Universidade de Brasília
Faculdade do Gama

Metodologia ativa de ensino de programação baseada em reconhecimento ótico de caracteres

Emanuel Holanda Barroso

PROJETO FINAL DE CURSO

ENGENHARIA DE *SOFTWARE*

Brasília

2023

Universidade de Brasília
Faculdade do Gama

Metodologia ativa de ensino de programação baseada em reconhecimento ótico de caracteres

Emanuel Holanda Barroso

Projeto Final de Curso submetido como requisito parcial para obtenção do grau de Engenheiro de *Software*

Orientador: Prof. Dr. Diogo Caetano Garcia
Coorientador: Yuri Monteiro

Brasília
2023

Holanda Barroso, Emanuel.
H000m Metodologia ativa de ensino de programação baseada em reconhecimento ótico de caracteres / Emanuel Holanda Barroso; orientador Diogo Caetano Garcia; coorientador Yuri Monteiro. -- Brasília, 2023.
48 p.

Projeto Final de Curso (Engenharia de *Software*) -- Universidade de Brasília, 2023.

1. Educação. 2. Ensino de programação. 3. Processamento de imagens. 4. Reconhecimento ótico de caracteres. I. Caetano Garcia, Diogo, orient. II. Monteiro, Yuri, coorient. III. Título

Universidade de Brasília
Faculdade do Gama

Metodologia ativa de ensino de programação baseada em reconhecimento ótico de caracteres

Emanuel Holanda Barroso

Projeto Final de Curso submetido como requisito parcial para obtenção do grau de Engenheiro de *Software*

Trabalho aprovado. Brasília, 15 de junho de 2023:

Prof. Dr. Diogo Caetano Garcia, UnB/FGA
Orientador

Prof. Dr. Cristiano Jacques Miosso, UnB/FGA
Examinador interno

Prof. Dr. Daniel Mauricio Munoz Arboleda, UnB/FGA
Examinador interno

Brasília
2023

Este trabalho é dedicado às crianças adultas que,
quando pequenas, sonharam em se tornar cientistas.

Agradecimentos

Agradeço a todas as pessoas que fazendo críticas construtivas, elogiando e principalmente me aceitando como sou, possibilitaram o meu ingresso, minha permânencia e minha conclusão do curso. Tenho muito orgulho em tê-los ao meu lado nesta caminhada.

“Educação não transforma o mundo.
Educação muda as pessoas.
Pessoas transformam o mundo”
Paulo Freire

Resumo

A educação é um processo contínuo, estando presente entre os povos desde os primórdios. Ela é dinâmica, evoluindo de acordo com o desenvolvimento da sociedade. A tecnologia de informação, fruto da terceira revolução industrial, tem facilitado a implementação do processo educativo, como por exemplo na aplicação da educação à distância. Neste processo, surgiram várias metodologias e programas de computador para auxiliarem no aprendizado, inclusive no ensino de linguagens de programação.

Dentro deste contexto, é comum alunos utilizarem vídeo-aulas para estudarem tópicos de programação, especialmente com o crescimento do ensino à distância após a pandemia de 2020. Os alunos geralmente não têm acesso aos códigos apresentados visualmente nestes vídeos, necessitando copiar os códigos manualmente, ou então utilizar diversas ferramentas separadas. Neste trabalho, propõe-se um programa para automatizar este processo, reconhecendo códigos de programação em video-aulas utilizando um algoritmo de reconhecimento de caracteres (Tesseract, baseado em redes neurais LSTM), de acordo com a seleção do usuário. O texto é automaticamente carregado na área de transferência para uso do estudante. Além disso, o programa oferece uma opção de aprendizagem ativa, acrescentando erros uniformemente aleatórios, para que o aluno tenha de corrigi-los antes de utilizar o código.

Foram realizados testes com diversos códigos em linguagem *Python* disponíveis gratuitamente *online*, obtendo-se uma taxa de erros de caracteres de cerca de 15% (sem levar em conta espaços em branco entre o texto), mostrando grande potencial em auxiliar na aprendizagem ativa de linguagens de programação.

Palavras-chave: Educação. Ensino de programação. Processamento de imagens. Reconhecimento ótico de caracteres.

Abstract

Education is a continuous process, being present among peoples since the beginning. It is dynamic, evolving according to the development of society. The industrial revolution was one of the most important milestones for the development of education, when machines and technologies were developed and incorporated into the educational process. Information technology, the result of the third industrial revolution, has facilitated the implementation of the educational process, such as the application of distance education. In this process, several methodologies and computer programs emerged to assist in learning, including the teaching of programming languages.

Within this context, it is common for students to use video lessons to study programming topics, especially with the growth of distance learning after the 2020 pandemic. Students generally do not have access to the codes visually presented in these videos, needing to copy the codes manually, or to use several separate tools. In this work, we propose a program to automate this process, by recognizing programming codes in video lessons using a character recognition algorithm (Tesseract, based on LSTM neural networks), according to the user's selection. The text is automatically loaded into the clipboard for use by the student. In addition, the program offers an active learning option, adding uniformly random errors so that the student has to correct them before using the code.

Tests were carried out with several Python codes available for free online, obtaining a character error rate of about 15% (without taking into account white spaces between the text), showing great potential in assisting in the active learning of languages of programming.

Keywords: Education. Programming teaching. Image processing. Optical character recognition.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

CCD	*Charge-Coupled Device* (dispositivo de carga acoplada) 19
CER	*Character error rate* (taxa de erros de caracteres) 28
JPEG	*Joint Photographic Expert Group* (Grupo Conjunto de Especialistas em Fotografia) 20
OCR	*Optical Character Recognition* (reconhecimento ótico de caracteres) 21
PNG	*Portable Network Graphics* (gráficos portáteis de rede) 20
RGB	*Red Green Blue* (vermelho, verde, azul) 20
WER	*Word error rate* (taxa de erros de palavra) 28

Sumário

1 Introdução

Historicamente, tem existido uma intensa ligação entre educação e a sociedade. Segundo Godetti (1999), o pensamento pedagógico surgiu a partir de uma reflexão sobre as práticas e a necessidade de sistematizá-las, inicialmente ensinadas pelos mais velhos e de forma intuitiva e cultural. Segundo Lewian, a aprendizagem surgiu a partir de um contínuo intercâmbio entre teoria e prática, onde a primeira orienta a segunda, que por sua vez passa por um processo de reflexão, ocorrendo uma nova ação. Para Piaget, "o ser humano se move de um mundo concreto para um abstrato, sendo que a prática oferece as informações para a construção deste movimento, existindo uma dualidade entre as ideias vindas do mundo e aquelas adquiridas a partir da prática" (KOLB, 1984).

O aprendizado na Antiguidade estava diretamente relacionado às necessidades momentâneas e acontecia na rotina das famílias, onde os mais velhos levavam as crianças para o plantio e as crianças iam incorporando as maneiras de realizarem essas atividades. Na Grécia e Roma Antigas, conhecimentos como orátoria, retórica, filosofia, artes e literatura eram transmitidos em consonância com suas sociedades, com o objetivo de preparar o estudante para a vida política da época. Via-se assim o surgimento dos grandes filósofos como Sócrates, Platão e Aristóteles, dentre outros. Na Idade Média a educação sofreu grande influência da Igreja Católica, limitando-se a uma minoria. Na época moderna, com o surgimento do Iluminismo com os lemas liberdade, igualdade e fraternidade, a escola foi ampliada. Com o advento da Revolução Industrial, surgiram diversas máquinas, e com a necessidade de mão-de-obra para manuseá-las, tornou-se imprescíndivel a capacitação da mão-de-obra, facilitada pela adoção de escolas.

A necessidade conduz o indivíduo a agir. Sempre que há um desequilíbrio no organismo, por exemplo a fome, o indivíduo é impulsionado a buscar meios para equilibrá-lo, e a partir daí aprende a buscá-los sempre que houver necessidade. O processo de aprendizagem é assim impulsionado pelas próprias necessidades do indivíduo.

Os avanços em tecnologia pouco a pouco foram sendo utilizados pelas escolas. Inicialmente, máquinas enormes eram utilizadas para preparar material para as crianças, e aos poucos foram surgindo aparelhos menores e mais eficientes para ajudar no trabalho do professor. Vê-se assim um progresso contínuo na tecnologia e consequentemente no processo educativo. Nos países menos desenvolvidos economicamente, a educação nem sempre pode acompanhar o desenvolvimento tecnológico, pois é necessário investimento massivo tanto em instrumentos como em capacitação do professor para adequar o ensino às mudanças.

Na escola presencial a presença física do aluno é imprescindível e controlada, e a organização do espaço escolar, dos espaços de convivência e dos objetos dita muito do que

será realizado naquele espaço (KENSKI, 2003).

Durante muito anos o aprendizado foi feito em escolas presenciais, contudo com avanço da tecnologia e mais especificamente da tecnologia da informação surgiram as escolas virtuais. No ano de 2020, devido à pandemia de COVID-19, surgiu uma necessidade de distanciamento social, e as aulas *online* foram a principal forma de manter o ensino naquele ano, passando por uma forte expansão.

A indústria de *software*, uma das que compõem a indústria da tecnologia de informação, baseia-se na programação e representa atualmente mais de 55% da mão-de-obra de serviços. Têm sido feitos muitos estudos na área de metodologias inovadoras de ensino de programação.

Alunos de programação frequentemente desejam copiar o código apresentado na tela em vídeos *online* para facilitar o aprendizado e a prática. No entanto, não é tecnicamente possível copiar o código diretamente da tela, pois os códigos exibidos em vídeos são geralmente imagens estáticas, o que torna o processo de seleção e cópia do texto impossível por meio de métodos tradicionais. Para contornar essa limitação, é necessário recorrer a diversas ferramentas externas: programas de captura de tela, algoritmos de reconhecimento de caracteres em imagens (capazes de extrair o texto presente na imagem do código e transformá-lo em um formato editável) e programas de cópia do texto resultante para a área de transferência do sistema operacional. Essa abordagem permite aos alunos obterem o código em formato de texto, tornando-o mais acessível para manipulação e estudo.

Atualmente, existe uma variedade de ferramentas que realizam o reconhecimento ótico de caracteres, tanto em bibliotecas e *softwares* pagos como em código aberto (MARKETSPLASH TEAM, 2022). Alguns exemplos do primeiro caso incluem o ABBY FineReader, o Transym OCR, o OCR Space, a Google Cloud Vision, a Azure Cognitive Services (Microsoft) e o Textract (Amazon Web Services), que permitem não somente a extração de textos como também outros serviços. Dentre os serviços gratuitos, pode-se citar o Tesseract OCR, que é uma ferramenta gratuita, com código aberto e eficiente na extração de textos.

O presente trabalho tem como objetivo oferecer uma junção destas ferramentas de cópia de textos em videos para estudos de programação, facilitando o trabalho do aluno. O programa proposto também oferece a opção de gerar programas com alguns erros, para que posteriomente sejam corrigidos pelos estudantes. Esta proposta vai ao encontro de pesquisas que indicam que metodologias centradas na resolução de problemas e que incentivem o estudante a pensar oferecem aprendizado mais efetivo.

1.1 Objetivo geral

1. Desenolver uma ferramenta que auxilie no processo de aprendizagem de programação ao tornar mais fácil a reutilização de códigos-fontes presentes em videos *online*.

1.2 Objetivos específicos

1. Desenvolver uma aplicação que permita ao usuário selecionar uma porção da tela do computador, que será então processada por um algoritmo de identificação de texto em imagem, salvando o texto na área de transferência do sistema operacional para posterior uso em algum editor de texto.

2. Identificar possiveis erros que venham a ocorrer no processo de transformação de imagem em texto na ferramenta escolhida.

3. Permitir a inserção de erros ao texto detectado, para que a aplicação possa ser usada num contexto de educação, com o aluno corrigindo-os posteriormente.

1.3 Estrutura do trabalho

O Capítulo 2 apresenta a revisão bibliográfica do tema em estudo, incluindo as definições de educação e de aprendizagem, seu histórico e sua importância, detalhes do ensino de programação, paradigmas de programação e o processo produtivo da programação. Também são detalhados conceitos básicos de inteligência artificial, visão computacional, processamento digital de imagens e reconhecimento digital de imagens. O Capítulo 3 apresenta o sistema proposto com base em interface gráfica desenvolvida em linguagem Java e reconhecimento ótico de caracteres com o Tesseract. O Capítulo 4 apresenta resultados obtidos para uma série de códigos em linguagem Python disponíveis *online*, e o Capítulo 5 oferece as considerações finais do presente trabalho.

2 Revisão Bibliográfica

A tecnologia, segundo Bodgnov (BOGDANOV, 1980), é o conjunto de técnicas adquiridas em interação com o homem ou com o meio. A primeira tecnologia surgiu ainda na pré-historia, com a construção das primeiras ferramentas para caça, e posteriormente veio a fala. A técnica é o grupo de práticas adotadas para executar eficazmente uma atividade. (ABBAGNANO, 2012)

2.1 Educação e aprendizagem

Para Kant, "a educação é uma forma de transmitir a cultura de um povo para as gerações futuras. Neste sentido ela evita que ocorram transformações" (MAGALHÃES, 2010). Já para Dewey (DEWEY, 2007), a educação é um processo de reelaboração da experiência vivida, e que traça alicerces para o futuro. Sendo assim, ela é uma força transformadora da sociedade. A educação vive então um embate conceitual entre ser um agente transformador ou mantenedor da cultura de sociedade.

Inicialmente, as atividades laboriais eram mais simples, de forma que as etapas de aprendizagem e de atuação profissional estavam entrelaçadas. Com a crescente sofisticação das profissões, ocorreu a separação do trabalho da etapa formativa, estabelecendo-se escolas e locais de trabalho.

A partir da massificação dos ensinos fundamental, médio e superior, o trabalho de individuos passou a ser atribuído pelos centros educacionais e não mais pela família.O ensino superior tem como objetivo preparar pessoas para as atividades gerenciais e deve evitar reprovações e evasões através das mais variadas estrátégias, tais como curriculos flexiveis, além de novos métodos e adaptações no ensino (EDUCATION FOR THE TWENTY-FIRST CENTURY; DELORS; UNESCO, 1996).

Com o advento das tecnologias da informação e da comunicação, o ensino pode ser realizado de forma presencial ou virtual (KENSKI, 2003). Na escola presencial, os apetrechos, a proximidade e os intervalos entre as aulas facilitam a criação de laços sociais nos envolvidos no processo de ensino e aprendizagem. Assim, a escola se torna uma verdadeira organização social viva e pulsante (KENSKI, 2003). Na escola virtual, aquela feita a partir de aulas oferecidas *online*, a organização do *site*, dos quase objetos e dos seus endereços na rede ditam muito do que será feito ali. Ali, tem-se certa dificuldade no estabelecimento de vínculos sociais e emocionais entre os envolvidos no processo de ensino e aprendizagem, já que a comunicação é feita por meio de imagens, palavras, senhas e símbolos, e pode-se verificar dificuldade na reprodução das vozes e imagens, devido a problemas que possam ocorrer na

transmissão via internet.

Kenski salienta que é importante criar perfis que apresentem uma breve descrição das caracteristicas pessoais dos indivíduos envolvidos para gerar maior aproximação entre as pessoas que as frequentam (KENSKI, 2003). O autor sugere que a educação híbrida pode alcançar melhor qualidade no ensino, oferecendo momentos presenciais mas também a participação dos estudantes em grupos de debate.

O ano de 2020 foi marcado pela pandemia de COVID-19, o que levou ao fechamento das escolas e universidades e ao incremento do ensino à distância e das atividades *online*, levando as pessoas a repensarem antigas práticas de trabalho e de vida. Tornaram-se necessárias novas metodologias de ensino e aprendizagem em setores críticos para a sociedade, como o de tecnologia da informação.

2.2 Ensino de programação

O ensino de programação é tão antigo quanto os próprios computadores. Antes de 1950, o foco da programação era contornar as barreiras do *hardware* para resolver os problemas, pois os computadores eram lentos e tinham memória muito limitada. Nesta época a programação era vista como uma arte (SILVA et al., 2015).

As primeiras linguagens de programação desenvolvidas foram as linguagens de máquina, ou *assembly*, que são dependentes do processador onde serão utilizadas (TUCKER; NOONAN, 2009). A popularização do ensino de programação contudo só começou com o surgimento de linguagens mais próximas da linguagem humana, como as linguagens de programação imperativas e as orientadas a objetos (SILVA et al., 2015).

Programação imperativa é a construção de programas que tenham como referência a estrutura de Von Neumann. Os programas são executados em ordem sequencial, e tanto os programas como os dados ficam armazenados na memória do computador. Para ser imperativa, a linguagem tem que ter como centro o processo de atribuição de valores, com comandos condicionais, expressões, laços e abstração procedural. Para ser completa do ponto de vista de Turing, a linguagem deve apresentar ainda controle de erros, exceções e estruturas de dados. Um exemplo de linguagem imperativa é a linguagem C (TUCKER; NOONAN, 2009).

A programação orientada a objetos, ao contrário da programação imperativa, não tem centro nos procedimentos a serem realizados, e sim nos objetos que representam coisas do mundo real, com comportamentos e propriedades, além de poderem constituir relações de herança, subclasse, e agregação. Além de objetos, a orientação à objetos pode fazer uso também de interfaces ou classes abstratas, que estabelecem uma assinatura para um método a ser utilizada por diversas classes (TUCKER; NOONAN, 2009).

A maneira tradicional de começar o aprendizado em programação consiste em separar os algoritmos e dividi-los em problemas menores que sejam suportados pelas linguagens de programação, separados das estruturas de dados. Esta metodologia teve origem nos trabalhos de Dahl, Dijkstra, Hoare e Wirth (LÁSZLÓ, 2003). Os passos seguintes sugeridos por Furlan consistem na análise do problema, na escrita numa forma que todas as pessoas entendam e posteriormente, na escrita numa linguagem que será utilizada no computador (SOUZA et al., 2020). Por fim, procura-se resolver eventuais erros do programa e otimiza-lo visando diminuir o tempo de execução, ou atender outras demandas que possam vir a surgir.

Segundo Levitin (LEVITIN, 2000), a construção de algoritmos se inicia no conhecimento do problema e na formulação matemática para o mesmo, vindo a seguir os testes e os ajustes necessários e posteriormente a transcrição do mesmo para uma linguagem de programação. Neste sentido podemos visualizar dois pontos de dificuldade: o entendimento do problema por parte do estudante e a sua posterior transcrição em uma linguagem de programação. Segundo este autor, ao se traduzir as construções mátematicas para uma linguagem de programação deve-se considerar especificidades do *hardware* que será utilizado, assim como após está transcrição deve-se fazer ajustes buscando-se o máximo de eficiência.(LEVITIN, 2000)

O ensino de programação no Brasil está ainda, e em geral, restrito às pessoas que desejam formação especifica, em especial graduados da área de *software* e correlatas. Outros países, como a Estônia e os Estados unidos, têm se preocupado em capacitar alunos para esta área desde as séries iniciais (LEVITIN, 2000).

Pode-se perceber que as pesquisas na área de ensino têm enfocado bastante o uso de jogos para o aprendizado juntamente com novas metodologias de ensino. A abordagem construcionista, por exemplo, utiliza computadores para que o estudante alcance seu próprio aprendizado (LEVITIN, 2000). Procura-se também desenvolver habilidades e competências tais como a resolução de problemas, o estabelecimento de conclusões, o planejamento e a tomada de decisões, além de habilidades sociais como lidar com regras e a cooperação entre pessoas, e competências como a criatividade, a estruturação do pensamento, a responsabilidade, a curiosidade e o trabalho em equipe (LEVITIN, 2000).

É comum os alunos apresentarem dificuldade na compreensão dos problemas e no raciocínio logico, fatores para os quais o uso de jogos e linguagens mais visuais contribui para sua amenização (LEVITIN, 2000). O jogo é uma forma de preparar para uma profissão, que traz divertimento e pode servir para conquista de parceiros e construção de relacionamenmtos sociais.

2.3 Tecnologia da informação

Segundo o *Model Curriculum for K-12 Computer Science*, a grande maioria das profissões do século XXI requer conhecimentos na área de ciência da computação (TUCKER, 2003). Faz parte da tecnologia da informação e comunicação (TIC) o conjunto de *hardware*, *software* e dispositivos de comunicação (JÚNIOR, 2017). Segundo Ritto, as TICs são itens essenciais para empresas, assim como água e energia. Sendo assim, sua ausência pode comprometer o funcionamento desta (JÚNIOR, 2017).

2.3.1 Processamento digital de imagens

A captura e o registro direto de uma imagem foi realizado inicialmente em 1826 por Joseph Nicepore, utilizando betume, que demora 8 horas para exibir o resultado final (SÁ, 2021).

Na fotografia digital, sensores CCD (dispositivos de carga acoplada, ou *charged-coupled devices*) recebem fótons e os convertem em impulsos elétricos proporcionais a intensidade recebida nos diferentes pontos de sua superficie. O digitalizador, equipamento eletrônico acoplado a eles, converte os impulsos elétricos analógicos em dados digitais (RUSS; NEAL, 2018).

Após a aquisição, as imagens digitais são representadas por uma função bidimensional, denotada pelas coordenadas (x,y), da intensidade e/ou brilho naquele ponto (SÁ, 2021). Na prática, uma imagem pode ser definida como uma matriz de indices, em que cada elemento (*pixel*) representa sua posição horizontal e vertical em relação a uma origem (SÁ, 2021).

Para uma imagem ser processada computacionamente, ela precisa ser definida em amplitude e em resolução. A coleta dos pontos em intervalos regulares define a amostragem da imagem, e a quantificação destes pontos em números denomina-se quantificação. A resolução é dada tanto nas dimensões vertical e horizontal, e caso seja dado só um dos valores, assume-se que o outro é igual (SÁ, 2021).

A definição acima representa imagens em escala de cinza, e não em cores. Para imagens coloridas, são necessárias matrizes que, quando combinadas, formam um único ponto da imagem (SÁ, 2021).

A conversão de uma imagem em tons de cinza pode ocorrer de duas formas. A primeira constrói o mapa de cores para cinza com base na vizinhança, e a segunda constrói este mapa sem considerar a vizinhança, construindo um mapa mais homogêneo. A primeira forma realça as características das regiões, mas pode-se perder a visão do conjunto, já que dois pixels de mesma cor em regiões diferentes podem ser mapeados para cores diferentes. A segunda forma mantém o conjunto da imagem (SÁ, 2021).

A binarização consiste na representação da imagem de entrada em apenas dois valores possíveis. Pode-se, por exemplo, especificar um nível de cinza de referência, a partir do qual, os pixels que estiverem acima deste nível serão tratados como preto e os pixels que estiverem abaixo serão tratados como brancos. A binarização também é conhecida como limiarização (SÁ, 2021).

A interpolação ocorre quando é necessário aumentar a resolução de presentação da imagem. Geralmente isto é feito determinando valores para os pixels em áreas desconhecidas, tendo como base os dados de áreas conhecidas da imagem. Podem ser utilizadas ferramentas matemáticas como transformada de Fourier e a transformada discreta de cossenos, além de filtros digitais com resposta ao impulso finita ou infinita (SÁ, 2021).

O armazenamento de imagens em memória pode ser feita de diversas formas, com base em diferentes padrões de codificação e armazenamento. O padrão PNG (*Portable Network Graphics*), por exemplo, é baseado na representação RGB (*red*, *green and blue*, ou vermelho, verde e azul) com o possível acréscimo de uma quarta camada de transparência para cada pixel (SÁ, 2021).

Outro padrão para compressão de imagens digitais muito utilizado na atualidade é o JPEG, que é obtido a partir do processamento de blocos de 8×8 *pixels*, que são transformados e quantizados de acordo com uma tabela pré-definida, o que leva à perda da qualidade (RUSS; NEAL, 2018). Os blocos são lidos numa ordem de varredura em formato de zigue-zague, onde o primeiro pixel dos 64 coeficientes é proporcional à média do bloco, e os coeficientes restantes representam outras frequências presentes no sinal. A este sinal é aplicado o código de Huffman, que reduz a repreentação em bits do sinal transformado.

2.3.2 Inteligência Artificial

Inteligência artificial é a capacidade de sistemas de informação de exibir habilidades e comportamentos humanos (HAYKIN, 2001). A inteligência artificial tem se tornado um dos pontos de maior atenção na lida com sistemas de informação e comunicação (JÚNIOR, 2017). Um dos ramos da inteligencia artificial é a visão computacional, que consiste num modo de fazer com que um computador reconheça e processe imagens do mundo real.

Uma das formas de se implementar a inteligência artificial são as redes neurais artificiais, que se beneficiam bastante de grandes quantidades de informação, como no caso das imagens (HAYKIN, 2001). Os primeiros pesquisadores a pensarem nestas redes foram Mcculloth e Pitts, na década de 1940. Segundo eles, o cérebro é composto de neurônios que funcionam de forma binária, onde um sinal considerado forte ativa um neurônio, ou frente a qualquer inibição o sinal do neurônio não fica ativado, não alterando a estrutura da rede (HAYKIN, 2001).

As redes neurais de múltiplas camadas tem sido muito utilizadas em diversas aplica-

ções, em especial no reconhecimento de padrões e objetos (SZEGEDY et al., 2015). Nas redes convolucionais, a tendência tem sido o aumento no número de camadas e de parâmetros de entrada. O crescimento das redes neurais nestes dois sentidos tende a aumentar o número de parametros do modelo, dificultando o treinamento e gerando maior custo computacional (SZEGEDY et al., 2015).

O reconhecimento de padrões com redes neurais convolucionais tradicionalmente ocorre em três etapas. Na primeira etapa, a classificação de características é geralmente feita manualmente por operadores com conhecimentos específicos do domínio. Na segunda etapa, a rede neural é treinada com os dados de forma computacional, e na última etapa o modelo treinado é testado com um novo conjunto de dados (SZEGEDY et al., 2015).

As redes neurais recorrentes (RNNs) são uma classe especial de arquiteturas de redes neurais que lidam com dados sequenciais, levando a ordem e as dependências temporais em consideração. As RNNs possuem utilizam realimentações internas, permitindo que informações sejam propagadas de um passo de tempo para outro. Essa capacidade de manter estados ocultos e aprender com dados sequenciais torna as RNNs adequadas para uma variedade de tarefas, como análise de sentimento em texto, previsão de séries temporais, reconhecimento de fala e tradução automática.

As redes neurais *Long Short-Term Memory* (LSTM) são uma variante de redes neurais recorrentes, desenvolvidas para lidar com o problema do gradiente desvanecente, comum em RNNs tradicionais, de forma que informações relevantes de eventos passados sejam preservadas por meio de mecanismos de controle de memória.

A ResNet (Rede Neural Residual) também lida com o problema do gradiente desvanecente, só que através de blocos residuais que consistem em conexões de atalho (*skip connections*) que pulam camadas intermediárias, de forma que que os gradientes fluam diretamente dos estágios posteriores para os estágios anteriores da rede, reduzindo sua degradação.

O reconhecimento digital de imagens é uma das aplicações do reconhecimento de padrões, fazendo uso principalmente de ferramentas estatísticas para identificar diferentes padrões em imagens (BISHOP, 1995). Para melhorar o reconhecimento de padrões, uma boa estratégia é agrupa-los em conjuntos maiores de interesse para um determinada área (BISHOP, 1995). No caso deste estudo pode-se ter por base as palavras reservadas da linguagem de programação desejada.

2.3.3 Reconhecimento digital de caracteres em imagens

O reconhecimento digital de caracteres (ou OCR, do inglês *Optical Character Recognition*) permite que se detecte texto em imagens, gerando arquivos que possam ser entendidos por máquinas, e permitindo serem manipulados e pesquisados. Pode haver inconsistência

entre os dados percebidos pelo olho humano e os dados recuperados pelo computador, e estas inconsistências podem ser reduzidas com documentos bem iluminados, imagens de boa qualidade e vocabulários atualizados (AULA, 2021).

O OCR é uma tecnologia usada para converter texto impresso ou manuscrito em dados legíveis por máquina. É um processo de extração de informações textuais de documentos e imagens físicas ou digitais e sua conversão em texto editável e pesquisável. A tecnologia OCR usa várias técnicas para reconhecer e interpretar caracteres de documentos ou imagens digitalizadas, encontrando aplicações em vários campos, incluindo digitalização de documentos, automação de entrada de dados, extração de texto de imagens, sistemas de arquivamento e reconhecimento automático de matrículas (ANPR). Simplifica o processo de conversão de documentos físicos em formatos digitais e permite análise e manipulação de texto eficientes (SARFRAZ, 2020).

A primeira etapa é a digitalização, em que se captura uma versão digital de um documento físico, usando uma câmera ou outros dispositivos. A segunda etapa é o pré-processamento, para melhorar a qualidade e a legibilidade do texto. Isso pode envolver tarefas como redução de ruído, rotação de imagem, correção de distorção e ajuste de contraste. Nesta etapa de pré-processamento, costuma-se aplicar também o processo de binarização, que consiste em reduzir a representação da imagem a apenas dois valores, geralmente correspondendo a preto e branco. Contudo, este processo não tem bons resultados caso o plano de fundo da imagem não seja bem distinguível do objeto.

A terceira etapa é a localização de texto, analisando a imagem para definir áreas que potencialmente contenham texto. Identificam-se as regiões ou blocos de texto dentro da imagem. A quarta etapa é a segmentação de texto. Dentro de cada região de texto, o sistema OCR divide o texto em caracteres ou palavras individuais. Esta etapa envolve segmentar o texto e separar os caracteres uns dos outros (HEITLINGER, 2007).

A quinta etapa é a extração de recursos, onde o sistema extrai recursos específicos de cada caractere, como linhas, curvas, ângulos e outros atributos distintos. Esses recursos são usados para diferenciar caracteres. A sexta etapa é o reconhecimento de caracteres, em que são utilizados algoritmos de aprendizado de máquina ou técnicas de reconhecimento de padrões para comparar os recursos extraídos de cada caractere com um banco de dados de caracteres conhecidos. Esse processo atribui o valor de caractere mais provável a cada caractere extraído (HEITLINGER, 2007).

A sétima e última etapa é o pós-processamento. Percebe-se que uma vez que os caracteres são reconhecidos, técnicas de pós-processamento são aplicadas para refinar os resultados. Isso pode envolver correção de erros, verificação ortográfica e melhoria da precisão geral do texto reconhecido. Por fim, tem-se uma representação de texto legível por máquina do documento digitalizado ou capturado. Esse texto pode ser processado, editado, pesquisado ou armazenado digitalmente (HEITLINGER, 2007).

Tabela 2.1 – Modos de segmentação de texto (PSM) do Tesseract

Modo	Descrição
0	Detecção somente de orientação e de fonte utilizada (OSD - *Orientation and Script Detection*)
1	Segmentação automática com OSD
2	Não implementado
3	Segmentação de página sem OSD
4	Assumir uma coluna única de texto com tamanhos variáveis
5	Assuma um único bloco uniforme de texto alinhado verticalmente
6	Assuma um único bloco uniforme de texto
7	Trate a imagem como uma única linha de texto
8	Trate a imagem como uma única palavra
9	Trate a imagem como uma única palavra em um círculo
10	Trate a imagem como um único caractere
11	Texto esparso (encontrar o máximo de texto possível)
12	Texto esparso com OSD
13	Linha única de texto, ignorando os truques específicos do Tesseract

Existem diversas ferramentas de OCR disponíveis, tais como a Google Vision API[1], o EasyOCR[2] e o Tesseract (KAY, 2007). A Google Vision API é uma suíte de ferramentas de visão computacional amplamente utilizada, realizando a extração de textos inseridos em imagens, o reconhecimento de pessoas (rostos e demais características humanas), o reconhecimento de padrões de imagem e a classificação de conteúdo impróprio, dentre outras aplicações. Ela não possui descrição aberta, e seu uso é gratuito somente nas primeiras 1000 chamadas em cada mês.

O EasyOCR é uma biblioteca OCR de código aberto e uso gratuito que utiliza redes ResNet para detectar a presença de caracteres, e redes LSTM para reconhecer os caracteres.

O Tesseract é outra biblioteca OCR de código aberto, desenvolvida originalmente pela Hewlett-Packard (HP) nos anos 1980 e 1990. Ele é considerado o mais avançado entre os *softwares* gratuitos, e é capaz de reconhecer caracteres mesmo em imagens de menor resolução. O Tesseract utiliza redes LSTM no reconhecimento de caracteres, e também tem uso gratuito e código aberto.

O Tesseract é utilizado em milhares de projetos (TESSERACT OCR, 2015), e possui opções de calibração e ajuste, de acordo com o tipo de aplicação em que está sendo utilizado.

[1] https://cloud.google.com/vision
[2] https://github.com/JaidedAI/EasyOCR

É possível configurar camadas, densidade de pixels, esquema de cores, foco, resolução e outras possiblidades, para que se obtenha a maior acurácia possível. Dentre seus diversos parâmetros de configuração, destacamos:

- A língua do texto em questão (inglês, português etc.);
- O modo de segmentação PSM (*page segmentation mode*), que define a forma como o texto será procurado na imagem antes de ser reconhecido (Tabela 2.1);
- O modo de reconhecimento de caracteres OEM (*OCR Engine Mode*), que executa o OCR propriamente dito:
  - **Modo 0:** modo original;
  - **Modo 1:** modo baseado em redes neurais LSTM;
  - **Modo 2:** modos 0 e 1;
  - **Modo 3:** detecção automática de modo (o que estiver disponível).

3 Metodologia proposta

Dado este contexto de ensino de programação, neste trabalho propõe-se o desenvolvimento de uma ferramenta ativa para o ensino de uma linguagem de programação, baseada na identificação de textos em imagens. Dado que o aluno esteja estudando uma linguagem de programação em videos *online* e deseje copiar o código apresentado, foi desenvolvido um programa[1] para o usuário selecionar a área de interesse na tela do computador, onde um algoritmo de OCR é aplicado para detectar o conteúdo textual correspondente e copia-lo à área de transferência do sistema operacional. O usuário pode então copiar este conteúdo ao *software* de edição de texto desejado.

Além disso, o programa desenvolvido oferece a opção de acrescentar ao texto detectado erros cometidos tipicamente por alunos que estejam aprendendo aquela linguagem de programação, tais como esquecimento ou acréscimo de caracteres fundamentais daquela linguagem (dois pontos, ponto-e-vírgula etc.). Desta forma, o programa proposto pode ser usado como ferramenta de aprendizagem tanto pelo aluno como pelo professor, na forma de exercícios.

3.1 Ferramentas utilizadas

A fim de oferecer as funcionalidades já citadas, utilizou-se as seguintes bibliotecas e *softwares* no desenvolvimento do programa proposto:

- A captura de códigos voltada para a linguagem Python foi escolhida pois esta apresenta diversas vantagens para a aprendizagem de programação, tais como sintaxe fácil e clara, *feedback* rápido, semântica expressiva e ampla base de material dispónível na internet (GRANDELL et al., 2006).
- Optou-se por criar a interface gráfica e capturar a imagem da tela usando a linguagem Java, que possui diversas bibliotecas adequadas para esta tarefa.
  - Utilizou-se o *Java Development Kit 20* (ORACLE, 2023) e o *Apache Netbeans* (APACHE NETBEANS, 2023) para desenvolver a interface gráfica.
  - Utilizou-se as bibliotecas *Robot* (JAVA PLATFORM SE 8, 2023a), *Color* (JAVA PLATFORM SE 8, 2023b) e *Rectangle* (JAVA PLATFORM SE 8, 2023c) para ilustrar e capturar a região de interesse na tela do computador, pois elas oferecem uma interface mais adequada para o usuário visualizar a região selecionada, diferentemente de bibliotecas em outras linguagens, tais como Python.

[1] Disponível em `https://github.com/manueng/OCRPythonEdu`.

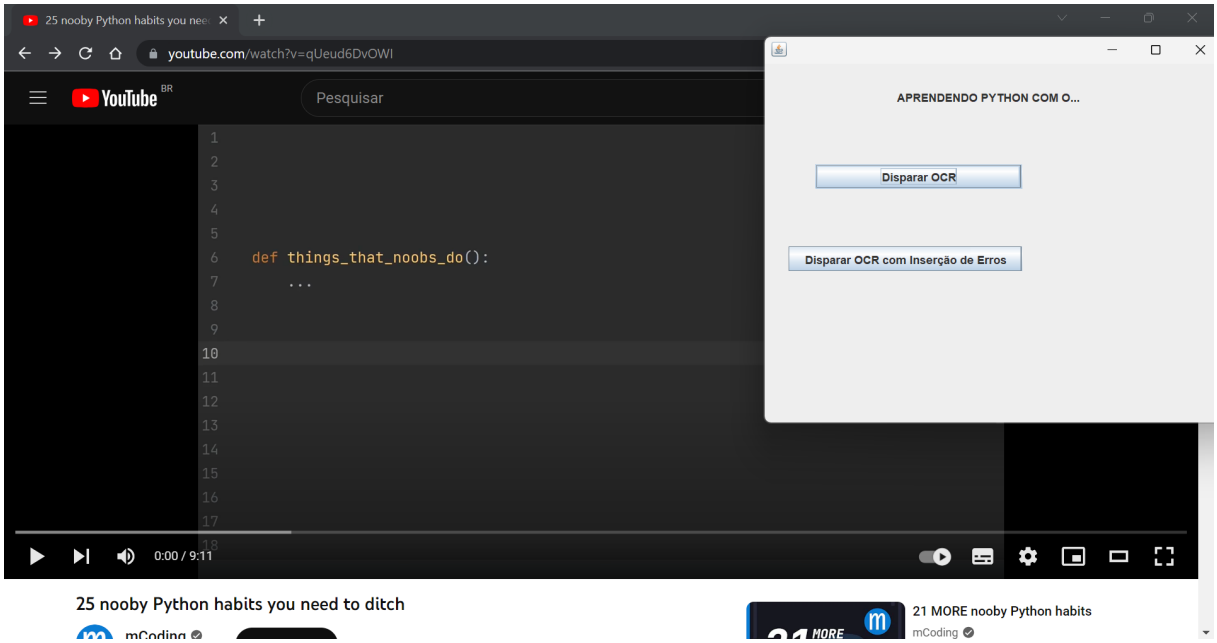

Figura 3.1 – Tela inicial do programa desenvolvido, com video-aula de programação Python ao fundo.

  - Foi necessário o uso de *listeners* (detectores de eventos) para os diversos componentes da tela, como botões e *mouse*, permitindo a realização de tarefas do usuário por meio da interface.

- Para a imagem capturada ser processada antes do reconhecimento de caracteres, foi utilizada a biblioteca OpenCV (OPENCV TEAM, 2023), de código aberto e uso bastante disseminado nas comunidades de visão computacional e de processamento de imagens e vídeo.
- Com relação à detecção de caracteres, optou-se pelo uso do programa Tesseract.
- Para copiar o texto detectado para a área de transferência, foi usada a biblioteca Python *Pyperclip* (AL SWEIGART, 2011).

3.2 Solução proposta

A Figura 3.1 apresenta a interface desenvolvida em Java. É possível ver que foram feitos dois botões para o usuário escolher se o reconhecimento de caracteres é feito com ou sem a inserção de erros típicos. Em ambos os casos, o seguinte processamento é realizado (Figura 3.2):

1. O usuário clica em um dos botões, e a janela da Fig. 3.1 desaparece.
2. O usuário clica com o botão esquerdo do *mouse* no canto superior esquerda da sua região de interesse na tela, segura o botão, arrasta o cursor do *mouse* até o canto

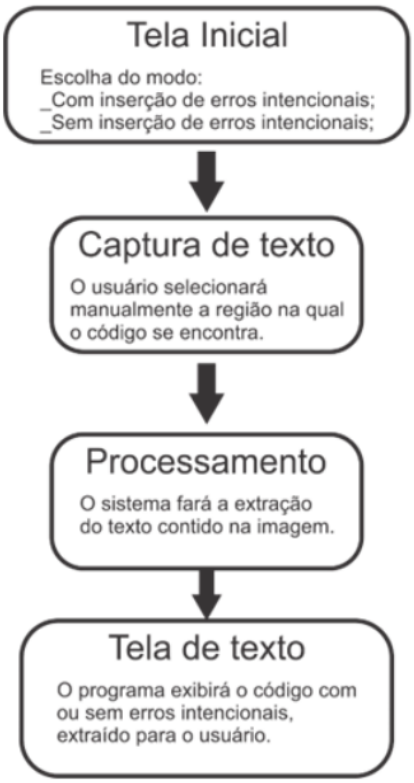

Figura 3.2 – Fluxograma da solução proposta para a extração de códigos em videos *online* por reconhecimento ótico de caracteres.

inferior direito da região de interesse, e solta o botão do *mouse*. Um retângulo com transparência é desenhado na tela para indicar a região selecionada, como indicado nas Figuras 3.3 (seleção parcial) e 3.4 (seleção final).

3. A região selecionada é capturada em uma imagem e salva em disco no formato PNG.

4. A imagem é lida em código Python com a biblioteca OpenCV, e sua resolução é aumentada em 3 vezes com interpolação Lanczos, o que melhora o desempenho do algoritmo OCR.

5. O programa Tesseract é chamado a partir da biblioteca Pytesseract para a imagem de resolução aumentada. Dado que o usuário seleciona a região de interesse por conta própria, não é necessário executar uma segmentação de página PSM muito complexa, de forma que o modo 4 (“assumir uma coluna única de texto com tamanhos variáveis”) foi utilizado. O modo de reconhecimento usado foi LSTM, com dados de treinamento disponibilizados pela equipe do Tesseract para a língua inglesa[2]. Limitou-se os caracteres passíveis de detecção àqueles utilizados na linguagem de programação Python: `0 1 2 3 4 5 6 7 8 9 q w e r t y u i o p a s d f g h j k l z x c v b n m Q W E R T Y U I O P A S D F G H J K L Z X C V B N M \’ ” . , : ( ) < > [ ] { } _ * / + - =`

[2] https://github.com/tesseract-ocr/tessdata/blob/main/eng.traineddata

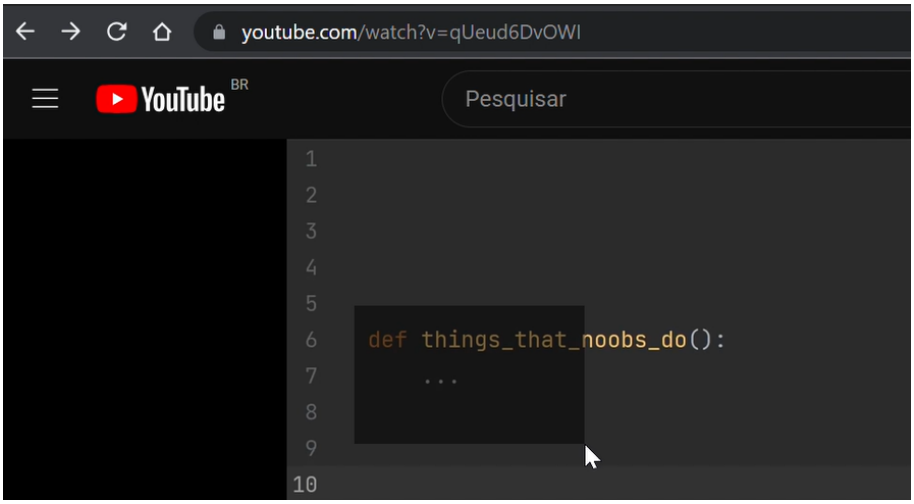

Figura 3.3 – Seleção parcial pelo usuário da região de interesse em video-aula de programação Python.

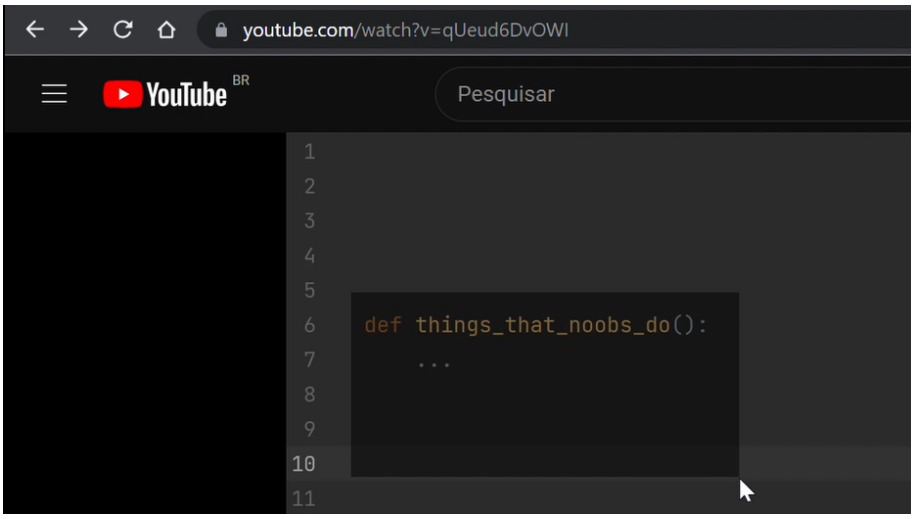

Figura 3.4 – Seleção final pelo usuário da região de interesse em video-aula de programação Python.

6. Caso o segundo botão da interface tenha sido pressionado, erros aleatórios de programação em Python são acrescentados ao texto de saída do Tesseract, com base na biblioteca *Random* do Python.

7. O texto detectado (com ou sem erros adicionados) é apresentado numa nova janela e copiado à área de transferência do sistema operacional com a biblioteca *Pyperclip*. A Figura 3.5 apresenta os resultados para a execução sem e com acréscimo de erros típicos, respectivamente.

3.3 Metodologia de análise

A avaliação de sistemas de reconhecimento de caracteres é tipicamente feita com base em duas métricas: a taxa de erros de caracteres *CER* (*Character Error Rate*) e a taxa de

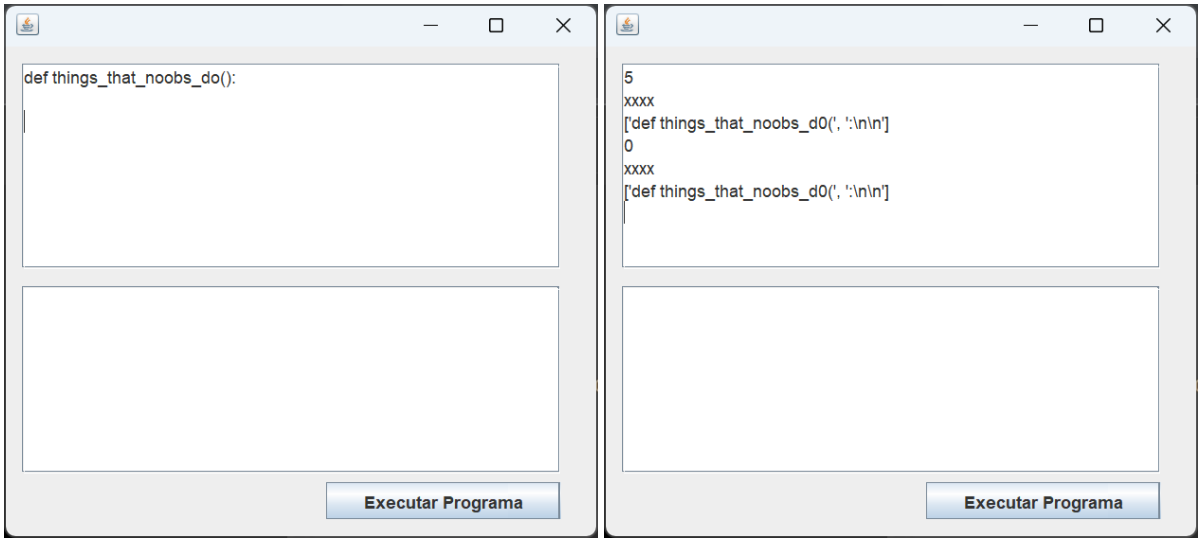

Figura 3.5 – Saída do programa sem e com acréscimo de erros típicos.

erros de palavras WER (*Word Error Rate*):

$$CER = \frac{Sc + Dc + Ic}{Sc + Dc + Cc} \tag{3.1}$$

$$WER = \frac{Sw + Dw + Iw}{Sw + Dw + Cw} \tag{3.2}$$

onde Sc, Dc e Ic são as quantidades de substituições, eliminações e inserções de caracteres, Sw, Dw e Iw são as quantidades de substituições, eliminações e inserções de palavras, e Cc e Cw são as quantidades de caracteres e palavras corretos, respectivamente (TORCHMETRICS, 2020a,b).

Tabela 3.2 – Métricas propostas

Métrica	**Descrição**
CER_1	CER considerando todos os textos de teste como um único texto
CER_2	Médias das CERs individuais
CER_3	CER_1 após a remoção de tabulações e de espaços em branco repetidos
CER_4	CER_2 após a remoção de tabulações e de espaços em branco repetidos
WER_1	WER considerando todos os textos de teste como um único texto
WER_2	Médias das WERs individuais
WER_3	WER_1 após a remoção de tabulações e de espaços em branco repetidos
WER_4	WER_2 após a remoção de tabulações e de espaços em branco repetidos

A fim de avaliar os resultados obtidos pelo programa proposto, foram utilizadas quatro variações destas métricas, como apresentado na Tabela 3.2. Para avaliar se há grandes

desequilíbrios entre os resultados, as métricas CER_1 e WER_1 concatenam todos os resultados em único texto, e as métricas CER_2 e WER_2 calculam médias das CERs e WERs de cada resultado, devendo ser próximas caso não existam grandes discrepâncias individuais. As métricas CER_3, WER_3, CER_4 e WER_4 calculam CERs e WERs sem levar em conta espaços em branco replicados, e foram usadas porque a linguagem Python é indentada, e os algoritmos de OCR tipicamente detectam caracteres, ignorando espaços em branco. Assim, estas métricas oferecem um panorama mais equilibrado do que é possível obter com o programa desenvolvido.

4 Resultados experimentais

4.1 Resultados médios

A fim de testar o programa proposto, foram feitas 21 detecções em um video sobre programação em linguagem Python em resolução HD[1]. Os testes foram realizados em um computador pessoal com processador Intel® Core™ i7-6700 de 3,4 GHz, com 8 GB de memória RAM, HD SSD Fanxiang de 512 GB e sistema operacional Windows™ 10 Pro. Todas as detecções estão disponíveis no Apêndice A, e a Tabela 4.3 apresenta os valores médios obtidos para as métricas propostas na seção anterior, bem como o desvio-padrão de cada medida. As métricas CER_1, CER_3, WER_1 e WER_3 não apresentam desvio-padrão porque todos os textos obtidos são concatenados em um único texto.

Tabela 4.3 – Resultados médios obtidos

Métrica	Média	Desvio-padrão
CER_1	26,1%	—
CER_2	26,5%	10,9%
CER_3	15,0%	—
CER_4	16,2%	10,5%
WER_1	77,6%	—
WER_2	73,7%	16,1%
WER_3	56,3%	—
WER_4	56,5%	17,3%

A partir da Tabela 4.3, é possível tirar uma série de conclusões:

- Os valores de CER são consideravelmente menores do que os valores de WER, o que se explica pelo fato de que a linguagem Python possui muitos caracteres especiais, como `{`, `}`, `[` e `]`. Além disso, definições de funções e de variáveis costumam concatenar palavras em uma única expressão, que é considerada como uma única palavra no cálculo de WER (por exemplo, a função `finally_instead_of_context_manager()` mostrada na Figura 4.6).
- Devido à falta de detecção de espaços em branco pelo Tesseract, as métricas CER_3, WER_3, CER_4 e WER_4 oferecem resultados consideravelmente melhores do que as

[1] Disponível em https://www.youtube.com/watch?v=qUeud6DvOWI.

métricas CER_1, WER_1, CER_2 e WER_2, de forma que fica a cargo do estudante inserir a indentação após a detecção do código.

- Não houveram diferenças significativas entre CER_1 e CER_2, e entre WER_1 e WER_2. Ou seja, não houveram grandes discrepâncias entre os resultados individuais.
- Os desvios-padrão das medidas é relativamente alto (especialmente para CER_4), indicando que o Tesseract pode ter resultados variados.

A Figura 4.6 apresenta o resultado da terceira e da nona detecções realizadas. É possível ver que o Tesseract ignora a indentação, e que ele tem dificuldade em detectar caracteres especiais, tais como `%`. Às vezes, alguns caracteres comuns são trocados, tais como `m`-`n` e `l`-`1`. Resultados para as mesmas telas e com o acréscimo de erros aleatórios são mostrados na Figura 4.7.

4.2 Tempos de execução

Além da análise de taxas de erro de caracteres e de palavras, foram feitas medições de tempo de execução das detecções disponíveis no Apêndice A, como se pode ver na Tabela 4.4. A detecção de caracteres levou uma média de 4,17 segundos, que é facilmente percebido pelo usuário, mas que não chega a ser um impeditivo para o uso do programa desenvolvido. A execução do Tesseract para OCR ocupa a maior parte deste tempo.

Tabela 4.4 – Tempos de execução

Resultado	**Tempo (s)**
1	4,21
2	3,48
3	3,67
4	5,05
5	3,02
6	4,13
7	4,54
8	3,48
9	4,59
10	4,92
11	5,00
12	5,12
13	5,13
14	5,32
15	4,86
16	4,34
17	3,86
18	2,89
19	2,96
20	3,49
21	3,42

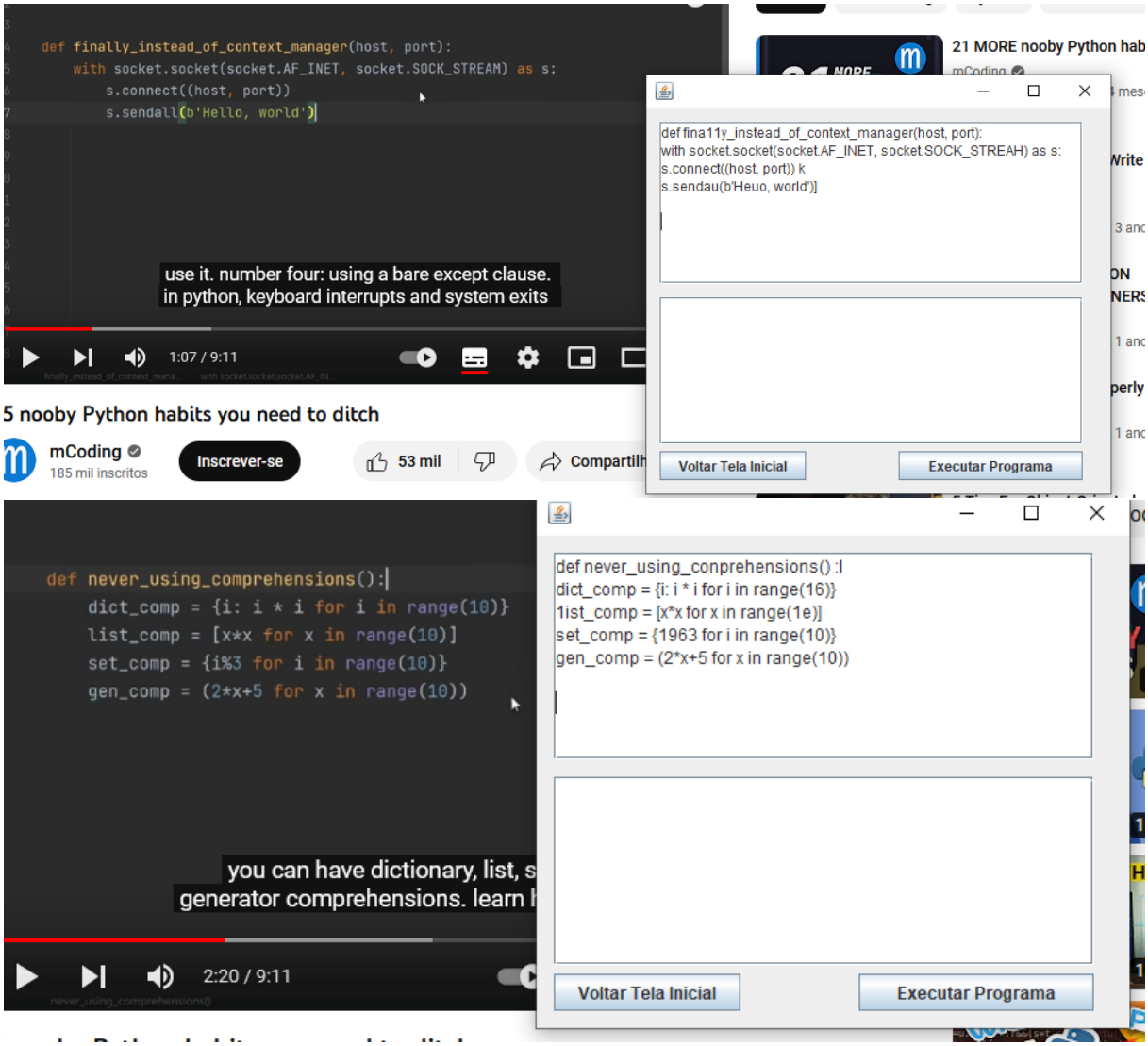

Figura 4.6 – Resultados 3 e 9 sem inserção de erros.

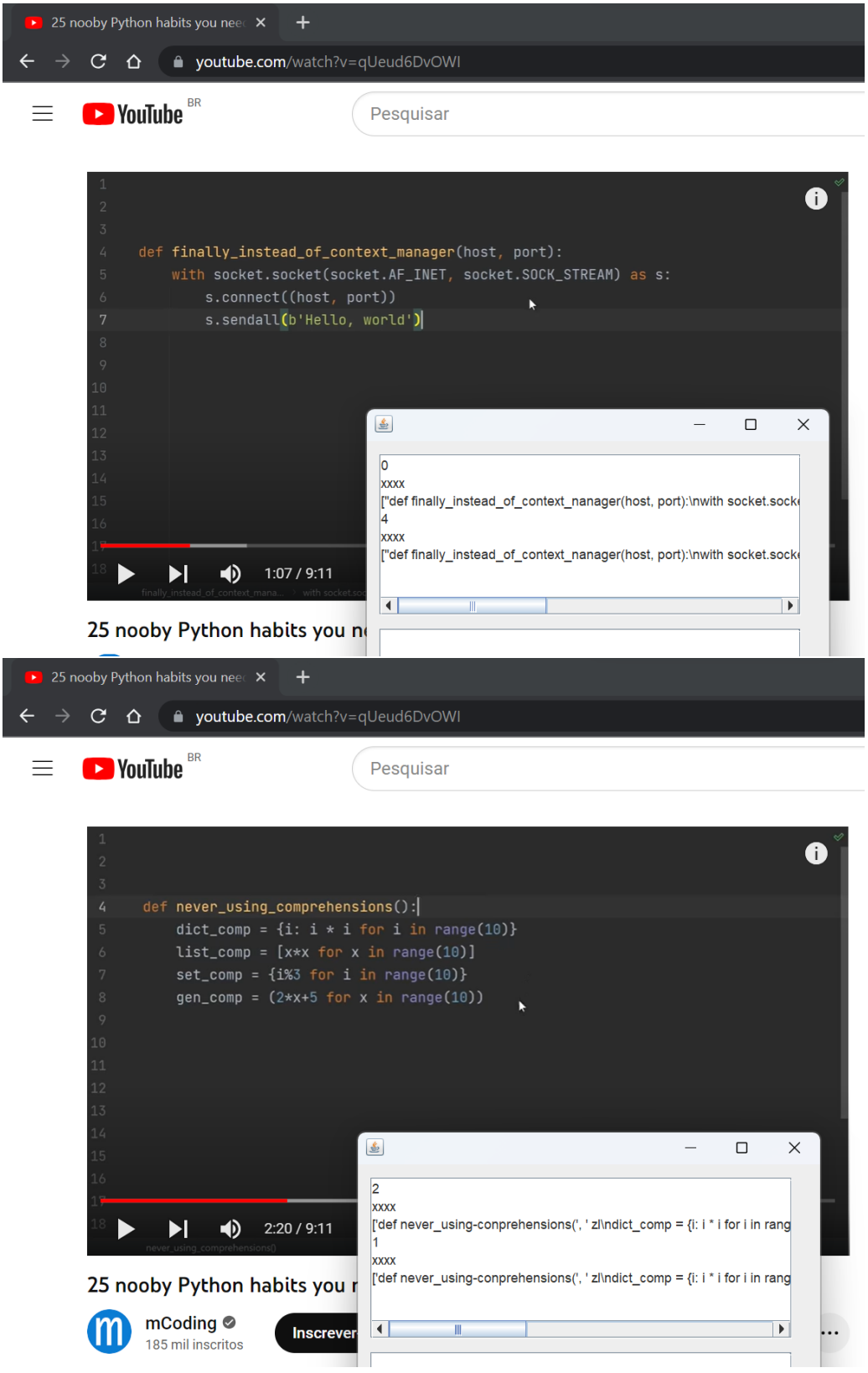

Figura 4.7 – Resultados 3 e 9 com inserção de erros.

5 Conclusão

5.1 Conclusões gerais

Neste trabalho, foi apresentado o desenvolvimento de um programa de detecção de códigos de programação presentes em video-aulas *online*, utilizando o reconhecimento ótico de caracteres em regiões de interesse definidas pelo usuário. Após a detecção, o texto foi apresentado em uma janela, além de automaticamente carregado na área de transferência para uso do estudante. O programa desenvolvido também ofereceu uma opção de aprendizagem ativa, com o acréscimo aleatório de erros, o que força o aluno a corrigi-los antes de utilizar o código. Testes com códigos disponíveis gratuitamente *online* mostraram uma taxa de erros de caracteres média de 15% sem levar em conta espaços em branco entre o texto, mostrando o potencial do sistema desenvolvido em auxiliar na aprendizagem ativa de linguagens de programação.

5.2 Passos futuros

Apesar de o programa proposto ter sido desenvolvido com sucesso, ele ainda possui uma série de pontos de melhoria:

- Melhorias no desempenho do algoritmo OCR:
  - Detecção de espaços em branco e indentação;
  - Treinamento *offline* de caracteres típicos para uso com as redes neurais do Tesseract, de forma a melhorar os resultados de CERs e WERs;
  - Uso de dicionários para eliminar erros que pouco agregam ao apreendizado;
  - Testes com outras ferramentas de OCR, como o EasyOCR e a Google Vision API;
- Melhorias na interface:
  - Permitir a compilação e execução de programas Python, que requeiram a interação do usuário;
  - Exibir erros mais significativos;
- Melhorias na metodologia pedagógica:
  - Aprimoramentos do algoritmo de inserção intencional de erros;
  - Aplicação de outras linguagens de programação;

- Realização de experimentos reais com alunos para avaliar a viabilidade do uso da ferramenta desenvolvida, utilizando:
  * Um grupo de controle sem acesso ao OCR,
  * Um grupo experimental com acesso ao OCR sem erros aleatórios, e
  * Um segundo grupo experimental com acesso ao OCR com erros aleatórios.

Referências

ABBAGNANO, N. Dicionário de filosofia. In. ISBN 9788578275211. Disponível em: <https://books.google.com.br/books?id=v6%5C_7zwEACAAJ>. Citado na p. 16.

AL SWEIGART. **Pyperclip**. 2011. https://github.com/asweigart/pyperclip. Acessado em 17 de julho de 2023. Citado na p. 26.

APACHE NETBEANS. Mai. 2023. https://netbeans.apache.org/download/index.html. Acessado em 17 de julho de 2023. Citado na p. 25.

AULA, L. **Improvement of Optical Character Recognition on Scanned Historical Documents Using Image Processing**. 2021. Citado na p. 22.

BISHOP, C. **Neural Networks for Pattern Recognition**. Clarendon Press, 1995. (Advanced Texts in Econometrics). ISBN 9780198538646. Disponível em: <https://books.google.com.br/books?id=T0S0BgAAQBAJ>. Citado na p. 21.

BOGDANOV, A. **Essays in Tektology**. Intersystems Publications, 1980. (Systems inquiry series). Disponível em: <https://books.google.com.br/books?id=B4Y-AQAAIAAJ>. Citado na p. 16.

DEWEY, J. **Democracia e educação: capıtulos essenciais**. Ática, 2007. (Ensaios comentados). ISBN 9788508114733. Disponível em: <https://books.google.com.br/books?id=tXKAPgAACAAJ>. Citado na p. 16.

EDUCATION FOR THE TWENTY-FIRST CENTURY, I. C. on; DELORS, J.; UNESCO. **Learning, the Treasure Within: Report to UNESCO of the International Commission on Education for the Twenty-first Century**. Unesco Pub., 1996. ISBN 9789231032745. Disponível em: <https://books.google.com.br/books?id=v4KcAAAAMAAJ>. Citado na p. 16.

GRANDELL, L.; PELTOMÄKI, M.; BACK, R.-J.; SALAKOSKI, T. Why Complicate Things? Introducing Programming in High School Using Python. In: PROCEEDINGS of the 8th Australasian Conference on Computing Education - Volume 52. Hobart, Australia: Australian Computer Society, Inc., 2006. (ACE '06), p. 71–80. ISBN 1920682341. Citado na p. 25.

HAYKIN, S. **Redes Neurais: Princıpios e Prática**. Bookman Editora, 2001. ISBN 9788577800865. Disponível em: <https://books.google.com.br/books?id=bhMwDwAAQBAJ>. Citado na p. 20.

HEITLINGER, P. A legibilidade das letras para leitura automática. In: 3. CADERNOS de Tipografia. 2007. Citado na p. 22.

JAVA PLATFORM SE 8. Abr. 2023a. https://docs.oracle.com/javase/8/docs/api/java/awt/Robot.html. Acessado em 17 de julho de 2023. Citado na p. 25.

JAVA PLATFORM SE 8. Abr. 2023b. https://docs.oracle.com/javase/8/docs/api/java/awt/Color.html. Acessado em 17 de julho de 2023. Citado na p. 25.

JAVA PLATFORM SE 8. Abr. 2023c. https://docs.oracle.com/javase/8/docs/api/java/awt/Rectangle.html. Acessado em 17 de julho de 2023. Citado na p. 25.

JÚNIOR, A. **Sistemas de segurança da informação na era do conhecimento**. InterSaberes, 2017. ISBN 9788559723021. Disponível em: <https://books.google.com.br/books?id=7fv-zwEACAAJ>. Citado nas pp. 19, 20.

KAY, A. Tesseract: An Open-Source Optical Character Recognition Engine. **Linux J.**, Belltown Media, Houston, TX, v. 2007, n. 159, p. 2, jul. 2007. ISSN 1075-3583. Citado na p. 23.

KENSKI, V. **Tecnologias e ensino presencial e a distância**. Papirus, 2003. (Prática Pedagógica). ISBN 9788530807085. Disponível em: <https://books.google.com.br/books?id=dWdWPHkGCEkC>. Citado nas pp. 14, 16, 17.

KOLB. **Experiental Learning Experience as the source of learning and Development**. 1984. v. 1, p. 1–10. Citado na p. 13.

LÁSZLÓ, S. P.-.-Z. Methods of teaching programming. **Teaching Mathematics and Computer Science**, v. 1, p. 247–258, 2003. Citado na p. 18.

LEVITIN, A. Design and analysis of algorithms reconsidered. In: PROCEEDINGS of the thirty-first SIGCSE technical symposium on Computer science education. 2000. P. 16–20. Citado na p. 18.

MAGALHÃES, J. **Da cadeira ao banco: escola e modernização : (séculos XVIII-XX)**. Educa, Unidade de I & D de Ciências da Educação, 2010. (Ciências da Educação / direcção de António Nóvoa). ISBN 9789898272102. Disponível em: <https://books.google.com.br/books?id=AFOwnQAACAAJ>. Citado na p. 16.

MARKETSPLASH TEAM. **"20 Melhor Software De OCR (Reconhecimento Óptico De Caracteres) Gratuito Pago"**. 2022. Acessado em 17 de julho de 2023. Disponível em: <https://marketsplash.com/pt/melhor-software-ocr/>. Citado na p. 14.

OPENCV TEAM. 2023. https://pypi.org/project/opencv-python/. Acessado em 17 de julho de 2023. Citado na p. 26.

ORACLE. Abr. 2023. https://www.oracle.com/java/technologies/downloads/. Acessado em 17 de julho de 2023. Citado na p. 25.

RUSS, J.; NEAL, F. **The Image Processing Handbook**. CRC Press, 2018. ISBN 9781498740289. Disponível em: <https://books.google.com.br/books?id=ROSYCgAAQBAJ>. Citado nas pp. 19, 20.

SÁ, Y. V. d. A. **Desenvolvimento de aplicações IA: robótica, imagem e visão computacional**. Platos Soluções Educacionais S.A., 2021. ISBN 9786589881681. Citado nas pp. 19, 20.

SARFRAZ, M. **Digital Imaging**. Rijeka: IntechOpen, mai. 2020. ISBN 978-1-78985-600-2. DOI: 10.5772/intechopen.83239. Disponível em: <https://doi.org/10.5772/intechopen.83239>. Citado na p. 22.

SILVA, T. R. da; MEDEIROS, T.; MEDEIROS, H.; LOPES, R.; ARANHA, E. Ensino-aprendizagem de programação: uma revisão sistemática da literatura. **Revista Brasileira de Informática na Educação**, v. 23, n. 01, p. 182, 2015. Citado na p. 17.

SOUZA, M. de; GOMES, M.; SOARES, M.; CONCILIO, R. **Algoritmos e lógica de programação: um texto introdutório para a engenharia**. Cengage Learning, 2020. ISBN 9788522128150. Disponível em: <https://books.google.com.br/books?id=VQpFzgEACAAJ>. Citado na p. 18.

SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V.; RABINOVICH, A. Going deeper with convolutions, p. 1–9, 2015. DOI: 10.1109/CVPR.2015.7298594. Citado na p. 21.

TESSERACT OCR. 2015. https://github.com/tesseract-ocr/tesseract. Acessado em 17 de julho de 2023. Citado na p. 23.

TORCHMETRICS. 2020a. https://torchmetrics.readthedocs.io/en/stable/text/char_error_rate.html. Acessado em 17 de julho de 2023. Citado na p. 29.

TORCHMETRICS. 2020b. https://torchmetrics.readthedocs.io/en/stable/text/word_error_rate.html. Acessado em 17 de julho de 2023. Citado na p. 29.

TUCKER, A.; NOONAN, R. **Linguagens de programação: princıpios e paradigmas**. McGraw-Hill, 2009. ISBN 9788577260447. Disponível em: <https://books.google.com.br/books?id=v3OrPgAACAAJ>. Citado na p. 17.

TUCKER, A. **A Model Curriculum for K–12 Computer Science: Final Report of the ACM K–12 Task Force Curriculum Committee**. New York, NY, USA, 2003. ISBN 1581138377. Citado na p. 19.

Apêndices

Apêndice A – Resultados individuais

Tabela A.5 – Resultados individuais obtidos

Resultados	*CER* **com espaços**	*CER* **sem espaços**	*WER* **com espaços**	*WER* **sem espaços**
1	52,5%	52,5%	71,4%	71,4%
2	6,6%	6,6%	22,2%	22,2%
3	24,0%	17,5%	89,3%	81,8%
4	51,1%	32,9%	94,4%	82,1%
5	13,4%	7,8%	60,0%	44,4%
6	21,4%	17,9%	75,0%	68,8%
7	26,5%	16,7%	85,4%	73,9%
8	21,2%	9,3%	76,0%	50,0%
9	14,3%	9,4%	56,5%	41,2%
10	22,1%	9,4%	60,6%	30,0%
11	19,5%	6,1%	65,9%	36,2%
12	28,7%	17,8%	84,3%	68,0%
13	23,2%	12,0%	79,2%	57,7%
14	29,5%	12,1%	85,4%	61,1%
15	30,4%	11,1%	93,3%	80,0%
16	39,6%	27,2%	83,3%	62,5%
17	27,9%	13,9%	81,5%	61,5%
18	27,9%	13,9%	81,5%	61,5%
19	28,3%	17,4%	69,0%	43,8%
20	27,0%	13,7%	71,1%	38,9%
21	20,9%	14,7%	63,2%	50,0%

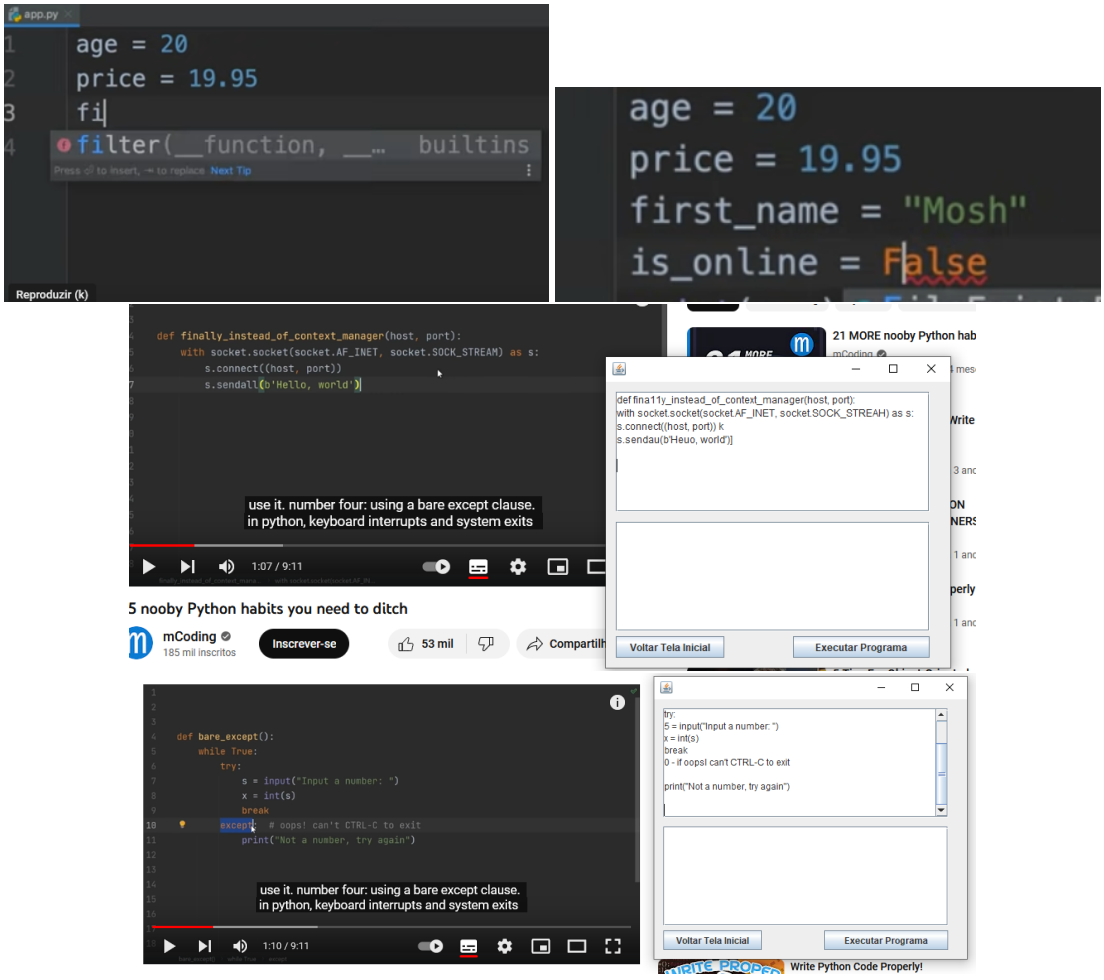

Figura A.8 – Resultados 1 a 4.

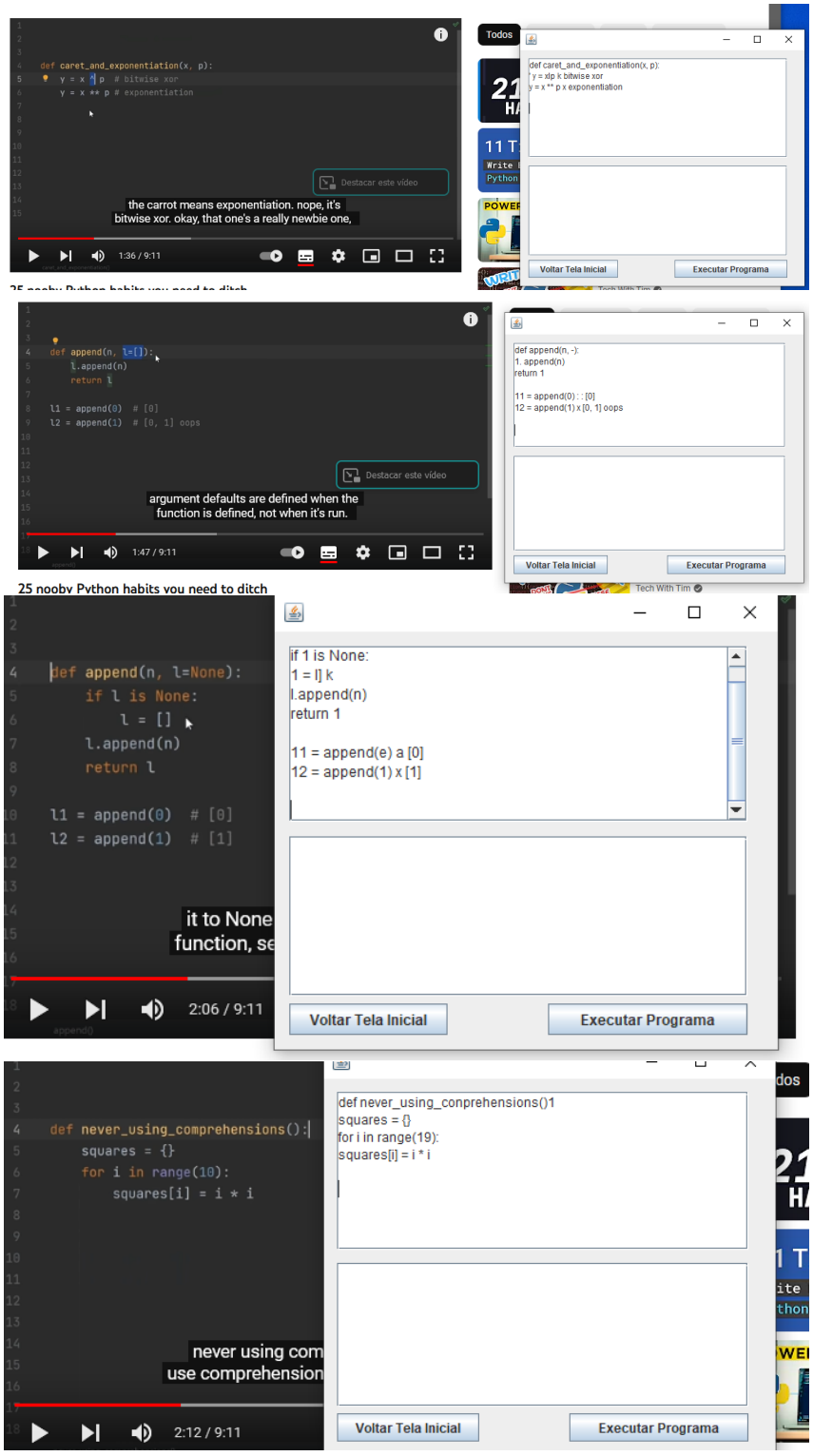

Figura A.9 – Resultados 5 a 8.

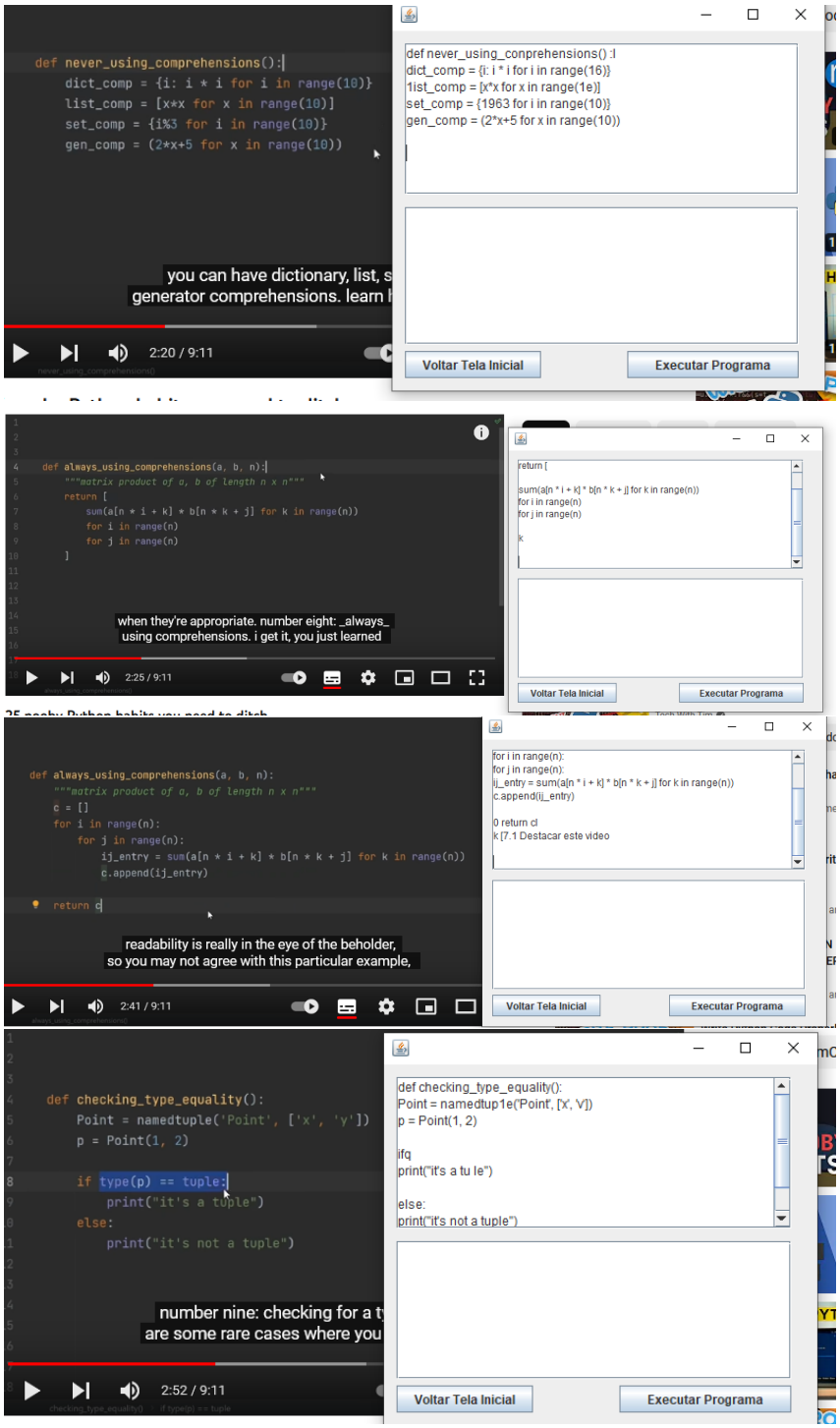

Figura A.10 – Resultados 9 a 12.

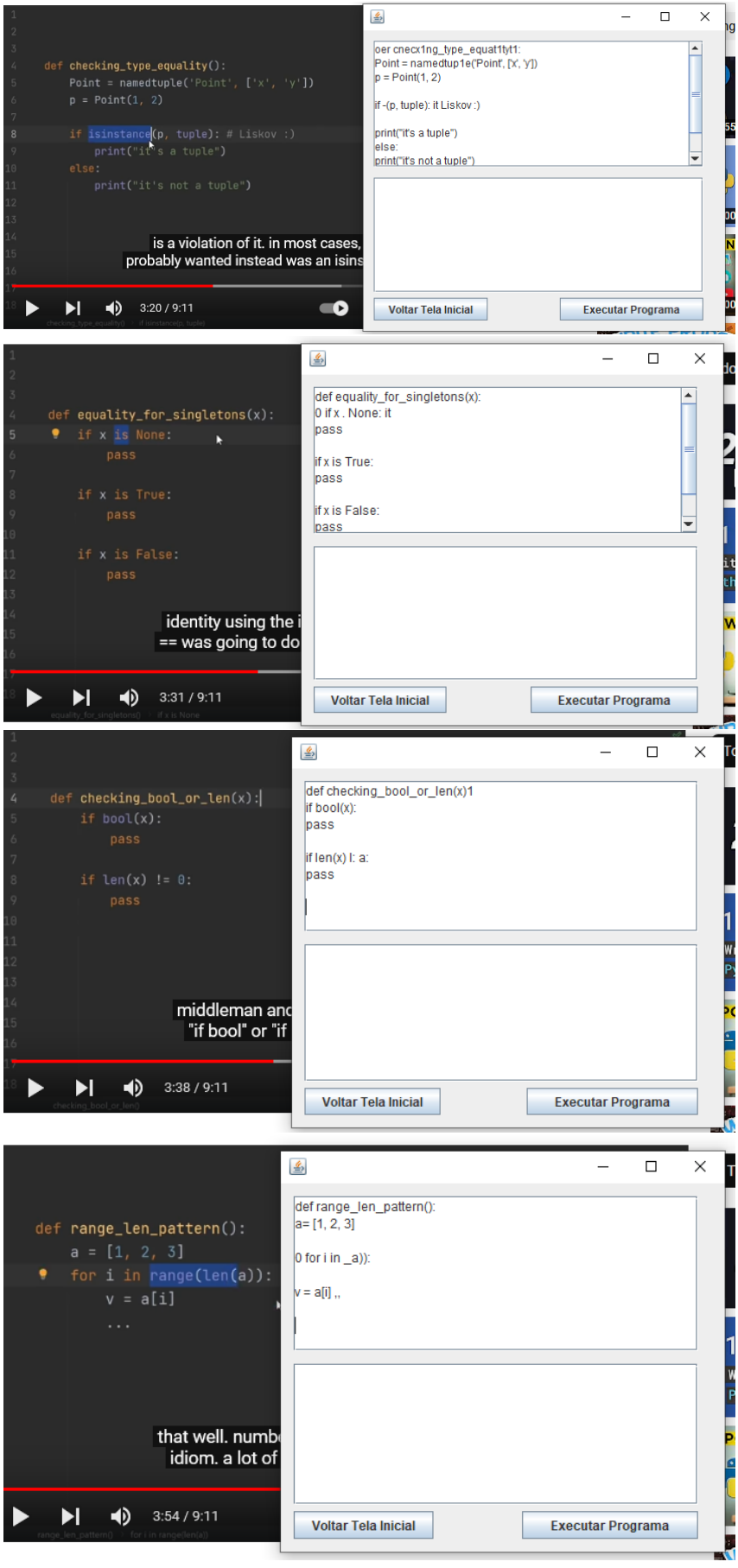

Figura A.11 – Resultados 13 a 16.

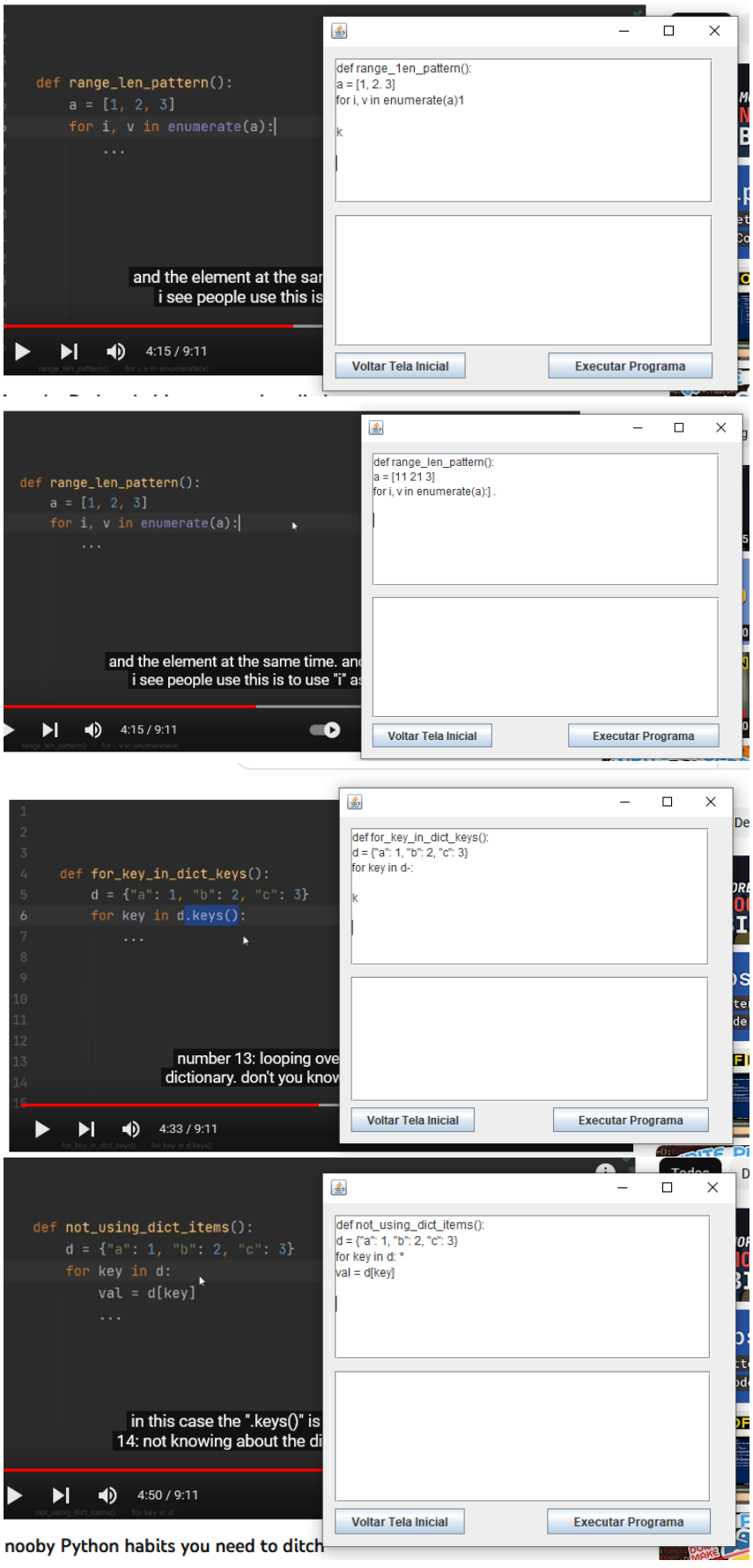

Figura A.12 – Resultados 17 a 20.

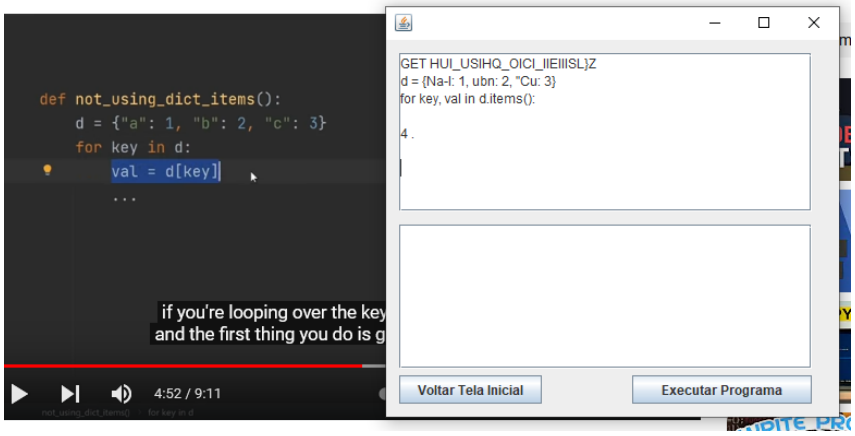

Figura A.13 – Resultado 21.