ReTEM
REVISTA TOCANTINENSE DE EDUCAÇÃO MATEMÁTICA

USO DE PYTHON NO ENSINO DE MATEMÁTICA: PYGGB E MANIM

USE OF PYTHON IN MATHEMATICS EDUCATION: PYGGB AND MANIM

USO DE PYTHON EN LA ENSEÑANZA DE LAS MATEMÁTICAS: PYGGB Y MANIM

Luis Andrés Castillo[*] Ivonne C. Sánchez[**]

RESUMO

Este trabalho explora as possibilidades de utilização do Python no ensino de Matemática, com foco nas ferramentas PyGgb e Manim. O PyGgb é um ambiente online que permite programar com objetos do GeoGebra e visualizar os resultados em uma interface gráfica, enquanto o Manim é uma biblioteca de Python voltada para a criação de animações matemáticas. O uso dessas ferramentas pode inovar a prática pedagógica, proporcionando maior interação visual e compreensão de conceitos complexos. No entanto, o acesso a materiais em português é limitado. Propomos o projeto AM^2: Animações Matemáticas com Manim, que visa oferecer formação para professores, criar recursos educacionais e facilitar o uso de tecnologias digitais no ensino de Matemática.

Palavras-chave: Python. PyGgb. Manim. Ensino. Matemática.

ABSTRACT

This paper explores the potential of Python in Mathematics education, focusing on the PyGgb and Manim tools. PyGgb is an online platform that allows programming with GeoGebra objects and visualizing results in a graphical interface, while Manim is a Python library for creating mathematical animations. These tools can innovate teaching practices by providing greater visual interaction and understanding of complex concepts. However, Portuguese-language materials are limited. We propose the AM^2 project: Mathematical Animations with Manim, aiming to offer teacher training, create educational resources, and facilitate the use of digital technologies in Mathematics teaching.

Keywords: Python. PyGgb. Manim. Teaching. Mathematics.

RESUMEN

Este trabajo explora las posibilidades de uso de Python en la enseñanza de Matemáticas, enfocándose en las herramientas PyGgb y Manim. PyGgb es una plataforma en línea que permite programar con objetos de GeoGebra y visualizar los resultados en una interfaz gráfica, mientras que Manim es una biblioteca de Python para la creación de animaciones matemáticas. Estas herramientas pueden innovar

[*] Mestre em Educação em Ciências e Matemáticas pela Universidade Federal do Pará (UFPA). Doutorando no Programa de Pós-Graduação em Educação em Ciências e Matemáticas da Universidade Federal do Pará (UFPA), Belém, Pará, Brasil. Endereço para correspondência: Rua Augusto Corrêa, 01, Campus Universitário do Guamá, Belém, Pará, Brasil, CEP: 66075-110. E-mail: luiscastleb@gmail.com.

[**] Mestra em Educação em Ciências e Matemáticas pela Universidade Federal do Pará (UFPA). Doutoranda no Programa de Pós-Graduação em Educação em Ciências e Matemáticas da Universidade Federal do Pará (UFPA), Belém, Pará, Brasil. Endereço para correspondência: Rua Augusto Corrêa, 01, Campus Universitário do Guamá, Belém, Pará, Brasil, CEP: 66075-110. E-mail: ivonne.s.1812@gmail.com.

la práctica pedagógica al proporcionar una mayor interacción visual y comprensión de conceptos complejos. Sin embargo, los materiales en portugués son limitados. Proponemos el proyecto AM²: Animaciones Matemáticas con Manim, que tiene como objetivo ofrecer formación docente, crear recursos educativos y facilitar el uso de tecnologías digitales en la enseñanza de las Matemáticas.

Palabras clave: Python. PyGgb. Manim. Enseñanza. Matemáticas.

1 INTRODUÇÃO

Na história da humanidade o transcurso do tempo fez revoluções que possibilitaram explorar as possibilidades de novas forma de representação de ideias e conceitos matemáticos. Alguns pontos de ignição de novas revoluções, tal como as industriais no século XVIII e XIX, e a revolução digital a partir de 1990 (Barros, 2023). Segundo Mendes (2023) destaca que, com o advento dessas revoluções tecnológicas, provocaram transformações ao longo do mesmo século, que ampliaram novos meios de representação da cultura digital e a (re)invenção das formas orais e escritas na produção de conhecimentos na sociedade, que posteriormente foram materializados pelos primeiros computadores de escritório até aos atuais smartphones.

Para Levy (1995, 2004) dessas Tecnologias e/ou Recursos Digitais que podem ser acessados via a internet, tem o potencial explorar até o limite das possibilidades semióticas e cognitivas oferecidas pela plasticidade indefinida das representações digitais produzidas por elas. Pois, uma imagem não só aminada como aquelas do cinema, mas, uma representação digital “mais interativa”, com a qual, além de poder fazer desfilar a imagem a uma velocidade pretendida, parar ou voltar atrás, interagir com ela em tempo real sobre seus parâmetros, por cores, tamanho, entre outros, em processo dialético entre a Tecnologia Digital e o sujeito no momento de agir, pensar, comunicar, ensinar e aprender com elas e nelas.

No cenário educativo, esses sujeitos, são os estudantes de hoje em dia que nascem e crescem num mundo informatizado e digital, tornando imprescindível que os professores e demais profissionais da educação se qualifiquem para inserir as Tecnologias Digitais no processo de ensino e aprendizagem. A forte presença das tecnologias digitais na sociedade, e consequentemente nas escolas, demanda novos olhares sobre a educação, considerando que o espaço de aprendizagem transcende a fisicalidade da sala de aula. Assim sendo, novos espaços são constituídos pelas tecnologias digitais, onde se configuram novos modos de agir, pensar, comunicar, aprender e ensinar com esses recursos.

Para Levy (2010), nesse processo o pensamento se dá em uma rede na qual, neurônios, módulos cognitivos, humanos, instituições de ensino, línguas, sistemas de escritas, livros e

computadores se interconectam, transformam e traduzem conceitos, definições e noções. Portanto, podemos considerar que a cognição humana é coletiva e influenciada pelas tecnologias, incluindo as digitais. Uma destas tecnologias são as linguagens de programação, as quais, possibilitam nos estudantes o desenvolvimento do pensamento computacional.

Segundo Base Nacional Comum Curricular – BNCC (2017) destaca a importância de do uso das linguagens de programação como uma ferramenta essencial para desenvolver habilidades para resolver problemas específicos ou semelhantes por meio de algoritmos, que podem ser descritos em diferentes formatos, como fluxogramas. A BNCC (2017) ressalta que o uso de programação vai além da tecnologia, pois ajuda os estudantes a estruturarem problemas em etapas lógicas e sequenciais, favorecendo e ampliando o repertório linguístico dos alunos para inseri-los no mundo digital.

Segundo o Índice TIOBE[1] para setembro de 2024, A linguagem Python se posiciona de número um desse ranking. Embora a sua concepção não seja recente, pois ela foi criada entre o fim de 1989 e o início dos anos 1990 como projeto pessoal de Guido van Rossum, que até hoje continua liderando seu desenvolvimento, contando com a colaboração de muitos desenvolvedores ao redor do mundo.

Pelo exposto anteriormente, surge o questionamento a seguir: quais possíveis desdobramentos no uso do Python o ensino de Matemática. Na literatura especializada encontramos o PyGgb e Manim. Segundo Montaner (2023) o PyGgb, é um ambiente de trabalho online no qual podemos programar em Python com objetos e comandos do GeoGebra e visualizar o resultado na visualização gráfica do GeoGebra. Além das janelas de programação e representação, PyGgb inclui uma terceira janela que permite visualizar outros dados relacionados à construção. No caso do Manim[2] (Mathematical Animation Engine), é uma biblioteca de Python que permite o desenvolvimento de vídeos de animação em matemática, gráficos e geometria usando a linguagem de programação Python, oferecendo um controle muito flexível e preciso de todos os elementos. Assim, nesse trabalho temos o objetivo de descrever o PyGGB e o Manim e as possibilidades deles para a abordagem de conteúdos de matemática.

Nas seguintes seções são dedicadas para aprofundar no Python, e suas conexões com o

[1] O índice da Comunidade de Programação TIOBE é um indicador da popularidade das linguagens de programação. O índice é atualizado uma vez por mês. https://www.tiobe.com/tiobe-index/
[2] https://docs.manim.community/en/stable/index.html

PyGGB e o Manim, e finalmente as considerações finais.

2 PYTHON E O ENSINO DE MATEMÁTICA

Na literatura, há diversos exemplos de como o Python é utilizado no ensino de matemática. Um deles é apresentado por Marcondes (2018) em seu livro *Matemática com Python: um guia prático*. O autor demonstra como essa linguagem pode ser empregada de forma direta e simples, tanto na busca por resultados quanto para facilitar a compreensão de conteúdos matemáticos, como operações básicas, matrizes, números complexos, equações, inequações, elaboração de gráficos e tópicos de cálculo diferencial e integral. Além disso, Marcondes (2018) afirma que o uso da linguagem de programação Python na resolução de problemas matemáticos revela o quão poderosa ela é no apoio a estudos, pesquisas e aplicações. O mais relevante é que seu uso não exige conhecimentos avançados de programação e algoritmos, sendo necessário apenas um entendimento básico da linguagem e de sua sintaxe.

Outro exemplo do uso do Python no ensino de matemática é encontrado em Lombardi (2022), em seu livro *Cálculo Numérico em Python*. O autor busca auxiliar tanto estudantes quanto aqueles que desejam iniciar ou aprofundar seus conhecimentos em Métodos Numéricos, incluindo programação. Com base na compreensão de problemas, sua resolução e implementação computacional, a obra aborda de maneira abrangente o conteúdo programático da disciplina, além de suas aplicações em problemas práticos.

A obra de Cervantes, Báez, Arízaga e Castillo (2017), *Python con aplicaciones a las matemáticas, ingeniería y finanzas*, também ilustra o uso da linguagem de programação Python, desde seus conceitos e características básicas até o desenvolvimento de programas de alta complexidade. Esses programas são aplicados em áreas como Ciências e Engenharias, bem como em Finanças, e a obra é direcionada a profissionais interessados em aprender a utilizar Python nessas áreas.

De acordo com Banin (2018), as características do Python que o tornam especialmente útil são: portabilidade, uma vez que seu código-fonte é escrito em ANSI C e o interpretador Python, juntamente com suas bibliotecas-padrão, está disponível para diversas plataformas, como Unix, Linux, Windows (todas as versões), macOS, BeOS e VMS, entre outras; ser código aberto (opensource), o que permite seu uso e distribuição livres; simplicidade e robustez, similar a linguagens de script como Perl e Scheme; além de mecanismos que facilitam a integração com softwares desenvolvidos em outras linguagens, como C. Python também é reconhecido

por ser fácil de aprender e de grande aplicabilidade, podendo ser utilizado em diversas áreas do conhecimento.

Lombardi (2022) e Marcondes (2018) destacam que muitas funções utilizadas na solução de problemas matemáticos são importadas de bibliotecas já existentes no ambiente Python. Operações mais complexas, que exigem funções específicas, não podem ser realizadas apenas no console. Entre essas funções, encontram-se as trigonométricas, exponenciais e logarítmicas, por exemplo. Uma biblioteca em Python é um conjunto de módulos e pacotes que contém funções, classes e variáveis pré-definidas, facilitando o desenvolvimento de software ao permitir que os programadores reutilizem código em vez de escrevê-lo do zero.

Algumas bibliotecas úteis para operações matemáticas são listadas a seguir:

- NumPy: Esta biblioteca é fundamental para a computação científica em Python. Oferece suporte para trabalhar com arrays e matrizes multidimensionais e uma ampla variedade de funções matemáticas para álgebra linear, estatísticas e transformadas. Seu principal benefício é o processamento rápido e eficiente de grandes volumes de dados numéricos.
- Math: Parte da biblioteca padrão do Python, a math oferece funções matemáticas básicas, como seno, cosseno, exponenciais e logaritmos. É útil para cálculos matemáticos tradicionais que envolvem números reais e operações elementares.
- Fractions: Esta biblioteca permite manipular frações de maneira exata, sem arredondamentos. Com ela, você pode criar e operar frações, representando números racionais de forma precisa, ideal para cenários em que a precisão é crítica, como cálculos financeiros ou álgebra.
- SymPy: Focada em matemática simbólica, permite a manipulação de expressões matemáticas de forma simbólica, o que significa que pode lidar com variáveis indefinidas, simplificação de expressões, derivadas, integrais e equações diferenciais. É amplamente utilizada em álgebra, cálculo e problemas que exigem manipulação simbólica.
- Pylab: É uma interface que combina funcionalidades do NumPy e Matplotlib, simplificando a criação de gráficos e a manipulação de dados numéricos. Usada principalmente para prototipagem e visualização rápida, a pylab permite gerar gráficos bidimensionais de maneira eficiente.
- Matplotlib: Uma biblioteca poderosa para a criação de gráficos e visualizações.

É altamente personalizável e permite a criação de gráficos complexos, como gráficos de linhas, barras, dispersão, histogramas e figuras tridimensionais. Usada extensivamente em ciência de dados e visualização de resultados.

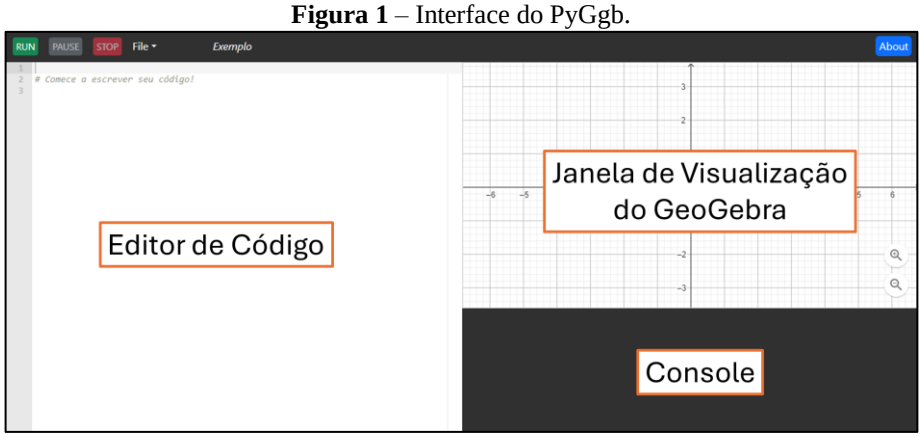
- Cmath: É uma extensão da biblioteca math, voltada para números complexos. Com cmath, é possível realizar operações como cálculo de ângulos, exponenciais, logaritmos e funções trigonométricas com números complexos, úteis em áreas como engenharia elétrica e física, onde números complexos são frequentemente utilizados.

Essas bibliotecas tornam Python uma linguagem poderosa para cálculos matemáticos e visualização de dados, desde problemas numéricos básicos até a resolução de equações simbólicas complexas.

3 PYGGB = PYTHON + GEOGEBRA

PyGgb[3] quer dizer Python + GeoGebra, este é um ambiente online onde você pode programar em Python e visualizar o resultado gráfico no GeoGebra. PyGgb é uma biblioteca Python que permite interagir com a aplicação de geometria dinâmica GeoGebra através de sua API. Com o PyGgb, os usuários podem criar programas Python que interagem com objetos e comandos do GeoGebra, possibilitando a criação de visualizações matemáticas complexas e a automação de tarefas repetitivas. PyGgb permite criar visualizações interativas e explorar conceitos matemáticos em um ambiente familiar como Python. PyGgb está atualmente em beta, portanto todos os recursos estão sujeitos a alterações. Na figura 1 se apresenta a estrutura no ambiente PyGgb.

Figura 1 – Interface do PyGgb.



Fonte: Elaboração própria dos Autores (2023).

[3] O acesso ao ambiente PyGgb é feito através do link https://www.geogebra.org/python

Nesta seção revisamos algumas das opções que o PyGgb nos oferece para abordar conteúdos matemáticos, a seguir mostramos algumas propostas:

3.1 Cálculo de Área de um Triangulo

Damos início com um problema simples, calcular área conforme os vértices de um triângulo são arrastados, e assim devemos escreva uma função em Python que é chamada quando um ponto na janela de visualização do GeoGebra é arrastado ao redor do plano. Para resolver o problema, primeiro o módulo math é importado para usar a função sqrt (raiz quadrada), essencial no cálculo da fórmula de Herão para a área de um triângulo. Depois usamos a classe Point para definir três vértices do triângulo com coordenadas iniciais: A(3, 4), B(0, 2) e C(5, 1). Esses pontos serão dinâmicos no ambiente GeoGebra, ou seja, eles podem ser arrastados pelo usuário. Depois, criamos a Função calcular_area(), de maneira a calcular as distâncias entre os pontos A, B e C usando a função Distance(). Isso dá os comprimentos dos lados do triângulo: AB, BC e CA.

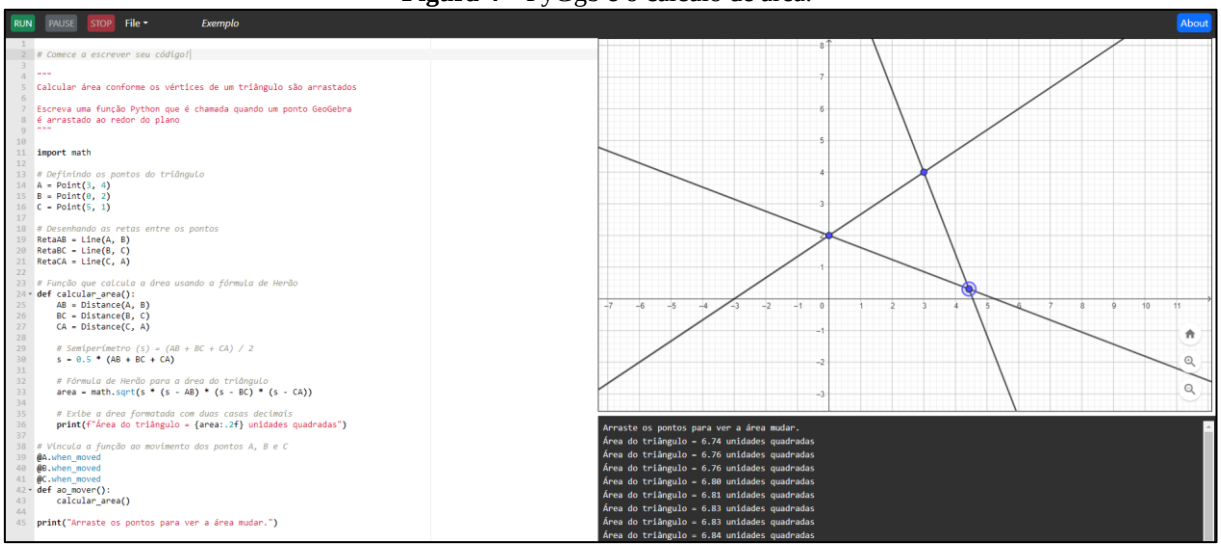
Os seguintes passos a seguir são é calcular o semiperímetro que chamamos de “s”, que é a metade da soma dos comprimentos dos três lados. Após isso, utilizamos a fórmula de Herão para calcular a área do triângulo. Onde “s” é o semiperímetro e AB, BC, CA são os comprimentos dos lados. Depois, definimos a Função ao_mover(), essa função é vinculada a @A.when_moved, @B.when_moved, e @C.when_moved, que significa que ela será chamada automaticamente toda vez que os pontos A, B ou C forem arrastados pelo plano. Dentro dela, chamamos calcular_area() para recalcular e mostrar a nova área do triângulo após cada movimento. Finalmente, na console, aparece a mensagem "Arraste os pontos para ver a área mudar" é exibida no console para orientar o usuário. Aqui definimos que a área calculada no console tem precisão de duas casas decimais.

Figura 3 – Script[4] no PyGgb.

```
# Comece a escrever seu código!

"""
Calcular área conforme os vértices de um triângulo são arrastados

Escreva uma função Python que é chamada quando um ponto GeoGebra
é arrastado ao redor do plano
"""

import math

# Definindo os pontos do triângulo
A = Point(3, 4)
B = Point(0, 2)
C = Point(5, 1)

# Desenhando as retas entre os pontos
RetaAB = Line(A, B)
RetaBC = Line(B, C)
RetaCA = Line(C, A)

# Função que calcula a área usando a fórmula de Herão
def calcular_area():
    AB = Distance(A, B)
    BC = Distance(B, C)
    CA = Distance(C, A)

    # Semiperímetro (s) = (AB + BC + CA) / 2
    s = 0.5 * (AB + BC + CA)

    # Fórmula de Herão para a área do triângulo
    area = math.sqrt(s * (s - AB) * (s - BC) * (s - CA))

    # Exibe a área formatada com duas casas decimais
    print(f"Área do triângulo = {area:.2f} unidades quadradas")

# Vincula a função ao movimento dos pontos A, B e C
@A.when_moved
@B.when_moved
@C.when_moved
def ao_mover():
    calcular_area()

print("Arraste os pontos para ver a área mudar.")
```

Fonte: Elaboração própria dos Autores (2023).

Figura 4 – PyGgb e o cálculo de área.



Fonte: Elaboração própria dos Autores (2023).

[4] https://ray.so/f6Z59aM

https://doi.org/10.63036/ReTEM.2965-9698.2023.v1.163 2965-9698

3.2 Plotagem de uma função exponencial

O exemplo a seguir mostra um script que define várias funções e realiza algumas operações para ajustar uma visualização e gerar pontos com base em uma função matemática. Primeiro, o módulo time é importado para permitir pausas e manipulação de tempo durante a execução. A função ZoomIn é definida para ajustar o zoom de uma visualização, mas no momento está incompleta e não realiza nenhuma ação (pass é um marcador de posição).

A função frange é uma implementação personalizada que gera uma sequência de números de ponto flutuante. Ela aceita três parâmetros: start (valor inicial), stop (valor final, não incluído) e step (intervalo entre valores). A função usa um loop infinito e o comando yield para gerar valores sucessivos até que a condição de parada seja atendida, permitindo iteração sobre números decimais, algo que o range() padrão não suporta.

A função a calcula a os pontos que pertem a função $f(x) = 2^n$ para um dado valor de n, utilizando a operação de exponenciação. É uma forma compacta de calcular potências de 2, alternativa ao uso da função pow do módulo math.

Depois de definir essas funções, o script chama ZoomIn para possivelmente redefinir o zoom e faz uma pausa de 0,5 segundos. Em seguida, o zoom é ajustado para um fator de 0,4 e outra pausa de 0,01 segundos é adicionada entre a criação de pontos.

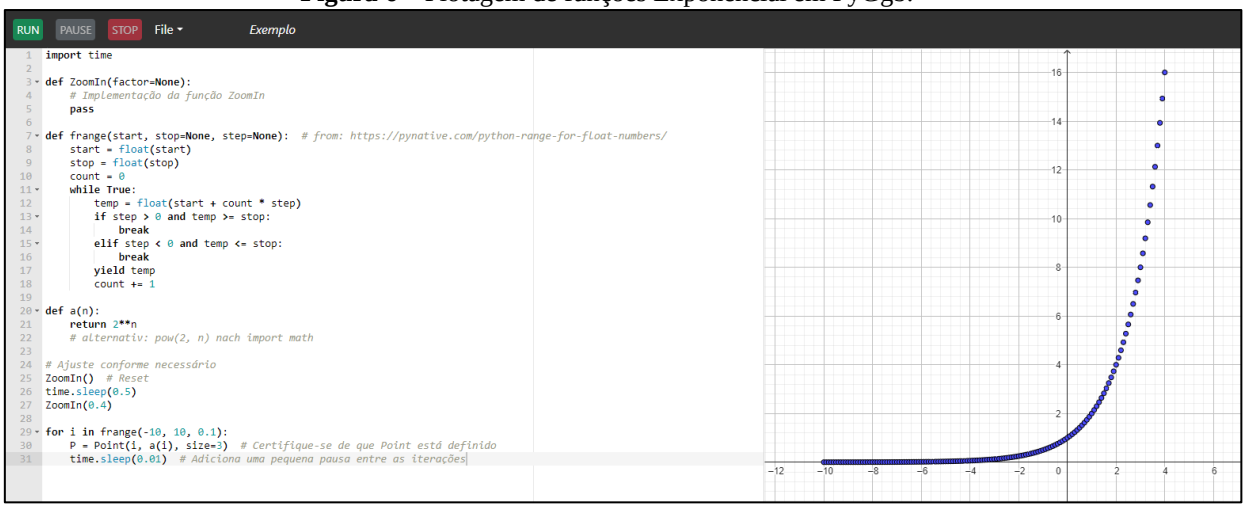
O loop for utiliza a função frange para iterar sobre um intervalo de valores de -10 a 10 com um passo de 0,1. Para cada valor gerado, a função a calcula 2^i, e um ponto é criado na posição $(i, a(i))$ com um tamanho especificado. Esta operação é repetida para todos os valores gerados, com uma pequena pausa entre cada iteração para evitar sobrecarga no processamento ou visualização dos pontos. No entanto, o script assume que a função Point está definida em outro lugar e que é responsável por criar os pontos na visualização desejada.

Figura 5 – Script[5] no PyGgb.

```python
import time

ZoomIn()  # Reset
time.sleep(0.5)
ZoomIn(0.4)

def frange(start, stop=None, step=None): # from: https://pynative.com/python-range-for-float-numbers/
    start = float(start)
    stop = float(stop)
    # print("start = ", start, "stop = ", stop, "step = ", step)
    count = 0
    while True:
        temp = float(start + count * step)
        if step > 0 and temp >= stop:
            break
        elif step < 0 and temp <= stop:
            break
        yield temp
        count += 1

def a(n):
  return 2**n
  # alternativ: pow(2,n) nach import math
for i in frange(-10,10,0.1):
  P = Point(i,a(i),size=3)
  time.sleep(0)
```

Fonte: Elaboração própria dos Autores (2023).

Figura 6 – Plotagem de funções Exponencial em PyGgb.



Fonte: Elaboração própria dos Autores (2023).

Com esses exemplos, evidenciamos que incorporar Python e o GeoGebra não só adiciona uma camada técnica interessante, mas também introduz os alunos à programação de

[5] https://ray.so/dwGQzkL

forma intuitiva e orientada a resultados, artefatos digitais concretos que podem perfeitamente ser o produto final de Situações de Aprendizagem de Competências. Isto não só fortalece as suas competências matemáticas, mas também desenvolve competências digitais que são cada vez mais necessárias no mundo de hoje.

4 MANIM

O Manim (*Mathematical Animation Engine*) foi criado por Grant Sanderson, conhecido pelo projeto 3Blue1Brown[6]. Essa biblioteca foi desenvolvida para criar animações precisas usando programação em Python. Devido à escassa documentação da versão original, diversas pessoas começaram a produzir tutoriais e materiais descritivos sobre a biblioteca. Isso deu origem a uma comunidade ativa, com centenas de pessoas contribuindo com documentação, tutoriais e exemplos.

Atualmente, existem três versões do Manim, **ManimGL**: *Manim Graphics Library* é a versão original criada por Grant Sanderson, também conhecido como 3Blue1Brown. **Manim Community Edition (ManimCE)** versão do ManimGL mantido pela comunidade. Esta versão tem como objetivo ser mais acessível e receber atualizações regulares e **ManimCairo** que é outra implementação do Manim que utiliza a biblioteca Cairo para renderização de gráficos em vez de OpenGL. Para a maioria dos usuários, especialmente iniciantes, ManimCE é a melhor escolha devido ao suporte comunitário ativo, melhor documentação e facilidade de instalação.

Nessa seção, apresentamos dois exemplos realizados com o Manim. No primeiro exemplo, criamos um círculo, um quadrado e um triângulo, cada um com sua respectiva legenda. Com o uso de transformações e animações que podem ser consultadas no site da documentação do Manim[7], esses Mobjects se transformam uns nos outros até chegar em uma mensagem final. Na Figura 7, encontrasse o registro do código[8] deste exemplo.

[6] https://www.3blue1brown.com/
[7] https://docs.manim.community/en/stable/index.html
[8] https://ray.so/hmWXhdV

Figura 7 – Script exemplo de Geometria.

```python
from manim import *

class ExemploGeometria(Scene):
    def construct(self):
        # Criar círculo e rótulo
        circulo = Circle(fill_color=GREEN, fill_opacity=1).shift(UP)
        circulo.set_stroke(WHITE, width=4)
        rotulo_circulo = Text('Círculo').next_to(circulo, DOWN)

        # Criar quadrado e rótulo
        quadrado = Square(fill_color=BLUE, fill_opacity=1).shift(UP)
        quadrado.set_stroke(WHITE, width=4)
        rotulo_quadrado = Text('Quadrado').next_to(quadrado, DOWN)

        # Criar triângulo e rótulo
        triangulo = Triangle(fill_color=YELLOW, fill_opacity=1).shift(UP)
        triangulo.set_stroke(WHITE, width=4)
        rotulo_triangulo = Text('Triângulo').next_to(triangulo, DOWN)

        # Criar mensagem final
        mensagem = Text('Completei minha primeira animação!').shift(2 * UP)

        # Mostrar círculo e rótulo
        self.add(circulo, rotulo_circulo)
        self.wait()

        # Transformar para quadrado e atualizar rótulo
        self.play(ReplacementTransform(circulo, quadrado),
                  ReplacementTransform(rotulo_circulo, rotulo_quadrado))
        self.wait()

        # Transformar para triângulo e atualizar rótulo
        self.play(ReplacementTransform(quadrado, triangulo),
                  ReplacementTransform(rotulo_quadrado, rotulo_triangulo))
        self.wait()

        # Desaparecer triângulo e rótulo, e depois mostrar a mensagem final
        self.play(FadeOut(triangulo, rotulo_triangulo))
        self.play(FadeIn(mensagem))
        self.wait(2)
```

Fonte: Elaboração própria dos Autores (2023).

No segundo exemplo foi produzida uma animação sobre a demonstração canônica da fórmula de Bhaskara, na qual, a partir da expressão canônica $ax^2 + bx + c = 0$, é trabalhada algebricamente por meio de transformações até chega na expressão que se usa comumente para determinar as raízes. Na Figura 8, encontrasse o registro do código[9] deste exemplo.

[9] https://ray.so/EFo92QP

https://doi.org/10.63036/ReTEM.2965-9698.2023.v1.163 2965-9698

Figura 8 – Script do exemplo da demonstração de Bhaskara.

```python
from manim import *


class BhaskaraProof(Scene):

    def equations_sequence(self, initial, equations):
        # Exibindo a equação inicial
        self.play(Write(initial))
        self.wait()

        # Transformando passo a passo
        current_step = initial
        for step in equations:
            self.play(ReplacementTransform(current_step, step))
            self.wait()
            current_step = step

        # Realçando o passo final
        self.play(Circumscribe(equations[-1]), buff=1.5)

    def construct(self):
        title = MathTex(r"\text{Demonstração Canônica}",
                        font_size=50, color=YELLOW)
        title.shift(0.5 * UP)
        self.play(Write(title))

        subtitle = MathTex(
            r"\text{Fórmula de Bhaskara}", font_size=40, color=BLUE)
        subtitle.next_to(title, DOWN, buff=0.75)
        self.play(Write(subtitle))
        self.wait()

        self.play(FadeOut(title), FadeOut(subtitle))

        """
        { function_description }
        """
        # Equação inicial:
        equation = MathTex(r"ax^2 + bx + c = 0")

        # Definindo as expressões intermediárias
        step1 = MathTex(r"ax^2 + bx = - c")
        step2 = MathTex(r"x^2 + \frac{b}{a}x = -\frac{c}{a}")
        step3 = MathTex(
            r"x^2 + \frac{b}{a}x + \frac{b^2}{4a^2} = -\frac{c}{a} + \frac{b^2}{4a^2}")
        step4 = MathTex(
            r"\left(x+\frac{b}{2a}\right)^2=-\frac{4ac}{4a^2}+\frac{b^2}{4a^2}")
        step5 = MathTex(r"x + \frac{b}{2a} = \pm \frac{\sqrt{b^2 - 4ac}}{2a}")
        step6 = MathTex(r"x = \frac{b \pm \sqrt{b^2 - 4ac}}{2a}")

        # Usando a função equations_sequence para mostrar a prova

        self.equations_sequence(equation, [
            step1, step2, step3, step4, step5, step6
        ])
```

Fonte: Elaboração própria dos Autores (2023).

https://doi.org/10.63036/ReTEM.2965-9698.2023.v1.163 2965-9698

5 CONSIDERAÇÕES FINAIS

Como discutido neste trabalho, vemos promissor o uso do Python no Ensino de Matemática. Embora isso possa representar desafios significativos para alguns professores, esses desafios podem ser superados com algumas horas de estudo utilizando os diversos cursos e tutoriais sobre Python disponíveis em plataformas digitais, como o YouTube. Além disso, há pouca literatura especializada sobre o uso das animações criadas com o Manim e do uso de PyGgb em sala de aula.

No caso do Manim, A comunidade do Manim oferece um guia completo de orientações, desde a instalação do Manim até como utilizá-lo. Embora o guia esteja em inglês, o que pode ser um desafio para alguns, pode ser uma vantagem para outros. No Brasil, já foi desenvolvido um manual[10] para que iniciantes em Python possam aprender a linguagem antes de se aventurarem no uso do Manim. Para mais detalhes, consulte Kishimoto e Coluci (2023). Este manual, no entanto, foi desenvolvido para a versão v0.11.0 do Manim. No caso de PyGgb, apenas o trabalho de Montaner (2023).

Nesse cenário, refletimos sobre necessidade de apresentar as potencialidades do Manim à comunidade de professores que buscam inovar em suas aulas de matemática com o apoio das tecnologias digitais. Trata-se de uma ferramenta de fácil aprendizado e baixo custo, proporcionando aos professores e futuros professores de matemática a capacidade de criar suas próprias animações de forma prática e acessível. Além disso, o Manim permite uma maior interatividade nas aulas, possibilitando a exploração visual de conceitos matemáticos complexos, tornando-os mais compreensíveis e atraentes para os estudantes.

Finalmente, consideramos a criação do projeto AM^2: Animações Matemáticas com Manim, com o intuito de proporcionar formação sólida e atualizada aos profissionais da educação matemática, que poderão usar tais conhecimentos em suas práticas pedagógicas e contribuir para a formação de futuros educadores matemáticos. Além disso, o projeto promoverá o desenvolvimento de recursos educacionais, guias e tutorias em português, o que facilitará o acesso e à compreensão dos materiais por parte dos professores. A seguir, apresentamos a logomarca (Figura 9) que representa o projeto.

[10] https://wordpress.ft.unicamp.br/explora/manual/

Figura 9 – Logo do Projeto AM^2

Fonte: Elaboração própria dos Autores (2023)

REFERÊNCIAS

BANIN, Sérgio Luiz. **python 3 conceitos e aplicações**: uma abordagem didática. 1. ed. São Paulo: Érica, 2018.

BARROS, José D'Assunção. A trama Polifônica das grandes Revoluções Tecnológicas – da revolução agrícola às revoluções industriais e à revolução digital. **Revista de Matemática, Ensino e Cultura - REMATEC**, Belém, v. 18, n. 44, e2023002, 2023. https://doi.org/10.37084/REMATEC.1980-3141.2023.n44.pe2023002.id505

CERVANTES, Ofelia; BÁEZ, David; ARÍZAGA, Antonio; CASTILLO, Esteban. **Python con aplicaciones a las matemáticas, ingeniería y finanzas**. 1. ed. México: Alfaomega, 2017.

KISHIMOTO, Eric Satoshi Suzuki; COLUCI, Vitor Rafael. Animações para o ensino de matemática usando o Manim-Python. **Professor de matemática online**, v. 11, n1, p. 50-64, 2023.

LÉVY, Pierre. **A ideografia dinâmica:** rumo a uma imaginação artificial? São Paulo: Edições Loyola, 2004.

LÉVY, Pierre. **A Máquina Universo:** Criação, Cognição e Cultura Informática. Lisboa: Instituto Piaget, 1995.

LÉVY, Pierre. **As tecnologias da inteligência**: O futuro do pensamento na era da informática. São Paulo: Editora 34, 2010.

LOMBALDO, Julio. **Cálculo Numérico em Python**. 1. ed. Porto Alegre: Ed. dos Autores, 2022.

MARCONDES, Guilherme Augusto Barucke. **Matemática com Python**: um guia prático. 1. ed. São Paulo: Novatec Editora, 2018.

MENDES, Iran Abreu. Sobre escritas ideográficas dinâmicas da história da Matemática. **Revista de Matemática, Ensino e Cultura - REMATEC**, Belém, v. 18, n. 44, e2023011, 2023. https://doi.org/10.37084/REMATEC.1980-3141.2023.n44.pe2023011.id515

MONTANER, Mónica Soler. GeoGebra con Python. UNIÓN - **Revista Iberoamericana de Educación Matemática**, v.19, n. 69, p. 1-8, 2023. Disponível em: https://union.fespm.es/index.php/UNION/article/view/1569

APÊNDICE 1 – INFORMAÇÕES SOBRE O MANUSCRITO

AGRADECIMENTOS
O presente trabalho foi realizado com apoio da Fundação Amazônia de Amparo a Estudos e Pesquisas do Pará (FAPESPA) e da Universidade Federal do Pará.

FINANCIAMENTO
Não se aplica.

CONTRIBUIÇÕES DE AUTORIA
Resumo/Abstract/Resumen: Luis Andrés Castillo e Ivonne C. Sánchez
Introdução: Luis Andrés Castillo e Ivonne C. Sánchez
Referencial teórico: Luis Andrés Castillo e Ivonne C. Sánchez
Discussão dos resultados: Luis Andrés Castillo e Ivonne C. Sánchez
Conclusão e considerações finais: Luis Andrés Castillo e Ivonne C. Sánchez
Referências: Luis Andrés Castillo e Ivonne C. Sánchez
Revisão do manuscrito: Luis Andrés Castillo e Ivonne C. Sánchez
Aprovação da versão final publicada: Luis Andrés Castillo e Ivonne C. Sánchez

CONFLITOS DE INTERESSE
Não se aplica.

DISPONIBILIDADE DE DADOS DE PESQUISA
Não se aplica.

PREPRINT
Não se aplica.

CONSENTIMENTO DE USO DE IMAGEM
Não se aplica.

APROVAÇÃO DE COMITÊ DE ÉTICA EM PESQUISA
Não se aplica.

COMO CITAR - ABNT
CASTILLO, Luis Andrés; SÁNCHEZ, Ivonne C. Uso de Python no Ensino de Matemática: PyGgb e Manim. **ReTEM – Revista Tocantinense de Educação Matemática**. Arraias, v. 1, *e*23006, jan./dez., 2023. https://doi.org/10.63036/ReTEM.2965-9698.2023.v1.163

COMO CITAR - APA
Castillo, L. A., & Sánchez, I. C. (2023). Uso de Python no Ensino de Matemática: PyGgb e Manim. *ReTEM – Revista Tocantinense de Educação Matemática*, *1*, *e*23006. https://doi.org/10.63036/ReTEM.2965-9698.2023.v1.163

HISTÓRICO
Submetido: 12 de setembro de 2023.
Aprovado: 25 de novembro de 2023.
Publicado: 31 de dezembro de 2023.