Números vs Números: Jogo Numérico da UEFS

Pedro Lucas F. de Souza

[1]UEFS – Universidade Estadual de Feira de Santana, Av. Transnordestina, s/n, Novo Horizonte, Feira de Santana – BA, Brasil – 44036-900

pedrofernandeseng.comp@gmail.com

Abstract. *This report aims to describe the phases of developing a numerical board game in Python interpretive language, whose motivation is the recruitment of the gaming league of the IEEE branch at Universidade Estadual de Feira de Santana. Methodologically, it's described the structures and concepts used for its development, as lists, matrices, dictionaries and matrices, such as the sparse matrix concept as a basis for building the board, in addition to file concepts for storing information, such as pickle and csv, and the management structures and task executions known as functions. There is a description of how these concepts were applied to each need of the problem. Some situations are considered when testing the game, involving its main parts, such as running the game, special moves and storing the game to be returned later. Errors were addressed in this document, describing their nature and proposing a computational solution for their resolution.*

Resumo. *Este relatório objetiva descrever as etapas do desenvolvimento de um jogo de tabuleiro numérico em linguagem interpretativa Python, tendo como motivação o recrutamento da liga de jogos do ramo IEEE da Universidade Estadual de Feira de Santana. Metodologicamente, descreve-se as estruturas e conceitos utilizados para seu desenvolvimento, como listas, matrizes, dicionários e tuplas, tal como o conceito de matriz esparsa como base para a construção do tabuleiro; além de conceitos de arquivos para armazenar informações, como pickle e csv, e as estruturas de gerenciamento e execuções de tarefas conhecidas como funções. Há descrição de como esses conceitos foram aplicados em cada necessidade do problema. São tomadas algumas situações amostrais para realizar os testes do jogo, envolvendo suas partes principais, como a execução do jogo, jogadas especiais e armazenamento do jogo para ser retomado posteriormente. Erros foram tratados nesse documento, descrevendo sua natureza e propondo uma solução computacional para sua resolução.*

1. Introdução

A humanidade, ao longo de sua história, sempre elaborou formas lúdicas de passar o tempo e integrar-se com a comunidade. Na antiguidade grega, a civilização helênica organizava jogos em momentos de trégua entre os conflitos das cidades-estados da região e era considerada uma forma de mostrar prestígio e força através das habilidades físicas dos participantes, o que atribuía aos jogos um aspecto de competitividade [Afonso 2024]. Com o passar das gerações, muitos jogos surgiram e alguns centravam-se na competição intelectual entre seus participantes. Um exemplo disso é o jogo de xadrez, surgido na

índia em meados do século VI d.C., que consiste em dois jogadores movendo peças com movimentos pré-definidos [Dantas 2024].

O advento da computação moderna na primeira metade do século XX e os avanços da eletrônica permitiram o surgimento de jogos eletrônicos. Na década de 50, o físico William Higinbotham desenvolveu o que pode ser considerado o primeiro jogo eletrônico: Tennis for Two. O jogo era jogado em um osciloscópio[1], em que dois jogadores controlavam a trajetória da bola, simulando um jogo de Tênis. Mais tarde, na década de 70, jogos como "Pong", que também simulava um jogo de tênis, marcaram a ascensão do entretenimento digital, então os jogos se desenvolveram cada vez mais conforme a computação e a eletrônica também evoluíam [Grupo EPJ 2024].

Nessa perspectiva, a liga de jogos do ramo IEEE da Universidade Estadual de Feira de Santana, a fim de recrutar novos integrantes, lançou o desafio de desenvolvimento de um jogo numérico chamado "Tabuleiro de Números", um jogo com tabuleiro de tamanho N por N casas e jogado por dois jogadores. Cada jogador, alternadamente, preenche as casas do tabuleiro com números inteiros positivos disponíveis, sendo o menor valor igual a 1; e o maior, N^2. Cada jogador recebe um objetivo, que pode ser igual ou não ao do adversário, entre quatro possíveis: formar sequências numéricas ordenadas ascendentes, descendentes, pares ou ímpares. A vitória é dada para aquele que cumprir seu objetivo primeiro; todavia, caso nenhum dos dois jogadores cumpra seu objetivo, o resultado do jogo é empate. O tamanho do tabuleiro é definido pelos níveis de dificuldade: fácil, normal e difícil, sendo seus tamanhos dados por 3 por 3, 4 por 4 e 5 por 5, respectivamente. Além disso, o jogo deve contar com dois modos de jogos: com ou sem jogada especial. A jogada especial, permitida ser jogada apenas uma vez por jogador, é a habilidade de eliminar uma linha ou coluna do tabuleiro seguida da jogada convencional. Ele deve apresentar interface interativa e intuitiva, a fim de ser atraente para os jogadores, além de possuir um *ranking* para adicionar o sentimento de competitividade entre os jogadores. Ele também deve conter a opção de ser salvo para ser retomado posteriormente.

Em aspectos técnicos, o código-fonte, escrito em linguagem interpretativa Python, deve ser estruturado e modularizado de modo a permitir atualizações e reúso de funções, além de apresentar consistência, robustez e confiabilidade para minimizar e evitar falhas. Nesse sentido, este relatório descreve a produção desse jogo, através da descrição ordenada em metodologia, referenciais teóricos, desenvolvimento, resultados, manual de uso, testes e erros encontrados, tal como as soluções pensadas para os mesmos.

2. Metodologia

2.1. Referencial teórico

O desenvolvimento do código-fonte, essencialmente, dependeu do uso de estruturas complexas capazes de armazenar informações, gerenciar e executar tarefas e guardar na memória do computador. As estruturas de armazenamento são listas, matrizes, dicionários e tuplas; as estruturas de gerenciamento e execução de tarefas são as funções; e as estruturas para guardar na memória são arquivos *csv* e *pickle*.

As listas são estruturas de dados capazes armazenar elementos ordenadamente conforme suas posições ao longo da estrutura. Quando uma lista armazena outras lis-

[1] Aparelho utilizado para fazer medições elétricas, sobretudo de fenômenos elétricos ondulatórios.

tas, esse aninhamento é chamado de matriz e possui as mesmas propriedades de uma lista [Menezes 2016]. Os dicionários, por outro lado, não são diretamente ordenáveis, e armazenam dados encadeados em pares chave-valor [Menezes 2016], e isso é útil para relacionar pares de dados entre si. As tuplas são semelhantes às listas, mas possuem a característica de imutabilidade, não podendo adicionar, alterar ou remover elementos [Menezes 2016].

As estruturas responsáveis gerenciar e executar tarefas são chamadas de funções. Elas executam tarefas em blocos de códigos que podem ou não depender de parâmetros, que são computados no escopo da função e podem ser transformados e retornados [Menezes 2016], e isso é interessante para transformar dados e estruturas.

Os arquivos *csv* e *pickle* são dois tipos de arquivos responsáveis por armazenar dados. Os arquivos do tipo *csv* armazenam dados de forma tabular, organizados por linha e separados por delimitadores, como vírgula [Python Software Foundation 2024]. Eles são frequentemente utilizados para armazenar informações que se relacionam entre si em uma mesma linha. Os arquivos *pickle*, por outro lado, são arquivos usados para salvar e carregar objetos em Python. Esse processo é conhecido como serialização e desserialização de objetos, em que os objetos são convertidos em binários e salvos e, depois, convertidos de volta à forma de objeto ao serem carregados. Eles são capazes de armazenar qualquer tipo de objeto de Python, como listas, dicionários, tuplas, inteiros e outros [Python Software Foundation 2024].

Para sortear os objetivos dos jogadores e quem iniciará a partida, a biblioteca *random* foi utilizada. Ela possui um método de comando chamado *randint* que permite retornar valores inteiros em um intervalo dado. No caso, a biblioteca retorna os valores inteiros 1 e 2 para decidir quem será o primeiro jogador a iniciar: jogador 1 ou 2. Para sortear os objetivos, ela retorna valores entre 1 e 4, em que cada inteiro representa um objetivo. Além disso, o método *try... except*, usado para lidar com exceções durante a execução do programa, foi essencial para lidar com erros do tipo *FileNotFoundError* [Python Software Foundation 2024].

O jogo também conta com a tabela ANSI, responsável por padronizar conjuntos de codificações simbólicas e modos gráficos. Ela foi útil para incluir cores e fazer diferenciação entre seus jogadores: o jogador 1 é representado pela cor azul; o jogador 2, pela cor vermelha. Essa tabela conta com códigos de cores e um código de *reset*, que retorna a cor padrão do terminal.

2.2. Desenvolvimento do jogo

As estruturas do código-fonte do jogo podem ser divididos em cinco grandes blocos de tarefas: análise, geração, interface, transformação e jogo. As estruturas de análise são responsáveis por validar e armazenar informações conforme os comandos dados pelo usuário. As estruturas de geração servem para gerar o tabuleiro e suas estruturas dependentes, essencialmente dicionários, conforme o nível de dificuldade decidido pelo usuário. Essas estruturas também retornam o objetivo dos jogadores e sorteia qual jogador iniciará a partida. As estruturas de interface são responsáveis por permitir a interatividade entre o programa e o usuário. A estrutura de transformação é responsável por alterar o tabuleiro conforme as jogadas feitas, como alterações de dicionários e listas de números possíveis. Por fim, a estrutura do jogo é a função geral que coordena as jogadas, sua verificação e

a interface. Ela é essencial para a modularização do jogo, pois seus parâmetros podem armazenados e retornados, como é o caso de salvar o jogo e retorná-lo.

As estruturas de análise são essencialmente matrizes, listas, e, em especial, os dicionários e as tuplas. Os dicionários e as tuplas em combinação podem formar uma matriz com índices, tendo as tuplas como chaves. Essa combinação é conhecida como matriz esparsa. No código-fonte, esse tipo de matriz é aplicada em dois contextos: a matriz de jogadas, em que as jogadas são armazenadas e possuem valores iniciais iguais a 100 negativo[2], e a matriz visual, que armazena a *string* "x", usada para representar o espaço vazio no jogo. As listas armazenam informações pertinentes aos jogadores, retornadas a partir de funções de geração, além de serem usadas para validar sequências nas linhas, colunas e diagonais para verificar vitória.

As listas também são responsáveis por implementar ou acrescentar os pontos de jogadores no *ranking*, que são escritos no arquivo *csv* para manter o *ranking* atualizado. A contabilidade dos pontos é feito a cada vitória, e é padronizado a pontuação baseada em múltiplos de 10, conforme os dados da tabela 1. As matrizes nesse contexto também armazenam listas com informações do *ranking* após a leitura do arquivo *csv* para serem impressas na tela.

Nível de dificuldade	Pontos
Fácil	10
Normal	20
Difícil	30

Tabela 1. Tabela de pontuação conforme nível de dificuldade.

As estruturas de geração geram o tabuleiro de jogos conforme o nível de dificuldade selecionado pelo usuário. Essas funções usam iteradores do tipo *for* para construir a matriz visual, usada para interagir com o usuário, e a matriz de jogadas, usada para realizar verificações. Essa construção itera os índices dos dicionários, que são tuplas, como em um sistema de coordenadas. A estrutura de geração também é responsável por gerar os números possíveis de serem jogados no nível selecionado.

As estruturas de interface são constituídas de funções que executam tarefas como impressão de menus, formatação de tabuleiro, números jogados e a disponibilidade de números para as próximas jogadas. As funções responsáveis pelos menus de opções na tela solicitam entradas numéricas correspondentes às opções disponíveis e retornam esses valores para serem computados. Essas funções são essencialmente compostas de comandos de *print* e *input*, responsáveis por imprimir as opções e solicitar a entrada do usuário, respectivamente. A formatação do tabuleiro é constituída por uma construção de uma matriz a partir do dicionário, que será impressa pela função responsável por imprimir o tabuleiro.

As estruturas de transformação essencialmente são usadas para modificar o tabuleiro de jogos. Duas funções são responsáveis por isso: a função de jogada e a função de jogada especial. Cada uma, em seu momento de execução, recebe como parâmetro as

[2]O valor é arbitrário e negativo o suficiente para não intervir nas verificações, contudo pode ser modificado a depender das necessidades do desenvolvedor em atualizações futuras.

matrizes de jogadas e visuais e seus dependentes, que têm seus elementos modificados conforme a jogada. No caso da função de jogada, ela recebe como outros parâmetros a linha, a coluna, que formam a tupla para indicar o índice do elemento; e o número a ser jogado. Esse número é incrementado em ambas as matrizes e passado para as estruturas de análise para verificações. Essas estruturas também validam se o número é possível de ser jogado. Em caso negativo, o sistema informa, então o jogador refaz sua jogada. No caso da função de jogada especial, ela recebe como parâmetros as matrizes visuais e de jogadas e a lista de números jogados. O usuário decide se deseja eliminar a linha ou coluna. Os valores da linha e da coluna eliminados são retornados para suas configurações iniciais em ambas as matrizes, e os números já jogados presentes na linha ou coluna eliminada retornam para os valores possíveis a serem jogados.

Além dessas, para salvar o jogo, existem duas funções: armazenar o jogo em um arquivo *pickle*, que o salva com as informações necessárias, tal como matrizes visuais e de jogadas, tamanho, jogador da vez, especiais e informações dos jogadores, que podem ser retomados com a leitura do arquivo feita pela segunda função e aplicados na opção de continuar o jogo salvo[3] caso o usuário deseje.

3. Resultados e discussões

3.1. Manual de uso

Para executar o programa, é necessário que haja um interpretador Python instalado na máquina. Segue adiante um passo-a-passo para a execução do jogo.

1. Abra o interpretador.
2. Navegue até o diretório onde está o código-fonte.
3. Execute o código-fonte com o interpretador.

A interface é intuitiva, os menus são compostos de opções vinculadas a números, bastando digitar o valor numérico correspondente à opção que se deseja. Caso deseja-se iniciar o novo jogo, selecione a opção, sua dificuldade e se deseja ter o modo com jogada especial. O jogo se inicia após a digitação dos nomes dos jogadores e o sorteio dos objetivos. A cada fim de jogada, o sistema pergunta se deseja salvar a partida. Em caso afirmativo, o jogo é fechado e salvo, e o jogo é retornado para o menu principal. O sistema também notifica caso não haja *ranking* e jogo salvo presentes. Caso não haja tais arquivos presentes, eles são criados e guardam suas respectivas informações.

3.2. Testes

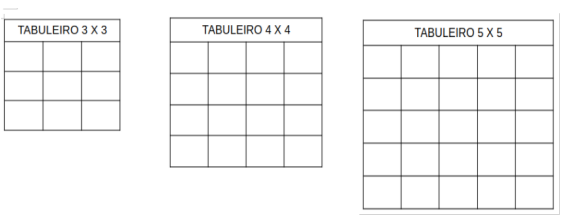

Figura 1. Tabuleiros

[3] A opção de continuar o jogo possui exatamente a mesma função da estrutura de jogo a fim de modularizar o código.

A figura 1 destaca os três tipos de tabuleiro disponíveis. Note que cada tabuleiro é dado por seu nível de dificuldade em relação ao tamanho. Para os testes, foram considerados algumas amostras situacionais: objetivos de jogo semelhantes, vitória, *ranking*, objetivos de jogo diferentes, empate e jogada especial. Considera-se um tabuleiro de tamanho 4 x 4 para os testes.

Considera-se os jogadores 1 e 2, respectivamente. com os nomes fictícios Fulano e Beltrano. Considera-se os sorteios feitos mostrados na figura 2 para o teste de objetivos semelhantes na partida.

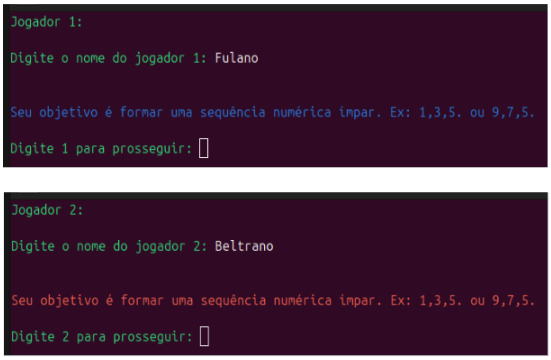

Figura 2. Jogo com objetivos iguais.

Ao fazer os jogadores jogarem alternadamente ao longo da diagonal principal, formando uma sequência de ímpares, sendo o jogador 2 sendo sorteado como o primeiro a fazer a jogada. Nota-se, portando, que o último jogador a fazer a jogada no último elemento da diagonal, fazendo uma leitura da esquerda para a direita, é o jogador 1, como mostra a figura 3. Nota-se que a vitória é dada para o jogador 1, pois ele concluiu o objetivo primeiro.

Figura 3. Fim de partida para o jogo com objetivos iguais.

Conforme a figura 4, observe que o *ranking* foi atualizado, e o jogador Fulano recebeu 20 pontos pela vitória, seguindo a padronização de pontos referenciados na tabela 1.

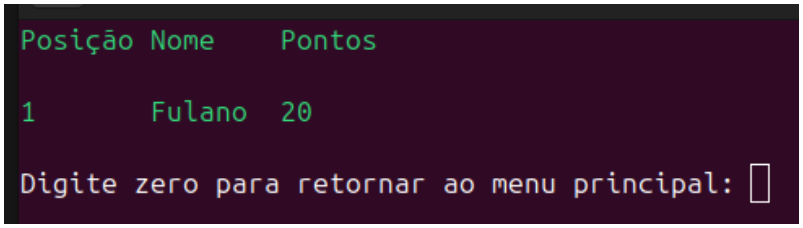

Figura 4. *Ranking* após a vitória do jogador 1.

Considera-se agora dois objetivos diferentes, conforme ilustra a figura 5.

Figura 5. Jogo com objetivos diferentes.

Aplicando-se o mesmo método de alternância aplicado ao exemplo anterior, tomando o jogador 2 como iniciador da partida, observa-se na figura 6 que, apesar do jogador 1 ter sido o último a jogar, ele completou o objetivo do jogador 2, dando-lhe a vitória.

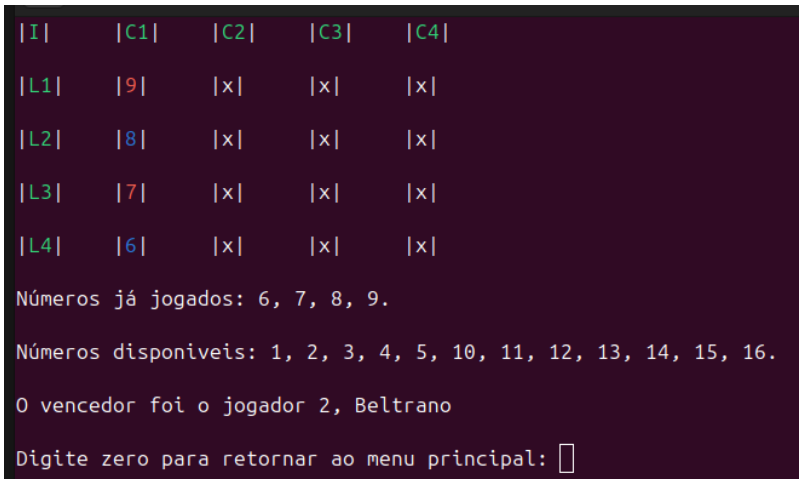

Figura 6. Fim da partida do jogo com objetivos diferentes.

Considerando agora que os jogadores não conseguiram completar seus objetivos[4], observa-se na figura 7 que os jogadores chegaram em um empate.

[4]Neste caso, ambos tinham o mesmo objetivo: formar sequência de ímpares.

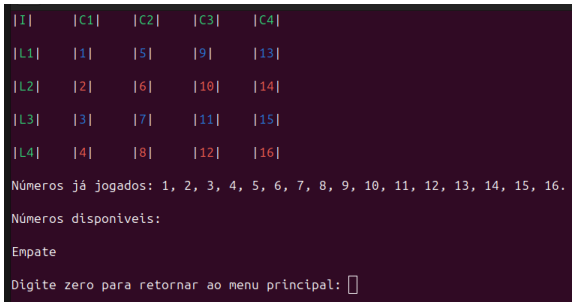

Figura 7. Jogo em caso de empate.

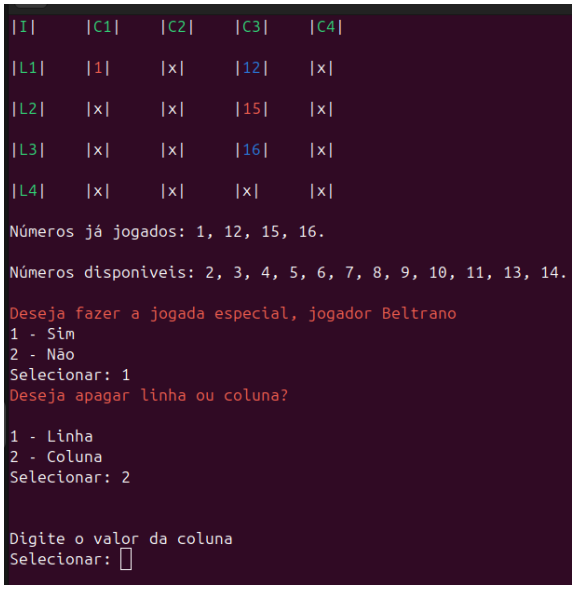

Figura 8. Jogo antes da realização da jogada especial.

Em caso de jogadas especiais, observa-se que o jogo mostrado na figura 8 mostra que o jogador 2 optou por fazer a jogada especial, eliminando os valores da coluna 3. Após a realização da jogada especial, o resultado é mostrado na figura 9.

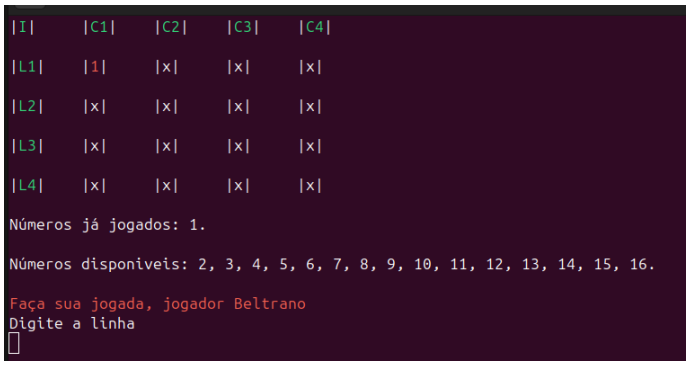

Figura 9. Jogo após a jogada especial.

3.3. Erros

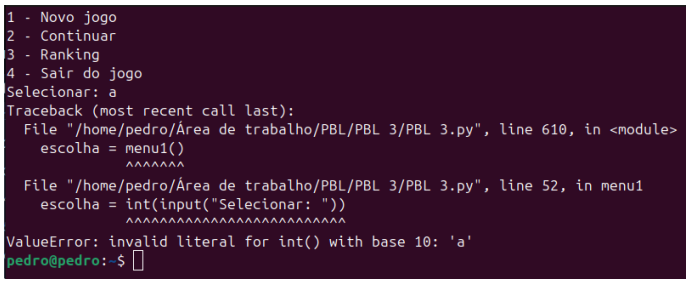

Figura 10. Erro ao fazer entrada incorreta de valores.

Os principais erros encontrados são referentes ás entradas de natureza diferente às solicitadas, como digitar uma *string*. Esse erro é chamado de *ValueError* e é referenciado na figura 10. Por isso, esse jogo foi pensado para ter suas entradas realizadas por usuários perfeitos. Este erro é comum em todas as partes do jogo em que se solicita uma entrada numérica.

4. Conclusão

Este relatório apresentou uma solução computacional em linguagem interpretativa Python de um jogo numérico. A solução resolve o problema proposto com a utilização de métodos e estruturas de dados, como listas, tuplas, matrizes e dicionários para armazenar e verificar as entradas. Ela traz as funções para dividir as tarefas em blocos para melhor compreensão do código e tornar a codificação mais organizada e modularizada, além de tornar o aspecto do jogo mais dinâmico, interativo e atraente através de menus e entradas de opções. A solução também foi pensada para armazenar informações na memória do computador, como os arquivos do tipo *pickle* para armazenar as informações do jogo, e os do tipo *csv* para armazenar e manter atualizado *ranking* dos jogadores.

O sistema possui erros que derivam de entradas incorretas, como objetos do tipo *string*. Versões futuras podem tratar melhor esse tipo de erro, através da implementação de um sistema de validações de entrada de objetos de tipos diferentes da solicitada pelo jogo, e tornar o código com menos falhas de execução, tornando a experiência do usuário mais segura e confortável para o jogo.

Referências

Afonso, L. (2024). Olimpíadas (jogos olímpicos). https://brasilescola.uol.com.br/educacao-fisica/olimpiadas.htm. Acesso em: 04 de Julho de 2024.

Dantas, P. L. (2024). Xadrez. https://mundoeducacao.uol.com.br/educacao-fisica/xadrez.htm#:~:text=O%20surgimento%20do%20xadrez%20se,que%20prov%C3%AAm%20o%20nome%20xadrez. Acesso em: 04 de Julho de 2024.

Grupo EPJ (2024). Videogame: Uma jornada épica pelo universo dos jogos eletrônicos. https://epraja.com.br/videogame-uma-jornada-epica-pelo-universo-dos-jogos-eletronicos/. Acesso em: 04 de Julho de 2024.

Menezes, N. N. C. (2016). *Introdução à programação com Python*. Novatec.

Python Software Foundation (2024). Python documentation. `https://docs.python.org/3.12/`. Acesso em: 01 de Julho de 2024.