

🧙 Aula 1 – Condições (if/else)

🎯 Objetivo da Aula

Aprender a tomar decisões no código usando estruturas condicionais em Python.

📖 Teoria

Um programa não precisa apenas executar instruções em sequência. Muitas vezes, precisamos tomar decisões:

- Se a condição for verdadeira → executa um bloco de código.
- Caso contrário → executa outro bloco.

Estrutura básica

```
if condição:  
    # bloco de código se a condição for verdadeira  
elif outra_condição:  
    # bloco de código se a primeira não for e esta for verdadeira  
else:  
    # bloco de código se nenhuma condição for verdadeira
```

Operadores relacionais

- == → igual
- != → diferente
- > → maior que
- < → menor que
- >= → maior ou igual
- <= → menor ou igual

Operadores lógicos

- and → e
 - or → ou
 - not → não
-

Exemplos práticos

Exemplo 1 – Verificação de idade

```
idade = int(input("Digite sua idade: "))

if idade >= 18:
    print("Você é maior de idade.")
else:
    print("Você é menor de idade.")
```

Exemplo 2 – Nota de aluno

```
nota = float(input("Digite a nota do aluno: "))

if nota >= 7:
    print("Aprovado!")
elif nota >= 5:
    print("Recuperação.")
else:
    print("Reprovado.")
```

Exemplo 3 – Login simples

```
usuario = input("Usuário: ")
senha = input("Senha: ")

if usuario == "admin" and senha == "1234":
    print("Login bem-sucedido!")
else:
    print("Usuário ou senha incorretos.")
```



Exercícios

1. Par ou ímpar: Peça ao usuário um número e diga se ele é par ou ímpar.
2. Maior de três números: Leia três números e mostre qual é o maior.
3. Desconto em loja: Se a compra for maior ou igual a R\$100,00, dê 10% de desconto. Caso contrário, preço normal.

4. Verificação de login: Crie um programa que só permita acesso se o usuário for "python" e a senha "123".

🌟 Desafio do Codemancer

Crie um programa que peça ao usuário sua idade e informe:

- Menor de 12 anos → “Você é uma criança.”
- Entre 12 e 17 anos → “Você é um adolescente.”
- Entre 18 e 59 anos → “Você é um adulto.”
- 60 anos ou mais → “Você é um idoso.”

Aula 2 – Laços de Repetição (for e while)

Objetivo da Aula

Aprender a repetir instruções automaticamente usando laços de repetição (`for` e `while`).

Teoria

O que são laços?

Laços (ou loops) permitem executar um mesmo bloco de código várias vezes, até que uma condição seja satisfeita.

Laço `for`

O `for` percorre um intervalo ou sequência.

```
for i in range(5):  
    print(i)
```

Saída:

```
0  
1  
2  
3  
4
```

- `range(5)` gera os números de 0 até 4.
- O `i` recebe cada número a cada volta.

Você pode personalizar o `range`:

```
for i in range(2, 10, 2): # começa em 2, vai até 9, de 2 em 2  
    print(i)
```

Laço `while`

O `while` repete enquanto a condição for verdadeira.

```
contador = 0  
while contador < 5:  
    print(contador)  
    contador += 1
```

➡ Saída:

```
0
1
2
3
4
```

⚡ Comandos de controle

- `break` → interrompe o loop imediatamente.
- `continue` → pula para a próxima iteração.

Exemplo:

```
for i in range(10):
    if i == 5:
        break    # para quando i == 5
    print(i)
```

```
for i in range(10):
    if i % 2 == 0:
        continue # pula os pares
    print(i)
```

💡 Exemplos práticos

Exemplo 1 – Tabuada (for)

```
n = int(input("Digite um número: "))

for i in range(1, 11):
    print(n, "x", i, "=", n * i)
```

Exemplo 2 – Contagem regressiva (while)

```
n = 5
while n > 0:
    print(n)
    n -= 1
print("Feliz Ano Novo!")
```

Exemplo 3 – Menu interativo

```
opcao = 0
while opcao != 3:
```

```
print("\nMenu:")
print("1 - Olá")
print("2 - Soma 2 + 2")
print("3 - Sair")

opcao = int(input("Escolha uma opção: "))

if opcao == 1:
    print("Olá, mundo!")
elif opcao == 2:
    print("2 + 2 =", 2 + 2)
elif opcao == 3:
    print("Saindo...")
else:
    print("Opção inválida.")
```



Exercícios

1. Contagem de 1 a 10 (for):

Faça um programa que mostre os números de 1 até 10.

2. Soma de 5 números (while):

Leia 5 números digitados pelo usuário e mostre a soma deles.

3. Tabuada personalizada:

Peça ao usuário um número e mostre sua tabuada de 1 a 10 usando `for`.

4. Contagem regressiva:

Peça um número e faça a contagem regressiva até 0, mostrando “Fim!” no final.

5. Menu de opções:

Crie um menu com 3 opções (ex: mostrar uma mensagem, calcular algo, sair) usando `while`.



Desafio do Codemancer

Crie um programa que peça números ao usuário até que ele digite 0.

- No final, mostre a soma e a quantidade de números digitados (excluindo o 0).



Aula 3 – Combinando Condições e Laços



Objetivo da Aula

Aprender a usar `if/else` dentro de laços de repetição e conhecer os comandos especiais `break` e `continue` para controlar a execução.



Teoria



Condições dentro de laços

Podemos usar `if` dentro de `for` ou `while` para criar testes em cada repetição.

Exemplo:

```
for i in range(1, 11):
    if i % 2 == 0:
        print(i, "é par")
    else:
        print(i, "é ímpar")
```



Comando `break`

Interrompe imediatamente o laço.

```
for i in range(1, 10):
    if i == 5:
        break
    print(i)
```



Para quando `i` for igual a 5.



Comando `continue`

Pula a iteração atual e vai para a próxima.

```
for i in range(1, 10):
    if i % 2 == 0:
        continue
    print(i)
```



Mostra apenas números ímpares.

Exemplos práticos

Exemplo 1 – Números pares e ímpares

```
for i in range(1, 11):
    if i % 2 == 0:
        print(i, "é par")
    else:
        print(i, "é ímpar")
```

Exemplo 2 – Busca até encontrar

```
nomes = ["Ana", "Pedro", "Maria", "Lucas"]

for nome in nomes:
    print("Verificando:", nome)
    if nome == "Maria":
        print("Maria encontrada!")
        break
```

Exemplo 3 – Pular valores

```
for i in range(1, 8):
    if i == 4:
        continue # pula o número 4
    print(i)
```

Exemplo 4 – Menu com saída

```
while True:
    print("\nMenu:")
    print("1 - Dizer Olá")
    print("2 - Mostrar quadrados de 1 a 5")
    print("3 - Sair")

    opcao = int(input("Escolha: "))

    if opcao == 1:
        print("Olá, mundo!")
    elif opcao == 2:
        for i in range(1, 6):
            print(i, "² =", i * i)
    elif opcao == 3:
        print("Saindo...")
        break
    else:
        print("Opção inválida!")
```



Exercícios

1. Números pares de 1 a 20 (for):

Mostre apenas os números pares de 1 até 20.

2. Soma de números positivos (while):

O programa deve ler números até que o usuário digite um número negativo.

No final, mostre a soma dos números positivos digitados.

3. Busca em lista:

Crie uma lista com 5 nomes. Peça ao usuário um nome e verifique se está na lista.

- Se encontrar, mostre “Nome encontrado!” e interrompa o laço.
- Se não encontrar, percorra até o final e mostre “Nome não encontrado.”

4. Pular múltiplos de 3 (for):

De 1 a 15, pule os números que são múltiplos de 3 (use `continue`).

5. Menu interativo:

Monte um menu com 3 opções:

- Mostrar “Bem-vindo”
- Mostrar tabuada de um número escolhido
- Sair



Desafio do Codemancer

Crie um jogo da adivinhação:

- O programa escolhe um número secreto (ex: 7).
- O usuário deve tentar adivinhar.
- A cada tentativa, o programa responde:
 - “Maior que o número secreto.”
 - “Menor que o número secreto.”
 - “Acertou!” e o jogo termina.



Aula 4 – Funções: a magia da reutilização



Objetivo da Aula

Aprender a criar e usar funções em Python para organizar, reaproveitar e deixar o código mais legível.



Teoria



O que é uma função?

Uma função é um bloco de código com um nome que executa uma tarefa específica.

- Podemos chamar a função várias vezes sem precisar reescrever o código.
 - Funções ajudam a organizar e reutilizar o código.
-



Estrutura básica

```
def nome_da_funcao(parametros):  
    # bloco de código  
    return resultado
```

- `def` → palavra-chave para definir a função
 - `parametros` → valores que a função recebe
 - `return` → devolve um resultado (opcional)
-



Função sem parâmetro e sem retorno

```
def saudacao():  
    print("Olá, mundo!")  
  
saudacao() # chamada
```



Função com parâmetro

```
def dobro(numero):  
    print("O dobro é:", numero * 2)  
  
dobro(5)
```

◆ Função com retorno

```
def soma(a, b):  
    return a + b  
  
resultado = soma(3, 7)  
print("Resultado:", resultado)
```

◆ Escopo de variáveis

- Variável local → existe apenas dentro da função.
- Variável global → existe fora e pode ser usada em todo o programa.

```
x = 10 # global  
  
def teste():  
    x = 5 # local  
    print("Dentro da função:", x)  
  
teste()  
print("Fora da função:", x)
```

💡 Exemplos práticos

Exemplo 1 – Cálculo de área

```
def area_retangulo(base, altura):  
    return base * altura  
  
print("Área:", area_retangulo(5, 3))
```

Exemplo 2 – Conversão de temperatura

```
def celsius_para_fahrenheit(c):  
    return (c * 9/5) + 32  
  
print("30°C =", celsius_para_fahrenheit(30), "°F")
```

Exemplo 3 – Função com decisão

```
def situacao_aluno(nota):  
    if nota >= 7:  
        return "Aprovado"  
    elif nota >= 5:  
        return "Recuperação"  
    else:
```

```
        return "Reprovado"

print("Aluno com nota 8:", situacao_aluno(8))
```



Exercícios

1. Função de saudação:

Crie uma função que receba o nome de uma pessoa e mostre uma mensagem de boas-vindas.

2. Dobro de um número:

Crie uma função que receba um número e retorne o dobro.

3. Cálculo de média:

Crie uma função que receba 3 notas e retorne a média.

- Se a média ≥ 7 , retorne “Aprovado”
- Se $5 \leq \text{média} < 7$, retorne “Recuperação”
- Se < 5 , retorne “Reprovado”

4. Maior de dois números:

Crie uma função que receba dois números e retorne o maior deles.

5. Tabuada em função:

Crie uma função que receba um número e mostre sua tabuada de 1 a 10.



Desafio do Codemancer

Crie um mini sistema de funções com as seguintes opções:

- 1 → Calcular área do retângulo
- 2 → Converter Celsius para Fahrenheit
- 3 → Calcular a média de 3 notas
- 4 → Sair

👉 Cada funcionalidade deve estar em uma função separada.

👉 O menu principal deve usar `while` + `if/elif` para chamar as funções.

Aula 5 – Escopo e Organização (módulos e boas práticas)

Objetivo da aula

- Entender escopo (variáveis locais vs globais).
 - Organizar código em funções e módulos (arquivos .py).
 - Usar `import`, `from ... import`, `alias` e o guardião `if __name__ == "__main__":`.
 - Aplicar boas práticas (nomes, docstrings, dicas de tipo).
 - Montar um mini-projeto com pastas.
-

Teoria

1) Escopo de variáveis

- Local: criada dentro de função — só existe ali.
- Global: definida no topo do arquivo — acessível no arquivo inteiro.
- Evite `global` quando possível; prefira passar parâmetros e retornar valores.

```
x = 10 # global

def f():
    x = 5 # local
    print("Dentro:", x)

f()
print("Fora:", x)
```

Regra de ouro: funções puras (sem depender de global) são mais fáceis de testar e reaproveitar.

2) Organização em funções

- Uma função deve fazer uma coisa bem (single responsibility).
- Nomeie com verbos e minúsculas com_: `calcular_media`, `converter_temperatura`.

```
def calcular_media(notas):
```

```
"""Retorna a média aritmética de uma lista de números."""  
return sum(notas) / len(notas)
```

3) Módulos (arquivos .py) e imports

- Cada arquivo .py é um módulo.
- Para reutilizar, importe.

```
# utils.py  
def dobro(n): return n * 2  
  
# app.py  
import utils  
print(utils.dobro(7))           # via módulo  
  
from utils import dobro  
print(dobro(7))                # import direto  
  
import utils as u  
print(u.dobro(7))              # com alias
```

4) O guardião mágico

Use o padrão abaixo para que código só rode quando o arquivo for executado diretamente, e não quando for importado:

```
def main():  
    print("Rodando app...")  
  
if __name__ == "__main__":  
    main()
```

5) Docstrings e dicas de tipo

- Docstring: explique o que a função faz (não “como”), parâmetros e retorno.
- Type hints ajudam legibilidade e ferramentas.

```
def c_para_f(c: float) -> float:  
    """Converte Celsius para Fahrenheit.  
    :param c: temperatura em °C  
    :return: temperatura em °F  
    """  
    return (c * 9/5) + 32
```

6) Estrutura inicial de projeto

```
meu_projeto/  
├─ app.py           # ponto de entrada (menu, I/O)  
├─ utils.py         # funções genéricas  
├─ conversores.py   # cálculos específicos  
└─ README.md
```

- Regra prática:
 - `app.py` → interação com o usuário (input/print).
 - Módulos → lógica e cálculo (sem `input()`/`print()` quando possível).
-

7) Boas práticas “Codemancer”

- PEP 8 (versão enxuta): nomes claros, linhas ≤ 79 –100 col, 4 espaços de indentação.
 - Evite duplicação (DRY): se repetiu 2–3 vezes, vire função.
 - Separe cálculo (puro) de apresentação (prints).
-



Exemplos práticos

Exemplo A — separar lógica em módulo

`conversores.py`

```
def c_para_f(c: float) -> float:  
    return (c * 9/5) + 32  
  
def km_para_milhas(km: float) -> float:  
    return km * 0.621371
```

`app.py`

```
from conversores import c_para_f, km_para_milhas  
  
def main():  
    print("1) Celsius → Fahrenheit")  
    print("2) Km → Milhas")  
    op = input("Escolha: ").strip()  
    if op == "1":  
        c = float(input("°C: "))  
        print("°F =", c_para_f(c))  
    elif op == "2":  
        km = float(input("Km: "))  
        print("Milhas =", km_para_milhas(km))  
    else:  
        print("Opção inválida.")  
  
if __name__ == "__main__":
```

```
main()
```

Exemplo B — utilitários e import com alias

utils.py

```
def ler_float(msg: str) -> float:
    while True:
        try:
            return float(input(msg))
        except ValueError:
            print("Valor inválido. Tente de novo.")
```

app.py

```
import utils as u
import conversores as conv

def main():
    print("Conversor")
    c = u.ler_float("°C: ")
    print("°F =", conv.c_para_f(c))

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



Exercícios

1. Refatorar com módulos

Crie `calculadora.py` com as funções: `somar(a, b)`, `subtrair(a, b)`, `multiplicar(a, b)`, `dividir(a, b)` (trate divisão por zero).

Crie `app.py` com um menu que usa essas funções. Use o guardião `if __name__ == "__main__":`.

2. Strings util

Em `strings_util.py`, crie funções:

- `contar_vogais(texto)` → retorna quantidade de vogais
- `eh_palindromo(texto)` → ignora espaços/maiúsculas
- `slugificar(texto)` → minúsculas, troque espaços por -, remova caracteres não alfanuméricos (básico)

Implemente testes simples em `app.py` chamando essas funções com entradas do usuário.

3. Lista de tarefas (mini to-do)

Mantenha a lista em `tarefas.py` (em memória).

Funções: `adicionar(t)`, `listar()`, `remover(indice)` (validando índice).

Em `app.py`, menu para gerenciar tarefas. Não use globais no `app.py`: passe a lista como parâmetro e retorne valores.

4. Docstrings + tipos

Adicione docstrings e type hints a todas as funções dos exercícios 1–3.

Crie um `README.md` curto explicando como rodar o projeto.

5. Refatoração DRY

Pegue um código seu com prints e cálculos misturados (pode ser da Aula 2/3) e:

- Extraia a lógica para funções puras (sem `print/input`).
- No `app.py`, só fica o fluxo e a apresentação.

Desafio do Codemancer (mini-projeto)

Crie a estrutura:

```
codemancer_toolbox/  
├── app.py  
├── conversores.py  
├── calculadora.py  
├── strings_util.py  
└── utils.py
```

Requisitos:

- `conversores.py`: `c_para_f`, `f_para_c`, `km_para_milhas`, `milhas_para_km`.
- `calculadora.py`: as quatro operações; `media(*nums)`; `maior_menor(nums)` retorna (`maior`, `menor`).
- `strings_util.py`: `contar_vogais`, `eh_palindromo`, `slugificar`.
- `utils.py`: `ler_float`, `ler_int`, `menu(opcoes: list[str]) -> int` (mostrar e retornar escolha válida).
- `app.py`: menu principal com 4 seções: Calculadora, Conversores, Strings, Sair.
 - Cada seção tem seu sub-menu chamando funções do módulo correspondente.
- Sem variáveis globais (exceto constantes).
- `if __name__ == "__main__":` em `app.py`.

Bônus (opcional):

- Salvar/ler tarefas em arquivo texto simples (`tarefas.txt`) no exercício 3.
- Criar pacote `codemancer_toolbox/` com subpasta `modulos/` e `__init__.py`.

- Adicionar verificação simples de argumentos da linha de comando (via `sys.argv`) para pular o menu e executar uma ação direta (ex.: `python app.py c2f 30`).
-



Checklist rápido

- Funções com nomes claros, docstrings e type hints
 - Sem lógica duplicada (DRY)
 - Módulos separados por responsabilidade
 - `app.py` só orquestra I/O e menus
 - `if __name__ == "__main__":` no arquivo de entrada
 - Pequenos testes manuais no final dos módulos (e rodar via `python -m`)
-



Aula 6 – Mini Projeto do Codemancer



Objetivo da aula

Construir um mini sistema completo em Python, com:

- Menu principal e submenus
 - Funções bem definidas
 - Módulos separados (`calculadora.py`, `conversores.py`, `strings_util.py`, `utils.py`, `app.py`)
 - Boas práticas: `if __name__ == "__main__":`, docstrings, type hints
-



Estrutura do Projeto

```
codemancer_toolbox/  
├─ app.py           # ponto de entrada (menu principal)  
├─ utils.py         # funções auxiliares (input, menu)  
├─ calculadora.py   # operações matemáticas  
├─ conversores.py   # conversões de unidades  
└─ strings_util.py  # manipulação de textos
```



Teoria aplicada

1. Modularização: cada parte do código em seu arquivo.
 2. Reutilização: funções chamadas várias vezes.
 3. Menus interativos: `while` + `if/elif`.
 4. Escopo controlado: variáveis locais sempre que possível.
 5. Boas práticas: nomes claros, docstrings, type hints.
-



Exemplos de código

`calculadora.py`

```
"""Operações matemáticas básicas."""  
from typing import Iterable, Tuple  
  
def somar(a: float, b: float) -> float:  
    return a + b
```

```

def subtrair(a: float, b: float) -> float:
    return a - b

def multiplicar(a: float, b: float) -> float:
    return a * b

def dividir(a: float, b: float) -> float:
    if b == 0:
        raise ValueError("Divisão por zero não permitida.")
    return a / b

def media(*nums: float) -> float:
    if not nums:
        raise ValueError("Informe ao menos um número.")
    return sum(nums) / len(nums)

def maior_menor(nums: Iterable[float]) -> Tuple[float, float]:
    lista = list(nums)
    if not lista:
        raise ValueError("Lista vazia.")
    return max(lista), min(lista)

```

conversores.py

```

"""Conversores de unidades comuns."""

def c_para_f(c: float) -> float:
    return (c * 9/5) + 32

def f_para_c(f: float) -> float:
    return (f - 32) * 5/9

def km_para_milhas(km: float) -> float:
    return km * 0.621371

def milhas_para_km(mi: float) -> float:
    return mi / 0.621371

```

strings_util.py

```

"""Funções utilitárias para manipulação de strings."""
import re

VOGAIS = set("aeiouáéíóúâêôãäöAEIOUÁÉÍÓÚÂÊÔÃÄÖ")

def contar_vogais(texto: str) -> int:
    return sum(1 for ch in texto if ch in VOGAIS)

def eh_palindromo(texto: str) -> bool:
    normal = re.sub(r"\s+", "", texto).lower()
    return normal == normal[::-1]

def slugificar(texto: str) -> str:

```

```
t = texto.lower().strip()
t = re.sub(r"\s+", "-", t)
t = re.sub(r"[^a-z0-9\-]", "", t)
t = re.sub(r"-{2,}", "-", t)
return t.strip("-")
```

utils.py

```
"""Funções auxiliares para entrada e menus."""
```

```
def ler_float(msg: str) -> float:
    while True:
        try:
            return float(input(msg))
        except ValueError:
            print("Valor inválido. Tente novamente.")

def ler_lista_float() -> list[float]:
    while True:
        try:
            entrada = input("Digite números separados por espaço: ")
            return [float(x.replace(",", ".")) for x in entrada.split()]
        except ValueError:
            print("Algum valor não é numérico. Tente novamente.")

def menu(opcoes: list[str], titulo: str = "Menu") -> int:
    print(f"\n=== {titulo} ===")
    for i, texto in enumerate(opcoes, start=1):
        print(f"{i}) {texto}")
    print("0) Voltar")
    while True:
        escolha = input("Escolha: ").strip()
        if escolha == "0":
            return 0
        if escolha.isdigit() and 1 <= int(escolha) <= len(opcoes):
            return int(escolha)
        print("Opção inválida.")
```

app.py

```
"""Aplicação principal do Codemancer Toolbox."""
import calculadora as calc
import conversores as conv
import strings_util as su
from utils import ler_float, ler_lista_float, menu

def submenu_calculadora():
    while True:
        op = menu([
            "Somar", "Subtrair", "Multiplicar", "Dividir",
            "Média (vários)", "Maior/Menor (vários)"
        ], "Calculadora")
        if op == 0: return
        try:
```

```

        if op in (1, 2, 3, 4):
            a, b = ler_float("a: "), ler_float("b: ")
            if op == 1: print("Resultado:", calc.somar(a, b))
            elif op == 2: print("Resultado:", calc.subtrair(a, b))
            elif op == 3: print("Resultado:", calc.multiplicar(a, b))
            elif op == 4: print("Resultado:", calc.dividir(a, b))
        elif op == 5:
            nums = ler_lista_float()
            print("Média:", calc.media(*nums))
        elif op == 6:
            nums = ler_lista_float()
            maior, menor = calc.maior_menor(nums)
            print(f"Maior: {maior} | Menor: {menor}")
    except Exception as e:
        print("Erro:", e)

def submenu_conversores():
    while True:
        op = menu([
            "Celsius → Fahrenheit", "Fahrenheit → Celsius",
            "Km → Milhas", "Milhas → Km"
        ], "Conversores")
        if op == 0: return
        try:
            if op == 1:
                c = ler_float("°C: ")
                print("°F =", conv.c_para_f(c))
            elif op == 2:
                f = ler_float("°F: ")
                print("°C =", conv.f_para_c(f))
            elif op == 3:
                km = ler_float("Km: ")
                print("Milhas =", conv.km_para_milhas(km))
            elif op == 4:
                mi = ler_float("Milhas: ")
                print("Km =", conv.milhas_para_km(mi))
        except Exception as e:
            print("Erro:", e)

def submenu_strings():
    while True:
        op = menu(["Contar vogais", "Verificar palíndromo", "Slugificar"], "Strings
Util")
        if op == 0: return
        txt = input("Texto: ")
        if op == 1: print("Vogais:", su.contar_vogais(txt))
        elif op == 2: print("É palíndromo?", su.eh_palindromo(txt))
        elif op == 3: print("Slug:", su.slugificar(txt))

def main():
    while True:
        op = menu(["Calculadora", "Conversores", "Strings"], "Codemancer Toolbox")
        if op == 0:
            print("Até a próxima!")
            return
        if op == 1: submenu_calculadora()
        elif op == 2: submenu_conversores()
        elif op == 3: submenu_strings()

```

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



Exercícios da Aula 6

1. Adicione um submenu de estatísticas com funções:

- somar lista
- maior e menor da lista
- média da lista

2. Acrescente ao `strings_util` uma função `inverter(texto)` que devolva o texto invertido, e use-a no menu.

3. No `conversores`, crie uma função `segundos_para_horas(seg)` que devolva em horas:minutos:segundos.

4. Crie testes manuais chamando funções diretamente do console Python (`python + import calculadora`).



Desafio do Codemancer

Permitir que o programa seja executado por argumentos de linha de comando:

```
python app.py soma 2 3  
python app.py c2f 30  
python app.py slug "Olá Mundo!"
```

Dica: use `import sys` e `sys.argv`.

Assinado por:

Zahroniel Syrran & Kael'Aran