lisboa (Portugal).

KAHN, D. (1968). *The Codebreakers*. The Macmillan Company, Nova Yorque.

LAMBERT, K. A. (2019). *Fundamentals of Python First Programs*. Cengage.

PONTES, E. A. S., SILVA, B. H. M. d. S., OLIVEIRA, E. G., LEITE, L., et al. (2022). Criptografia em funções polinomiais: Um processo de ensino e aprendizagem de matemática na educação básica. *The Journal of Engineering and Exact Sciences*, 8(6):14609–01e.

RODRIGUES, D. V. A. (2024). Uma abordagem da criptografia com tecnologia para o aprimoramento da aprendizagem de funções polinomiais. Dissertação de Mestrado, Universidad Federal de Mato Grosso).

SHOUP, V. e BONEH, D. (2023). A graduate course in applied cryptography. Disponível em `http://toc.cryptobook.us/` Acesso em: 19/03/2024.