



Computação 1 – Aula 1

Números, Algoritmos e Python

Prof. Cesar Raitz

1. Introdução

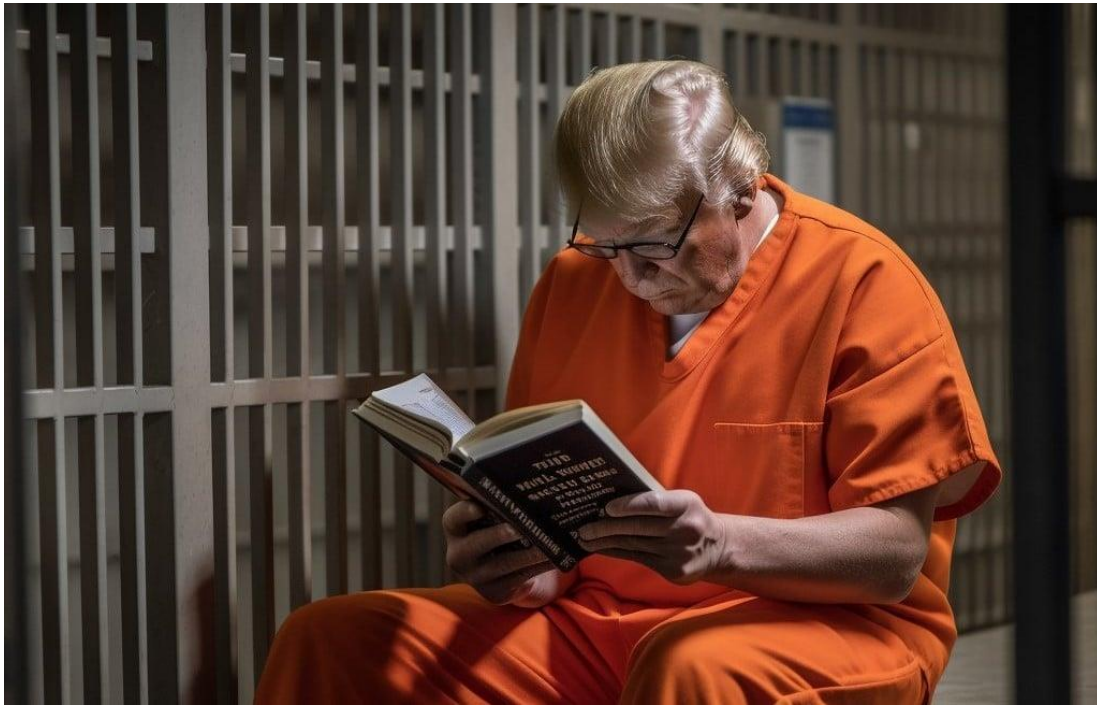
- Primeiro, faça essa conta:

$$1 + 1 = ?$$

- Agora tente essa:

$$132.576.932,234 \times 967.554.112,437 = ?$$

- A primeira é intuitiva e qualquer criança sabe responder.
A segunda requer muito esforço e é possível que até seu professor erre a conta. Por quê?
- Claro que um matemático habilidoso é capaz de calcular operações como essa em questão de segundos. Mas o cérebro não evoluiu com esse propósito.
- Desde a antiguidade, ferramentas foram construídas para poder abstrair as contas e nos ater à resolução de problemas.
 - Sistemas numéricos (de contagem)
 - Instrumentos de calcular
 - Técnicas de cálculo
 - Formas de resolver problemas
- Chegamos ao ponto da IA (inteligência artificial) gerar sons de vozes ou imagens realistas. Você chegou a ver os *deep fakes* do Trump e do Harry Potter (2022/23)?



Fonte: Facebook

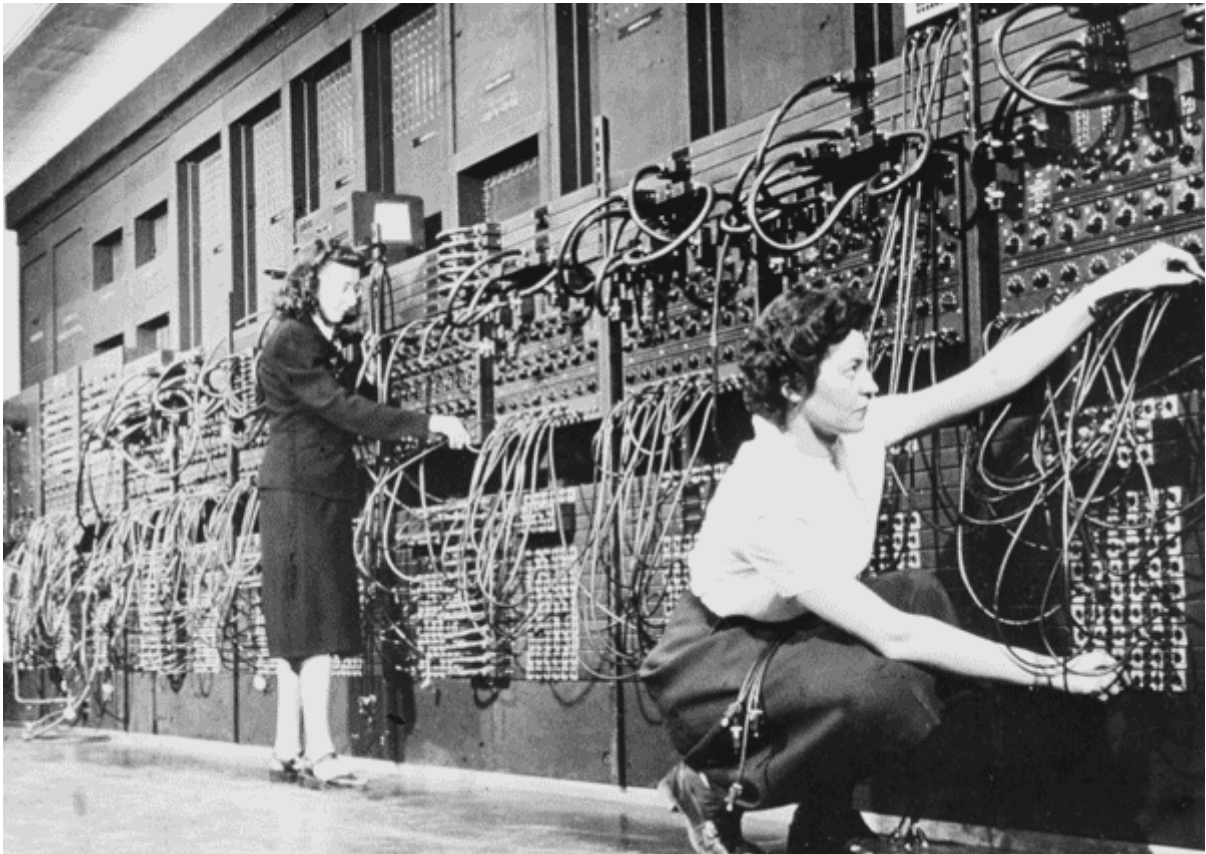
2. Sistemas de contagem

- Estamos acostumados a contar até 9, depois temos uma dezena, até 9 de novo e depois duas dezenas. Depois de 9 dezenas e 9 unidades, temos uma centena. Essa é a **base decimal**.
- O ábaco é bom para somar e subtrair, mas para multiplicar?



Fonte: Wikipedia

- Charles Babbage até tentou, mas eram tantas engrenagens...
- Durante a corrida espacial, já havia calculadoras mecânicas, só que as matemáticas eram mais rápidas (você deve ter visto em “Estrelas Além do Tempo”).
- O ENIAC surgiu quase ao mesmo tempo.
- Os matemáticos já sabiam que é possível decompor todo tipo de informação na **base binária**, com apenas dois algarismos: 0 e 1.
- Os engenheiros perceberam que é mais fácil construir computadores com **bits** (dígitos binários ou *binary digits*).



Fonte: Wikipedia

3. Algoritmos

- Somar, multiplicar e dividir, sabemos o que é, mas precisamos de uma receita para fazer certo?

Um algoritmo é uma sequência de instruções para resolver um problema da forma eficiente.

- Podemos pensar num algoritmo como uma receita de bolo:
 - A lista de ingredientes é o que você tem inicialmente, a **entrada**
 - O passo-a-passo é o que gera o resultado, são as **instruções**
 - O bolo é o resultado, ou a **saída**

- Quer um exemplo?

Algoritmo: Soma de números

1. Ler dois números a e b .
2. Pegar um dígito mais à direita de a e um dígito mais à direita de b .
3. Somar os dois dígitos.
4. Coloque a unidade da soma no resultado, na mesma posição dos dígitos somados.
5. **Se** a soma passar de 9 **então** guarde 1 para a próxima soma.
6. **Se** não houver mais dígitos em a ou b , **terminar**.
7. Pegar os próximos dígitos mais à direita de a e b .
8. Somar os dígitos e adicionar 1, **caso** esteja guardado.
9. **Voltar** ao passo 4.

- Repare que quase todas as instruções começam com verbos, indicando **ação**.
- Algumas instruções dependem de uma condição, como as 5, 6 e 8. São chamadas **estruturas condicionais**.
- Algumas instruções indicam repetição como a 9. São chamadas **estruturas de repetição**.
- O algoritmo acima foi escrito em **pseudo-código** que tende a ser mais verbal do que linguagens de programação de computadores.

4. Linguagem de máquina

- O algoritmo da soma não vai funcionar em computador nenhum se não for traduzido antes para linguagem de máquina.
- Suponha que eu tenha um computador (chamado *CompUm*) com *instruções de apenas um bit*. O manual especifica as instruções:

0 = acende a lâmpada
1 = apaga a lâmpada

- Então, o que acontece quando o *CompUm* executa os seguintes códigos? (leia da esquerda para a direita)
 - 0.0.0.0
 - 0.1.0.1
 - 0.0.1.1

5. Python

- Comp.1 é um curso para você aprender a criar estratégias para solucionar problemas.
- Claro que não queremos escrever apenas **pseudo-código**, ficaria muito abstrato.
- Então, usaremos a linguagem de programação Python para escrever programas.
- É uma das linguagens mais acessíveis e usadas hoje em dia.

- Veja um exemplo:

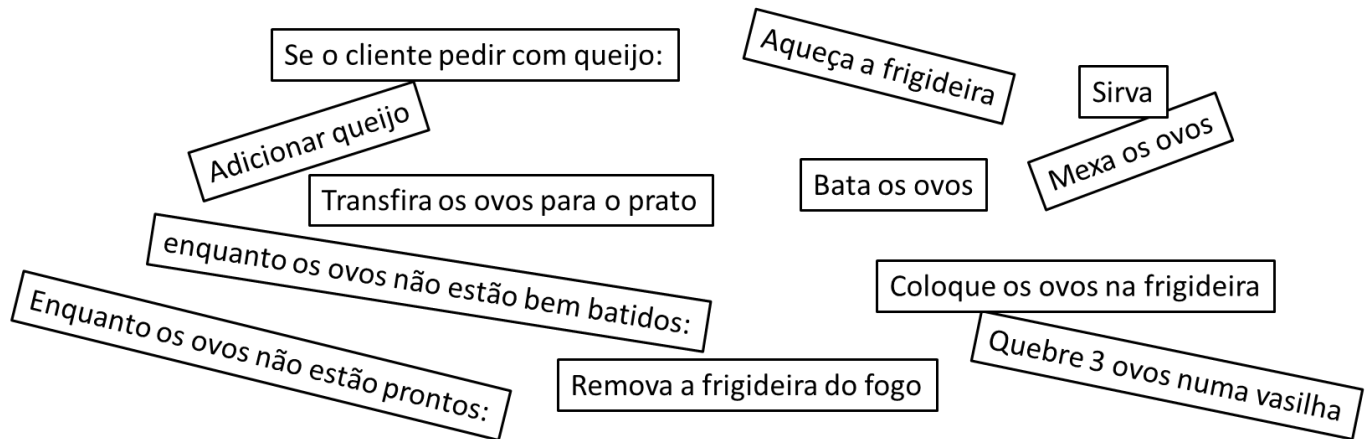
calculadora.py	
1.	<code>print("Calculadora")</code>
2.	
3.	<code>while True:</code>
4.	<code> a = float(input("Digite a:"))</code>
5.	<code> b = float(input("Digite b:"))</code>
6.	
7.	<code> print("Escolha a opção:")</code>
8.	<code> print("1 - somar")</code>
9.	<code> print("2 - subtrair")</code>
10.	<code> print("3 - multiplicar")</code>
11.	<code> print("4 - dividir")</code>
12.	<code> print("0 - sair")</code>
13.	<code> op = int(input("? "))</code>
14.	
15.	<code> if op == 0:</code>
16.	<code> print("Tchau!")</code>
17.	<code> break</code>
18.	<code> elif op == 1:</code>
19.	<code> print(a+b)</code>
20.	<code> elif op == 2:</code>
21.	<code> print(a-b)</code>
22.	<code> elif op == 3:</code>
23.	<code> print(a*b)</code>
24.	<code> elif op == 4:</code>
25.	<code> print(a/b)</code>

- Neste código, há **instruções** e **variáveis**, que veremos em detalhes ao longo do curso.

6. Exercícios

Exercício 1. Algoritmo da omelete

Você deve recriar um algoritmo perfeito para uma omelete partindo das instruções a seguir, deixadas por um famoso Chef:



Exercício 2. Pseudo-código do semáforo

Você deve programar um semáforo para controlar o fluxo de carros num cruzamento. Não instruções prontas então você deve pensar na sequência de instruções para o semáforo. Deve acender qual luz? Esperar quanto tempo?

Exercício 3. Semáforo em linguagem de máquina

Finalmente você recebeu um computador capaz de operar o semáforo! Cada instrução ocupa 4 bits na memória. O manual mostra quais são as instruções:

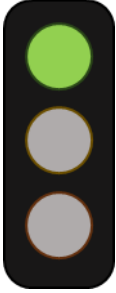
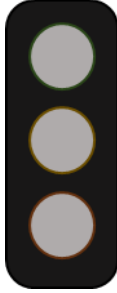
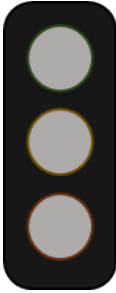
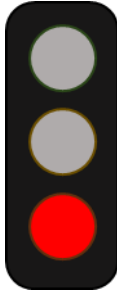
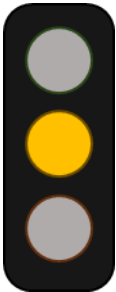
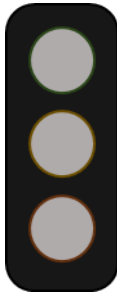
0100 = acende luz vermelha	0101 = aguarda 30 segundos	0111 = apaga todas as luzes
0010 = acende luz amarela	1010 = aguarda 3 minutos	1011 = acende todas as luzes
0001 = acende luz verde	0000 = interrompe o programa	1000 = volta à primeira instrução

Só falta escrever o código!

7. Soluções

Exercício 2. Pseudo-código do semáforo

Eis o que o semáforo deve fazer:

<p>1. Ligar a luz verde 2. Esperar 5 min</p> 	<p>6. Apagar a luz amarela</p> 
<p>3. Apagar a luz verde</p> 	<p>7. Ligar a luz vermelha 8. Esperar 3 min</p> 
<p>4. Ligar a luz amarela 5. Esperar 30 seg</p> 	<p>9. Apagar a luz vermelha 10. Voltar ao passo 1</p> 

Exercício 3. Semáforo em linguagem de máquina

Esse código será uma tradução quase direta do pseudo-código. Note que aguardar e esperar são sinônimos. Precisamos adaptar algumas instruções para operar neste computador, sabendo que $5 \text{ min} = 3 \text{ min} + 4 \times 30 \text{ seg}$, e que podemos apagar qualquer luz com a instrução 0111.

Sua solução pode variar um pouco, sem esquecer de apagar as luzes e usar 1000 no final, para continuar alternando as luzes!

Aula 1 – Números, Algoritmos e Python

Endereço na memória	Instrução	O que faz?
0	0001	Acende a luz verde
1	1010	Aguarda 3 min
2	0101	Aguarda 30 seg
3	0101	Aguarda 30 seg
4	0101	Aguarda 30 seg
5	0101	Aguarda 30 seg
6	0111	Apaga as luzes
7	0010	Acende a luz amarela
8	0101	Aguarda 30 seg
9	0111	Apaga as luzes
10	0100	Acende a luz vermelha
11	1010	Aguarda 3 min
12	0111	Apaga as luzes
13	1000	Volta ao início

} = 5min