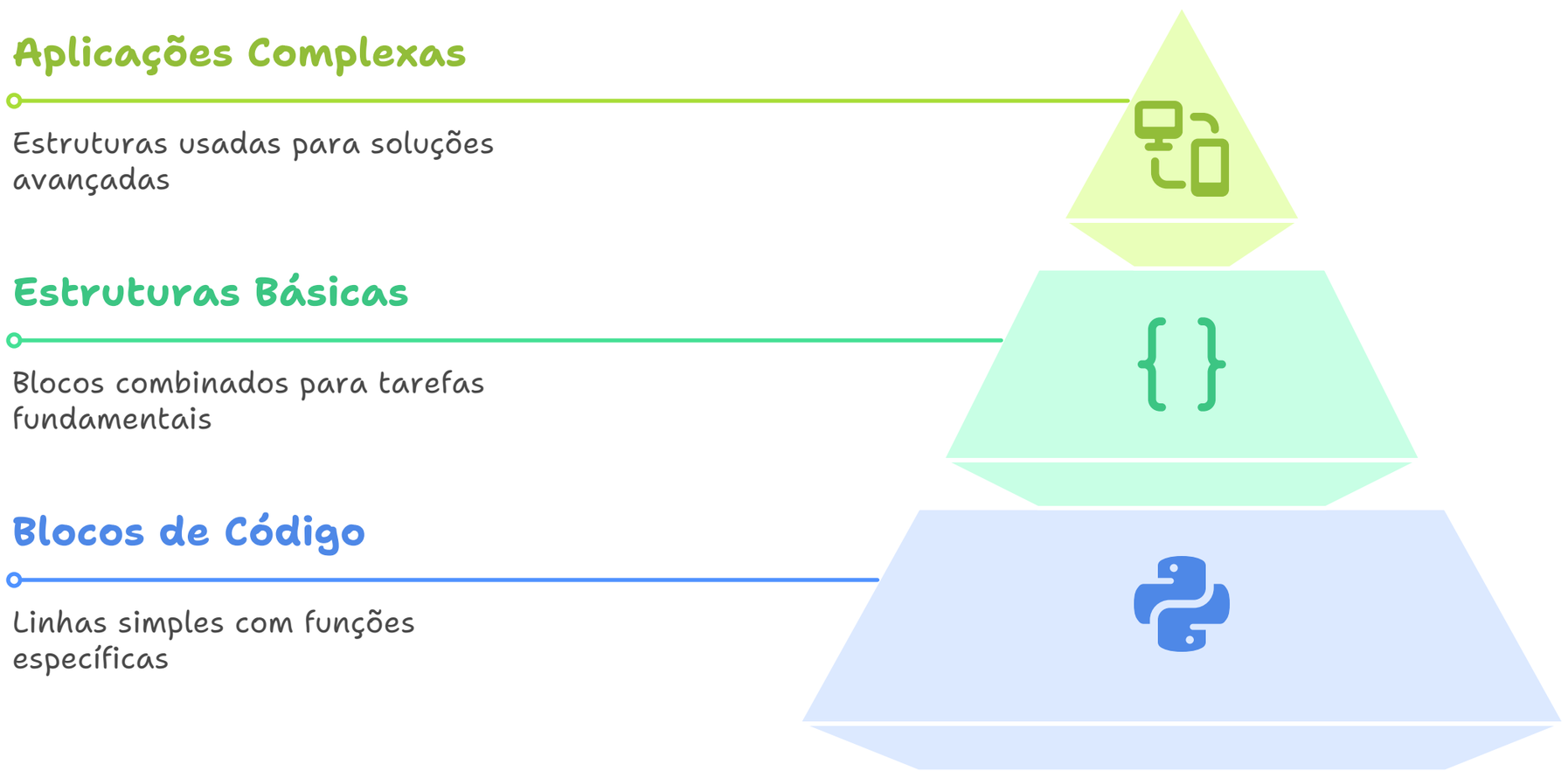
Introdução Python

1. Introdução ao Python

O que é?
Python é uma linguagem de programação de alto nível, fácil de aprender, usada em diversas áreas como web, análise de dados e automação.

Analogia Visual:
Imagine Python como um LEGO: blocos simples que se encaixam para criar estruturas complexas. Cada bloco (linha de código) tem uma função específica.



Variáveis e Operadores

- **Variáveis:** São como caixas que armazenam dados.

```
nome = "Ana"  # String (texto)
idade = 30    # Inteiro (número)
```

- **Operadores:** Ferramentas para manipular dados.
    - Aritméticos: **+**, **/**
    - Comparação: **>**, **==**, **!=**

Exercício Resolvido:
Calcular o tempo de viagem:

```
distancia = float(input("Distância (km): "))
velocidade = float(input("Velocidade média (km/h): "))
tempo = distancia / velocidade
print(f"Tempo estimado: {tempo:.2f} horas")
```

🎯 2. Estruturas Condicionais

O que é?
Decisões no código com **if**, **elif** e **else**.

Analogia Visual:
Como um semáforo:

- **if**: Siga se o sinal estiver verde.
- **else**: Pare se estiver vermelho.

Exemplo:

```
idade = int(input("Sua idade: "))
if idade >= 18:
    print("Maior de idade")
else:
    print("Menor de idade")
```

Exercício Resolvido:

Verificar se um número é par ou ímpar:

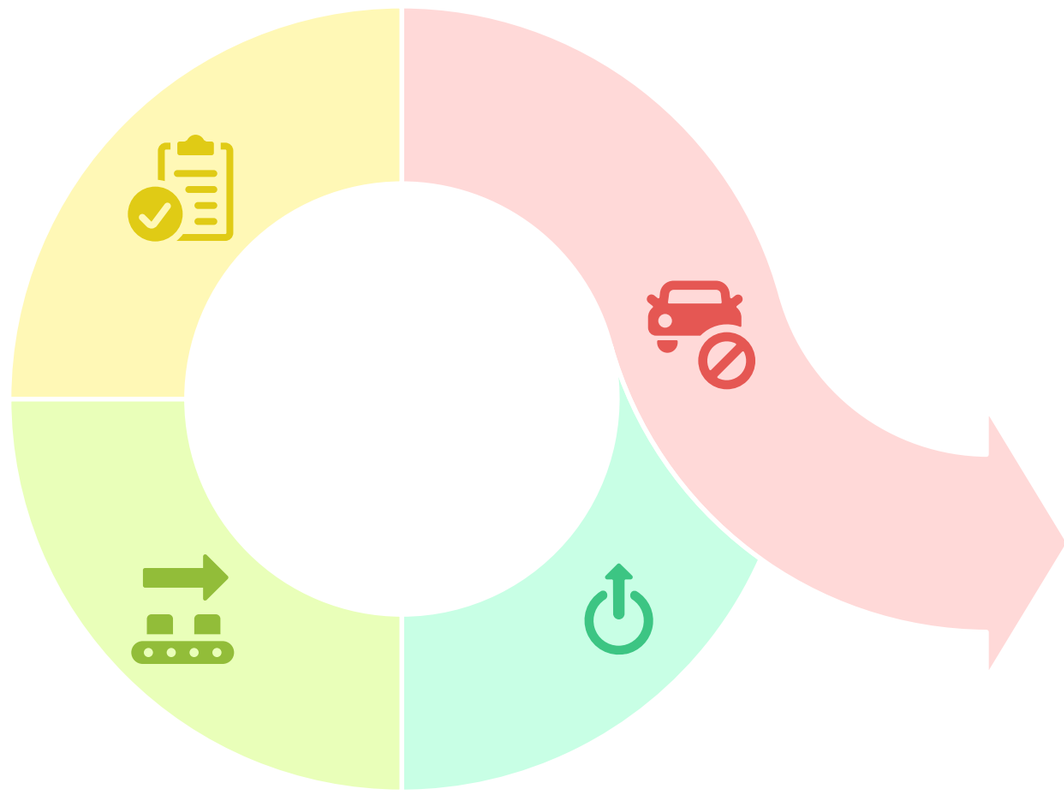
```
numero = int(input("Digite um número: "))
if numero % 2 == 0:
    print("Par")
else:
    print("Ímpar")
```

🔁 3. Estruturas de Repetição

O que é?
Repetir ações com **while** e **for**.

Analogia Visual:
Como uma esteira de produção: repete tarefas até cumprir uma condição.

Ciclo de Estruturas de Repetição



1

Iniciar Tarefa

Começar a tarefa repetitiva

2

Executar Tarefa

Realizar a tarefa repetidamente

3

Verificar Condição

Avaliar se a condição foi cumprida

4

Cumprir Condição

A condição é atendida, parar a tarefa

Exemplo com for:

```
for i in range(5):  # Repete 5 vezes
    print(f"Execução {i+1}")
```

Exercício Resolvido:

Somar números até digitar 0:

```
soma = 0
while True:
    numero = int(input("Digite um número (0 para sair): "))
    if numero == 0:
        break
    soma += numero
print(f"Soma total: {soma}")
```

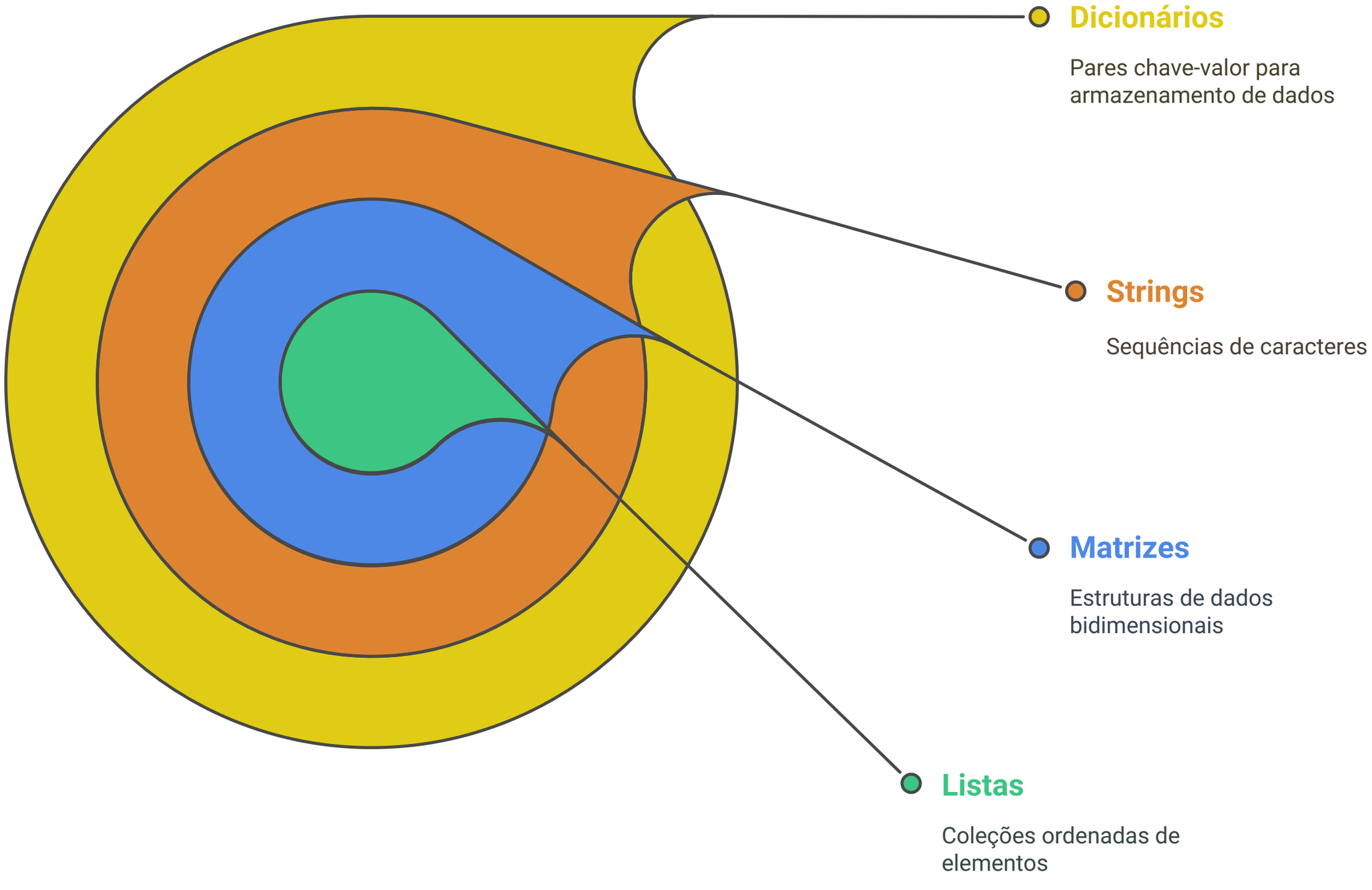
📚 4. Listas/Vetores

O que é?

Coleções ordenadas de elementos.

Analogia Visual:

Uma fila de pessoas, uma pilha de roupas, onde cada item está atrás de um próximo.



Métodos Úteis:

- **append()**: Adiciona um item.
- **remove()**: Remove um item.

Exercício Resolvido:

Separar pares e ímpares:

```
numeros = [2, 5, 8, 3, 10]
pares = [num for num in numeros if num % 2 == 0]
impares = [num for num in numeros if num % 2 != 0]
print(f"Pares: {pares}, Ímpares: {impares}")
```

🧮 5. Matrizes

O que é?
Tabelas bidimensionais (linhas e colunas).

Analogia Visual:
Uma planilha do Excel, onde cada célula tem coordenadas (linha, coluna).

Exemplo:

```
matriz = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
print(matriz[1][2])  # Acessa a linha 1, coluna 2 → 6
```

Exercício Resolvido:

Somar diagonal principal:

```
soma = 0
for i in range(3):
    soma += matriz[i][i]
print(f"Soma da diagonal: {soma}")
```

🔤 6. Strings

O que é?

Sequências de caracteres.

Métodos Úteis:

- **upper()**: Converte para maiúsculas.
- **split()**: Divide em partes.

Exercício Resolvido:

Verificar palíndromo:

```
texto = input("Digite uma palavra: ").lower()
if texto == texto[::-1]:
    print("É palíndromo!")
else:
    print("Não é palíndromo.")
```

🛠️‼️ 7. Funções

O que é?

Blocos de código reutilizáveis.

Analogia Visual:

Uma receita de bolo: você define os passos (função) e usa sempre que precisar.

Exemplo:

```
def calcular_area(raio):
    return 3.14 * raio ** 2
print(calcular_area(5))  # 78.5
```

Exercício Resolvido:

Calcular IMC:

```
def calcular_imc(peso, altura):
    return peso / (altura ** 2)
print(f"IMC: {calcular_imc(70, 1.75):.2f}")
```

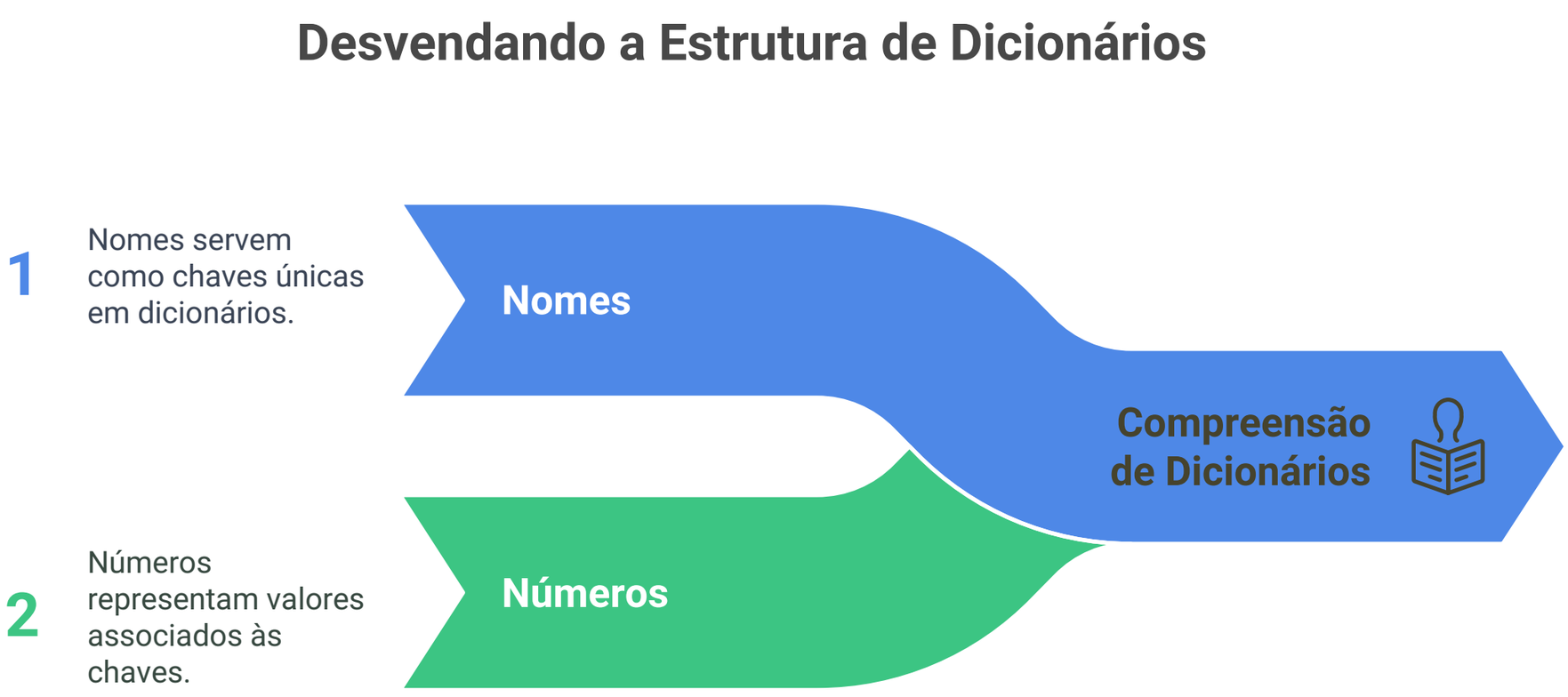
📖 8. Dicionários

O que é?

Coleções de pares chave-valor.

Analogia Visual:

Uma agenda telefônica: o nome (chave) liga ao número (valor).



Exemplo:

```
agenda = {"João": 99999, "Maria": 88888}
print(agenda["João"])  # 99999
```

Exercício Resolvido:

Contar vogais em um texto:

```
texto = input("Digite um texto: ").lower()
vogais = {"a": 0, "e": 0, "i": 0, "o": 0, "u": 0}
for letra in texto:
    if letra in vogais:
        vogais[letra] += 1
print(vogais)
```

1. Manipulação de Arquivos

O que são?

Arquivos são estruturas para armazenar dados de forma persistente. Em Python, você pode ler e escrever em arquivos de texto ou binários.

Analogia Visual:

Pense em um arquivo como um caderno:

- **Abrir (open): Pegar o caderno da mochila.**
- **Ler/Escrever (read/write): Ler anotações ou escrever algo novo.**
- **Fechar (close): Guardar o caderno de volta.**

Modos de Abertura

Modo Descrição **r** Leitura (padrão). **w** Escrita (sobrescreve o arquivo). **a** Escrita (adiciona ao final do arquivo).

Exemplo: Escrever em Arquivo

```
arquivo = open("exemplo.txt", "w")
arquivo.write("Olá, mundo!\\\\n")
arquivo.close()
```

Exemplo: Ler Arquivo

```
arquivo = open("exemplo.txt", "r")
conteudo = arquivo.read()
print(conteudo)
arquivo.close()
```

2. Tratamento de Erros (Try/Except)

O que é?

Mecanismo para lidar com exceções (erros) durante a execução do programa, evitando que ele pare abruptamente.

Analogia Visual:

Imagine dirigir um carro:

- **Try: Tentar seguir o caminho.**
- **Except: Desviar de um obstáculo (erro) e avisar o motorista.**

Estrutura Básica

```
try:
    # Código que pode gerar erro
except TipoErro:
    # Ação se o erro ocorrer
```

Exemplo: Verificar Número Primo

```
def verificar_primo(numero):
    try:
        if numero <= 1:
            return False
        for i in range(2, int(numero**0.5) + 1):
            if numero % i == 0:
                return False
        return True
    except TypeError:
        print("Erro: Insira um número inteiro!")
    except:
        print("Erro desconhecido!")

# Teste
print(verificar_primo(7))  # Saída: True
print(verificar_primo(8))  # Saída: False
```

Exercício Resolvido

Enunciado:

Desenvolva uma função **verificar_primo(numero)** que verifica se um número é primo, tratando erros de entrada.

Código Completo:

```
def verificar_primo(numero):
    try:
        numero = int(numero)
        if numero <= 1:
            print(f"{numero} não é primo.")
            return
        for i in range(2, int(numero**0.5) + 1):
            if numero % i == 0:
                print(f"{numero} não é primo.")
                return
        print(f"{numero} é primo.")
    except ValueError:
        print("Erro: Digite um número inteiro válido!")

# Entrada do usuário
entrada = input("Digite um número inteiro: ")
verificar_primo(entrada)
```

Testes:

1. **Entrada: 7 → Saída: 7 é primo.**
2. **Entrada: 8 → Saída: 8 não é primo.**
3. **Entrada: abc → Saída: Erro: Digite um número inteiro válido!**