UNIVERSIDADE ESTADUAL DA PARAÍBA
CAMPUS I – CAMPINA GRANDE
CENTRO DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE COMPUTAÇÃO
CURSO DE LICENCIATURA PLENA EM COMPUTAÇÃO

JOAO PAULO DE ANDRADE COUTINHO

UTILIZANDO A FERRAMENTA SCRATCH COMO FACILITADOR NO ENTENDIMENTO DE CONCEITOS BÁSICOS DE COMPONENTES DA LÓGICA DE PROGRAMAÇÃO

CAMPINA GRANDE
2023

JOÃO PAULO DE ANDRADE COUTINHO

UTILIZANDO A FERRAMENTA SCRATCH COMO FACILITADOR NO ENTENDIMENTO DE CONCEITOS BÁSICOS DE COMPONENTES DA LÓGICA DE PROGRAMAÇÃO

Trabalho de Conclusão de Curso (Artigo) apresentado ao Departamento do Curso de Computação da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de Licenciado em Computação.

Orientador: Prof. Me. Edson Holanda Cavalcante Júnior.

CAMPINA GRANDE
2023

C871u Coutinho, Joao Paulo de Andrade.

Utilizando a ferramenta Scratch como facilitador no entendimento de conceitos básicos de componentes da lógica de programação [manuscrito] / Joao Paulo de Andrade Coutinho. - 2023.

31 p. : il. colorido.

Digitado.

Trabalho de Conclusão de Curso (Graduação em Computação) - Universidade Estadual da Paraíba, Centro de Ciências e Tecnologia, 2023.

"Orientação : Prof. Me. Edson Holanda Cavalcante Júnior, Coordenação do Curso de Computação - CCT. "

1. Scratch. 2. Python. 3. Lógica de Programação. I. Título

21. ed. CDD 005.13

Elaborada por Camile de A. Gomes - CRB - 15/559 BC/UEPB

JOAO PAULO DE ANDRADE COUTINHO

UTILIZANDO A FERRAMENTA SCRATCH COMO FACILITADOR NO ENTENDIMENTO DE CONCEITOS BÁSICOS DE COMPONENTES DA LÓGICA DE PROGRAMAÇÃO

Trabalho de Conclusão de Curso de Graduação em Ciência da Computação da Universidade Estadual da Paraíba, como requisito à obtenção do título de Bacharel em Ciência da Computação.

Aprovada em 03 de Março de 2023.

Prof. Me. Edson Holanda Cavalcante Júnior (DC - UEPB)
Orientador(a)

Prof. Dr. Wellington Candeia de Araujo (DC - UEPB)
Examinador(a)

Prof. Me. Francisco Anderson Mariano da Silva (CCEA/UEPB)
Examinador(a)

LISTA DE ILUSTRAÇÕES

SUMÁRIO

UTILIZANDO A FERRAMENTA SCRATCH COMO FACILITADOR NO ENTENDIMENTO DE CONCEITOS BÁSICOS DE COMPONENTES DA LÓGICA DE PROGRAMAÇÃO

João Paulo de Andrade Coutinho[1]

RESUMO

A lógica de programação é uma habilidade fundamental para qualquer pessoa que deseja aprender a programar. Ela envolve o entendimento de como pensar de forma lógica, ordenada e estruturada com o objetivo de desenvolver algoritmos e resolver problemas de maneira eficiente. No entanto, o desenvolvimento destes conceitos no dia a dia pode acabar criando uma barreira na evolução de muitos alunos, pois em vários casos, requer a capacidade de abstrair situações e pensar de forma abstrata, algo que muitos não estão acostumados a fazer por causa do modo de ensino que foram expostos em anos anteriores na escola. A fim de contribuir para a diminuição dessa problemática, foram realizadas pesquisas que investigaram esse cenário, e paralelamente, a busca de ferramentas alternativas que pudessem aproximar mais o aluno desses conceitos e minimizar as lacunas causadas por eles. Para isso, utilizou-se uma abordagem de comparação, resolvendo alguns exemplos detalhados entre a linguagem de programação Python - que hoje é utilizada na disciplina de Algoritmos - abarcando os componentes da lógica de programação do Bacharelado em Computação da Universidade Estadual da Paraíba e o Scratch, uma ferramenta escolhida por possuir um ambiente de programação rico em mídia e propício à quebra das abstrações. Os exemplos foram trabalhados nas duas linguagens e detalhados linha por linha, mostrando que o Scratch pode ser utilizado como facilitador no ensino dos componentes da lógica em conjunto com outras linguagens.

Palavras-chave: Scratch. Python. Lógica de Programação. Facilitador.

ABSTRACT

Programming logic is a fundamental skill for anyone who wants to learn to program. It involves understanding how to think in a logical, orderly, and structured way in order to develop algorithms and solve problems efficiently. However, the development of these concepts on a daily basis can end up creating a barrier in the evolution of many students, because in several cases, it requires the ability to abstract situations and think abstractly, something that many are not used to doing because of the way they were taught in previous years at school. In order to contribute to the reduction of this problem, research was conducted to investigate this scenario, and, in parallel, the search for alternative tools that could bring the student closer to these concepts and minimize the gaps caused by them. For this, a comparison approach was used, solving some detailed examples between the Python programming language - which is currently used in the Algorithms discipline - covering the programming logic components of the Bachelor's Degree in Computer Science at the Paraíba State University, and Scratch, a tool chosen for having a media-rich programming environment that is conducive to breaking abstractions. The examples were worked

in both languages and detailed line by line, showing that Scratch can be used as a facilitator to teach logic components in conjunction with other languages.

Keywords: Scratch. Python. Programming Logic. Facilitator.

[1] Universidade Estadual da Paraíba (UEPB) – Campina Grande – PB – Brasil
joao.paulo.coutinho@aluno.uepb.edu.br

1. INTRODUÇÃO

O estudo da Lógica de Programação é considerado por muitos, um dos principais componentes curriculares nas grades dos cursos que formam a área de Tecnologia, destacando-se por fundamentar, como pré-requisito, as disciplinas de desenvolvimento de sistemas, focando na iniciação do conhecimento para a programação. Conforme nos indica Puga (2008, p. 9), esse estudo é tido como base para a criação de algoritmos, que são sequências de instruções finitas as quais levam à realização de tarefas definidas.

> Algoritmos e Lógica de programação são disciplinas indispensáveis para a área de informática, tanto no bacharelado quanto nos cursos tecnológicos. Sua importância reside em desenvolver a lógica e o raciocínio do estudante, que normalmente sente dificuldade para elaborar e desenvolver algoritmos. (PUGA; RISSETTI, 2008, p. XI).

Os algoritmos são criados para resolver problemas computacionais de forma eficiente, clara e precisa; para isso é essencial que se tenha um bom entendimento da lógica de programação, visto ser possível, através dela, adquirir a habilidade de organizar as ideias e estruturar de forma coerente à problemática, desse modo o aluno terá capacidade de analisar e dividir um problema em blocos menores facilitando o gerenciamento e consequentemente a sua resolução.

Conseguir entender e aplicar os conceitos estudados da Lógica de Programação é de fundamental importância para que o aluno possa avançar com uma base sólida no decorrer do curso.

Destaco abaixo alguns conceitos abordados dentro do estudo da lógica que norteiam esse desenvolvimento:

1. **Estruturas de Controle de Fluxo de Execução** – São blocos de códigos utilizados para controlar o fluxo de execução de um programa, permitindo que sejam tomadas decisões baseado em determinadas condições podendo ser: Sequencial – Quando os comandos são executados numa sequencia pré-estabelecida, um após o término do outro. Decisão – Quando o fluxo de instruções a ser seguido é escolhido em função da análise de uma ou mais condições. Repetição – Quando se faz necessário a repetição de um determinado trecho do código por algum motivo específico.

2. **Constantes e Variáveis** – São elementos fundamentais da programação, uma constante possui um valor fixo que não pode ser alterado durante o processo de execução, por isso o nome constante. Variáveis são utilizadas para armazenar informações que podem ser alteradas durante a execução do programa.

3. **Tipos de Dados** – São características importantes das constantes e variáveis, pois definem o tipo de valor que será armazenado e como serão manipulados. Podendo ser numérico, booleano, texto e outros.

4. **Operadores Lógicos** – São símbolos usados para combinar ou modificar expressões (AND, NOT, OR).

5. **Operadores Matemáticos** – São símbolos usados para realização de operações matemáticas (+, -, *, /).
6. **Funções e Procedimentos** – São estruturas que realizam tarefas específicas e também permitem a reutilização de códigos dentro do programa, evitando repetições desnecessárias e tornando o programa mais organizado para ser mantido. Respectivamente, funções são utilizadas em situações que precisamos retornar algum valor para manipulação, como o resultado de um cálculo e os procedimentos são usados quando não precisamos desse retorno para manipulação, como no caso da impressão de uma mensagem na tela.

O termo 'lógica' em alguns dicionários online é definido como: "Modo de raciocinar coerente que expressa uma relação de causa e consequência; raciocínio, método" ou simplesmente "Sequência coerente de ideias" (LÓGICA, 2023). Forbellone et al. (2005, p.1), por sua vez, sintetiza a lógica como:

> A "arte de pensar bem", que é a "ciência das formas do pensamento". Podendo dizer ainda que a lógica tem em vista a "ordem da razão", dando a entender que a nossa razão pode funcionar desordenadamente e por isso, a lógica estuda e ensina a colocar "ordem no pensamento". (FORBELLONE; EBERSPACHER, 2005, p 1).

Inferimos então, baseado nos significados expostos, que o estudo da lógica funciona como uma técnica capaz de desenvolver uma forma de ordenar as ideias e quando voltada para a programação, possui a finalidade de estruturar o passo a passo para que possamos escrever uma determinada instrução, onde o objetivo é chegar a um resultado da forma mais eficiente possível.

O estudo da lógica, já no inicio do curso, se torna necessário para que o aluno seja preparado para resolver problemas computacionais que podem conter um alto grau de complexidade; ele deverá ter a capacidade de analisar, definir e organizar ideias e instruções e a partir daí buscar uma solução que traga uma harmonia para aquilo que ele está pensando e o que está fazendo. Para Santos et al. (2015), o aprendizado desses conceitos exige do aluno o domínio de habilidades específicas relacionadas à abstração e ao raciocínio lógico, aptidões que são pouco exploradas no ensino médio.

1.1 Contexto e Motivação

Como os conceitos da lógica de programação se mostram repletos de definições que podem ser consideradas abstratas, principalmente para aqueles alunos que nunca tiveram contato com esse tipo de conteúdo, a forma de transmissão é de fundamental importância, pois o aluno precisa entender a essência e a necessidade de estudá-las. Quando isso não acontece, acaba interferindo diretamente na sua motivação e consequentemente no desempenho.

> A disciplina de Algoritmos e Programação nas universidades tem sido uma problemática abordada em frequentes discussões de trabalhos científicos, congressos e fóruns. Este cenário é reflexo da dificuldade no entendimento aliado ao mau desenvolvimento do raciocínio lógico, além do que, este problema é responsável pela evasão nos cursos e reprovação dos alunos na disciplina. (SILVA; SALES; AMORIM, 2020, p.408).

Esse cenário foi confirmado em diversos artigos estudados durante o desenvolvimento deste trabalho, onde as deficiências que o aluno acumula, ano após ano, são agravadas pela complexidade dos conteúdos expostos nos componentes da Lógica de Programação ou em disciplinas iniciais nos cursos de tecnologia, que acabam alimentando um alto índice de reprovação e abandono de curso, conforme no diz Coutinho et al. (2017, p. 2):

> No início do ensino superior é comum que estudantes da área da computação e áreas afins tenham dificuldade no raciocínio lógico, conteúdo necessário e essencial para o projeto e criação de soluções, uma vez que, no ensino médio, tais estudantes estavam acostumados a apenas aplicar fórmulas previamente existentes na solução de problemas. (COUTINHO; LIMA; SANTOS, 2017, p.2).

Por se tratar de um cenário problemático e desafiador, é premente buscar metodologias e ferramentas que possam aproximar o aluno do conceito de lógica e assim facilitar sua compreensão, somando-as ao processo de ensino aprendizagem, com o objetivo de potencializar o aprendizado do mesmo com recursos que possam ser mais atrativos.

Durante a pesquisa, foram identificados e analisados estudos que trabalharam ferramentas no intuito de tentar contribuir com a assimilação dos conteúdos e tratar os conceitos de forma mais clara e objetiva, de modo que os alunos consigam trabalhar situações que antes estavam no campo da abstração, com metodologias ativas que proporcionem o desenvolvimento de habilidades necessárias para a sequencia do curso.

Sobre isso, afirma Peixoto (2022, p. 22):

> Uma estratégia que pode ser utilizada para mudar este cenário, já discutido previamente, é a utilização de ferramentas didáticas compostas principalmente por Tecnologias de Informação e Comunicação (TIC's), que oferecem recursos para a elaboração de atividades que visam à aproximação e melhor interação entre o docente e seus alunos. São destacadas diversas classes de ferramentas, que podem ser físicas ou digitais (softwares) visando à elaboração e correção de códigos, recursos que são baseados em jogos e programas de apoio ao ensino da programação. (PEIXOTO, 2022, p. 22)

Entre as possibilidades analisadas, foi visto que a Ferramenta Scratch possui vários estudos que a apresentam como porta de entrada para o mundo da programação e é voltada para desenvolvimento do pensamento computacional, resolução de problemas, aprendizagem criativa e compartilhada, tendo como um de seus principais pontos positivos a possibilidade de se programar sem a preocupação imediata com a sintaxe, o que não acontece com as outras linguagens de programação.

Para Ventura et al. (2019), o Scratch pode realmente ser um importante facilitador de aprendizagem, tendo sua benéfica associada não apenas aos futuros programadores, mas também a quem não consegue compreender a lógica em tarefas cotidianas. Para essa comprovação, foi desenvolvido um projeto na disciplina de Lógica de Programação em um curso técnico, onde o Scratch era utilizado como ferramenta de intermédio e obteve 74% de aceitação dos alunos envolvidos.

Já para Ferreira et al. (2021), que apresentam uma revisão literária em publicações da SBIE no período de 2001 a 2019, voltada para o ensino de programação na educação básica com uso da ferramenta Scratch, foram

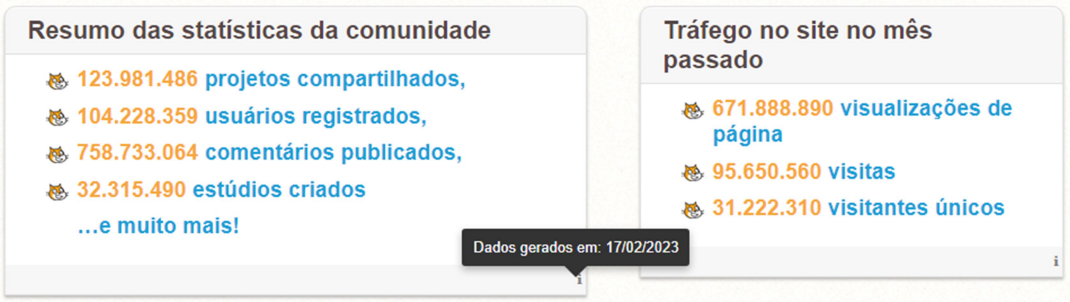
considerados exitosos os trabalhos nos quais a ferramenta consegue capacitar e auxiliar na resolução de problemas de uma forma diferente.

Viana et al. (2019) fizeram um comparativo de uso entre três ferramentas, o Scratch, Visualg e Robocode abordando o contexto de apoio a introdução da lógica e de algoritmos e o Scratch ficou em segundo lugar com uma pequena diferença em relação à ferramenta Visualg com melhor média. A pesquisa de Viana et al (2019) aponta que esse tipo de ferramenta pode ser um forte aliado no processo de ensino aprendizagem.

Torna-se válido informar, como reforço que motivou na escolha da Ferramenta Scratch como objeto de estudo, que com o passar do tempo ela se tornou algo bem maior que apenas um ambiente de programação, tornou-se uma comunidade que conta com milhões de usuários por todo o mundo, que criam seus projetos, compartilham e tem a possibilidade de editar estes projetos compartilhados de acordo com seus interesses. Todo esse ambiente de interação e suas infinitas possibilidades de criação podem ser benéficos se incorporados de forma correta ao processo.

Abaixo, segue uma estatística (Figura 1) retirada do site da ferramenta que mostra a quantidade de interações realizadas na plataforma até a data 18/02/2023.

Figura 1 - Troca de informações entre membros do Scratch.



Fonte: Site da ferramenta. (https://scratch.mit.edu/statistics/)

1.2 Objetivos

A pesquisa em questão tem como objetivo mostrar que a Ferramenta Scratch pode ser utilizada, em conjunto com outras linguagens de programação, na compreensão dos conceitos iniciais da Lógica de Programação de maneira lúdica, interativa e desprendida do processo de conhecimento de sintaxe.

Como retratado anteriormente, foram analisados estudos e levantamentos sobre a importância de se ter ferramentas didáticas incorporadas ao processo de ensino aprendizagem, aplicadas, em nosso caso, ao ensino dos conceitos iniciais da Lógica de Programação; grande parte desses estudos tiveram suas exposições de metodologias e resultados realizadas apenas em modo de texto, onde os autores detalhavam o que foi trabalhado na pesquisa e no que aquela ação resultou.

Dessa forma, o trabalho em questão propõe fazer um complemento dos artigos estudados dando uma experiência visual nesse processo de interação, mostrando na prática o detalhamento linha por linha de como ficaria a comparação do desenvolvimento entre duas linguagens de programação, o Scratch e Python.

A linguagem Python foi escolhida por ser utilizada na Disciplina de Algoritmos que engloba a Logica de Programação no curso de Ciência da Computação da

Universidade Estadual da Paraíba (UEPB), instituição cujo presente trabalho é vinculado.

2. CONHECENDO O SCRATCH

O Scratch é uma plataforma gratuita que utiliza a programação visual com blocos lógicos para criação de objetos virtuais, possui uma interface simples e de fácil compreensão permitindo aos usuários um ambiente interativo para resoluções de problemas, criação de jogos, histórias e animações digitais.

O funcionamento lembra a montagem de um quebra cabeça em que o usuário encaixa uma sequencia de blocos e assim consegue passar comandos para determinadas situações, possibilitando a resolução de desafios usando raciocínio lógico, a criatividade e o pensamento computacional de forma dinâmica, criativa e divertida. O Scratch também permite o compartilhamento dos projetos, promovendo uma ação colaborativa entre os usuários. Na figura 2, podemos observar no slogan tudo que foi dito acima resumido em três palavras que são a base do Scratch: Imagine, Programe e Compartilhe.

Figura 2 - Slogan do Scratch.

Fonte: Site da ferramenta. (https://scratch.mit.edu/)

Foi desenvolvido pelo grupo *Lifelong Kindergarten* criado e coordenado pelo pesquisador Mitchel Resnick e pertencente ao *Media Lab* do *Massachusetts Institute of Technology* (MIT), com a finalidade de ensinar lógica de programação para crianças e adolescentes com idades entre oito e dezesseis anos, mas que pode ser utilizado por pessoas de todas as faixas etárias.

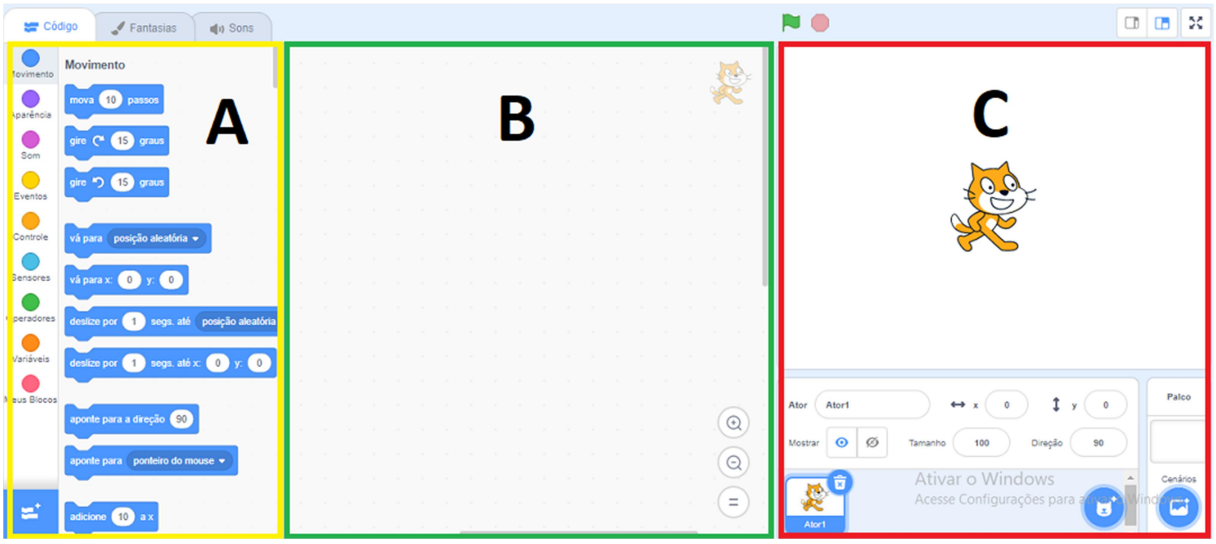
Sua interface pode ser dividida em três blocos, são eles:

A. **Área de comandos** – Onde ficam localizados os comandos a serem utilizados nos projetos e que é disposta a partir das funcionalidades (Movimento, Aparência, Som, Eventos, Controle, Sensores, Operadores, Variáveis e Meus Blocos).

B. **Área para disposição dos blocos** – Nesta local ficam os blocos arrastados e montados na ordem correta de execução. Podemos dizer que é o ambiente de programação.

C. **Área de palco** – Onde as simulações dos eventos programados serão executadas. Neste local o usuário vai ter a exata noção se o que ele programou está executando da maneira correta ou não.

Figura 3 - Interface do Scratch.



Fonte: Site da ferramenta. (https://scratch.mit.edu/projects/editor/?tutorial=getStarted)

Para um melhor entendimento no que a ferramenta é capaz de oferecer, também se faz necessário conhecer mesmo que superficialmente os componentes que fazem parte da área de comandos, representados na figura 03 na opção A, onde ficam disponibilizados os blocos que darão sentido a lógica do Scratch. São eles:

Blocos de Movimento – Estes blocos são responsáveis pela movimentação e posições dos atores no Palco, com eles podemos dizer para onde o ator deve se deslocar, se ele vai girar e determinar ação para quando ele tocar na borda do palco. Possui um total de 18 blocos.

Figura 4 - Scratch: Exemplo de blocos de movimento.

Fonte: Site da ferramenta. (https://scratch.mit.edu/projects/editor/?tutorial=getStarted)

Blocos de Aparência – Estes blocos são responsáveis pela parte de interação visual, tratam de tamanho, cor, aparecer ou desaparecer e falas. Possui um total de 20 blocos.

Figura 5 - Scratch: Exemplo de blocos de aparência.

Fonte: Site da ferramenta. (https://scratch.mit.edu/projects/editor/?tutorial=getStarted)

Blocos de Som – Estes blocos são responsáveis por atribuir sons aos eventos desejados, podendo utilizar os disponíveis na plataforma ou fazer o upload de algum som e até editá-lo dentro da plataforma. Possui um total de 9 blocos.

Figura 6 - Scratch: Exemplo de blocos de Som.

Fonte: Site da ferramenta. (https://scratch.mit.edu/projects/editor/?tutorial=getStarted)

Blocos de Eventos – São responsáveis por dar inicio ou determinar uma condição para que a execução seja iniciada. Possui um total de 8 blocos.

Figura 7 - Scratch: Exemplo de blocos de Eventos.

Fonte: Site da ferramenta. (https://scratch.mit.edu/projects/editor/?tutorial=getStarted)

Blocos de Controles – São blocos responsáveis por organizar a execução das ações de um determinado componente, são os blocos responsáveis pelas repetições e condicionais. Possui um total de 11 blocos.

Figura 8 - Scratch: Exemplo de blocos de Controle.

Fonte: Site da ferramenta. (https://scratch.mit.edu/projects/editor/?tutorial=getStarted)

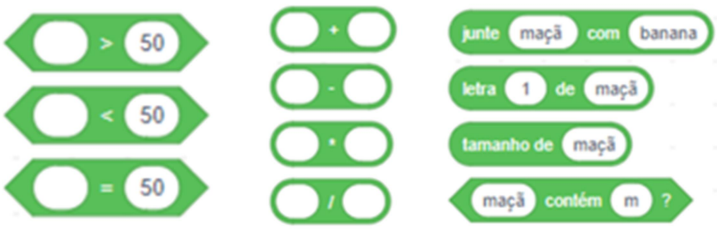
Bloco de Sensores – São blocos responsáveis por executar interações. Alguns destes blocos já indicam que possuem locais específicos para encaixe em outro blocos. Por exemplo: o bloco "*tocando na cor...?"* deve ser encaixado aos blocos de controle para determinar o que ocorrerá se o ator em questão tocar na cor escolhida. Possui 18 blocos.

Figura 9 - Scratch: Exemplo de blocos de Sensores.

Fonte: Site da ferramenta. (https://scratch.mit.edu/projects/editor/?tutorial=getStarted)

Bloco de Operadores – São responsáveis pelas funções matemáticas e lógicas no Scratch devendo ser associados com demais grupos. Podemos visualizar que os blocos responsáveis pelas interações matemáticas possuem as bordas arredondadas e os blocos lógicos possuem o formato de um losango. Possui 18 blocos.

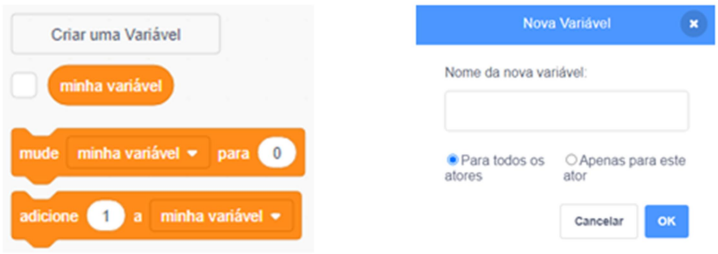
Figura 10 - Scratch: Exemplo de blocos de Operadores.



Fonte: Site da ferramenta. (https://scratch.mit.edu/projects/editor/?tutorial=getStarted)

Bloco de Variáveis – São blocos utilizado para armazenamento de valores e criação de variáveis. Possui 5 blocos por padrão, mas essa quantidade pode se indefinida, visto que o usuário pode criar a quantidade de variáveis que desejar.

Figura 11 - Scratch: Exemplo de blocos Variáveis



Fonte: Site da ferramenta. (https://scratch.mit.edu/projects/editor/?tutorial=getStarted)

Meus Blocos - Área destinada para criação de blocos específicos. Aqui o usuário pode formatar a criação de um bloco específico para sua necessidade caso os blocos padrões não consigam atende-lo. Possui por padrão apenas um botão que inicia a produção do bloco, mas essa quantidade pode se indefinida, visto que o usuário pode criar a quantidade de blocos que desejar.

Figura 12 - Scratch: Exemplo Meus Blocos.

Fonte: Site da ferramenta. (https://scratch.mit.edu/projects/editor/?tutorial=getStarted)

3. CONHECENDO A LINGUAGEM DE PROGRAMAÇÃO PYTHON

Python é uma linguagem de programação de alto nível, interpretada e orientada a objetos, muito conhecida por sua simplicidade e facilidade de uso, que a coloca como uma linguagem muito acessível para iniciantes. Ela é de alto nível por possuir uma linguagem mais “próxima” ao entendimento humano, diferente da linguagem de máquina. Ela é interpretada porque não precisa ser compilada para ser executada, basta o código ser processado por um interpretador que o traduz em tempo de execução para a linguagem de maquina entendida pelo computador. Ela é orientada a objetos porque permite que seus usuários organizem as informações e funções em objetos, tornando a programação mais fácil de entender.

Foi criada em 1991 pelo holandês Guido van Rossun e atualmente é uma linguagem muito popular. Apesar de sua simplicidade, o Python pode ser empregada

em rotinas bem robustas e utilizada para desenvolvimento web, análise de dados, desenvolvimento de programas entre outras. Possui como principais características:

- Uma sintaxe mais simples, limpa e intuitiva, tornando o código mais leve para leitura e para ser mantido.

- Permite expressar conceitos em menos linhas de código quando comparada com as outras linguagens de programação.

- Um formato de declaração de variáveis que não necessita especificar o tipo antes do uso, chamada de tipagem dinâmica.

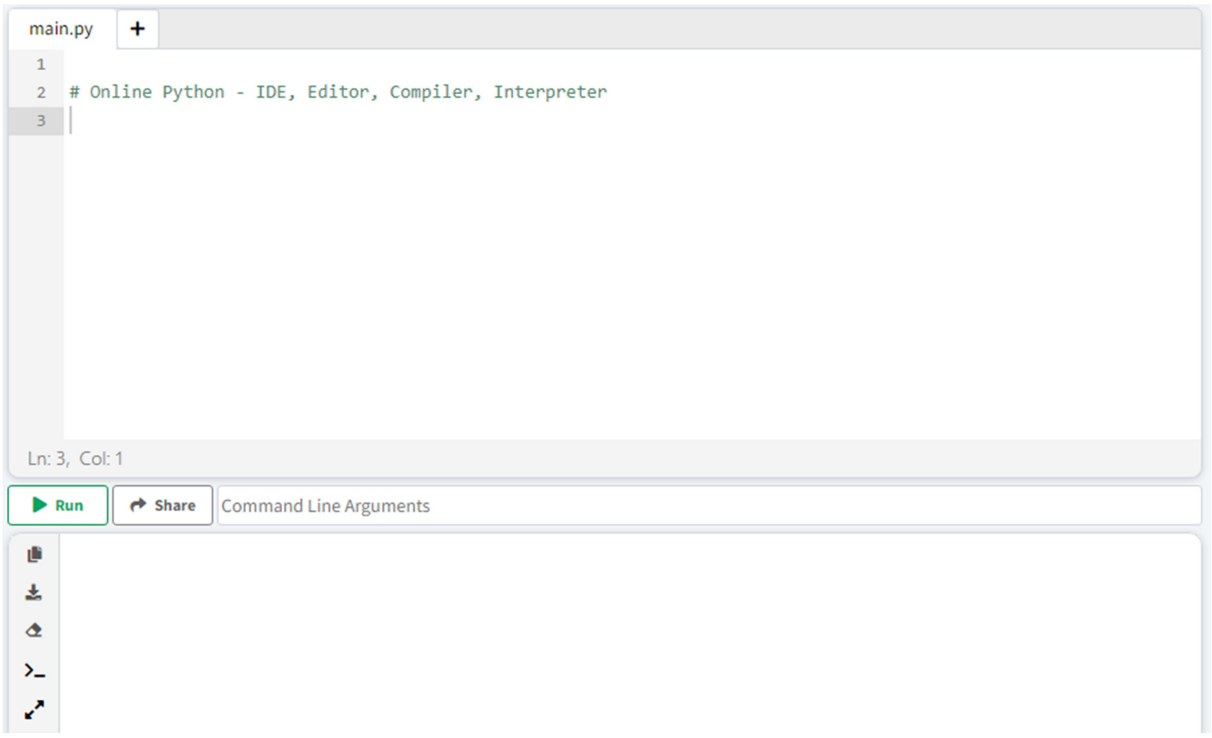
- Como é uma linguagem interpretada, Python é uma linguagem multiplataforma, pois sua execução não depende de compilação, apenas da interpretação.

- Comunidade ativa, onde os usuários mantém uma vasta quantidade de recursos e documentações, tornando mais fácil encontrar ajuda e informações para resolver problemas.

Figura 13 - Identificação visual para a linguagem de programação Python.

Fonte: Site da fundação. (https://www.python.org/community/logos/)

Figura 14 - IDE para desenvolvimento em Python utilizada no trabalho.



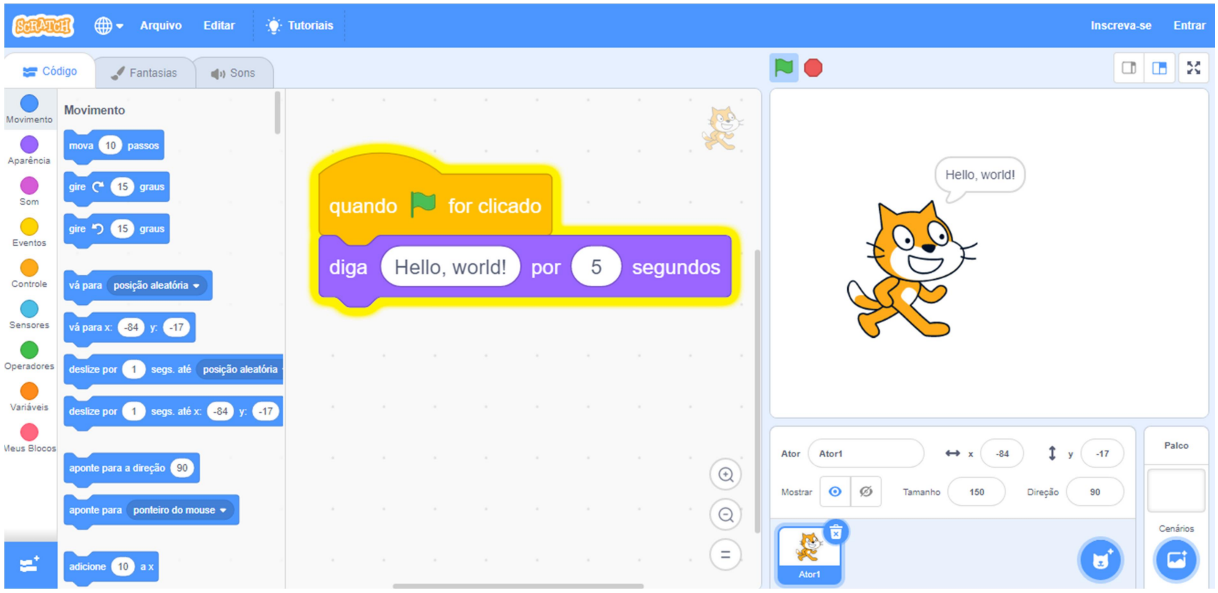
Fonte: https://www.online-python.com/

4. PROGRAMANDO E COMPARANDO O SCRATCH COM O PYTHON

O Scratch foi desenvolvido para que programar seja tão fácil que até uma criança possa fazer, basta encaixar os blocos corretos para que o ator no palco execute o que foi definido. Mesmo que o processo seja fácil de fazer, é necessário que o aluno planeje todos os passos, ou seja, ordene suas ideias antes, pois assim quando ele for para o ambiente de comando já terá em mente o começo, meio e fim da execução.

O ambiente de programação do Scratch é encontrado no site da comunidade https://scratch.mit.edu/projects/editor/?tutorial=getStarted, que também disponibiliza uma infinidade de materiais, fóruns, orientações, modelos, entre outras ajudas que facilitam o entendimento e o acesso à ferramenta. Para a execução dos exemplos no Python, utilizamos uma IDE online no endereço https://www.online-python.com/.

Figura 15 - Exemplo simples de escrita com retorno no Scratch.



Fonte: Estudo do autor.

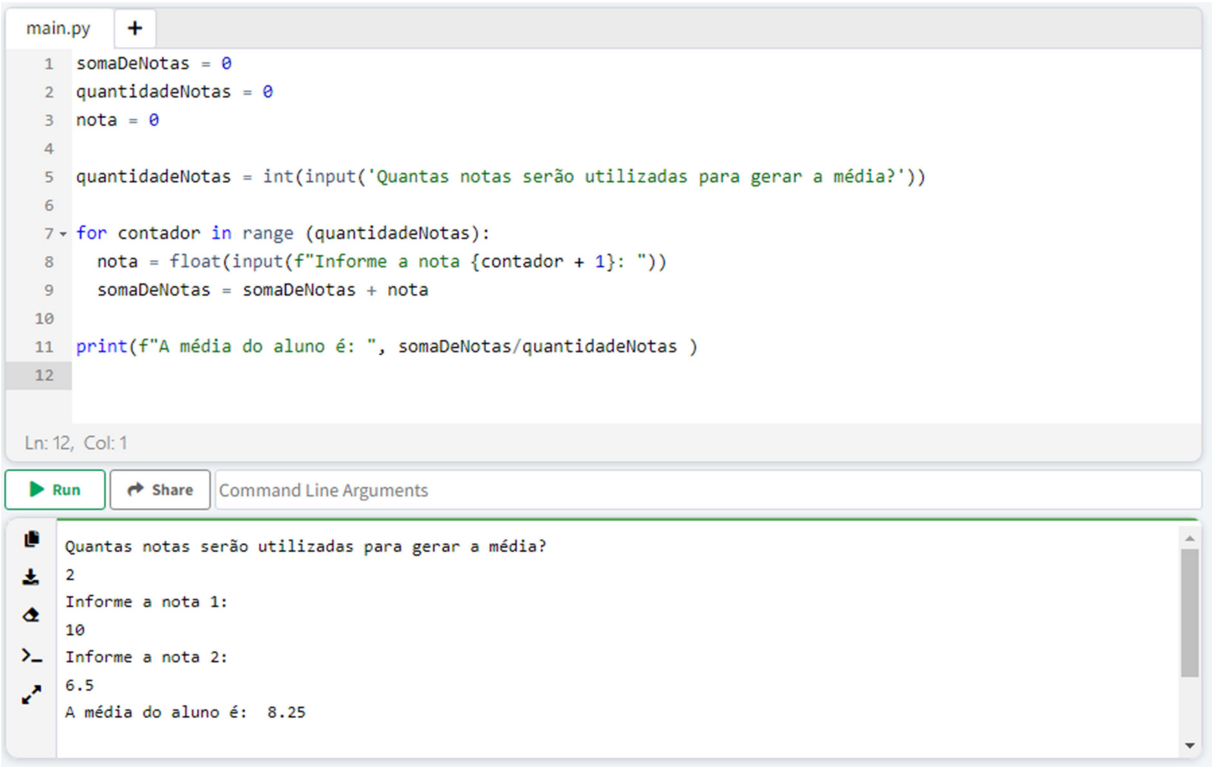
Figura 16 - Exemplo simples de escrita com retorno no Python.

Fonte: Estudo do autor.

Como o intuito do trabalho é reforçar que a utilização de uma ferramenta mais interativa e visualmente mais agradável pode ajudar na abstração dos conceitos e auxiliar no processo de ensino aprendizagem, faremos daqui pra frente comparações com alguns exemplos simples resolvidos no Python e no Scratch, trazendo para temática do estudo literalmente o passo a passo visual da resolução de cada problemática, mostrando através de imagens como ficou representada cada instrução. Para tomada de amostragem trabalharemos com o conceito de algumas Estruturas de Controle de Fluxo de Execução.

No exemplo 1, a problemática se dar em desenvolver um código onde seja retornada a média de um aluno. Para isso, será necessário informar quantas notas farão parte da média, esta informação será repassada via IDE juntamente com as notas do aluno. Neste exemplo, vamos conhecer a estrutura de repetição PARA (FOR), que é utilizada quando sabemos o número exato de vezes que a instrução será repetida. Na figura 17, temos o código desenvolvido em Python e na figura 18 no Scratch.

Figura 17 - Exemplo 1 desenvolvido no Python. Entendendo linha por linha.



Fonte: Estudo do autor.

Linha 01 – Foi declarada a variável "*somaDeNotas*" e definido inicialmente o valor zero. Esta variável irá receber a soma das notas que o usuário irá repassar, ou seja, o seu valor vai ser alterado durante a execução do programa. Notem que não se faz necessário em Python especificar o tipo de dados que a variável irá receber.

Linha 02 – Foi declarada a constante "*quantidadeNotas*" e definido inicialmente o valor zero. Esta constante irá receber a quantidade de notas que irá fazer parte da formação da média e após ser definida não irá mais ser alterada durante a execução do programa.

Linha 03 – Foi declarada a variável "*nota*" e definido inicialmente o valor zero. Esta variável irá receber as notas do aluno para ter a média calculada, ou seja, o seu valor vai ser alterado durante a execução do programa.

Linha 05 – A variável "*quantidadeNotas*" vai receber a quantidade de notas que serão utilizadas para formação da média. Essa informação é digitada pelo usuário.

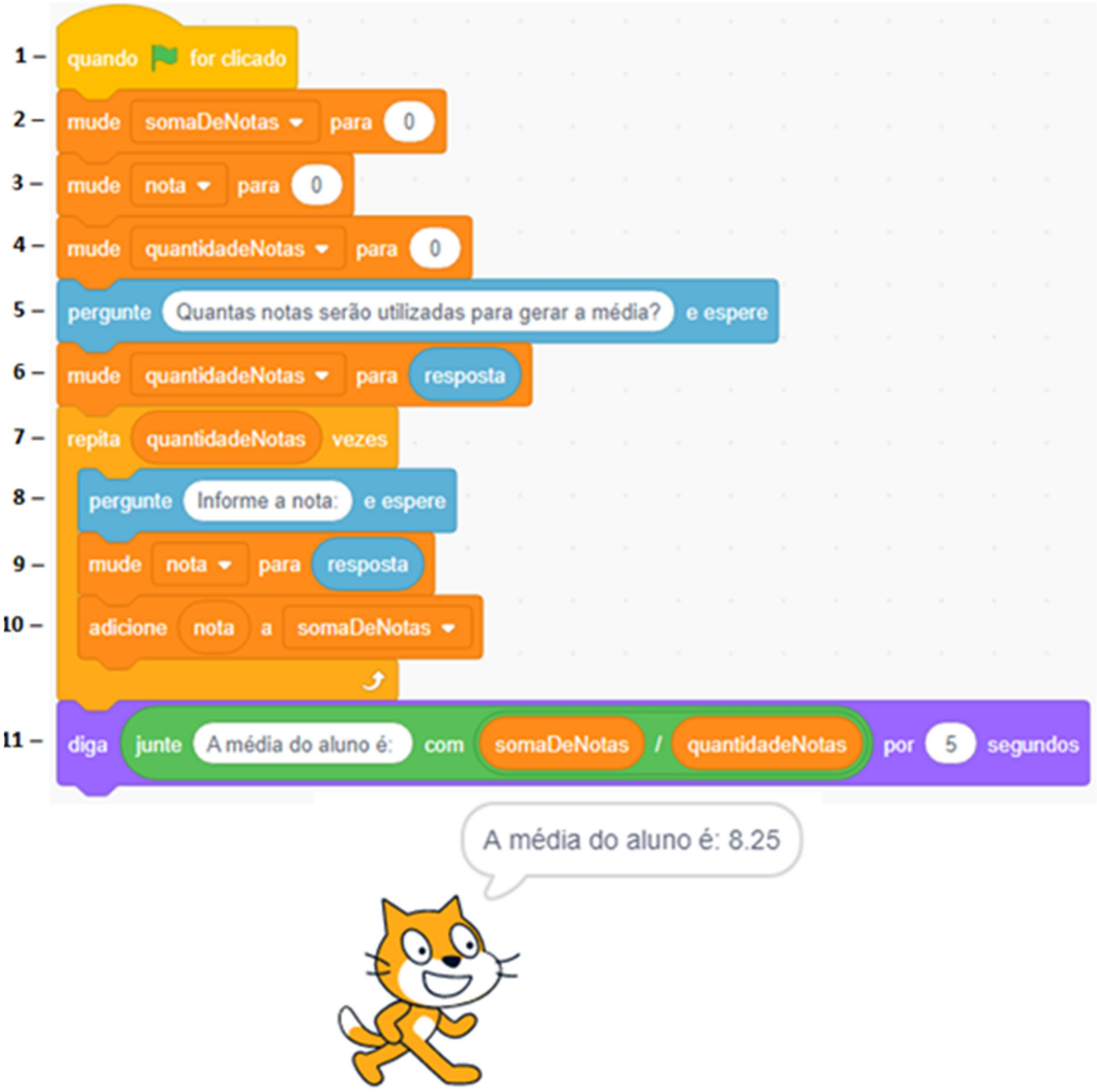
Linha 07 – Entra na estrutura de repetição FOR e vai executar aquela instrução até que a quantidade de notas informadas seja igual ao valor armazenado na constante "*quantidadeNotas*".

Linha 08 – A variável "*nota*" recebe o valor informado pelo usuário de cada nota.

Linha 09 – A variável "*somaDeNotas*" adiciona o valor da variável "*nota*" ao valor que já estava nela.

Linha 11 – Imprime em tela uma mensagem concatenada com o cálculo da média, que no nosso exemplo seria o valor da variável "*somaDeNotas*" dividida pelo valor da constante "*quantidadeNotas*".

Figura 18 - Exemplo 1 desenvolvido no Scratch. Entendendo linha por linha.



Fonte: Estudo do autor.

Linha 01 – Utiliza um *bloco de eventos* para iniciar a execução do código. Inicia o programa quando o usuário clica na bandeira verde. Pode ser visualizada na figura 03 acima do bloco C.

Linha 02 – Utiliza um *bloco de variáveis* para criar uma variável. Foi criada a variável "*somaDeNotas*" e está recebendo o valor zero. Essa variável vai receber durante a execução a soma das notas digitadas.

Linha 03 – Utiliza um *bloco de variáveis* para criar uma variável. Foi criada a variável "*nota*" e está recebendo o valor zero. Essa variável vai receber durante a execução a nota informada pelo usuário;

Linha 04 – Utiliza um *bloco de variáveis* para criar uma variável. Foi criada a variável "*quantidadeNotas*" e está recebendo o valor zero. Essa variável vai receber durante a execução a quantidade de notas que serão utilizadas para realizar a média geral. No Scratch o termo constante não é abordado, mas pela definição como essa variável irá receber um valor e não será mais alterado, configura-se como uma constante.

Linha 05 – Utiliza um *bloco de sensores* para definir uma interação com o usuário. Pergunta ao usuário quantas notas serão utilizadas para gerar a média e espera até que o usuário informe.

Linha 06 – Utiliza um *bloco de variáveis* para alterar o valor de uma variável. Vai definir a variável "*quantidadeNotas*" para o valor informado na linha 05, que foi a resposta do usuário.

Linha 07 – Utiliza um *bloco de controle* para executar um determinado número de vezes uma instrução. Entra na estrutura de repetição e vai executar aquela instrução até que a quantidade de notas informada seja igual ao valor armazenado na variável "*quantidadeNotas*", ou seja, nós sabemos quantas vezes ela vai executar, pois o usuário passou essa informação no inicio da execução.

Linha 08 – Utiliza um *bloco de sensores* para definir uma interação com o usuário. Solicita que usuário passe o valor de uma nota.

Linha 09 – Utiliza um *bloco de variáveis* para alterar o valor de uma variável. Altera o valor da variável "*nota*" para resposta do usuário.

Linha 10 – Utiliza um *bloco de variáveis* para alterar o valor de uma variável. Soma o valor da variável "*nota*" ao valor existente na variável "*somaDeNotas*".

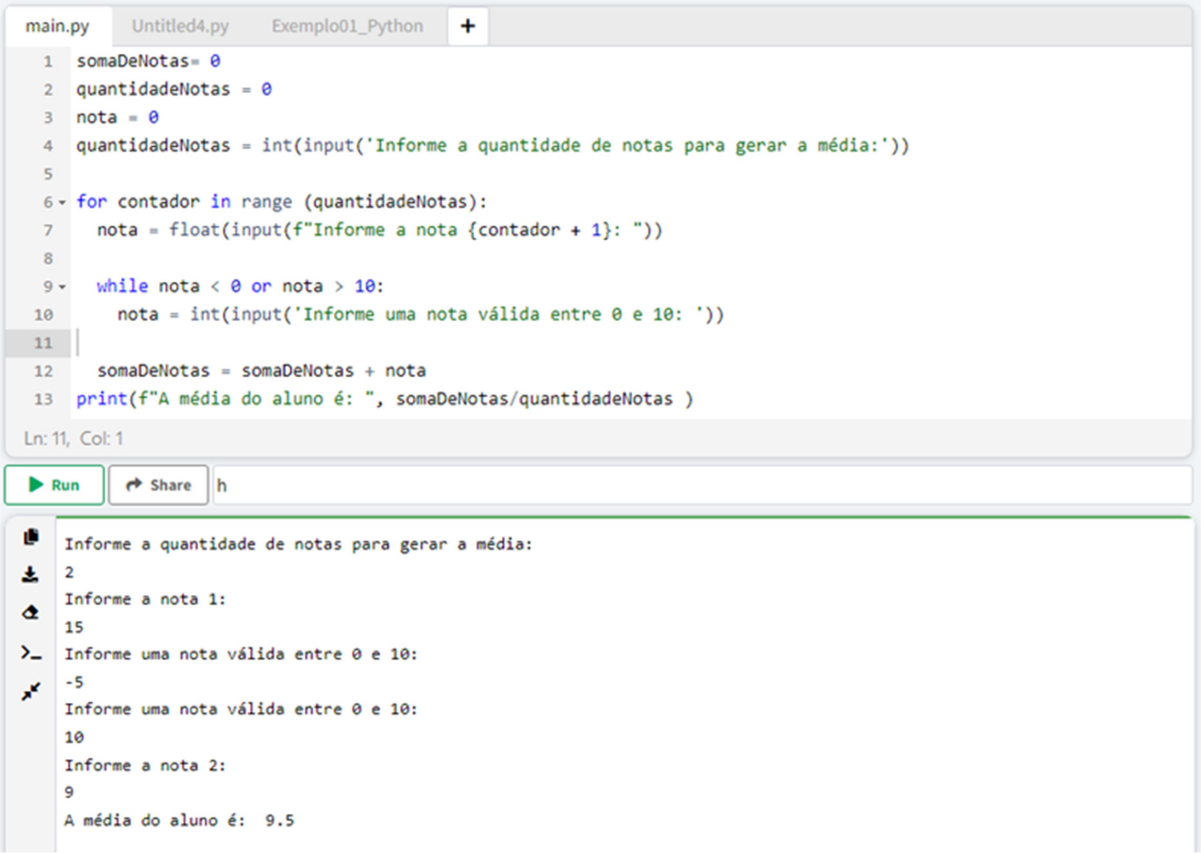
Linha 11 – Utiliza um *bloco de aparência* para passar o resultado desejado. Nessa linha será transmitida a mensagem da média concatenada com o cálculo da média, que no nosso exemplo seria o valor da variável "*somaDeNotas*" dividida pelo valor da variável *"quantidadeNotas"*, isto ficará em tela por 5 segundos e a execução para de rodar. No Scratch não conseguimos ter uma sequencia impressa da execução, como podemos visualizar nos exemplos do Python.

No exemplo 2, utilizaremos o primeiro exemplo como base, mas iremos validar as notas repassadas que só serão utilizadas se satisfazerem as condições definidas, ou seja, o sistema irá repetir essa instrução enquanto a informação repassada for verdadeira.

A estrutura de repetição ENQUANTO é utilizada para testar uma condição inicialmente, e se o resultado for verdadeiro, o bloco de instrução será executado.

Na figura 19, temos o código desenvolvido em Python e na figura 20 no Scratch.

Figura 19 - Exemplo 2 desenvolvido no Python. Entendendo linha por linha.



Fonte: Estudo do autor.

Linha 01 – Foi declarada a variável "*somaDeNotas*" e definido inicialmente o valor zero. Esta variável irá receber a soma das notas que o usuário irá repassar, ou seja, o seu valor vai ser alterado durante a execução do programa. Notem que não se faz necessário em Python especificar o tipo de dados que a variável irá receber.

Linha 02 – Foi declarada a constante "*quantidadeNotas*" e definido inicialmente o valor zero. Esta constante irá receber a quantidade de notas que irá fazer parte da formação da média e após ser definida não irá mais ser alterada durante a execução do programa.

Linha 03 – Foi declarada a variável "*nota*" e definido inicialmente o valor zero. Esta variável irá receber as notas do aluno para ter a média calculada, ou seja, o seu valor vai ser alterado durante a execução do programa.

Linha 04 – A variável "*quantidadeNotas*" vai receber a quantidade de notas que serão utilizadas para formação da média. Essa informação é digitada pelo usuário.

Linha 06 – Entra na estrutura de repetição FOR e vai executar aquela instrução até que a quantidade de notas informadas seja igual ao valor armazenado na constante "*quantidadeNotas*".

Linha 07 – A variável "*nota*" recebe o valor informado pelo usuário de cada nota.

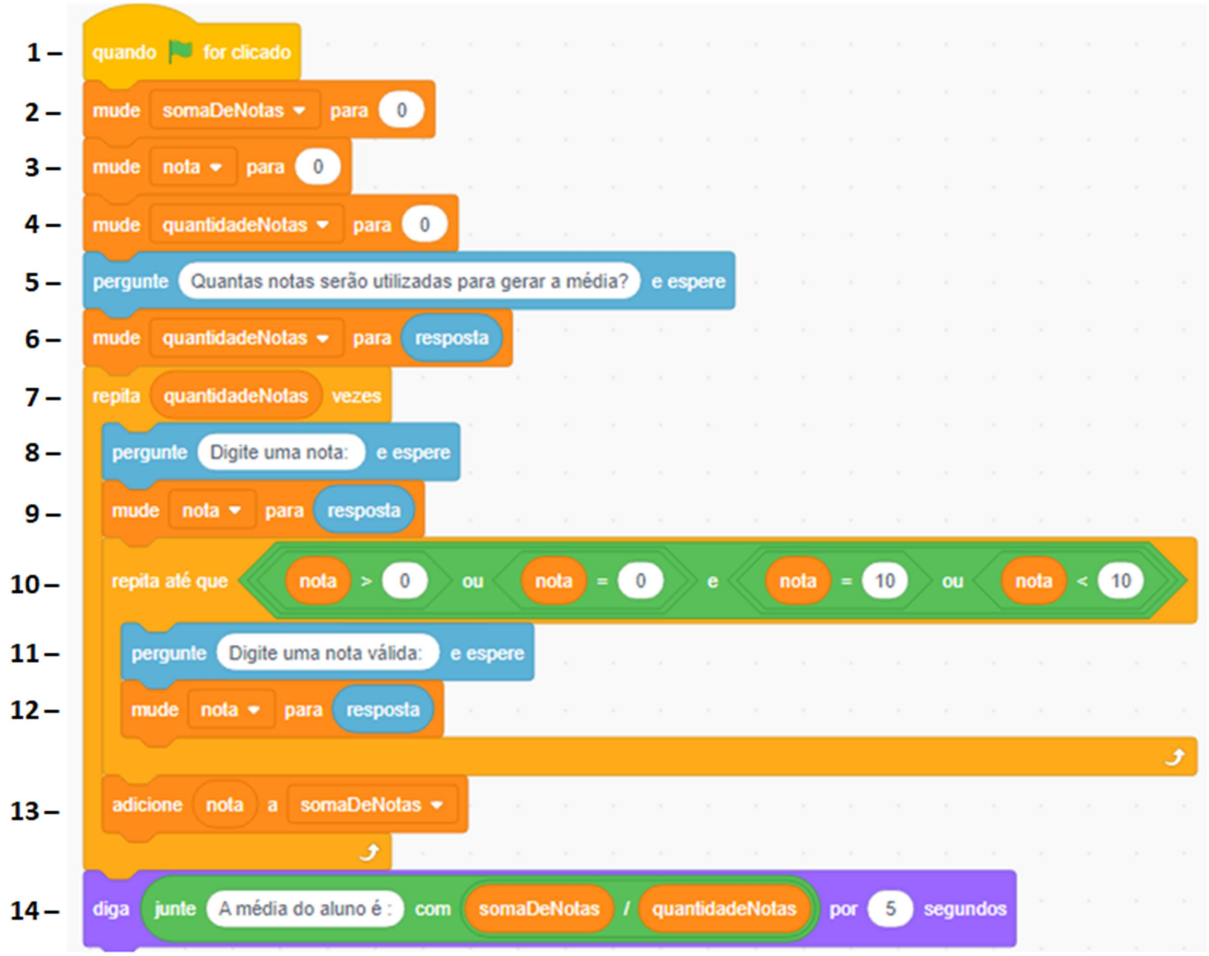
Linha 09 – Se as condições da nota inserida estiverem dentro dos parâmetros informados, a execução passa para a linha 12, caso contrário passará para linha 10. Para delimitar as condições para a nota informada pelo usuário, neste exemplo temos de forma simplificada a condição para a nota < 0 ou Nota > 10, ou seja, só será verdadeira quando estiver fora desse intervalo, entrando assim na estrutura do WHILE.

LINHA 10 – Solicita ao usuário uma nota válida, pois só será executada se a nota informada estiver fora do intervalo delimitado na linha 09.

Linha 12 – A variável "*somaDeNotas*" adiciona o valor da variável "*nota*" ao valor que já estava nela.

Linha 13 – Imprime em tela uma mensagem concatenada com o cálculo da média, que no nosso exemplo seria o valor da variável "*somaDeNotas*" dividida pelo valor da constante "*quantidadeNotas*".

Figura 20 - Exemplo 2 desenvolvido no Scratch. Entendendo linha por linha.



Fonte: Estudo do autor.

Linha 01 – Utiliza um *bloco de eventos* para iniciar a execução do código. Inicia o programa quando o usuário clica na bandeira verde. Pode ser visualizada na figura 03 acima do bloco C.

Linha 02 – Utiliza um *bloco de variáveis* para criar uma variável. Foi criada a variável "*somaDeNotas*" e está recebendo o valor zero. Essa variável vai receber durante a execução a soma das notas digitadas.

Linha 03 – Utiliza um *bloco de variáveis* para criar uma variável. Foi criada a variável "*nota*" e está recebendo o valor zero. Essa variável vai receber durante a execução a nota informada pelo usuário;

Linha 04 – Utiliza um *bloco de variáveis* para criar uma variável. Foi criada a variável "*quantidadeNotas*" e está recebendo o valor zero. Essa variável vai receber durante a execução a quantidade de notas que serão utilizadas para realizar a média geral. No Scratch o termo constante não é abordado, mas pela definição como essa variável irá receber um valor e não será mais alterado, configura-se como uma constante.

Linha 05 – Utiliza um *bloco de sensores* para definir uma interação com o usuário. Pergunta ao usuário quantas notas serão utilizadas para gerar a média e espera até que o usuário informe.

Linha 06 – Utiliza um *bloco de variáveis* para alterar o valor de uma variável. Vai definir a variável "*quantidadeNotas*" para o valor informado na linha 05, que foi a resposta do usuário.

Linha 07 – Utiliza um *bloco de controle* para executar um determinado número de vezes uma instrução. Entra na estrutura de repetição e vai executar aquela instrução até que a quantidade de notas informada seja igual ao valor armazenado na variável "*quantidadeNotas*", ou seja, nós sabemos quantas vezes ela vai executar, pois o usuário passou essa informação no inicio da execução.

Linha 08 – Utiliza um *bloco de sensores* para definir uma interação com o usuário. Solicita que usuário passe o valor de uma nota.

Linha 09 – Utiliza um *bloco de variáveis* para alterar o valor de uma variável. Altera o valor da variável "*nota*" para resposta do usuário.

Linha 10 – Utiliza um *bloco de controle* para executar uma instrução, onde a nota vai precisar ser repetida enquanto ela for menor que zero ou maior que 10. Se a nota inserida estiver dentro dos parâmetros informados, a execução passa para a linha 13, caso contrário passará para as linhas 11 e 12, e pela validação novamente. No Scratch não existe os operadores lógicos maior ou igual (>=) e menor ou igual (<=).

Linha 11 – Só será executada caso a nota informada seja menor que 0 ou maior que 10. Utiliza um *bloco de sensores* para definir uma interação com o usuário. Solicita ao usuário uma nota válida e espera.

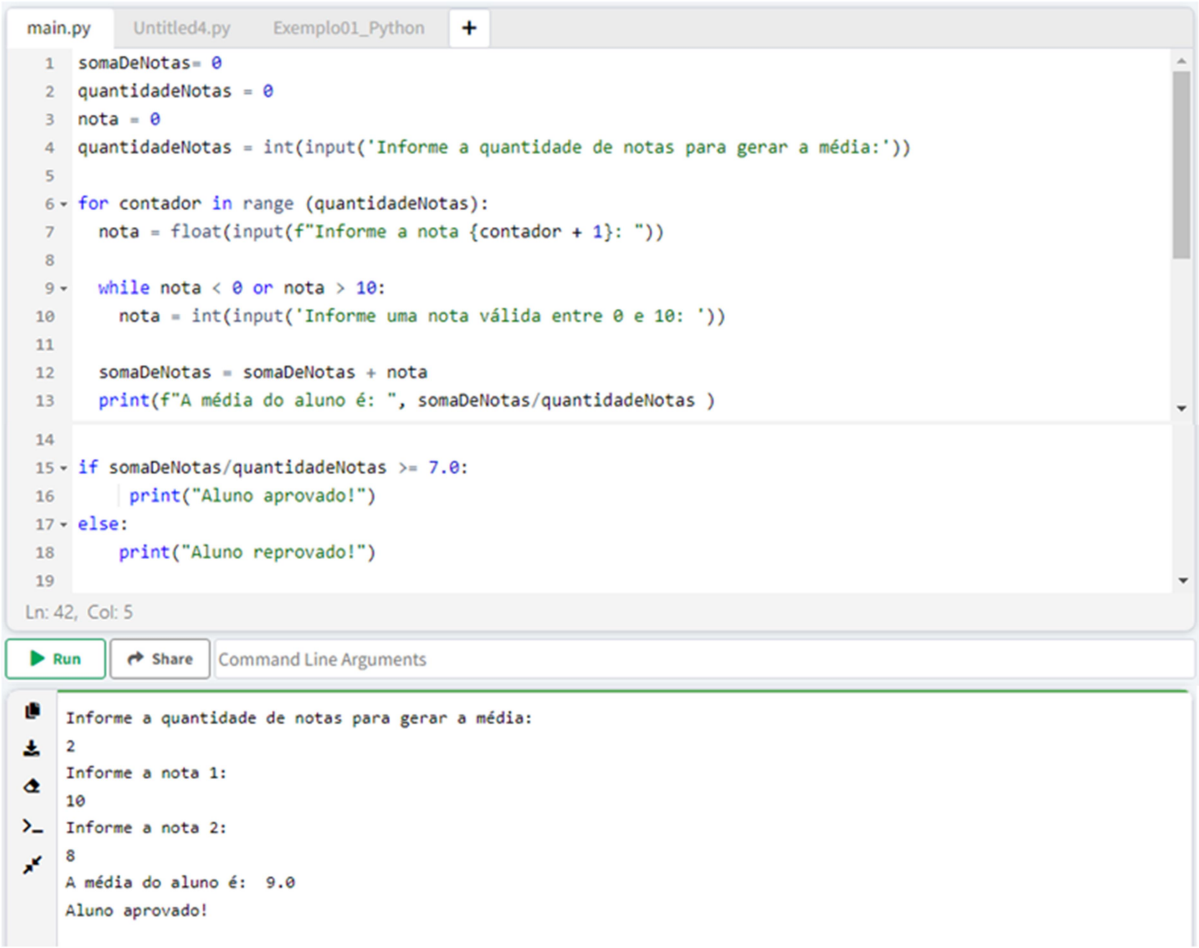
Linha 12 – Utiliza um *bloco de variáveis* para alterar o valor de uma variável. Vai definir a variável *“nota”* para o valor informado na linha 11, que foi a resposta do usuário.

Linha 13 – Utiliza um *bloco de variáveis* para alterar o valor de uma variável. Soma o valor da variável *“nota”* ao valor existente na variável *“somaDeNotas”.*

Linha 14 – Utiliza um *bloco de aparência* para passar o resultado desejado. Nessa linha será transmitida a mensagem da média concatenada com o cálculo da média, que no nosso exemplo seria o valor da variável *“somaDeNotas”* dividida pelo valor da variável *“quantidadeNotas”*, isto ficará em tela por 5 segundos e a execução para de rodar. No Scratch não conseguimos ter uma sequencia impressa da execução, como podemos visualizar nos exemplos do Python.

No exemplo 3, utilizaremos o segundo exemplo como base e iremos validar a média do aluno, informando se ele está “Aprovado” ou “Reprovado”, tomando como parâmetro se a média for maior igual a 7.0, o programa imprime “Aluno Aprovado!”, caso contrário imprime “Aluno Reprovado!”. Para isso, vamos utilizar a estrutura condicional IF- ELSE. Na figura 21, temos o código desenvolvido em Python e na figura 22 no Scratch.

Figura 21 - Exemplo 3 desenvolvido no Python. Entendendo linha por linha. Como o exemplo executa basicamente todo o código do exemplo 2, iremos apenas detalhar as linhas referentes à condicional.



Fonte: Estudo do autor.

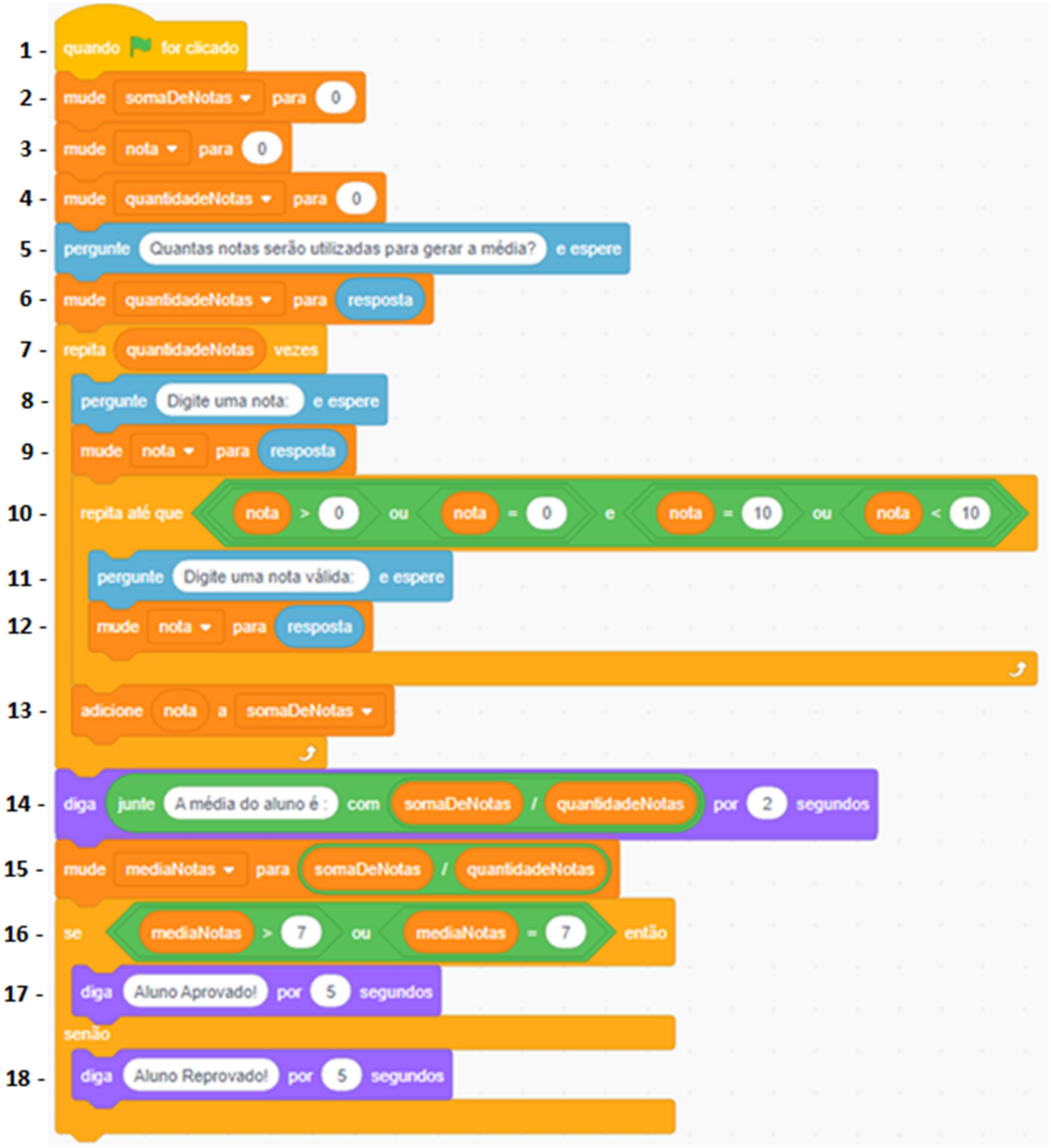
Linha 15 – Entra na estrutura condicional e verifica se a média é maior ou igual a 7.0. Com o cálculo da média, que no nosso exemplo seria o valor da variável "*somaDeNotas*" dividida pelo valor da constante "*quantidadeNotas*".

Linha 16 – Imprime na tela a mensagem de "Aluno Aprovado!", caso a média do aluno seja maior ou igual a 7.0.

Linha 17 – Sintaxe da linguagem para caso a primeira condição não seja aceita.

Linha 18 – Imprime na tela a mensagem de "Aluno Reprovado!", caso a média do aluno seja menor que 7.0.

Figura 22 - Exemplo 3 desenvolvido no Scratch. Entendendo linha por linha. Como o exemplo executa basicamente todo o código do exemplo 2, iremos apenas detalhar as linhas referentes à condicional.



Fonte: Estudo do autor.

Linha 15 – Foi criada uma variável chamada *"mediaNotas"*, essa variável vai receber o resultado do cálculo da média, que no nosso exemplo seria o valor da variável "*somaDeNotas*" dividida pelo valor da constante "*quantidadeNotas*". Entra na estrutura condicional e verifica se a média é maior ou igual a 7.0. No Scratch não existe os operadores lógicos maior ou igual (>=) e menor ou igual (<=).

Linha 16 – Imprime na tela a mensagem de "Aluno Aprovado!", caso a média do aluno seja maior ou igual a 7.0.

Linha 17 – Sintaxe da linguagem para caso a primeira condição não seja aceita.

Linha 18 – Imprime na tela a mensagem de “Aluno Reprovado!”, caso a média do aluno seja menor que 7.0.

5. CONSIDERAÇÕES FINAIS

A pesquisa apresentou uma problemática referente à dificuldade encontrada por novos alunos em relação a conceitos considerados abstratos no estudo da Lógica de Programação. Foi possível observar pela metodologia abordada, que a utilização de uma ferramenta alternativa pode ser uma boa ideia para os conteúdos introdutórios, tratando as abstrações de uma forma mais lúdica e tangível, fazendo a transição para outra linguagem suavemente sem grandes impactos. A ideia é de que uma linguagem possa complementar a outra e não substituir.

Com o Scratch a introdução dos conceitos podem se tornar mais visuais e sem a necessidade de explanações longas preparando o aluno para a entrada da linguagem de programação propriamente dita, que no caso da UEPB seria o Python.

Diferentemente do que ocorre com outras linguagens de programação, no Scratch não se faz necessário o uso de sintaxe propriamente dita, deixando a atenção inicial apenas para a resolução dos problemas e para alguns o fato da plataforma disponibilizar o uso em português, acaba deixando a experiência mais positiva nesse primeiro contato.

A abordagem utilizada com a resolução de exemplos linha por linha e com comparações por imagens da programação realizada se diferencia de outros estudos analisados que acabavam passando apenas as impressões de questionários ou detalhamento de metodologias, a intenção foi realmente mostrar como acontece em cada uma e que o leitor pudesse ter a noção de como é realizado o processo de programação dentro do ambiente Scratch.

REFERÊNCIAS

BAPTISTA, João Álvaro de Sousa. Programação com Scratch: Desenvolvendo Raciocínio Lógico. Instituto Nacional de Matemática Pura e Aplicada (IMPA). 13/07/2017.

COUTINHO, Emmanuel Ferreira; LIMA, Ernesto de; SANTOS, Clemilson Costa. Revista Tecnologia na Educação: Um panorama sobre o desempenho de uma disciplina inicial de programação em um curso de graduação. Ano 9 / Volume 19. 2017.

FERREIRA, Luisa S.; SANTOS, Sylvana Karla S. L.; BONFIM, Cristiane Jorge L.. Pensamento Computacional e Programação Scratch: uma revisão de literatura do SBIE. *In*: ENCONTRO NACIONAL DE COMPUTAÇÃO DOS INSTITUTOS FEDERAIS (ENCOMPIF), 8. , 2021, Evento Online. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2021. p. 5-8. ISSN 2763-8766.
DOI: https://doi.org/10.5753/encompif.2021.15942.

FORBELLONE, André Luiz Villar; EBERSPACHER, Henri Frederico. Lógica de Programação: A construção de algoritmos e estruturas de dados. 3ª Edição. Editora Pearson Prentice Hall, 2005.

LÓGICA. In: DICIO, Dicionário Online de Português. Disponível em: https://www.dicio.com.br/logica/ Acesso em: 11/01/2023.

LÓGICA. In: MICHAELIS, Dicionário Online da Língua Portuguesa. Disponível em: https://michaelis.uol.com.br/busca?r=0&f=0&t=0&palavra=l%C3%B3gica. Acesso em: 11/01/2023.

PEIXOTO, Macilon Arruda. Uma Revisão Bibliográfica Acerca do uso de Ferramentas Didáticas no Ensino de Lógica de Programação. 2022.

PUGA, Sandra; RISSETTI Gerson. Lógica de programação e estruturas de dados com aplicações em Java. 2ª edição. Editora Pearson Prentice Hall, 2008.

RAFALSKI, Jadson do Prado; SANTOS, Otavio Lube dos. Uma experiência com a Linguagem Scratch no Ensino de Programação com Alunos do Curso de Engenharia Elétrica. Anais do Workshop de Informática na Escola, [S.l.], p. 612-620, nov. 2016. ISSN 2316-6541. Disponível em:
http://ojs.sector3.com.br/index.php/wie/article/view/6868/4746. Acesso em: 13 fev. 2023. DOI: http://dx.doi.org/10.5753/cbie.wie.2016.612.

SANTOS, Antunes; GORGÔNIO, Arthur; LUCENA, Amarildo; GORGÔNIO, Flavius. A Importância do Fator Motivacional no Processo Ensino-Aprendizagem de Algoritmos e Lógica de Programação para Alunos Repetentes. *In*: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO (WEI), 23. , 2015, Recife. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2015. p. 168-177. DOI: https://doi.org/10.5753/wei.2015.10233. Acesso 15/01/2023.

SILVA, Dário Gean da; SALES, Reinaldo Eduardo da Silva; AMORIM, Franciel da Silva. O Ensino de Lógica de programação por meio da gamificação. Instituto Federal do Pará (IFPA), Editora Científica Digital, p. 404-421, 08/12/2020.

SILVA, F. dos S.; ALMEIDA, A. C. F. de; GODOI E SILVA, K. A. O Desenvolvimento do pensamento computacional com a integração do Software Scratch no Ensino Superior. Revista Observatório, *[S. l.]*, v. 5, n. 1, p. 276–298, 2019. DOI: 10.20873/uft.2447-4266.2019v5n1p276. Disponível em: https://sistemas.uft.edu.br/periodicos/index.php/observatorio/article/view/4740. Acesso em: 02 fev. 2023.

Site do Scratch. Disponível em https://scratch.mit.edu/. Acessado em 13/01/2023.

VALENTE, J. A.; ALMEIDA, E. B. DE; GERALDINI, A. F. S. Metodologias ativas: das concepções às práticas em distintos níveis de ensino. Revista Diálogo Educacional, v. 17, n. 52, p. 455–478, 2017.

VENTURA, L. M.; BIANCHINI, L. G. B.; KIRNEW, L. C. P. Scratch e a possibilidade de novos sentidos sobre o ensino da Lógica de Programação. Educitec - Revista de Estudos e Pesquisas sobre Ensino Tecnológico, Manaus, Brasil, v. 5, n. 11, 2019. DOI: 10.31417/educitec. v5i11.702. Disponível em: https://sistemascmc.ifam.edu.br/educitec/index.php/educitec/article/view/702. Acesso em: 01/02/ 2023.

VIANA, G. A.; PORTELA, C. dos S. O Uso de Softwares Educativos para Introdução de Lógica de Programação no Ensino de Base e Superior. Informática na educação: teoria e prática, Porto Alegre, v. 22, n. 1, 2019. DOI: 10.22456/1982-1654.86079. Disponível em: https://seer.ufrgs.br/index.php/InfEducTeoriaPratica/article/view/86079. Acesso em: 29 jan. 2023.