ESTGL – ESCOLA SUPERIOR DE GESTÃO DE LAMEGO

COMPUTAÇÃO CLOUD

Miguel Ângelo Correia Pereira – 11177

Relatório de projeto final no âmbito do CTESP em Computação Cloud

Moimenta da Beira, janeiro 2025

ESTGL – ESCOLA SUPERIOR DE GESTÃO DE LAMEGO

COMPUTAÇÃO CLOUD

Trabalho elaborado no âmbito da unidade curricular de Princípios de Programação.

Miguel Ângelo Correia Pereira – 11177

Moimenta da Beira, janeiro 2025

RESUMO

Este relatório apresenta o desenvolvimento de um sistema de *slot machine* programado em *Python*, com uma abordagem modular e foco na interatividade do utilizador.

Segundo (Lutz, 2013)" a linguagem *Python* é amplamente reconhecida pela sua simplicidade e versatilidade, sendo uma escolha popular para o desenvolvimento de aplicações interativas".

O projeto teve como objetivo explorar conceitos fundamentais de programação, como controle de fluxo, geração de números aleatórios, manipulação de estruturas de dados, e interação em tempo real, dentro de um ambiente de jogo digital.

O sistema foi concebido com uma estrutura que prioriza a separação de responsabilidades, dividindo as funcionalidades em módulos independentes. A lógica do jogo centra-se na simulação de uma máquina de *slot* com cinco colunas e combinações baseadas em símbolos gráficos. A funcionalidade principal inclui:

1. Gestão de saldo do utilizador: O sistema permite ao utilizador depositar fundos, levantar dinheiro e consultar o saldo atual. Essas operações são realizadas com validações rigorosas, como limites de depósitos e restrições para evitar saldos negativos.
2. Mecânica de jogo: Após definir o valor da aposta, o utilizador inicia a *slot machine*, que gera combinações aleatórias de símbolos. Estas combinações são verificadas para identificar padrões vencedores, e os ganhos são calculados com base em uma tabela de multiplicadores associada a cada símbolo.
3. Interface interativa: A interação com o utilizador ocorre por meio de menus dinâmicos apresentados em linha de comando. Estes menus são desenhados para facilitar a navegação entre as funcionalidades, com mensagens claras e opções organizadas.

A tabela de pagamentos do jogo foi desenvolvida para garantir equilíbrio entre risco e recompensa, utilizando probabilidades personalizadas para cada símbolo. Símbolos como cerejas oferecem recompensas menores, mas aparecem com maior frequência, enquanto símbolos raros como diamantes oferecem pagamentos substancialmente maiores, mas são menos prováveis.

Foram empregues diversas bibliotecas nativas do *Python* para enriquecer a experiência do utilizador. A biblioteca *random* foi utilizada para gerar as combinações de símbolos; os e

shutil permitiram melhorar a apresentação no terminal, centralizando os resultados e oferecendo uma experiência visual mais imersiva.

O desenvolvimento do projeto também envolveu desafios relacionados com a verificação de ganhos. O sistema analisa as combinações geradas para identificar sequências vencedoras, tanto em linhas horizontais quanto em combinações mais complexas, atribuindo pagamentos adequados conforme a aposta e os multiplicadores.

Embora funcional e interativo, o projeto apresenta espaço para melhorias significativas. Entre as sugestões para desenvolvimentos futuros destacam-se:

Interface Gráfica (GUI): A transição de uma interface em linha de comando para uma gráfica utilizando bibliotecas como *tkinter* ou *PyQt* tornaria o sistema mais intuitivo e visualmente apelativo.

Persistência de Dados: A inclusão de um banco de dados para registar o progresso do utilizador, estatísticas e histórico de apostas aumentaria a longevidade e o envolvimento com o jogo.

Implementar múltiplas linhas de pagamento e mecanismos adicionais, como bónus ou jackpots, traria maior complexidade e diversão ao sistema. Como afirma (Sweigart, 2015)" o desenvolvimento de jogos digitais requer uma abordagem modular, onde a lógica do jogo é separada da interface do utilizador".

ABSTRACT

This report presents the development of a slot machine system programmed in Python, employing a modular approach with a focus on user interactivity. The project aimed to explore fundamental programming concepts, such as control flow, random number generation, data structure manipulation, and real-time interaction, within the context of a digital game.

The system was designed with a structure that prioritizes the separation of responsibilities, dividing functionalities into independent modules. The game logic revolves around simulating a slot machine with five columns and combinations based on graphical symbols. The primary functionality includes:

1. User balance management: The system allows users to deposit funds, withdraw money, and check their current balance. These operations are carried out with strict validations, such as deposit limits and safeguards to prevent negative balances.
2. Game mechanics: After defining the bet amount, users spin the slot machine, generating random symbol combinations. These combinations are checked for winning patterns, and payouts are calculated based on a predefined multiplier table associated with each symbol.
3. Interactive interface: User interaction occurs through dynamic command-line menus. These menus are designed to simplify navigation between functionalities, with clear messages and organized options.

The game's payout table was developed to balance risk and reward, using customized probabilities for each symbol. Symbols like cherries offer smaller rewards but appear more frequently, while rare symbols like diamonds provide significantly higher payouts but are less likely to occur.

Various Python libraries were utilized to enhance the user experience. The random library was employed to generate symbol combinations; os and shutil improved the terminal display by centralizing results and offering a more immersive visual experience.

The project also addressed challenges related to verifying winnings. The system analyzes generated combinations to identify winning sequences, both in horizontal lines and more complex patterns, assigning payouts according to the bet and multipliers.

Although functional and interactive, the project offers significant opportunities for further development.

Proposed enhancements include:

1. Graphical User Interface (GUI): Transitioning from a command-line interface to a graphical interface using libraries like tkinter or PyQt would make the system more intuitive and visually appealing.
2. Data Persistence: Adding a database to record user progress, statistics, and betting history would increase engagement and replayability.
3. Improved Game Logic: Implementing multiple paylines and additional mechanisms, such as bonus rounds or jackpots, would introduce greater complexity and enjoyment.

This project demonstrates the practical application of fundamental programming knowledge, serving as a didactic tool to consolidate theoretical concepts in a playful context. The use of Python was essential to the project's success, owing to its simplicity, versatility, and extensive library resources.

The system provides a solid proof of concept for building applications that combine user interaction, probability, and computational logic, serving as a foundation for more advanced projects in the future.

AGRADECIMENTOS

Gostaria de expressar a minha profunda gratidão à ESTGL por me proporcionar os recursos e o ambiente necessário para o desenvolvimento deste projeto, permitindo-me transformar ideias em realidade e aprofundar os meus conhecimentos.

Aos professores, quero agradecer pela paciência, pela disponibilidade constante e por cada momento de aprendizagem partilhado. O vosso compromisso e incentivo foram fundamentais para que este trabalho fosse possível. Obrigado por sempre me desafiarem a ir mais longe e por partilharem o vosso entusiasmo pelo conhecimento.

Dedico um agradecimento muito especial ao meu orientador, Professor Cristiano, por acreditar no meu potencial desde o início. A sua orientação foi muito mais do que técnica; foi uma fonte constante de inspiração e criatividade. A sua abordagem "fora da caixa" e o seu olhar atento ajudaram-me a superar desafios e a atingir resultados que inicialmente pareciam inalcançáveis.

Finalmente, agradeço também aos meus colegas, amigos e família que, direta ou indiretamente, me deram forças e motivação para levar este projeto a bom porto. Este trabalho é fruto não só do meu esforço, mas também do apoio incondicional de todos os que estiveram ao meu lado nesta jornada. Obrigado por acreditarem em mim.

ÍNDICE

INTRODUÇÃO

Este projeto consiste no desenvolvimento de um jogo de *Slot Machine* em *Python*, criado para oferecer uma experiência de jogo interativa e divertida, ao mesmo tempo que explora conceitos fundamentais da programação. O programa simula uma *slot machine* digital, onde os utilizadores podem realizar depósitos, definir o valor das apostas, jogar rodadas, verificar o saldo e levantar dinheiro, permitindo ainda acompanhar os ganhos e as perdas de forma simples e intuitiva.

A lógica do jogo é baseada em probabilidades e combinações típicas de máquinas de apostas reais. Os símbolos apresentados na *slot* têm diferentes valores associados, com pagamentos que variam consoante a raridade e o tipo de combinação obtida. O jogo inclui funcionalidades que tornam a experiência envolvente, como rolos dinâmicos que exibem símbolos de forma aleatória e pagamentos calculados automaticamente com base nos resultados.

Entre as principais funcionalidades do programa estão o menu principal, onde o utilizador pode navegar entre diferentes opções, como efetuar depósitos, levantar dinheiro, verificar o saldo ou iniciar o jogo. O sistema de depósito e levantamento permite que o jogador adicione fundos ao saldo disponível ou retire valores acumulados, assegurando que o saldo nunca seja negativo. A *slot machine* em si é a peça central do projeto, oferecendo uma experiência visual que simula os rolos clássicos de máquinas de apostas, com combinações aleatórias e ganhos proporcionais às apostas realizadas.

A lógica de jogo integra pagamentos baseados em probabilidades, com cada símbolo a ter um valor específico e combinações únicas que oferecem multiplicadores que aumentam significativamente os ganhos. Para além disso, o utilizador pode consultar o saldo em qualquer momento e acompanhar o desempenho financeiro global, o que inclui uma visão clara das suas apostas e dos ganhos ou perdas ao longo das rodadas.

O projeto foi desenvolvido utilizando apenas bibliotecas padrão do *Python*, como a biblioteca *random* para gerar combinações aleatórias. O terminal foi configurado para apresentar os resultados de forma organizada e visualmente apelativa, com os rolos da *slot* centralizados e animações simples que simulam o funcionamento de uma máquina real.

Este jogo não só funciona como uma aplicação prática dos conceitos de programação, mas também como uma plataforma para desenvolver competências em manipulação de

dados, lógica de jogo e interação com o utilizador. Além disso, o projeto apresenta grande potencial para expansões futuras, como a integração de gráficos mais avançados, interfaces gráficas ou funcionalidades adicionais que enriquecem ainda mais a experiência do utilizador. Este trabalho reflete uma abordagem criativa e prática, ideal para quem procura aprender e aplicar conceitos fundamentais de programação enquanto desenvolve algo divertido e funcional.

OBJETIVOS

O principal objetivo deste projeto é simular o funcionamento de uma slot machine digital, reproduzindo de forma fiel a lógica de jogo, incluindo a geração de combinações aleatórias de símbolos, a aplicação de probabilidades para cada um deles e o cálculo automático de ganhos com base nas apostas realizadas. Pretende-se criar uma experiência de jogo interativa e envolvente, utilizando uma interface baseada em texto, com animações simples para os rolos da máquina e menus organizados e intuitivos que facilitem a interação do utilizador.

Outro objetivo importante é desenvolver funcionalidades de gestão financeira que permitam ao utilizador realizar depósitos e levantamentos de forma simples e segura, assegurando o controlo do saldo disponível e validando os valores inseridos para evitar erros ou ações inválidas, como apostas superiores ao saldo. Adicionalmente, o projeto incorpora um sistema de probabilidades e pagamentos, garantindo que os resultados sejam justos, mas desafiantes, e que os ganhos sejam calculados com base em combinações específicas e multiplicadores predefinidos.

O programa visa também oferecer feedback financeiro claro ao utilizador, apresentando informações como o saldo atual, os ganhos e perdas acumulados e o impacto de cada rodada no desempenho financeiro de forma simples e acessível, promovendo transparência e realismo. Além disso, o projeto tem como objetivo garantir a robustez do sistema, implementando mecanismos de validação de todas as entradas do utilizador, como depósitos, levantamentos e valores de aposta, evitando erros ou interrupções no fluxo do programa.

Outro aspeto fundamental é criar uma base escalável para futuras expansões, permitindo que o sistema possa ser facilmente adaptado para integrar funcionalidades

adicionais, como gráficos mais avançados, interfaces gráficas ou novos modos de jogo. O projeto também se propõe a explorar e aplicar conceitos fundamentais de programação, como geração aleatória de dados, manipulação de estruturas de dados, controlo de fluxo, validação de entradas e interação com o utilizador, servindo como uma ferramenta prática para consolidar conhecimentos.

Por fim, o objetivo é criar um programa que combine entretenimento e aprendizado, oferecendo uma aplicação prática e criativa que demonstre como ferramentas simples podem ser utilizadas para desenvolver produtos interativos e funcionais. Este projeto reflete, assim, uma abordagem inovadora e prática, integrando conhecimentos técnicos e promovendo uma experiência divertida e enriquecedora.

ANÁLISE E CONCEÇÃO DO SISTEMA

O sistema desenvolvido para a slot machine digital segue uma arquitetura simples e eficiente, composta por menus organizados de forma hierárquica, funcionalidades independentes e uma lógica robusta para a geração de combinações aleatórias. Este sistema foi implementado utilizando a linguagem Python, tirando partido de bibliotecas nativas para criar funcionalidades e simular um ambiente de jogo interativo. Segundo (Beazley, (2013)) e Jones (2013), "a utilização de bibliotecas como *random* e time permite criar efeitos visuais dinâmicos, melhorando a experiência do utilizador". A arquitetura geral do sistema está estruturada em torno de um menu principal, que funciona como ponto de entrada para todas as operações disponíveis. Este menu apresenta ao utilizador diversas opções, como efetuar depósitos, realizar levantamentos, iniciar o jogo, verificar o saldo atual, consultar os ganhos e perdas acumulados, bem como a opção de sair do programa. O fluxo de navegação é simples e intuitivo, permitindo ao utilizador selecionar a operação desejada através da introdução de um número correspondente à opção pretendida.

O sistema de depósitos foi concebido para permitir que o utilizador adicione fundos ao saldo disponível, sendo efetuadas validações para garantir que apenas valores positivos e válidos são aceites. Existe também um limite máximo para os depósitos, de modo a evitar erros ou valores irreais que possam comprometer a experiência de jogo. A funcionalidade de levantamento foi implementada para possibilitar a retirada de fundos do saldo acumulado, garantindo que o utilizador não pode levantar montantes superiores ao saldo disponível. Caso o saldo seja insuficiente para o levantamento solicitado, o sistema apresenta uma mensagem informativa e redireciona o utilizador ao menu principal, assegurando uma experiência de utilização clara e sem erros.

A funcionalidade principal do sistema, o jogo em si, foi desenvolvida com base numa lógica de geração de combinações aleatórias de símbolos, inspirada na estrutura tradicional das *slot machines*. Antes de iniciar o jogo, o utilizador tem a opção de definir o valor da aposta, que deve ser inferior ou igual ao saldo disponível, evitando situações de saldo negativo. A mecânica do jogo utiliza a biblioteca random para gerar combinações aleatórias de símbolos dispostos em rolos. Cada símbolo tem uma probabilidade associada, definida para equilibrar o desafio e a recompensa no jogo, assegurando que combinações mais valiosas, como cinco diamantes consecutivos, são mais raras.

Adicionalmente, foi implementada uma lógica de cálculo de ganhos com base nas combinações obtidas e no valor da aposta definida pelo utilizador. Os ganhos são calculados multiplicando o valor da aposta por fatores associados a cada símbolo, conforme uma tabela de pagamentos predefinida. Combinações de três ou mais símbolos consecutivos resultam em ganhos proporcionais ao valor e à raridade dos símbolos. Caso não sejam obtidos ganhos numa rodada, o saldo do utilizador é atualizado apenas com a subtração da aposta realizada.

Por fim, a estrutura modular do sistema permite uma navegação eficiente entre menus e funcionalidades, garantindo que o utilizador possa gerir facilmente as suas ações e consultar informações relevantes, como o saldo atual e o desempenho financeiro acumulado. A arquitetura do sistema foi projetada para oferecer uma experiência de jogo equilibrada, envolvente e funcional, permitindo ainda futuras expansões ou melhorias, como a integração de novos modos de jogo ou a introdução de uma interface gráfica para melhorar a apresentação visual e a interação com o utilizador.

IMPLEMENTAÇÃO

A implementação do sistema de *slot machine* foi realizada utilizando a linguagem *Python*, tirando partido de bibliotecas nativas como *random*, time, os e *shutil* para criar um programa interativo, dinâmico e funcional. O código foi organizado de forma modular, com funções específicas que permitem a separação lógica das diferentes funcionalidades, facilitando a leitura, a manutenção e a escalabilidade do projeto.

A biblioteca *random* desempenha um papel central no sistema, sendo utilizada para a geração de combinações aleatórias dos símbolos apresentados nos rolos da *slot machine*. A função *random.choice* é responsável por selecionar aleatoriamente os símbolos, garantindo uma distribuição justa baseada nas probabilidades atribuídas a cada um. A biblioteca *time* foi empregada para introduzir pequenas pausas no jogo, melhorando a experiência do utilizador com efeitos visuais de movimento nos rolos. Além disso, as bibliotecas os e *shutil* foram utilizadas para limpar e formatar o terminal, contribuindo para uma interface mais limpa e organizada.

O código foi estruturado em funções específicas, que desempenham papéis claros no funcionamento global do sistema. Por exemplo, a função ***menu_principal*** apresenta as opções iniciais ao utilizador e direciona para as funcionalidades desejadas. Outras funções, como deposito e levantar, tratam da gestão financeira, permitindo ao utilizador adicionar ou

retirar fundos do saldo disponível. Estas funções incluem validações para garantir que apenas valores válidos são aceites, evitando entradas inválidas, como números negativos ou caracteres alfabéticos. A validação é feita através de blocos ***try-except***, que lidam com exceções de entrada e apresentam mensagens informativas ao utilizador.

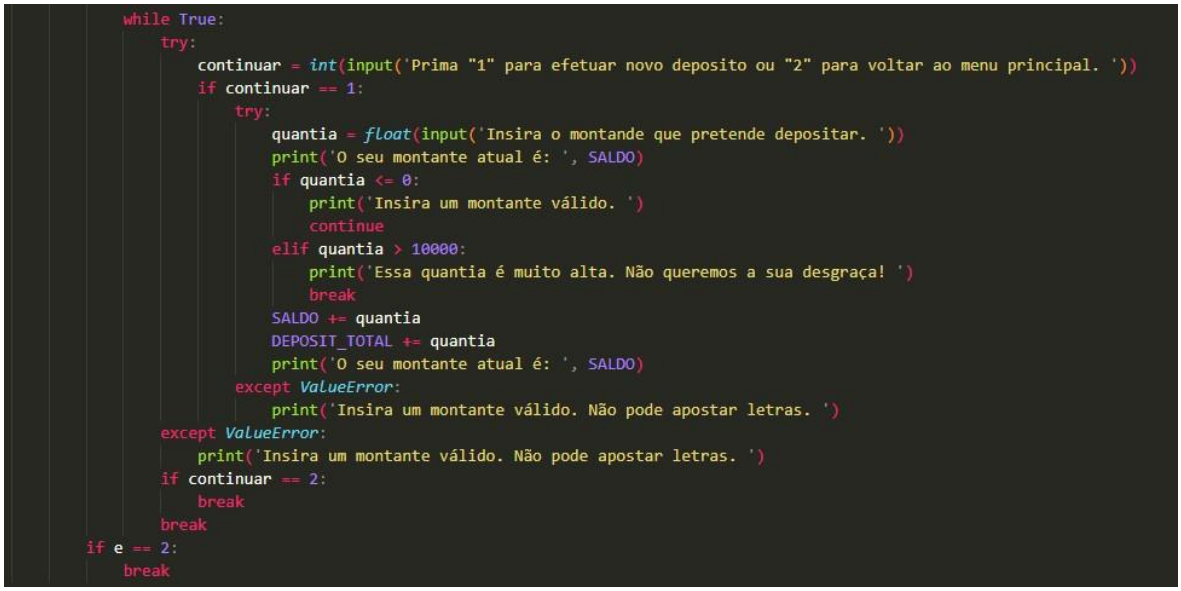
Segue-se um exemplo da função de depósito:

```
def deposito():
    global SALDO, DEPOSIT_TOTAL
    while True:
        try:
            e = int(input('Bem-vindo ao menu de deposito. Prima "1" para prosseguir para o deposito ou prima "2" para sair.'))
            if e > 2:
                print('Insira uma opção válida. "1 ou 2."')
                continue
            elif e < 1:
                print('Insira uma opção válida. "1 ou 2."')
                continue
        except ValueError:
            print('Insira uma opção válida. "1 ou 2."')
            continue
```

Figura 1 - A primeira parte da função solicita ao utilizador que escolha entre duas opções: "1" para prosseguir com o depósito ou "2" para sair.

```
        if e == 1:
            while True:
                try:
                    quantia = float(input('Insira o montande que pretende depositar. '))
                    if quantia <= 0:
                        print('Insira um valor positivo para o depósito.')
                        continue
                    elif quantia > 10000:
                        print('Essa quantia é muito alta. Não queremos a sua desgraça! ')
                        break
                    SALDO += quantia
                    DEPOSIT_TOTAL += quantia
                    print(f'O seu montante atual é: {SALDO:.2f} euros.')
                except ValueError:
                    print('Insira um montante válido. ')
                    continue
                break
```

Figura 2 - Este bloco permite ao utilizador introduzir o montante a depositar, validando se o valor é positivo e inferior a 10.000 euros. O saldo é atualizado apenas com valores válidos.



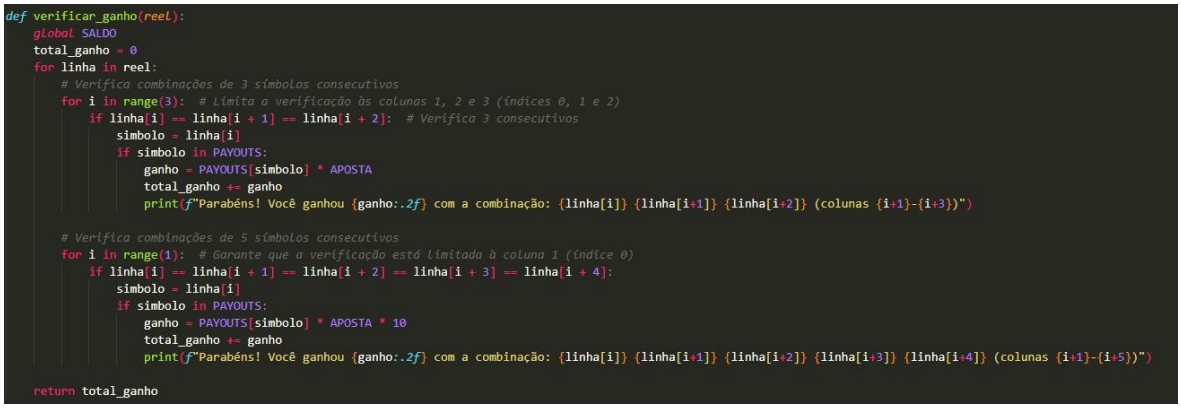
```
            while True:
                try:
                    continuar = int(input('Prima "1" para efetuar novo deposito ou "2" para voltar ao menu principal. '))
                    if continuar == 1:
                        try:
                            quantia = float(input('Insira o montande que pretende depositar. '))
                            print('O seu montante atual é: ', SALDO)
                            if quantia <= 0:
                                print('Insira um montante válido. ')
                                continue
                            elif quantia > 10000:
                                print('Essa quantia é muito alta. Não queremos a sua desgraça! ')
                                break
                            SALDO += quantia
                            DEPOSIT_TOTAL += quantia
                            print('O seu montante atual é: ', SALDO)
                        except ValueError:
                            print('Insira um montante válido. Não pode apostar letras. ')
                except ValueError:
                    print('Insira um montante válido. Não pode apostar letras. ')
                if continuar == 2:
                    break
                break
        if e == 2:
            break
```

Figura 3 - Este bloco verifica se o montante é válido, rejeitando entradas inválidas, como letras, e permite sair do menu.

Esta função valida as entradas, assegurando que apenas números positivos são aceites para o depósito e limita o valor máximo a 10.000 euros, promovendo uma interação segura e realista.

A função principal do jogo, ***game_logic***, utiliza as combinações geradas aleatoriamente para simular os rolos da ***slot machine***. Esta função utiliza um ciclo que exibe as combinações de símbolos enquanto calcula os ganhos do utilizador com base na aposta.

A lógica de cálculo dos ganhos é gerida pela função ***verificar_ganho***, que verifica combinações de símbolos consecutivos e aplica os multiplicadores definidos numa tabela de pagamentos.



```
def verificar_ganho(reel):
    global SALDO
    total_ganho = 0
    for linha in reel:
        # Verifica combinações de 3 símbolos consecutivos
        for i in range(3):  # Limita a verificação às colunas 1, 2 e 3 (índices 0, 1 e 2)
            if linha[i] == linha[i + 1] == linha[i + 2]:  # Verifica 3 consecutivos
                simbolo = linha[i]
                if simbolo in PAYOUTS:
                    ganho = PAYOUTS[simbolo] * APOSTA
                    total_ganho += ganho
                    print(f"Parabéns! Você ganhou {ganho:.2f} com a combinação: {linha[i]} {linha[i+1]} {linha[i+2]} (colunas {i+1}-{i+3})")

        # Verifica combinações de 5 símbolos consecutivos
        for i in range(1):  # Garante que a verificação está limitada à coluna 1 (índice 0)
            if linha[i] == linha[i + 1] == linha[i + 2] == linha[i + 3] == linha[i + 4]:
                simbolo = linha[i]
                if simbolo in PAYOUTS:
                    ganho = PAYOUTS[simbolo] * APOSTA * 10
                    total_ganho += ganho
                    print(f"Parabéns! Você ganhou {ganho:.2f} com a combinação: {linha[i]} {linha[i+1]} {linha[i+2]} {linha[i+3]} {linha[i+4]} (colunas {i+1}-{i+5})")

    return total_ganho
```

Figura 4 - Função que verifica combinações vencedoras de símbolos na slot machine e calcula os ganhos correspondentes.

1. Função da logica do jogo

A base da lógica de uma *slot machine* reside na geração de sequências de números aparentemente aleatórias. Assim a função ***game_logic*** é o coração da lógica do jogo da *slot machine*. Ela controla o fluxo do jogo, desde a verificação do saldo e aposta até a geração dos rolos, o cálculo dos ganhos e a decisão do jogador de continuar ou não.

Definição da Aposta:

Antes de iniciar o jogo, o utilizador deve definir o valor da aposta. Este valor é validado para garantir que não excede o saldo disponível e que é um valor positivo. Caso o utilizador tente apostar mais do que o saldo disponível, o sistema informa que não há fundos suficientes e redireciona o utilizador para o menu de depósito.

```
def game_logic(duration=2):
    global SALDO, APOSTA, TOTAL_BETS, TOTAL_WINS

    running = True
    while running:
        if SALDO <= 0:
            print("Você não tem saldo suficiente para continuar a jogar.")
            time.sleep(2)
            break
        if APOSTA <= 0:
            print('Por favor, defina uma aposta antes de continuar.')
            time.sleep(2)
            break
```

Figura 5 - função game_logic

Após a definição da aposta, o sistema gera combinações aleatórias de símbolos. Para isso, utiliza a biblioteca ***random***, que seleciona aleatoriamente símbolos da lista ***SYMBOLS***. A função cria uma matriz de 5 colunas e 5 linhas, onde cada célula contém um símbolo aleatório.

```
    num_columns = 5
    num_lines = 5
    end_time = time.time() + duration

    horizontal_border = "✨" + "═" * (num_columns * 4 - 1) + "✨"
    frame_width = len(horizontal_border)
    terminal_size = shutil.get_terminal_size((80, 20))
    terminal_width = terminal_size.columns
    terminal_height = terminal_size.lines
    padding_left = (terminal_width - frame_width) // 2
    padding_top = (terminal_height - (num_lines + 2)) // 2

    while time.time() < end_time:
        reel = [[random.choice(SYMBOLS) for _ in range(num_columns)] for _ in range(num_lines)]
        os.system("cls" if os.name == "nt" else "clear")
        print("\n" * padding_top)
        print(" " * padding_left + horizontal_border)
        for linha in reel:
            row = "  |  ".join(linha)
            print(" " * padding_left + row.center(frame_width))
        print(" " * padding_left + horizontal_border)
        time.sleep(0.1)
```

Figura 6 - continuação game logic

Após a rotação dos rolos, o sistema verifica se há combinações vencedoras. Para isso, a função ***verificar_ganho*** é chamada. Esta função analisa as combinações geradas e verifica se há sequências de símbolos iguais. Se forem encontradas combinações vencedoras, o ganho é calculado multiplicando o valor da aposta pelo multiplicador correspondente ao símbolo.

```
def verificar_ganho(reel):
    global SALDO
    total_ganho = 0
    for linha in reel:
        # Verifica combinações de 3 símbolos consecutivos
        for i in range(3):  # Limita a verificação às colunas 1, 2 e 3 (índices 0, 1 e 2)
            if linha[i] == linha[i + 1] == linha[i + 2]:  # Verifica 3 consecutivos
                simbolo = linha[i]
                if simbolo in PAYOUTS:
                    ganho = PAYOUTS[simbolo] * APOSTA
                    total_ganho += ganho
                    print(f"Parabéns! Você ganhou {ganho:.2f} com a combinação: {linha[i]} {linha[i+1]} {linha[i+2]} (colunas {i+1}-{i+3})")

        # Verifica combinações de 5 símbolos consecutivos
        for i in range(1):  # Garante que a verificação está limitada à coluna 1 (índice 0)
            if linha[i] == linha[i + 1] == linha[i + 2] == linha[i + 3] == linha[i + 4]:
                simbolo = linha[i]
                if simbolo in PAYOUTS:
                    ganho = PAYOUTS[simbolo] * APOSTA * 10
                    total_ganho += ganho
                    print(f"Parabéns! Você ganhou {ganho:.2f} com a combinação: {linha[i]} {linha[i+1]} {linha[i+2]} {linha[i+3]} {linha[i+4]} (colunas {i+1}-{i+5})")

    return total_ganho
```

Figura 7 – verificar ganho

2. Menu principal

A função ***menu_principal*** é a interface inicial do sistema da ***slot machine***, onde o utilizador pode escolher entre várias opções para interagir com o jogo. Esta função desempenha um papel crucial na navegação do utilizador, permitindo-lhe aceder a diferentes funcionalidades do sistema, como depositar fundos, levantar dinheiro, iniciar o jogo, verificar o saldo, consultar ganhos/perdas e sair do programa. A navegação é feita através da introdução de números correspondentes às opções disponíveis. A função começa por apresentar um menu com as opções disponíveis ao utilizador. Cada opção é numerada, e o utilizador deve introduzir o número correspondente à funcionalidade que deseja utilizar. O menu é apresentado de forma clara e organizada, facilitando a navegação.

```
def menu_principal():
    print('\n Bem-vindo ao menu principal da slot! ')
    print('\n')
    print('1 - Escolha o montante a depositar! ')
    print('2 - Escolha o montante que pretende levantar. ')
    print('3 - Iniciar o jogo. ')
    print('4 - Verificar saldo. ')
    print('5 - Total de ganhos/percas.')
    print('6 - Sair.')
```

Figura 8 - Menu principal

3. Função para levantamento

A função levantar é uma das funcionalidades centrais do sistema de *Slot Machine*, permitindo ao utilizador retirar fundos do saldo acumulado. Esta função foi desenvolvida com o objetivo de garantir que o utilizador possa realizar levantamentos de forma segura e eficiente, ao mesmo tempo que o sistema valida todas as entradas para evitar operações inválidas, como levantamentos que resultem em saldos negativos ou que excedam o saldo disponível. A função levantar é chamada quando o utilizador seleciona a opção 2 no menu principal. A função começa por apresentar um menu de levantamento, onde o utilizador pode escolher entre proceder ao levantamento ou sair e voltar ao menu principal.

```python
def levantar():
    global SALDO, WITHDRAW_TOTAL
    while True:
        try:
            l = int(input('Bem vindo ao menu de levantamento. '
                          'Prima "1" para proceder ao levantamento, ou "2" para sair: '))
        except ValueError:
            print('Opção inválida. Tente novamente.')
            continue
```

Figura 9 – Função levantar

Em seguida, o sistema valida a entrada do utilizador para garantir que apenas valores válidos são aceites. Se o utilizador introduzir um valor inválido (por exemplo, um caractere não numérico), o sistema apresenta uma mensagem de erro e solicita novamente a entrada.

```python
if l == 1:
    try:
        tirar = float(input('Quanto dinheiro pretendes levantar? '))
    except ValueError:
        print('Valor inválido.')
        continue
```

Figura 10 - Continuação função levantar

Após o utilizador introduzir o valor que deseja levantar, o sistema realiza várias validações:

1. **Valor Positivo:**
   O sistema verifica se o valor introduzido é positivo. Caso o utilizador insira um valor negativo ou zero, o sistema apresenta uma mensagem de erro e solicita novamente a entrada.
2. **Saldo Suficiente:**
   O sistema verifica se o valor solicitado não excede o saldo disponível. Caso o saldo seja insuficiente, o sistema informa o utilizador e redireciona-o para o menu principal.

```
        if tirar <= 0:
            print('Valor de levantamento deve ser positivo.')
            continue

        if tirar > SALDO:
            print('Não tem saldo suficiente para efetuar o levantamento!')
            print('A voltar para o menu principal...')
            time.sleep(2)
            break
```

Figura 11 - Verificação de valor função levantar

Se o valor de levantamento for válido, o sistema atualiza o saldo do utilizador, subtraindo o valor levantado, e regista o valor total levantado na variável *WITHDRAW_TOTAL*. O sistema também fornece feedback ao utilizador, informando-o sobre o valor levantado e o saldo restante.

```
    else:
        SALDO -= tirar
        WITHDRAW_TOTAL += tirar
        print(f'Levantou {tirar} euros. Restam {SALDO:.2f} euros no saldo.')
        time.sleep(2)
    break
```

Figura 12

Após o levantamento ser processado com sucesso, o sistema sai do menu de levantamento e volta ao menu principal. Caso o utilizador selecione a opção 2 (Sair) no menu de levantamento, o sistema também volta ao menu principal sem realizar qualquer operação.

```
        elif l == 2:
            break
        else:
            print('Opção inválida. Tente novamente.')
```

Figura 13

4. Função para ver saldo

A função *versaldo* permite ao utilizador consultar o saldo atual de forma rápida e simples. Quando chamada, exibe o saldo armazenado na variável global SALDO. Caso o saldo seja negativo (situação improvável devido às validações do sistema), ele é automaticamente ajustado para zero, garantindo a integridade do sistema.

Após mostrar o saldo, o utilizador é solicitado a pressionar "1" para voltar ao menu principal. O sistema valida a entrada, garantindo que apenas o número "1" seja aceite. Se o utilizador inserir um valor inválido (como letras ou outros números), uma mensagem de erro é exibida, e o sistema solicita novamente a entrada.

A função é simples, mas eficaz, fornecendo feedback claro ao utilizador e permitindo uma navegação intuitiva. Além disso, a estrutura modular facilita futuras expansões, como a inclusão de um histórico de transações ou notificações de saldo baixo.

```python
def versaldo():
    global SALDO
    while True:
        print('O teu saldo atual é: ', SALDO)
        if SALDO < 0:
            SALDO = 0
        try:
            a = int(input('Prima 1 para voltar ao menu principal. '))
            if a == 1:
                break
            elif a < 1:
                print('Insira uma opção válida. ')
                time.sleep(2)
                continue
            elif a > 1:
                print('Insira uma opção válida. ')
                time.sleep(2)
                continue
        except ValueError:
                print('Insira uma opção válida. ')
                print('Tenta de novo... ')
                time.sleep(2)
```

Figura 14 - Função ver saldo

5. Menu do jogo

O *game_menu* é a interface principal do jogo, onde o utilizador pode interagir com as funcionalidades específicas da *slot machine*. Esta função apresenta um menu com quatro opções: iniciar uma rodada, definir o valor da aposta, ver as regras do jogo ou sair. A função também valida se o utilizador tem saldo suficiente para realizar a aposta, garantindo que o jogo só é iniciado se houver fundos disponíveis. Caso o saldo seja insuficiente, o utilizador é informado e redirecionado para o menu principal.

Nesta parte, o menu é apresentado ao utilizador, com as opções numeradas de 1 a 4. O sistema solicita que o utilizador insira um número correspondente à opção desejada. A entrada do utilizador é validada para garantir que seja um número inteiro. Caso o utilizador insira um valor inválido (por exemplo, uma letra ou um número fora do intervalo), o sistema exibe uma mensagem de erro e retorna ao menu principal.

```
def game_menu():
    global SALDO, APOSTA
    print('💎 Bem-vindo à SLOT-BONANZA! 💎')
    print('1 - Iniciar o jogo.')
    print('2 - Definir a quantidade da aposta.')
    print('3 - Sair do jogo.')
    print('4 - Ver as regras do jogo.')
    print('')

    try:
        e_1 = int(input('Pressione um dos números do menu: '))
    except ValueError:
        print('Opção inválida.')
        return
```

Figura 15 - Menu de jogo

Se o utilizador escolher iniciar o jogo (opção 1), a função ***game_logic*** é chamada. Se escolher definir a aposta (opção 2), o sistema solicita o valor da aposta e valida se o utilizador tem saldo suficiente. Caso o saldo seja insuficiente, o utilizador é informado e redirecionado para o menu principal. Se o utilizador escolher ver as regras (opção 4), a função ***regras_do_jogo*** é chamada. A opção 3 permite sair do jogo e voltar ao menu principal.

```
while True:
    if e_1 == 1:
        game_logic()
    elif e_1 == 2:
        while True:
            try:
                APOSTA = float(input('Insira o valor que pretendes apostar: '))
            except ValueError:
                print('Valor inválido, tente novamente.')
                continue

            if APOSTA <= 0:
                print('A aposta deve ser maior que zero.')
                continue
            if APOSTA > SALDO:
                print('Não tem dinheiro suficiente. Faça um depósito para jogar.')
                print('A voltar para o menu principal...')
                time.sleep(2)
                return
            else:
                print(f'Aposta definida: {APOSTA:.2f}. Saldo: {SALDO:.2f}')
                # Volta ao menu do jogo para iniciar
                game_menu()
            break
    elif e_1 == 4:
        regras_do_jogo()
        time.sleep(2)
        return
    elif e_1 == 3:
        return
    break
```

Figura 16 - Logica do menu de jogo

O sistema garante que o utilizador só possa iniciar o jogo se tiver saldo suficiente para apostar. Caso o saldo seja insuficiente, o utilizador é informado e redirecionado para o menu principal. Além disso, o sistema fornece feedback claro sobre o valor da aposta e o saldo atual, permitindo que o utilizador tome decisões informadas.

```
            if APOSTA <= 0:
                print('A aposta deve ser maior que zero.')
                continue
            if APOSTA > SALDO:
                print('Não tem dinheiro suficiente. Faça um depósito para jogar.')
                print('A voltar para o menu principal...')
                time.sleep(2)
                return
            else:
                print(f'Aposta definida: {APOSTA:.2f}. Saldo: {SALDO:.2f}')
                # Volta ao menu do jogo para iniciar
                game_menu()
            break
    elif e_1 == 4:
        regras_do_jogo()
        time.sleep(2)
        return
    elif e_1 == 3:
        return
    break
```

Figura 17

6. Relatório de ganhos/perdas

A função ***ganhos_percas*** é responsável por apresentar um resumo detalhado das operações financeiras do utilizador ao longo do jogo. Este relatório é essencial para que o utilizador possa acompanhar o seu desempenho financeiro, incluindo os depósitos, levantamentos, apostas realizadas, ganhos obtidos e o saldo atual. Além disso, a função calcula o lucro ou prejuízo do utilizador, tanto no contexto do jogo como no geral, proporcionando uma visão clara e organizada das suas finanças.

A função utiliza variáveis globais para armazenar os valores totais de depósitos, levantamentos, apostas e ganhos, permitindo que o relatório seja atualizado dinamicamente à medida que o utilizador interage com o jogo. Vamos detalhar o código em três partes principais:

Nesta parte, a função calcula dois valores principais:

1. **Lucro/Prejuízo do Jogo (net_game):**
   Este valor é obtido subtraindo o total apostado (***TOTAL_BETS***) do total ganho em prémios (***TOTAL_WINS***). Se o resultado for positivo, o utilizador teve lucro no jogo; se for negativo, teve prejuízo.

2. Lucro/Prejuízo Total (net_overall):
   Este valor reflete a situação financeira global do utilizador, considerando o saldo atual, os levantamentos realizados e os depósitos efetuados. É calculado somando o saldo atual (SALDO) e o total levantado (WITHDRAW_TOTAL), e subtraindo o total depositado (DEPOSIT_TOTAL).

```
def ganhos_percas():

    global SALDO, DEPOSIT_TOTAL, WITHDRAW_TOTAL, TOTAL_BETS, TOTAL_WINS

    net_game = TOTAL_WINS - TOTAL_BETS
    net_overall = SALDO + WITHDRAW_TOTAL - DEPOSIT_TOTAL
```

Figura 18 - Função ganhos e perdas

Nesta parte, a função ganhos_percas exibe um relatório detalhado e organizado das operações financeiras do utilizador. O relatório é apresentado de forma clara e estruturada, permitindo que o utilizador compreenda facilmente o seu desempenho financeiro ao longo do jogo. O relatório inclui as seguintes informações:

1. **Total Depositado:**
   Este valor representa a soma de todos os depósitos realizados pelo utilizador ao longo do jogo. É armazenado na variável ***DEPOSIT_TOTAL*** e reflete o montante total que o utilizador colocou no sistema para jogar.
2. **Total Levantado:**
   Este valor corresponde ao total de fundos que o utilizador retirou do sistema. É armazenado na variável ***WITHDRAW_TOTAL*** e mostra quanto dinheiro o utilizador já levantou do seu saldo acumulado.
3. **Total Apostado:**
   Este valor indica o montante total que o utilizador apostou em todas as rodadas do jogo. É armazenado na variável ***TOTAL_BETS*** e reflete o risco assumido pelo utilizador ao jogar.
4. **Total Ganho em Prémios:**
   Este valor representa a soma de todos os prémios que o utilizador ganhou ao longo

do jogo. É armazenado na variável ***TOTAL_WINS*** e mostra o retorno financeiro obtido com as apostas realizadas.

5. **Lucro/Prejuízo do Jogo:**
   Este valor é calculado subtraindo o total apostado (***TOTAL_BETS***) do total ganho em prémios (***TOTAL_WINS***). Se o resultado for positivo, significa que o utilizador teve lucro no jogo; se for negativo, indica que teve prejuízo. Este cálculo é armazenado na variável ***net_game***.
6. **Saldo Atual:**
   Este valor reflete o saldo disponível do utilizador no momento da consulta do relatório. É armazenado na variável ***SALDO*** e mostra quanto dinheiro o utilizador tem disponível para continuar a jogar ou para levantar.
7. **Lucro/Prejuízo Total:**
   Este valor representa a situação financeira global do utilizador, considerando o saldo atual, os levantamentos realizados e os depósitos efetuados. É calculado somando o saldo atual (***SALDO***) e o total levantado (***WITHDRAW_TOTAL***), e subtraindo o total depositado (***DEPOSIT_TOTAL***). Este cálculo é armazenado na variável net_overall.

```
print('\n===== RELATÓRIO DE GANHOS/PERDAS =====')
print(f'Valor total depositado:      {DEPOSIT_TOTAL:.2f}')
print(f'Valor total levantado:       {WITHDRAW_TOTAL:.2f}')
print(f'Total apostado (rodadas):    {TOTAL_BETS:.2f}')
print(f'Total ganho em prêmios:      {TOTAL_WINS:.2f}')
print(f'-------------------------------------')
print(f'Lucro/Prejuízo do jogo:      {net_game:.2f}')
print(f'Saldo atual:                 {SALDO:.2f}')
print(f'Lucro/Prejuízo TOTAL:        {net_overall:.2f}\n')
input('Pressione Enter para voltar ao menu principal... ')
```

Figura 19

7. Função regras de jogo

A função ***regras_do_jogo*** serve para explicar ao utilizador, de forma clara e acessível, como funciona o jogo de ***Slot Machine***. Através de um texto bem estruturado, esta função descreve as regras básicas do jogo, as combinações de símbolos que dão direito a prémios e os multiplicadores associados a cada símbolo. O objetivo é que o utilizador entenda exatamente como o jogo funciona, para que possa fazer apostas de forma consciente e informada.

```
def regras_do_jogo():
    texto_regras = """
    ========================= REGRAS DO JOGO =========================
    1. Define o valor da tua aposta antes de iniciares o jogo.
    2. A slot vai girar 5 colunas, cada uma exibindo um símbolo aleatório.
    3. Se conseguires:
       - 3 símbolos iguais consecutivos na mesma linha, ganhas um multiplicador
         relativo a esse símbolo (por exemplo, 3 cerejas pagam 3x a tua aposta).
       - 5 símbolos iguais consecutivos na mesma linha, ganhas um prémio ainda maior
         (por exemplo, 5 símbolos iguais aplicam um multiplicador especial).
    4. Cada giro custa o valor da aposta definida. O teu saldo é atualizado a cada rodada.
    5. Se ficares sem saldo, terás de depositar novamente para continuares a jogar.
    6. Podes sair a qualquer momento, e o teu saldo ficará registado (desde que não o levantes).
    7. Diverte-te e joga com responsabilidade!
    ==================================================================
    """
    print(texto_regras)
```

Figura 20 - função regras de jogo

8. Função main

A função ***main*** é o ponto de entrada do programa, responsável por controlar o fluxo principal de execução. Ela exibe o menu principal repetidamente, permitindo que o utilizador escolha entre várias opções: depositar dinheiro, levantar fundos, iniciar o jogo de ***slots***, verificar o saldo, consultar ganhos e perdas, ou sair do programa.

O programa valida a entrada do utilizador, garantindo que apenas números de 1 a 6 sejam aceitos. Se o utilizador inserir uma opção inválida, uma mensagem de erro é exibida, e o programa retorna ao menu principal após uma breve pausa. Dependendo da escolha, a função chama outras funções específicas, como ***deposito(), levantar(), game_menu(), versaldo(), ou ganhos_percas()***.

O ***loop infinito (while True)*** mantém o programa em execução até que o utilizador selecione a opção ***6 (quit()),*** que encerra o programa. Essa estrutura garante uma experiência

interativa e contínua, permitindo que o utilizador realize múltiplas operações sem reiniciar o programa.

```
def main():

    while True:
        menu_principal()
        try:
            menu = int(input('Escolhe uma opção: '))
            if menu <0:
                print('Insira uma opção válida. "De 1 a 6. "')
                print('Voltando ao menu principal...')
                time.sleep(2)
            elif menu > 6:
                print('Insira uma opção válida. "De 1 a 6. "')
                print('Voltando ao menu principal...')
                time.sleep(2)
        except ValueError:
            print('Insira uma opção válida. "De 1 a 6. "')
            print('Voltando ao menu principal...')
            time.sleep(2)
            continue
        if menu == 1:
            deposito()
        elif menu == 2:
            levantar()
        elif menu == 3:
            game_menu()
        elif menu == 4:
            versaldo()
        elif menu == 5:
            ganhos_percas()
        elif menu == 6:
            quit()
```

Figura 21 - Função main

CONCLUSÃO

O desenvolvimento deste projeto de ***slot machine*** em ***Python*** permitiu explorar e aplicar conceitos fundamentais de programação, como a manipulação de estruturas de dados, geração de números aleatórios, validação de entradas e interação com o utilizador. O sistema foi concebido com uma arquitetura modular, que separa claramente as funcionalidades de gestão financeira (depósitos, levantamentos e consulta de saldo) da lógica do jogo, garantindo uma estrutura organizada e de fácil manutenção.

A utilização de bibliotecas nativas do ***Python***, como ***random***, ***time***, ***os*** e ***shutil***, permitiu criar uma experiência de jogo interativa e visualmente apelativa, com animações simples que simulam o funcionamento de uma máquina de ***slots*** real. A lógica de jogo, baseada em probabilidades e combinações de símbolos, foi desenvolvida para garantir um equilíbrio entre risco e recompensa, oferecendo ao utilizador uma experiência desafiadora e divertida.

O sistema de gestão financeira foi implementado com validações rigorosas, assegurando que o utilizador não possa realizar operações inválidas, como depósitos ou levantamentos que resultem em saldos negativos. Além disso, a funcionalidade de consulta de ganhos e perdas permite ao utilizador acompanhar o seu desempenho financeiro de forma clara e transparente.

Embora o projeto tenha atingido os objetivos propostos, existem várias oportunidades para melhorias e expansões futuras. A implementação de uma interface gráfica (GUI), utilizando bibliotecas como ***tkinter*** ou ***PyQt***, tornaria o sistema mais intuitivo e visualmente atrativo. A inclusão de um sistema de persistência de dados, como um banco de dados, permitiria registar o progresso do utilizador, estatísticas e histórico de apostas, aumentando a longevidade e o envolvimento com o jogo. Adicionalmente, a introdução de novas mecânicas, como múltiplas linhas de pagamento, bónus ou jackpots, poderia enriquecer a experiência do utilizador e adicionar complexidade ao jogo.

Em suma, este projeto demonstra a aplicação prática de conhecimentos teóricos em programação, servindo como uma ferramenta didática e lúdica para consolidar competências técnicas. A utilização de ***Python*** revelou-se uma escolha acertada, dada a sua simplicidade, versatilidade e vastos recursos de bibliotecas. O sistema desenvolvido oferece uma base sólida para futuras expansões, podendo servir como ponto de partida para projetos mais complexos e ambiciosos.

1 BIBLIOGRAFIA

Beazley, D. &. ((2013)). *Python Cookbook: Recipes for Mastering Python 3 (3ª ed.).* O'Reilly Media.

Lutz. (2013). *Learning Python (5ª ed.).* O'Reilly Media.

Sweigart. (2015). *Invent Your Own Computer Games with Python (4ª ed.).* No Starch Press.