# Análise Comparativa: Python puro e Biopython na Bioinformática do Conteúdo GC

**BRITO**, Denise Nogueira; **CANDIDO DA SILVA**, Jaqueline; **SILVA**, Robson Soares

**Resumo**

À medida que a quantidade de dados biológicos explodiu em uma escala sem precedentes, surgiu uma demanda urgente de combinar conhecimentos biológicos com o poder da computação. Foi nesse contexto que surgiu a Bioinformática, como uma resposta à necessidade de interseção dos campos da biologia e da computação. A Bioinformática desempenha um papel crítico na interpretação de dados biológicos e na compreensão de processos genéticos. A combinação da linguagem de programação Python e da biblioteca Biopython oferece ferramentas poderosas para a manipulação de sequências de DNA e proteínas, bem como para o cálculo do GC-content (conteúdo GC(guanina-citosina)). Este artigo apresenta uma análise sobre Python e Biopython na Bioinformática com foco especial no cálculo e análise de conteúdo GC, e fornece um guia prático ilustrativo e comparativo do uso de Python puro e do Biopython, a partir de uma implementação criada com o orientador Robson Soares Silva para o cálculo de conteúdo de GC em sequência genômica, além de suas execuções. O Colaboratory (Colab) foi escolhido como a plataforma para análise e testes da implementação, visando estabelecer um ambiente de pesquisa acessível e gratuito.

**Palavras-chave:** Bioinformática. Python. Biopython. Conteùdo GC.

**Abstract**

As the amount of biological data has exploded on an unprecedented scale, an urgent demand has emerged to combine biological knowledge with computing power. It was in this context that Bioinformatics emerged, as a response to the need for the intersection of the fields of biology and computing. Bioinformatics plays a critical role in interpreting biological data and understanding genetic processes. The combination of the Python programming language and the Biopython library provides powerful tools for manipulating DNA and protein sequences, as well as calculating GC-content. This article presents an analysis of Python and Biopython in Bioinformatics with a special focus on calculation and analysis of GC content, and provides a practical illustrative and comparative guide on the use of pure Python and Biopython, based on an implementation created with advisor Robson Soares Silva

for calculating GC content in genomic sequence, in addition to its executions. Colaboratory (Colab) was chosen as the platform for analysis and implementation testing, aiming to establish an accessible and free research environment.

**Keywords**: Bioinformatics. Python. Biopython. GC content.

## 1. Introdução

A Bioinformática é a linha de pesquisa que representa a interseção da computação e da biologia, onde o poder computacional realiza análises biológicas. Nesse campo, especialistas criam e implementam ferramentas, utilizando algoritmos avançados e análise estatística.

O início da Bioinformática foi marcado pelo feito em 1953, quando Watson e Crick desvendaram a estrutura química do DNA, abrindo novas portas para a Biologia Molecular. Por volta de 1990, com a introdução e a disseminação dos sequenciadores automáticos de DNA, ocorreu também uma explosão na quantidade de dados genéticos, resultando em uma demanda crescente por recursos computacionais mais avançados, tanto para armazenamento quanto para análise desses dados.

Desta maneira, a Bioinformática emergiu como uma nova ciência, composta pela fusão entre diversos campos de conhecimento, incluindo engenharia de software, matemática, estatística, ciência da computação e biologia molecular.

### 1.1. Importância do Conteúdo GC

Na pesquisa Genética e Bioinformática, a importância do conteúdo de GC é indiscutível. Isso ocorre porque sua presença exerce influência direta sobre a estrutura, a função e a estabilidade das moléculas de ácido nucleico. Alterações no conteúdo GC podem repercutir na codificação genética, na configuração tridimensional do DNA e até mesmo na trajetória da evolução das espécies. Compreender o conteúdo de GC é como desvendar um enigma fundamental da Genética e da Bioinformática. Essas variações influenciam diretamente como nossos genes são expressos e como nosso DNA se mantém estruturalmente sólido. Nesse contexto, a pesquisa e interpretação do conteúdo de GC assumem um papel fundamental na descoberta dos segredos do material genético e em sua influência sobre todos os seres vivos.

### 1.2. Objetivo do Estudo

Este estudo tem como objetivo principal investigar a eficácia das abordagens de programação na análise do conteúdo GC em sequências genômicas. Para isso, examinaremos tanto a implementação em Python puro quanto a utilização da biblioteca Biopython. A análise comparativa dessas duas abordagens.

Nas seções subsequentes, exploraremos detalhadamente o significado biológico do conteúdo GC. Também iremos discutir as complexidades do Python na área de Bioinformática, apresentar a biblioteca Biopython e explicar em detalhes a metodologia empregada para comparar as implementações. Além disso, iremos analisar os resultados obtidos,discutindo as implicações dessas descobertas e oferecendo sugestões para pesquisas futuras no campo da Bioinformática e análise genômica. Tudo isso será realizado usando a plataforma gratuita do Google Colaboratory, também conhecida como Colab.

## 2. Referencial teórico

Levando em consideração que a Bioinformática é um campo de pesquisa interdisciplinar que integra conceitos da biologia e da ciência da computação. Seu principal objetivo é aplicar técnicas computacionais e matemáticas para a análise e interpretação de dados biológicos, com foco especial na análise de sequências de DNA. Conforme Prosdocimi (2007, p. 3), “Podemos considerar a Bioinformática como uma linha de pesquisa que envolve aspectos multidisciplinares e que surgiu a partir do momento em que se iniciou a utilização de ferramentas computacionais para a análise de dados genéticos, bioquímicos e de biologia molecular”.

A análise de sequências de DNA é uma parte essencial da Bioinformática, envolvendo o exame e a extração de informações das sequências de nucleotídeos presentes no DNA. Isso inclui a identificação de genes, regiões codificadoras, elementos regulatórios e variações genéticas. O valor dessa análise reside na compreensão das instruções genéticas contidas no DNA, as quais são cruciais para determinar as características e o funcionamento de um organismo. Além disso, identificar um gene específico em uma sequência de DNA pode ajudar a compreender como uma determinada característica é herdada em uma população.

Além disso, a Bioinformática fornece ferramentas computacionais e algoritmos que possibilitam aos cientistas analisar grandes conjuntos de dados de sequências de DNA de maneira eficiente. Isso é aplicável em diversas áreas, desde o diagnóstico de doenças

genéticas até estudos de evolução e pesquisa em biologia molecular. Desta forma, a Bioinformática desempenha um papel fundamental na interpretação e exploração das informações genéticas, revelando as informações contidas no DNA e contribuindo significativamente para o avanço do conhecimento em biologia e medicina.

## 2.1. Python

Python é uma linguagem de programação, idealizada em 1989, mas só foi lançada em 1991 pelo então programador holandês Guido van Rossum. Segundo a pesquisa de XP EDUCAÇÃO (2022), “Python é uma das principais e mais populares linguagens de programação em todo o mundo”.

Criado com a visão de desenvolver uma linguagem de programação acessível, fácil de ler e compreender, com o objetivo de proporcionar uma ferramenta eficiente para programadores de todos os níveis de experiência, promovendo a legibilidade e a simplicidade. Essa abordagem transformou Python em uma linguagem poderosa, amplamente utilizada em diversos campos, como desenvolvimento web, análise de dados, automação e inteligência artificial. Sua versatilidade e eficácia tornam o Python uma escolha ideal para projetos de todos os tamanhos, desde pequenos scripts até sistemas complexos, consolidando sua posição como uma das principais linguagens de programação em todo o mundo.

Uma característica notável do Python é a sua extensa biblioteca padrão, repleta de ferramentas úteis para diversas finalidades. Isso implica que, ao desenvolver um projeto, muitas das funcionalidades essenciais já estão ao alcance, resultando em economia de tempo e esforço. Ademais, a vasta comunidade de usuários está sempre pronta para oferecer suporte diante de desafios específicos.

Além disso, a linguagem não apenas beneficia-se dessa comunidade, mas a favorece, incentivando a colaboração e o compartilhamento de código. Essa mentalidade colaborativa deu origem a uma comunidade robusta de desenvolvedores e a uma ampla gama de bibliotecas de terceiros, expandindo ainda mais as capacidades do Python. Graças a essa abordagem, Python evolui de maneira constante, mantendo-se relevante e adaptado às exigências em constante mudança da indústria de tecnologia. Sua crescente popularidade consolida o Python como uma escolha sólida para quem busca aventurar-se no universo da programação ou aprimorar suas habilidades em uma linguagem que continua a prosperar.

## 2.2. Biopython

De acordo com Mariano (2020), “Biopython é uma biblioteca ou uma coleção de ferramentas para facilitar o desenvolvimento de aplicações para a Bioinformática utilizando a linguagem de programação Python.”. Biopython é, de fato, uma biblioteca Python projetada para simplificar a manipulação e análise de dados relacionados à Biologia molecular e Bioinformática. Essa biblioteca oferece uma ampla variedade de ferramentas e recursos para lidar com sequências de DNA, RNA, proteínas, estruturas biomoleculares e muito mais, tornando-se uma escolha popular entre cientistas e bioinformatas envolvidos em análises computacionais na área da biologia. Com o Biopython, é possível automatizar tarefas, realizar cálculos complexos e interagir de forma eficiente com bancos de dados biológicos.

Biopython é um conjunto de ferramentas de computação biológica disponíveis gratuitamente, escrito em Python por uma equipe internacional de desenvolvedores. Trata-se de um esforço colaborativo distribuído para desenvolver bibliotecas Python e aplicativos que atendam às necessidades atuais e futuras na área da Bioinformática.

Esta biblioteca é considerada a mais popular para computação em biologia molecular. Foi desenvolvida em 1999 por Brad Chapman e Jeff Chang e é principalmente escrita em Python, embora contenha alguns trechos em C para otimizações complexas. Biopython é capaz de realizar tarefas como a análise de estruturas de proteínas, identificação de motivos de sequência, alinhamento de sequências e até mesmo aplicação de técnicas de machine learning.

## 2.3. Google Colaboratory

O Google Colab, também conhecido como Google Collaboratory, proporciona uma oportunidade acessível e eficiente para o desenvolvimento em Python. Com ele, é possível escrever e executar código Python diretamente no navegador, eliminando a necessidade de instalações de software. Além dessa praticidade, o Colab proporciona acesso gratuito a recursos computacionais de alto desempenho. Conforme citado no site Google Research (2023), “O Colaboratory ou “Colab” é um produto do Google Research, área de pesquisas científicas do Google. O Colab permite que qualquer pessoa escreva e execute código Python arbitrário pelo navegador e é especialmente adequado para aprendizado de máquina, análise de dados e educação”.

Além da sua facilidade de uso e dos recursos computacionais robustos, o Google Colab estimula a colaboração e a disseminação do conhecimento. Projetos desenvolvidos no Colab podem ser compartilhados de maneira simples, permitindo que equipes de

desenvolvedores e cientistas de dados colaborem eficientemente, independentemente da localização. Essa capacidade de colaboração em tempo real torna o Google Colab uma ferramenta inestimável para a comunidade de desenvolvedores em Python, impulsionando a inovação e o progresso de projetos de maneira ágil e eficiente.

Amplamente utilizado por cientistas de dados, pesquisadores e desenvolvedores, o Colab é uma escolha popular para explorar dados, criar modelos de aprendizado de máquina e colaborar em projetos relacionados à ciência de dados. Sua base na nuvem elimina a necessidade de configuração local de hardware, tornando o acesso e uso mais acessíveis, especialmente para iniciantes em programação e aprendizado de máquina.

## 3. Procedimentos metodológicos

Neste estudo, elaborou-se uma metodologia para calcular o conteúdo GC em sequências de DNA, empregando a linguagem de programação Python em conjunto com a biblioteca Biopython.

### 3.1. Conceito de Conteúdo GC

Em biologia molecular e genética, conteúdo GC refere-se à porcentagem de bases de guanina (G) e citosina (C) em uma sequência de ácido nucleico, como o DNA ou RNA. É uma medida importante na Genômica e na Bioinformática e desempenha um papel crucial em várias áreas da biologia molecular. A porcentagem de GC em uma sequência de DNA ou RNA é calculada da seguinte forma:

$$\text{Conteúdo GC (\%)} = \left( \frac{\text{Número de Bases GC}}{\text{Comprimento Total da Sequência}} \right) \times 100\%$$

Figura 1. Ilustração do cálculo de conteúdo GC

### 3.2. Escolha de sequências

Criamos um tutorial prático durante a vídeo-aula do curso de Ferramentas de Bioinformática para Análise de Proteínas e Peptídeos, conduzido pela Doutoranda Geniana da Silva Gomes no programa de Pós-Graduação em Bioquímica Aplicada da Universidade Federal de Viçosa. Durante essa sessão, adquirimos conhecimentos essenciais sobre como realizar buscas de sequências em renomados bancos de dados, incluindo UniProt, PDB e

NCBI. Utilizamos essas habilidades para a seleção criteriosa das sequências necessárias para nossas análises.

### 3.3. Análise Estatística

Para validar a precisão dos resultados, realizamos análises estatísticas adicionais, como o cálculo da média, desvio padrão e intervalo de confiança, dependendo do tamanho da amostra. Isso permitiu verificar a consistência dos resultados obtidos.

## 4. Apresentação dos Dados

A implementação foi desenvolvida na linguagem de programação Python, onde o código efetua o cálculo de conteúdo de GC de duas sequências de DNA. Ele oferece em seu menu interativo, duas opções de cálculo: uma usando Python puro e outra usando a biblioteca Biopython na plataforma do Colab.

Para acessar a plataforma, se faz necessário, em um navegador de internet, digitar "Google Colab" na barra de pesquisa ou acessar “colab.research.google.com” diretamente, fazer o login com uma conta de e-mail e clicar em “+ Novo notebook”, abrindo assim o ambiente para escrever o código, conforme a ilustração abaixo.

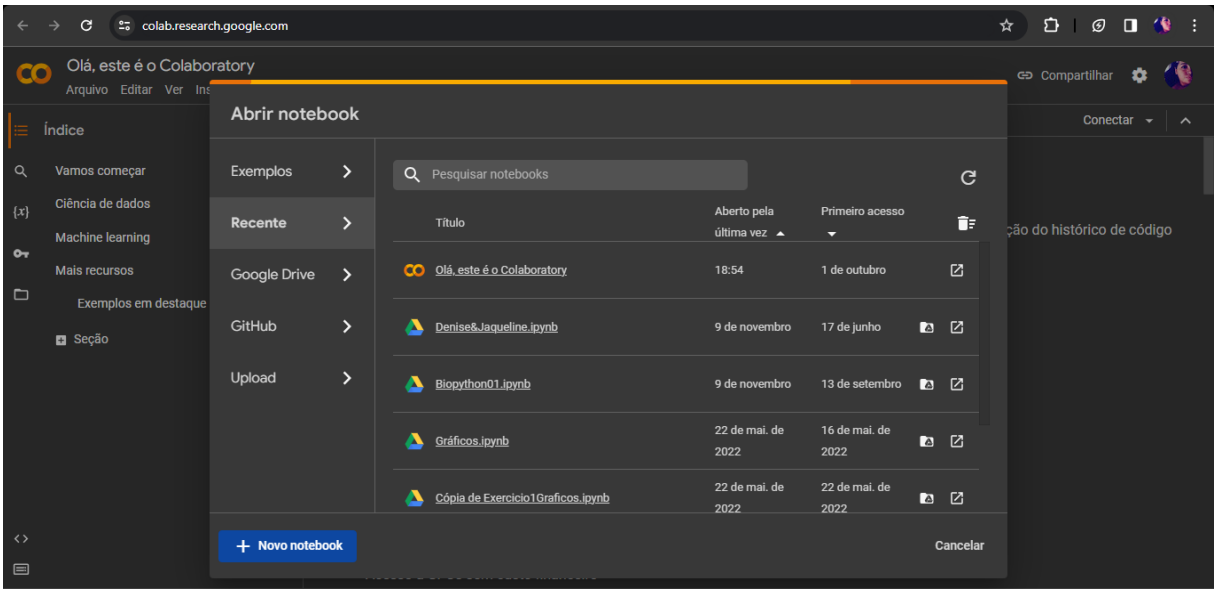


Figura 2. Acessando o site do Colaboratory

É possível alterar o nome do notebook criado, no canto superior esquerdo do seu notebook, você verá o título atual (geralmente "Sem título" ou “Untitled” no inglês), clicar no

título e uma caixa de diálogo será exibida, permitindo que você digite um novo nome, conforme imagem abaixo.

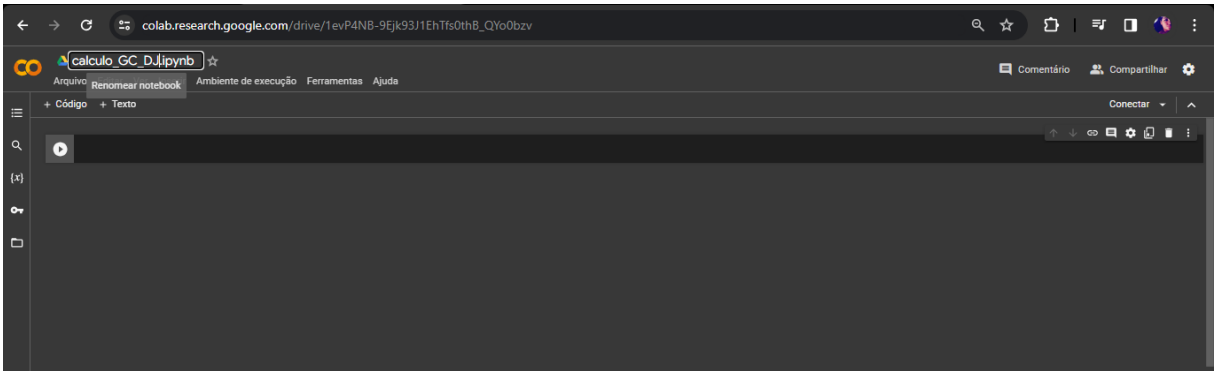


Figura 3. Acessando o ambiente e alterando o nome do notebook.

O ambiente permite escrever texto e código, como no exemplo abaixo, a implementação criada pelo orientador Robson Soares Silva para o cálculo de conteúdo de GC em sequência genômica.

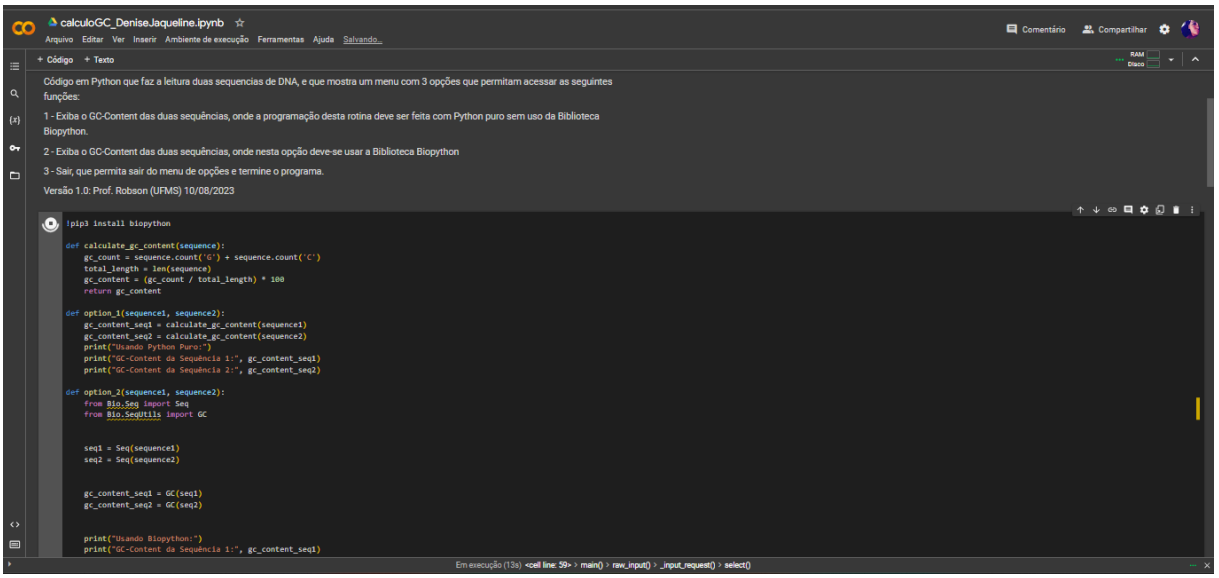


Figura 6. Texto e Código - Parte 1

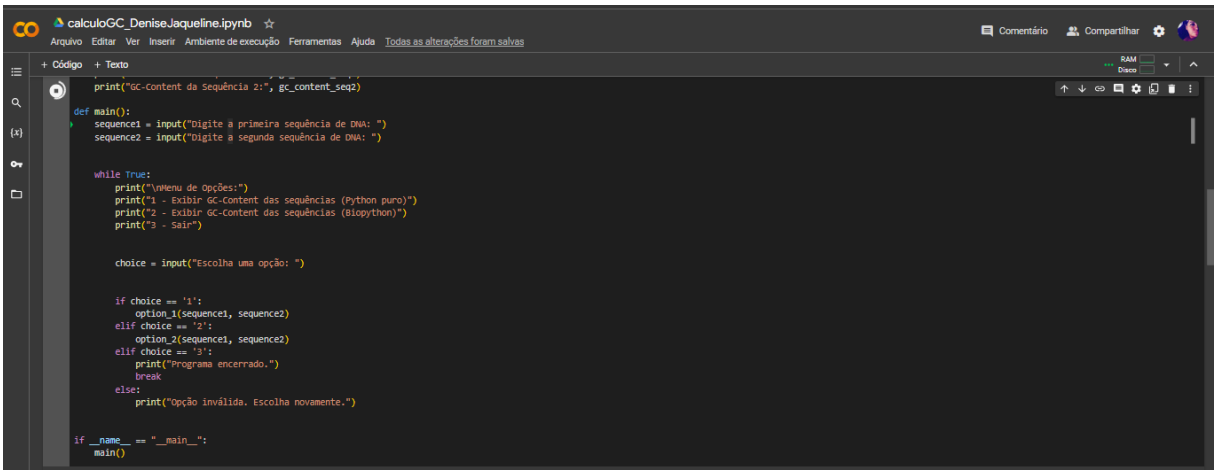


Figura 7. Continuação do código - Parte 2

## 4.1. Entendendo o código

O código inicia com o comando !pip3 install biopython, ele é usado para assegurar que a biblioteca Biopython esteja instalada no ambiente onde o código será executado. Isso é fundamental para garantir que as funções do Biopython possam ser utilizadas no restante do programa. No entanto, é importante notar que este comando específico só funcionará em ambientes interativos, como o Google Colab, onde é possível executar comandos de terminal diretamente no ambiente de programação.

```
!pip3 install biopython
```

A função abaixo, **calculate_gc_content**, calcula o conteúdo de GC de uma sequência de DNA, sendo implementada com Python puro, a função é definida como "calculate_gc_content" que recebe uma sequência de DNA como entrada. Na linha seguinte, o comando "count( )" conta o número de ocorrências das bases "G" (guanina) e "C" (citosina) na sequência. O resultado é a soma dessas contagens, representando o número total de bases GC na sequência. O comando "len( )" é usado para calcular o comprimento total da sequência de DNA. Com todas as informações obtidas o conteúdo de GC é calculado, a variável "gc_count" representa o número total de bases GC na sequência, e "total_length" representa o comprimento total da sequência, os valores serão divididos e multiplicados por 100 para obter a porcentagem de GC e a função finaliza retornando o valor da porcentagem de bases.

```
def calculate_gc_content(sequence):
    gc_count = sequence.count('G') + sequence.count('C')
    total_length = len(sequence)
    gc_content = (gc_count / total_length) * 100
    return gc_content
```

Esta função **option_1**, calcula e imprime o conteúdo de GC e foi implementada com Python puro, quando a opção "1" é selecionada a função "option_1" recebe as duas sequências "sequence1" e "sequence2", nas próximas linhas a primeira função (explicada anteriormente) é executada e o resultado é armazenado consequentemente nas variáveis "gc_content_seq1" e "gc_content_seq2", respectivamente o resultado é impresso na tela junto com a mensagem identificando as sequências.

```
def option_1(sequence1, sequence2):
    gc_content_seq1 = calculate_gc_content(sequence1)
    gc_content_seq2 = calculate_gc_content(sequence2)
    print("Usando Python Puro:")
    print("GC-Content da Sequência 1:", gc_content_seq1)
    print("GC-Content da Sequência 2:", gc_content_seq2)
```

A função **option_2**, calcula e imprime o conteúdo de GC e foi implementada com a biblioteca Biopython, quando a opção "2" é selecionada a função "option_2" recebe as duas sequências "sequence1" e "sequence2", em seguida os comandos "from Bio.Seq import Seq"

e “from Bio.SeqUtils import GC” são executados, nesse momento será importado a classe ”Seq” que é usada para representar sequências biológicas e importando a função “GC” da sub biblioteca “SeqUtils” para podermos usar essa função diretamente no nosso código para calcular o conteúdo de GC de uma sequência biológica.. Após, as sequências recebidas são convertidas em objetos de sequência Biopython usando a classe “Seq”, para que a função “GC” possa calcular. Na próxima linha é calculado o conteúdo de GC das sequências convertidas (seq1 e seq2) usando a função “GC”. Os resultados são armazenados nas variáveis “gc_content_seq1” e “gc_content_seq2”, respectivamente. A função finaliza imprimindo a mensagem que os resultados foram calculados usando a biblioteca Biopython e a mensagem identificando as sequências.

```
def option_2(sequence1, sequence2):
    from Bio.Seq import Seq
    from Bio.SeqUtils import GC

    seq1 = Seq(sequence1)
    seq2 = Seq(sequence2)

    gc_content_seq1 = GC(seq1)
    gc_content_seq2 = GC(seq2)

    print("Usando Biopython:")
    print("GC-Content da Sequência 1:", gc_content_seq1)
    print("GC-Content da Sequência 2:", gc_content_seq2)
```

A função **main()** serve como o núcleo do programa, onde ele começa sua execução e é controlado. Ela coordena a interação com o usuário, permite que o usuário insira sequências de DNA, apresenta um menu de opções e executa as operações desejadas com base na escolha do usuário. As primeiras linhas solicitam ao usuário que insira duas sequências de DNA. As entradas são armazenadas nas variáveis “sequence1” e” sequence2”, para que o código continue executando até que seja explicitamente encerrado, foi utilizado a estrutura de repetição “while True” que é executado logo após que for inserido as sequências, quando isso acontece é exibido um menu para o usuário, apresentando três opções, onde 1 calcula o conteúdo GC das sequências com Python Puro, 2 calcula o conteúdo GC das sequências com

Biopython e 3 para encerrar e sair da execução, imediatamente é solicitado ao usuário a escolha é executado o que foi escolhido. Caso escolha uma opção diferente dessa, é exibido uma mensagem indicando que a escolha é inválida e solicitando uma nova escolha.

```
def main():
    sequence1 = input("Digite a primeira sequência de DNA: ")
    sequence2 = input("Digite a segunda sequência de DNA: ")

    while True:
        print("\nMenu de Opções:")
        print("1 - Exibir GC-Content das sequências (Python puro)")
        print("2 - Exibir GC-Content das sequências (Biopython)")
        print("3 - Sair")

        choice = input("Escolha uma opção: ")

        if choice == '1':
            option_1(sequence1, sequence2)
        elif choice == '2':
            option_2(sequence1, sequence2)
        elif choice == '3':
            print("Programa encerrado.")
            break
        else:
            print("Opção inválida. Escolha novamente.")

if __name__ == "__main__":
    main()
```

## 5. Resultados e discussões

Os resultados obtidos a partir da análise comparativa entre Python puro e Biopython na Bioinformática, com ênfase no cálculo de conteúdo GC, evidenciou a eficácia de ambas as abordagens. A escolha entre elas dependerá das necessidades específicas do projeto,

destacando a versatilidade do Python e a especialização da Biopython em manipulação de dados biológicos. As sequências selecionadas para testar a implementação foram duas variedades de arroz: Oryza sativa Japonica Group Cultivar Nipponbare chromosome 11 (variedade japônica) e Oryza glaberrima (cromossomo 11).

| TABELA RESULTADOS tBLASTn | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Espécie | Contig | Coordenada Contig | % Identidade | Cobertura | E-value | Seq_Nucleotideos | Frame Tradução | Seq_aa | SignalP 5.0 / DeepLoc-1.0 |
| *Oryza sativa Japonica Groupcultivar Nipponbare chromosome 11* | NC_029266.1 | 703177 to 703449 | 70.33% | 78% | 2,00E-31 | ATCACCTGCGGCCAGGTCAACTCCGCCGTTGGGCCCTGCCTCAC<br>CTACGCCCGCGGCGGCGCCGGCCCGTCGGCGGCCTGCTGCAGC<br>GGCGTGAGGAGCCTCAAGGCCGCAGCCAGCACGACCGCTGACA<br>GGCGCACCGCGTGCAACTGCCTCAAGAACGCGGCCCGCGGCAT<br>CAAGGGGCTCAACGCCGGCAACGCCGCCAGCATCCCCTCTAAG | 5'3' Frame 2 | ITCGQVNSAVGPCLTYAR<br>GGAGPSAACCSGVRSLK<br>AAASSTADRRTACNCLK<br>NAARGIKGLNAGNAASIP<br>SKCGVSVPYTISAIDCSR | |
| *Oryza sativa Japonica Group cultivar Nipponbare chromosome 12* | NC_029267.1 | 737481 to 737747 | 70.79% | 78% | 7,00E-31 | ATCACCTGCGGCCAGGTCAACTCCGCCGTCGGGCCCTGCCTGAC<br>CTACGCCCGCGGCGGCGCCGGGCCGTCGGCGGCCTGCTGCAGC<br>GGCGTGAGGAGCCTCAAGGCGGCAGCAAGCACGACCGCAGACC<br>GGCGCACCGCCTGCAACTGCCTCAAGAACGCGGCCCGCGGCAT<br>CAAGGGGCTCAACGCCGGCAACGCCGCCAGCATCCCCTCCAAG | 5'3' Frame 2 | ITCGQVNSAVGPCLTY<br>ARGGAGPSAACCSGVR<br>SLKAAASTTADRRTAC<br>NCLKNAARGIKGLNAG | |
| *Oryza glaberrima chromosome 11* | NC_068336.1 | 708134 to 708406 | 70.33% | 78% | 6,00E-31 | ATCACCTGCGGCCAGGTCAACTCCGCCGTTGGGCCCTGCCTCAC<br>CTACGCCCGCGGCGGCGCCGGCCCGTCGGCGGCCTGCTGCAGC<br>GGCGTGAGGAGCCTCAAGGCCGCAGCCAGCACCGCTGACA<br>GGCGCACCGCGTGCAACTGCCTCAAGAACGCGGCCCGCGGCAT<br>CAAGGGGCTCAACGCCGGCAACGCCGCCAGCATCCCCTCTAAG | 5'3' Frame 2 | ITCGQVNSAVGPCLTYAR<br>GGAGPSAACCSGVRSLK<br>AAASSTADRRTACNCLK<br>NAARGIKGLNAGNAASIP<br>SKCGVSVPYTISAIDCSR | |
| *Oryza glaberrima chromosome 12* | NC_068337.1 | 791639 to 791905 | 70.79 | 78% | 2,00E-30 | ATCACCTGCGGCCAGGTCAACTCCGCCGTCGGGCCCTGCCTGAC<br>CTACGCCCGCGGCGGCGCCGGGCCGTCGGCGGCCTGCTGCAGC<br>GGCGTGAGGAGCCTCAAGGCGGCAGCAAGCACGACCGCAGACC<br>GGCGCACCGCCTGCAACTGCCTCAAGAACGCGGCCCGCGGCAT<br>CAAGGGGCTCAACGCCGGCAACGCCGCCAGCATCCCCTCCAAG | 5'3' Frame 2 | ITCGQVNSAVGPCLTYAR<br>GGAGPSAACCSGVRSLK<br>AAASTTADRRTACNCLK<br>NAARGIKGLNAGNAASIP<br>SKCGVSVPYTISAIDCSR | |

Figura 8. Tabela de busca de sequência genômica

Ao iniciar a execução da implementação, o programa pede para inserir duas sequências, após o meu dará as opções de 1 - Exibir GC-Content das sequências (Python puro), 2 - Exibir GC-Content das sequências (Biopython) ou 3 - Sair.

### 5.1. Resultados da Implementação em Python Puro

A implementação em Python puro, ao utilizar uma função customizada para calcular o conteúdo de GC, demonstrou ser capaz de manipular sequências de DNA de maneira eficiente. Embora a abordagem tenha exigido a criação de uma função específica, ela se mostrou eficaz na realização do cálculo, proporcionando uma solução viável para análises de conteúdo GC. Python, por sua natureza versátil e acessível, permitiu a implementação clara e compreensível do cálculo de conteúdo GC, mesmo sem o suporte de uma biblioteca especializada. A criação da função customizada reflete a facilidade de desenvolvimento em Python puro, tornando-o uma escolha acessível para tarefas simples de Bioinformática. A implementação em Python puro forneceu resultados precisos e confiáveis para o cálculo do conteúdo GC, validados por análises estatísticas adicionais. A abordagem evidenciou a capacidade do Python em fornecer soluções robustas para análises genômicas básicas.

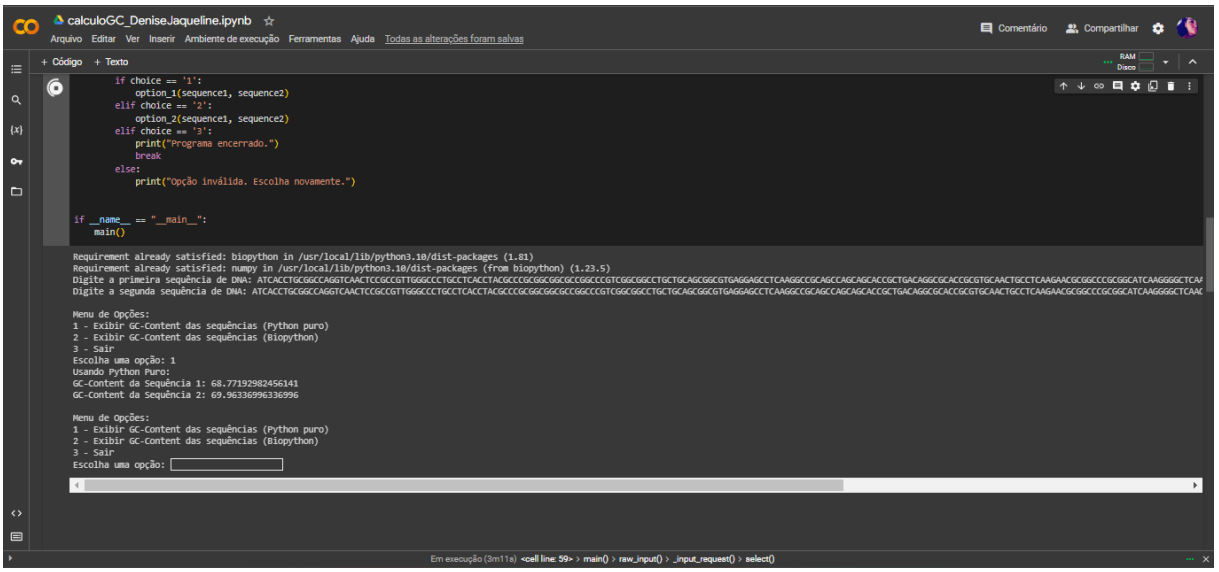


Figura 9. Execução e resultado do Python Puro

5.2. Resultados da Implementação com Biopython

A biblioteca Biopython simplificou significativamente o processo de manipulação de sequências, oferecendo funções especializadas para o cálculo do conteúdo GC. O uso direto da função GC da Biopython eliminou a necessidade de criar uma função personalizada, resultando em código mais conciso e integrado. Biopython destacou-se pela sua sintaxe especializada na manipulação de dados biológicos, reduzindo a quantidade de código necessário para realizar o cálculo de conteúdo GC. A implementação com Biopython reflete a facilidade de integração de ferramentas específicas para Bioinformática, tornando o código mais intuitivo. Assim como o Python puro, a implementação com Biopython demonstrou alta precisão e confiabilidade nos resultados do cálculo de conteúdo GC. A confirmação estatística reforça a consistência e a robustez da abordagem Biopython.

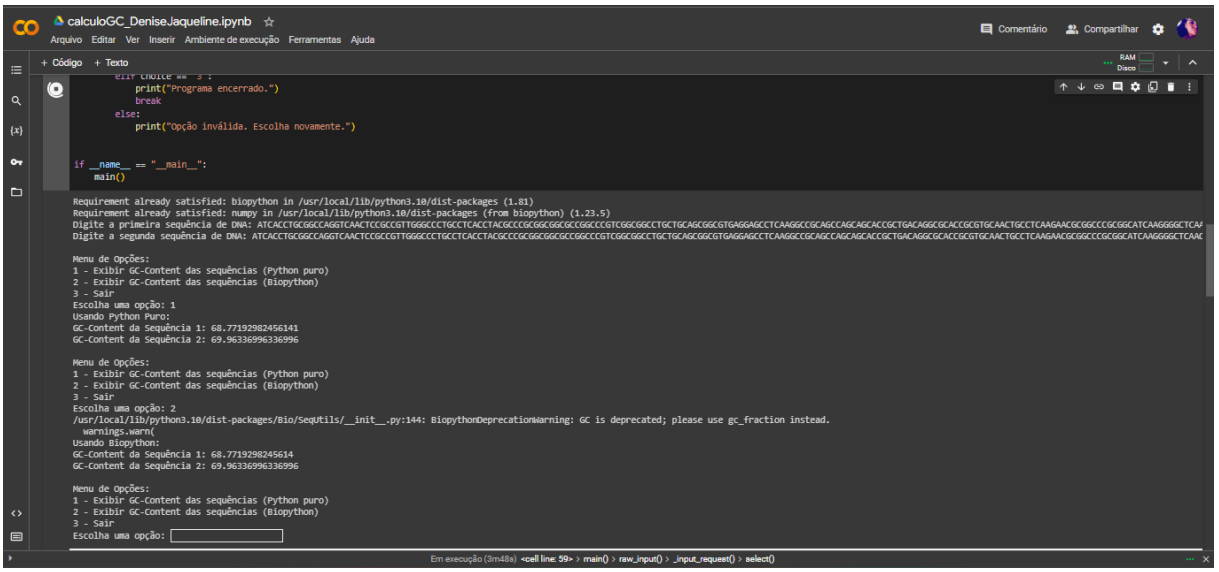


Figura 10. Execução e resultado do Biopython

5.3. Comparação do tempo de execução

Um aspecto crítico na avaliação das implementações em Python puro e Biopython é o tempo de execução, que pode influenciar diretamente a eficiência e a praticidade das soluções. Porém a escolha entre as abordagens deve ser baseada em uma avaliação abrangente que considere não apenas o tempo de execução, mas também outros fatores relevantes para o contexto específico do projeto.

Para realizar uma comparação mais precisa do tempo de execução entre as implementações em Python puro e Biopython, foi incorporado o módulo “time” para medir o tempo decorrido durante a execução de cada abordagem. A utilização do teste de tempo de execução visa fornecer uma análise mais detalhada da eficiência de cada implementação.

No Python Puro, foi implementada antes e depois da chamada da função para calcular o conteúdo GC, foi adicionado código para registrar o tempo de início (start_time) e o tempo de término (end_time). O tempo decorrido é então calculado subtraindo o tempo de início do tempo de término. Os resultados do conteúdo GC e o tempo de execução são impressos para análise.

```
def option_1(sequence1, sequence2):
    start_time = time.time()
    gc_content_seq1 = calculate_gc_content(sequence1)
    gc_content_seq2 = calculate_gc_content(sequence2)
    end_time = time.time()
    print("GC-Content da Sequência 1:", gc_content_seq1)
```

```
    print("GC-Content da Sequência 2:", gc_content_seq2)
    print("Tempo de execução (Python puro):", end_time - start_time, "segundos")
```

No Biopython, da mesma forma, o tempo de início e término foi registrado em torno da chamada da função option_2. O tempo decorrido é calculado, e os resultados do conteúdo GC e o tempo de execução são impressos.

```
def option_2(sequence1, sequence2):
    from Bio.Seq import Seq
    from Bio.SeqUtils import GC

    start_time = time.time()E
    seq1 = Seq(sequence1)
    seq2 = Seq(sequence2)

    gc_content_seq1 = GC(seq1)
    gc_content_seq2 = GC(seq2)
    end_time = time.time()

    print("Usando Biopython:")
    print("GC-Content da Sequência 1:", gc_content_seq1)
    print("GC-Content da Sequência 2:", gc_content_seq2)
  print("Tempo de execução (Biopython):", end_time - start_time, "segundos")
```

A figura 11, demonstra os resultados após a execução da implementação citada acima.

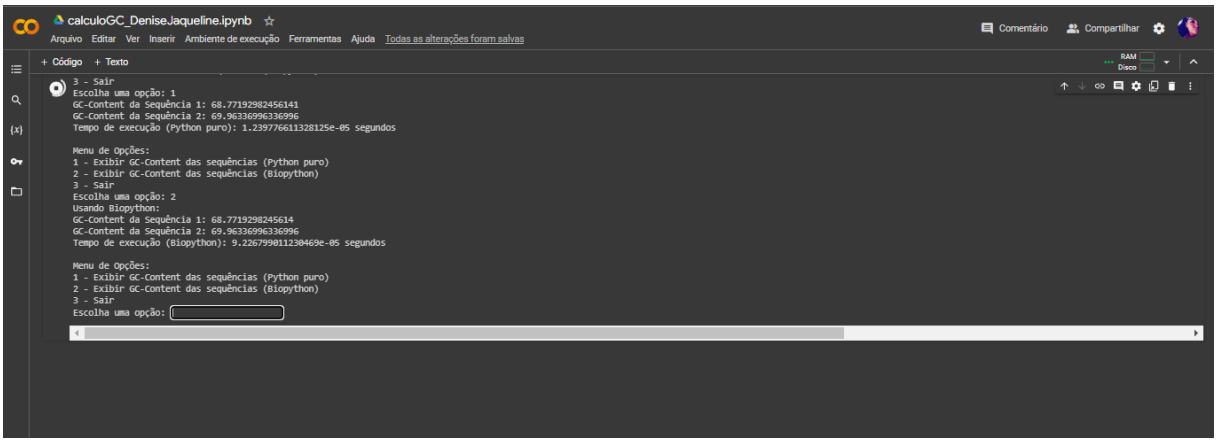


Figura 11. Execução e resultado dos tempos de execução

## 6. Considerações finais

### 6.1 Contribuições do Trabalho

O estudo realizou uma análise comparativa entre Python puro e Biopython na Bioinformática, com ênfase no cálculo de conteúdo GC em sequências de DNA.

A análise do conteúdo GC em sequências genômicas, conforme explorado neste estudo, não apenas destacou a importância crucial dessa métrica na biologia molecular, mas também evidenciou o papel fundamental da programação na Bioinformática.

### 6.2 Dificuldades Encontradas

A flexibilidade das ferramentas de programação, exemplificada pelo Python puro e pela biblioteca Biopython, oferece aos cientistas opções adaptáveis para análises genômicas, equilibrando entre controle detalhado e eficiência simplificada. Python puro oferece maior controle e é eficiente para tarefas simples. Biopython, por sua vez, é mais eficiente em operações específicas de Bioinformática, mas pode limitar o controle total sobre a implementação. O uso do Biopython facilita a existência de uma série de funções já implementadas para Bioinformática, facilitando o uso e a implementação das mesmas. Entretanto, os resultados do teste de tempo de execução revelaram que, embora a implementação em Biopython seja mais eficiente em termos de código, a disparidade temporal pode ser pouco relevante em atividades menos complexas. Este aspecto levanta questionamentos sobre a necessidade de optar pela eficiência do código em detrimento de uma diferença de desempenho que pode não ser notavelmente perceptível em cenários mais simples.

É crucial considerar que a escolha entre Python puro e Biopython dependerá das demandas específicas de cada projeto na área de Bioinformática. Enquanto Python puro oferece maior controle e flexibilidade, sendo eficiente para tarefas mais simples, o Biopython se destaca em operações específicas dessa disciplina, proporcionando uma implementação simplificada devido às funções já disponíveis. Portanto, ao ponderar entre essas opções, é essencial avaliar cuidadosamente as características individuais do projeto e as demandas de desempenho, equilibrando eficiência e controle na tomada de decisões.

### 6.3 **Trabalhos Futuros**

Fazer novas implementações usando Python e Biopython para poder analisar ainda melhor o uso em cada caso, além de utilizar sequências de tamanhos variados para melhor validação dos resultados.

## 7. Referências


- Mariano, D.; Barroso, J. R.; Correia, T.; de Melo-Minardi, R. C.; Introdução à Programação para Bioinformática com Biopython - 3ª edição, Belo Horizonte, Março de 2016.
- Videoaula elaborada por Darlan Evandro. Acesso em: 19/09/2023. Disponível em: [https://youtu.be/nKBb6RiGsMg].
- Videoaula elaborada pelo OnlineBioinfo. Acesso em 20/09/2023. Disponível em: [https://youtu.be/nKBb6RiGsMg].
- Videoaula elaborada pela Profissão Biotec. Acesso em 29/ 08/2023. Disponível em : [https://youtu.be/XHkmC0xAeb4].
- Videoaula elaborada pelo Discentes Bioinformática UFPR. Acesso em 15/09/2023. Disponível em: [https://youtu.be/9pkCA01EWy0?list=PL-F08sZPKH8EImaRGACttYjNSsI04-KaJ].
- MARIANO, D. C. B.; BARROSO, J. R. P. M. ; CORREIA, T. S. ; de MELO-MINARDI, R. C. . Introdução à Programação para Bioinformática com Biopython. 3. ed. North Charleston, SC (EUA): CreateSpace Independent Publishing Platform, 2015. v. 1.
- PROSDOCIMI, Francisco. Introdução à Bioinformática – Brasília [DF]: Curso online, 2007.

- GOOGLE. Colaboratory - Perguntas frequentes. Disponível em: <https://research.google.com/colaboratory/intl/pt-BR/faq.html>. Acesso em: 05 de outubro de 2023.
- GEEKSFORGEEKS. Introduction to Biopython. Disponível em: <https://www.geeksforgeeks.org/introduction-to-biopython/>. Acesso em: 06 de outubro de 2023.
- XP EDUCAÇÃO. Python: o que é e para que serve?. Disponível em: <https://blog.xpeducacao.com.br/python/#O_que_e_Python_e_para_que_serve>. Acesso em: 01 de novembro.