



PROJETO INTERDISCIPLINAR

Conversor de Base

Números decimais para outras bases

Alunos:

RGM	Nome
26109638	Gabriel Gomes da Silva
26235552	Gustavo Henrique de Paula Assis
27127516	Isabela Cristina da Silva
25971441	Vinicius Aparecido Lourenço Souza

São Paulo

2021

UNIVERSIDADE CRUZEIRO DO SUL

PROJETO INTERDISCIPLINAR

Conversor de Base

Números decimais para outras bases

Trabalho apresentado como parte do requisito para aprovação na Disciplina de Projeto Interdisciplinar do curso de Programação de Computadores e de Organização e Arquitetura de Computadores da Universidade Cruzeiro do Sul.

Orientadores: Prof. Douglas Almendro e Prof. Jean Rosa.

São Paulo

2021

Sumário

1. Apresentação:	4
1.1 Justificativa e Motivação.....	4
1.2 Dados do Programa	4
2 Requisitos de Programação de Computadores.....	5
3 Requisitos de Organização e Arquitetura de Computadores	8
4 Consideração finais.....	10
5 BIBLIOGRAFIA	11

1. APRESENTAÇÃO:

1.1 Justificativa e Motivação

A escolha do grupo sobre a atividade proposta foi a **opção 1** (Conversão de decimal para as bases binário, hexadecimal e octadecimal). Preferimos esta opção pois seus requisitos são as que melhores desafiam e instigam nosso conhecimento sobre a linguagem ensinada.

Acreditamos que, após a conclusão deste projeto, nos tornaremos pessoas mais competentes em termos de lógica e sintaxe de programação.

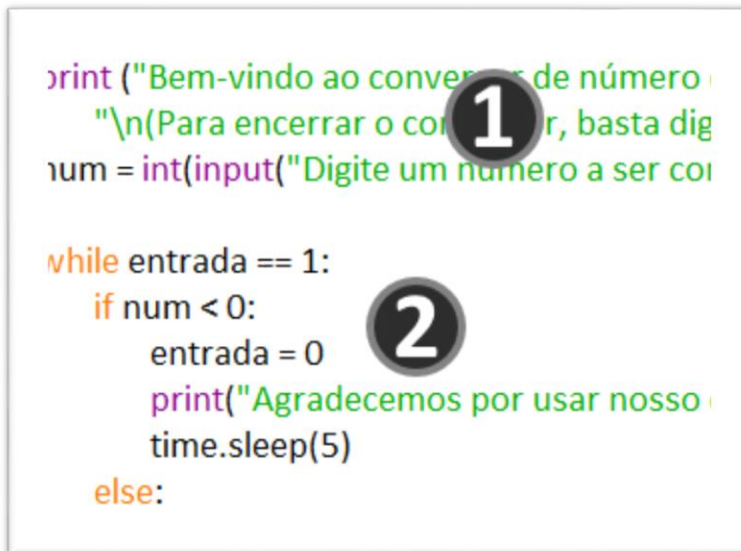
1.2 Dados do Programa

O objetivo deste programa é converter números inteiros de bases decimais para bases binárias, hexadecimais e octais mantendo o foco em facilidade e agilidade na interação com o usuário.

Em pouco segundos e com a inserção de apenas um único dado, o programa retorna o valor introduzido em diferentes bases, com a possibilidade da adição de dados em um modo contínuo.

2 REQUISITOS DE PROGRAMAÇÃO DE COMPUTADORES

Nosso código é inteiramente escrito na linguagem Python usando o software Python 3.8 IDLE. Abaixo descrevemos alguns pontos importantes discutidos durante o planejamento e a execução de nosso programa:



```
print ("Bem-vindo ao conversor de número |
      "\n(Para encerrar o código, basta dig
num = int(input("Digite um número a ser coi

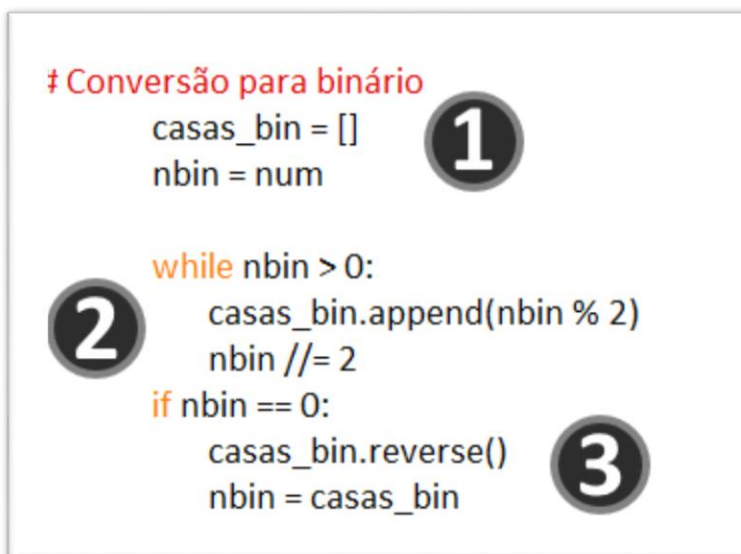
while entrada == 1:
    if num < 0:
        entrada = 0
    print("Agradecemos por usar nosso |
    time.sleep(5)
else:
```

The image shows a snippet of Python code for a number converter. It includes a welcome message, a prompt for the user to enter a number, and a loop that checks if the input is negative. If negative, it sets 'entrada' to 0 and prints a thank you message with a 5-second delay. The code is annotated with a '1' in a circle next to the input line and a '2' in a circle next to the loop condition.

Figura 1 - Entrada de dados. Fonte: conversor de bases.

Como mostrado na figura 1, o software se auto descreverá em poucas palavras e solicitará ao usuário um único valor. Uma vez que as conversões são feitas apenas em cima de números inteiros maiores ou iguais à 0, caso um número negativo seja inserido, o programa entenderá que o usuário está satisfeito e não necessita mais realizar nenhuma conversão, portanto a variável “entrada” mudará seu valor para 0 e, com isso, a aplicação se encerrará.

Através da biblioteca “time”, pudemos adicionar um temporizador de 5 segundos a fim de permitir/facilitar a leitura do texto mostrado na tela.



```
# Conversão para binário
casas_bin = []
nbin = num

while nbin > 0:
    casas_bin.append(nbin % 2)
    nbin //= 2
if nbin == 0:
    casas_bin.reverse()
    nbin = casas_bin
```

The image shows the binary conversion logic. It initializes a list 'casas_bin' and sets 'nbin' to the input number. A while loop processes the number by taking modulo 2 and integer division by 2. When the number reaches 0, the list is reversed and assigned back to 'nbin'. The code is annotated with a '1' in a circle next to the initialization, a '2' in a circle next to the while loop, and a '3' in a circle next to the reversal step.

Figura 2 - Conversão para binário. Fonte: conversor de bases.

Observando a figura 2, podemos perceber a conversão sendo realizada basicamente em três etapas: 1- definição de variáveis e listas, 2- cálculo de resto e divisão exata e, por último, 3- correção da ordem dos valores inseridos e registrados.

Nesta parte, buscamos um código que fosse otimizado e direto ao ponto. Primeiro o programa calcula o resto da divisão por 2 e adiciona este resultado na lista “casas_bin”, depois, divide o valor anteriormente calcula por 2. Através do laço “while”, este processo é continuamente realizado até que a variável “nbin” se torne 0.

Devemos observar que, ao fazermos a conversão de bases manualmente, nós sempre coletamos o resto da divisão em uma ordem **invertida**, portanto, a fim de imitar este processo, utilizamos do método “reverse()” para inverter a ordem dos dados inseridos.

Este procedimento de conversão é repetido para todas as outras bases mudando apenas o divisor.

```
for i in range(len(casas_hex)):
    if casas_hex[i] == 10:
        casas_hex[i] = "A"
    elif casas_hex[i] == 11:
        casas_hex[i] = "B"
    elif casas_hex[i] == 12:
        casas_hex[i] = "C"
    elif casas_hex[i] == 13:
        casas_hex[i] = "D"
    elif casas_hex[i] == 14:
        casas_hex[i] = "E"
```

Figura 3 - Ajuste de dados. Fonte: conversor de bases.

Devemos destacar durante a conversão para bases hexadecimais o uso de letras A, B, C, D, E e F para representar os números 10, 11, 12, 13, 14 e 15, respectivamente.

Como visto na figura 3, utilizamos o laço “for” para detectar e realizar a mudança em determinado dado da lista quando necessário.

```
resultado_bin = ""  
resultado_oct = ""  
resultado_hex = ""  
  
for i in range(len(casas_bin)):  
    resultado_bin += str(casas_bin[i])  
  
for i in range(len(casas_oct)): 2  
    resultado_oct += str(casas_oct[i])  
  
for i in range(len(casas_hex)):
```

Figura 4 – Dados em lista para *string*. Fonte: conversor de bases.

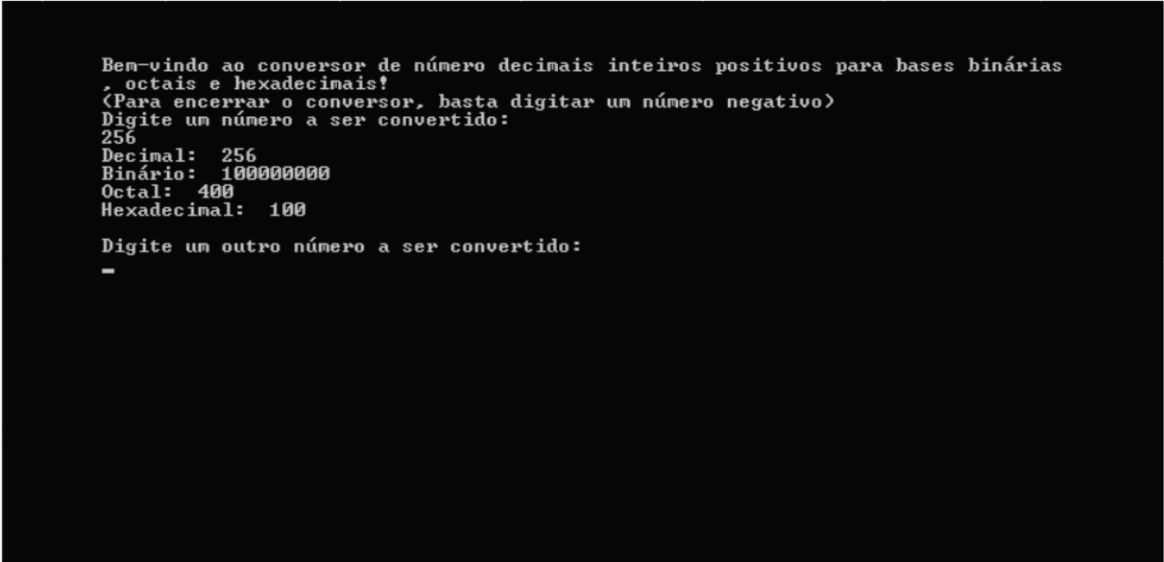
Seguidamente da conversão e do registro de dados, destinamos uma parte do código para “converter” os valores em lista para simples *strings*.

Conforme mostrado na figura 4, através da estrutura “*for*” conseguimos coletar e adicionar os dados de uma lista para uma variável *string*. Desse modo, a aplicação conseguirá exportar dados mais amigáveis e menos poluentes ao usuário.

Em resumo, a programação é feita de forma organizada e otimizada. Quando analisado de perto, notamos que cada linha do código foi cuidadosamente programada para um propósito e, assim, não desperdiça desempenho/espço.

3 REQUISITOS DE ORGANIZAÇÃO E ARQUITETURA DE COMPUTADORES

Com todos os dados devidamente calculados e registrados nas variáveis, a clareza dos dados a serem apresentados é de suma importância, afinal o propósito inteiro do conversor se baseia em volta disso.



```
Bem-vindo ao conversor de número decimais inteiros positivos para bases binárias
, octais e hexadecimais!
(Para encerrar o conversor, basta digitar um número negativo)
Digite um número a ser convertido:
256
Decimal: 256
Binário: 100000000
Octal: 400
Hexadecimal: 100

Digite um outro número a ser convertido:
-
```

Figura 5 – Resultado. Fonte: conversor de bases.

A seguir, explicaremos linha por linha o resultado mostrado vide figura 6. Primeiro, é exibido o número digitado pelo usuário em decimal, logo abaixo podemos ver o equivalente em bases binárias, octais e hexadecimais, respectivamente.

Preferimos mostrar todas as conversões em uma única tela a fim de facilitar o uso da aplicação.

Por último, o software solicita ao usuário um novo valor a ser convertido. Lembrando que, para finalizar, basta digitar um número negativo.

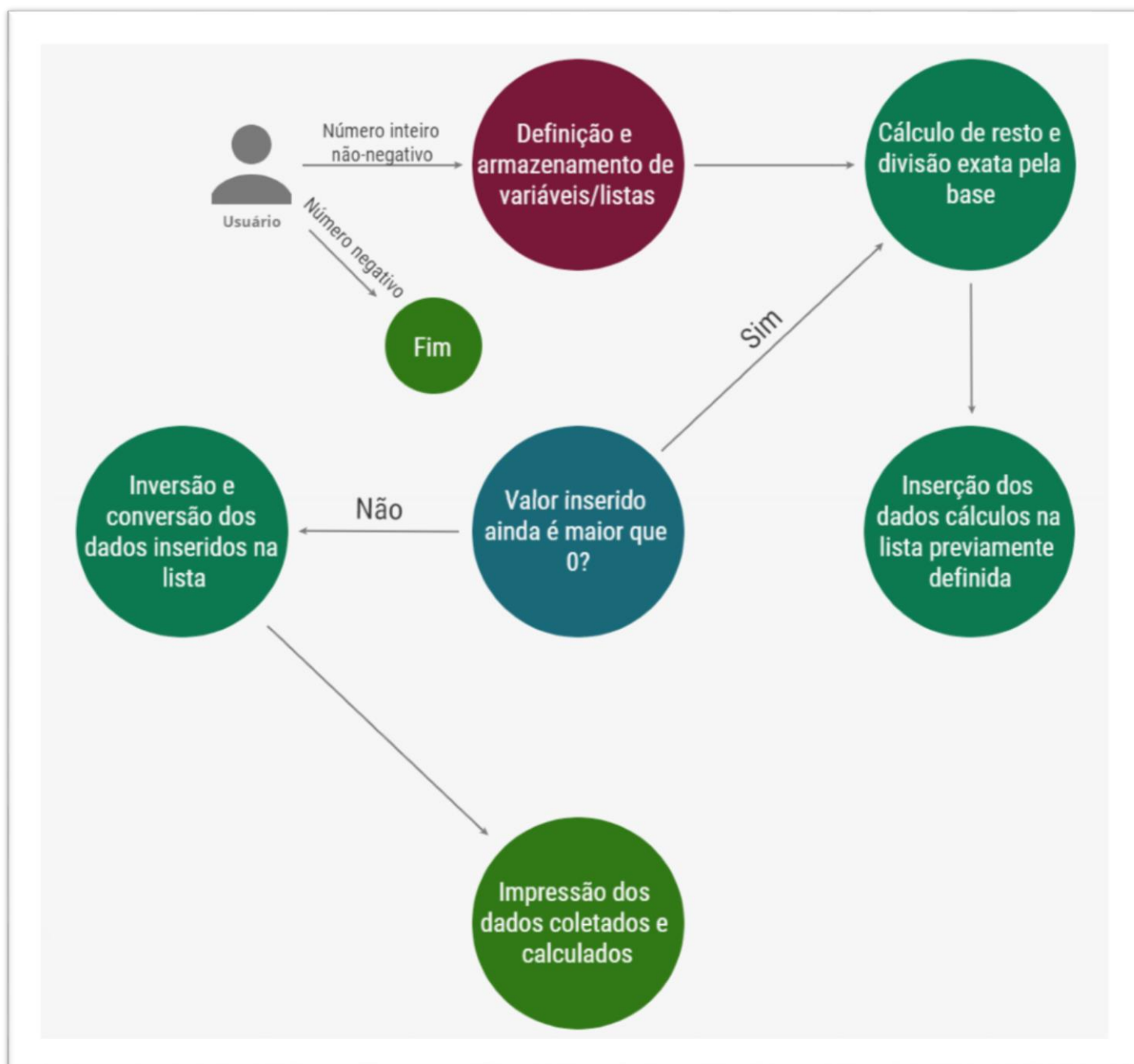


Figura 6 – Diagrama de Cálculos. Fonte: conversor de bases.

Na figura 6, podemos ver um diagrama descrevendo o procedimento de conversão conforme explicado no capítulo 2.

No geral, a interação entre o usuário e o código é feita de maneira simples e rápida. Não há motivos para dificultar a vida do usuário com múltiplas requisições de dados quando o propósito pode ser cumprido de maneira compreensível e amigável.

4 CONSIDERAÇÃO FINAIS

Inicialmente, pensávamos que um conversor de bases numéricas requeria um número superabundante de linhas de código, portanto decidimos parar de pensar na programação em si e analisar como é feita a conversão manuscrita.

Com o raciocínio lógico pronto, ficou claro como seria o passo-a-passo da sintaxe que usaríamos. Assim, fomos capazes de codificar de maneira otimizada sem grandes dificuldades.

5 BIBLIOGRAFIA

INC., Venngage. Criador de Infográficos. **Venngage**, 2021. Disponível em: <https://pt.venngage.com/>. Acesso em: 16 mai. 2021.

SOFTWARE FOUNDATION, Python. Programming Language. **Python**, 2021. Disponível em: <https://www.python.org/>. Acesso em: 16 mai. 2021.

DO SUL, Cruzeiro. Cursos presenciais e EADs. **Cruzeiro do Sul**, 2021. Disponível em: <https://www.cruzeirodosul.edu.br/>. Acesso em: 16 mai. 2021.