

AT — Maioria absoluta em Python: um programa de eleições

10.º Ano MACS 3 páginas Atividade de tecnologia

Objetivo: Construir um programa simples em Python (iniciação à linguagem) que, com as votações de 3 candidatos, determine quem garante a maioria absoluta — ou convoca a 2.ª volta — praticando abstração, decomposição, algoritmo e depuração (pensamento computacional).

Material

Material	Nota
Computador com editor de Python (IDLE, Thonny ou ambiente online)	1 por aluno/grupo
Folha com os casos de teste (página 2)	entradas e saídas esperadas
Calculadora	verificação de percentagens

Normas de segurança e de trabalho

- Guardar o ficheiro frequentemente (**ficheiro** → **guardar**) e trabalhar com **cópias de segurança** por caso de teste.
- Escrever **comentários** (#) a explicar cada bloco — o código é para ser lido por outros.
- Rigor de sintaxe: pontos-e-vírgula não existem em Python; **recuo** (**indentação**) **obrigatório** nos blocos `if/elif/else`.
- Nunca copiar a solução sem a executar e explicar — o objetivo é o processo, não o texto.

Procedimento

1. **Decompor o problema** em passos: ler votos → somar → calcular o limiar (mais de metade) → decidir → imprimir.
2. Escrever o **pseudocódigo** desses passos no caderno antes de programar.
3. Programar a **leitura** dos votos de A, B e C (valores inteiros) e o **total**.
4. Programar o **limiar**: `limiar = total / 2` — a maioria absoluta exige **mais** que metade (`>`, nunca `>=`).
5. Programar a decisão com **if / elif / else**: se algum candidato tem `votos > limiar`, é eleito; senão, imprimir «2.ª volta».
6. Acrescentar as **percentagens** de cada candidato (`votos ÷ total × 100`).
7. **Testar** com os 3 casos da página 2 — incluindo o caso de fronteira — e **depurar** até as saídas coincidirem com as esperadas.
8. Alterar o programa para **4 candidatos** e registar o que mudou.

Código de referência (executar e compreender linha a linha):

```
# Maioria absoluta com 3 candidatos - MACS 10.º ano
a = int(input("Votos A: "))
b = int(input("Votos B: "))
c = int(input("Votos C: "))
total = a + b + c
limiar = total / 2                # maioria absoluta = mais de metade
print("Total:", total, "| precisa de mais de", limiar)

if a > limiar:
    print("Eleito na 1.a volta: A -", a / total * 100, "%")
elif b > limiar:
    print("Eleito na 1.a volta: B -", b / total * 100, "%")
elif c > limiar:
    print("Eleito na 1.a volta: C -", c / total * 100, "%")
else:
    print("Sem maioria absoluta -> 2.a volta")
    print("Melhor classificado:", max(a, b, c), "votos")
```

Registo de dados — casos de teste

Executa o programa com cada conjunto de entradas e regista a saída observada. Compara com a **saída esperada** — se divergirem, depura (revê a comparação, a soma e o recuo).

Tabela 1 — Execuções

Execução	Entradas (A, B, C)	Total	Limiar (total ÷ 2)	Saída esperada	Saída observada
R1	700, 300, 200			Eleito A (58,3 %)	
R2	550, 430, 220			Sem maioria absoluta → 2.ª volta	
R3 (fronteira)	600, 400, 200			Sem maioria absoluta → 2.ª volta	

Porque R3 é o caso-chave: A tem exatamente 50,0 % — e 50 % **não** é maioria absoluta (é preciso **mais de** metade). O operador > recusa-o; um >= erraria a eleição.

Tratamento de resultados

(a) Execução R2 — passo a passo (entradas 550, 430, 220):

$total = 550 + 430 + 220 = 1200 \cdot limiar = 1200 \div 2 = 600$

$550 \leq 600 \text{ (não)} \cdot 430 \leq 600 \text{ (não)} \cdot 220 \leq 600 \text{ (não)} \rightarrow \text{else: sem maioria absoluta} \rightarrow 2.^a \text{ volta}$

(b) Percentagens de R2 (base = total):

$A: 550 \div 1200 \times 100 = 45,8 \% \cdot B: 430 \div 1200 \times 100 = 35,8 \% \cdot C: 220 \div 1200 \times 100 = 18,3 \%$

Soma = 99,9 % — a diferença é **arredondamento** a 1 decimal, não erro de contagem (a soma exata é 100 %).

(c) Votos necessários — «mais de metade»:

Total de votos expressos	Metade	Votos mínimos para maioria absoluta
1200 (par)	600	601
75 (ímpar)	37,5	
1000 (par)	500	

(d) Extensão a 4 candidatos — o que mudaste: +1 variável e +1 bloco `elif`; o limiar e a regra não mudam (o algoritmo é **reutilizável** — é isso que o programa tem de demonstrar).

Análise e conclusão

1. Porque é que o programa usa `> limiar` e não `>= limiar`? Usa o caso R3 para justificar.
2. Em Python 3, `total / 2` devolve sempre um número com decimal. Isso atrapalha a comparação em R3 (`600 > 600,0`)? E que cuidado de rigor impõe na leitura do limiar?
3. Caso de fronteira com total ímpar: com 75 votos, o limiar é 37,5 — quantos votos mínimos garante a maioria absoluta? E o candidato com 38 votos foi eleito mesmo com só 50,7 %? Explica.
4. **Depuração:** um colega escreveu `if a >= limiar: print("Eleito A")` e o R3 elegeu A. Diagnostica o erro e escreve a linha correta. Que caso de teste é este e porque é que só ele apanha o erro?
5. Como alterarias o programa para que, na 2.ª volta, imprima os **dois primeiros** candidatos (em ordem)? Indica a construção necessária (lista/tupla + ordenação) sem escrever o código todo.
6. Mapeia as 4 práticas do pensamento computacional no procedimento desta AT: onde está a **abstração**, a **decomposição**, o **algoritmo** e a **depuração**?

Conclusão-chave: a regra cívica da maioria absoluta (*mais de metade dos votos expressos*) traduziu-se num algoritmo de 6 linhas — e os testes de **fronteira** (R3: exatamente 50 %) são o que garante que a lei está bem programada. Programar é decidir com **casos**: escrever → executar → comparar com o esperado → depurar → estender.

Nota: esta AT concretiza o Pensamento Computacional das Aprendizagens Essenciais de MACS (abstração, decomposição, reconhecimento de padrões, algoritmos, depuração e otimização) — a linguagem é meio, o raciocínio é o objetivo. Próxima aplicação: a folha de cálculo na AT de partilha (item 11).