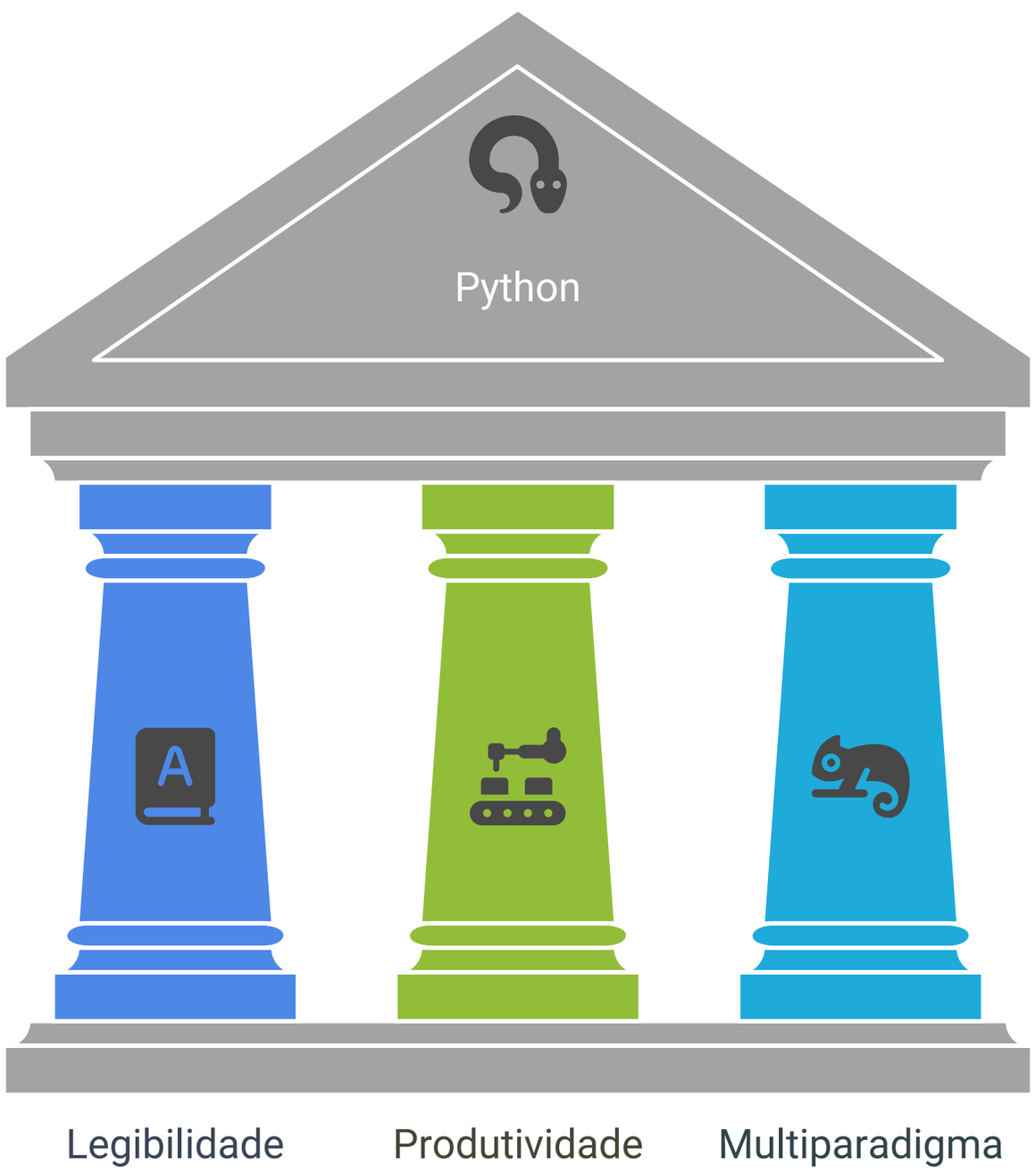
Aula de Fundamentos de Programação: Introdução a Python

Aula 05: Introdução a Python

1. O Que é Python?

Python é uma linguagem de programação de alto nível criada por **Guido van Rossum** e lançada em 1991. Seus pilares são:

- **Legibilidade: Sintaxe clara e intuitiva**
- **Produtividade: Código conciso e eficiente**
- **Multiparadigma: Suporta programação estruturada, orientada a objetos e funcional**



Por que aprender Python?

- Linguagem mais usada em cibersegurança e ciência de dados
- Utilizada por gigantes: Google, Netflix, NASA, Instagram
- Ideal para iniciantes pela sintaxe simplificada

2. Variáveis e Tipos de Dados

Variáveis são "containers" que armazenam valores na memória. Python usa **tipagem dinâmica**:

```
nome = "Ana"       # string (texto)
idade = 30         # int (número inteiro)
altura = 1.75      # float (número decimal)
ativo = True       # bool (valor lógico)3. Operadores
```

2.1 . Operadores

Aritméticos:

```
5 + 3   # Soma → 8
10 / 2  # Divisão → 5.0
2 ** 3  # Potência → 8
10 % 3  # Resto → 1
```

Relacionais (sempre retornam **True**/**False**):

```
5 > 3   # True
idade >= 18  # Verifica maioridade
```

Lógicos:

```
(idade >= 18) and (tem_carteira == True)  # AND
(nota >= 7) or (prova_extra == True)      # OR
```

4. Entrada/Saída de Dados

- **input(): Captura dados do usuário**
- **print(): Exibe informações na tela**

```
nome = input("Seu nome: ")
print(f"Olá, {nome}!")  # f-string formata valores
```

Caracteres especiais:

- **\n: Nova linha**
- **\t: Tabulação**

Entrada/Saída de dados

Característica	`input()`	`print()`
Função	Captura dados do usuário	Exibe informações
Caracteres Especiais	Nenhum	`\n`: Nova linha, `\t`: Tabulação

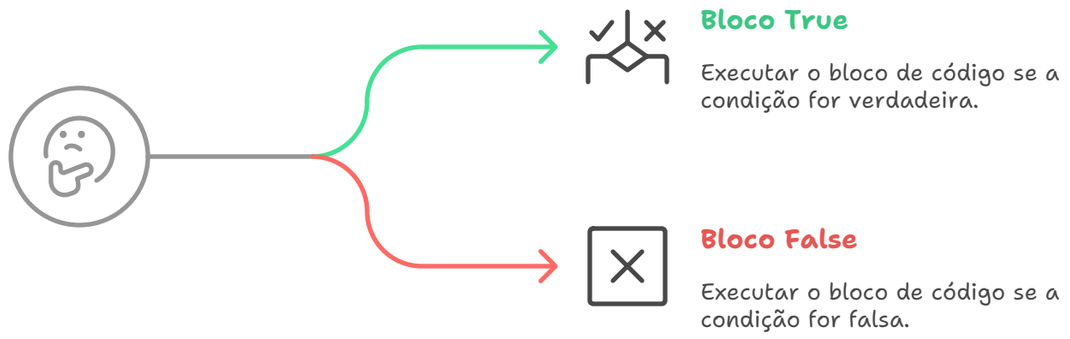
Aula 06: Estruturas Condicionais

1. Tomada de Decisões

Estruturas condicionais permitem que programas tomem decisões com base em condições:

Fluxograma Básico:

Qual bloco de código deve ser executado com base na condição?



2. Tipos de Condicionais

a) Simples (if):

```
if temperatura > 30:
    print(”Dia quente!”)
```

b) Composta (if-else):

```
if idade >= 16:
    print(”Pode votar!”)
else:
    print(”Não pode votar!”)
```

c) Aninhada (if-elif-else):

```
if nota >= 9:
    conceito = ”A”
elif nota >= 7:
    conceito = ”B”
else:
    conceito = ”C”
```

3. Operadores Relacionais Avançados

- **and**: Ambas condições devem ser verdadeiras

```
if (idade >= 18) and (cnh_valida):
    print(”Pode dirigir”)
```

- **or**: Pelo menos uma condição verdadeira

```
if (dia == "sábado") or (dia == "domingo"):
    print("Final de semana!")
```

4. Boas Práticas

- Sempre use **identação correta** (4 espaços)
- Evite condições complexas demais
- Teste casos limite (ex: valor 0, números negativos)

Aula 07: Estruturas de Repetição

1. O Poder da Repetição

Laços de repetição automatizam tarefas repetitivas:

```
print("Bom dia!")

print("Bom dia!")

print("Bom dia!")
```

```
for _ in range(3):
        print("Bom dia!")
```

2. Loop `while`

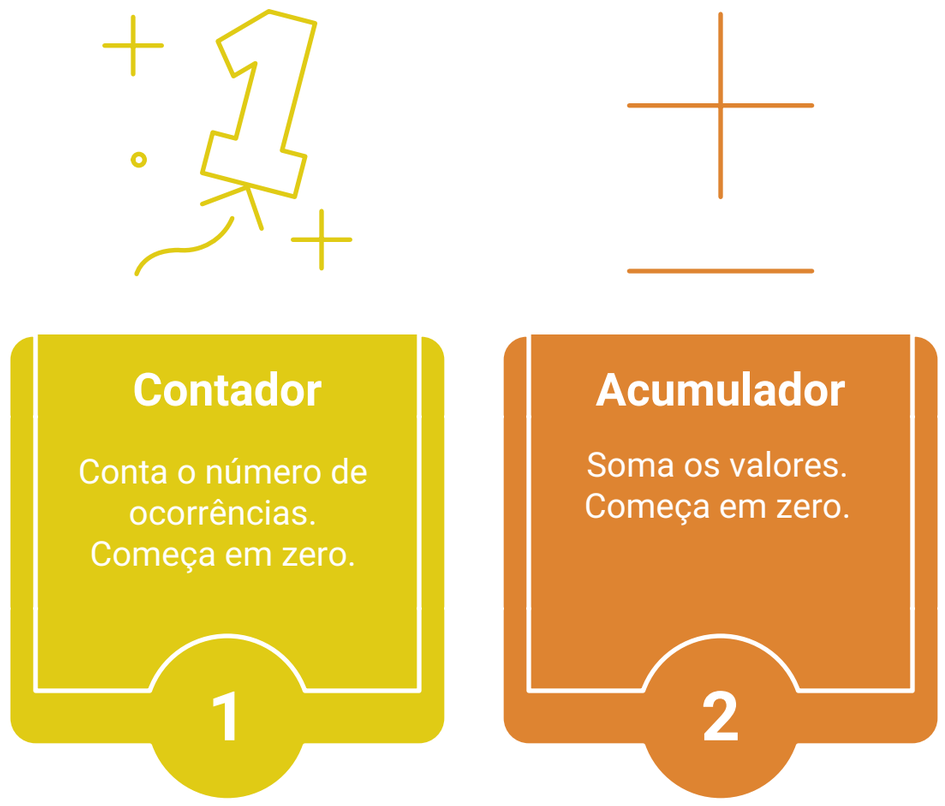
Executa um bloco **enquanto** uma condição for verdadeira:

```
contador = 1
while contador <= 5:
    print(contador)
    contador += 1  # Incremento ESSENCIAL!
```

Componentes chave:

- **Inicialização**: **contador = 1**
- **Condição**: **contador <= 5**
- **Atualização**: **contador += 1**

3. Contadores vs Acumuladores



Exemplo combinado:

```
soma = 0
cont = 0
while cont < 5:
    num = float(input("Número: "))
    soma += num
    cont += 1
print("Média:", soma/cont)
```

4. Controle de Fluxo

- **break**: Sai imediatamente do loop

```
while True:
    resposta = input("Sair? (s/n): ")
    if resposta == 's':
        break  # Sai do loop
```

- **continue**: Pula para próxima iteração

```
num = 0
while num < 10:
    num += 1
    if num % 2 == 0:
        continue  # Pula números pares
    print(num)
```