Introdução a Linguagem de Programação Back-End

Prof. Wellington S. S. Silva

Linguagem de Programação: Back End

Introdução: O que é a Linguagem de Programação Back End?

Back-End - por que ele é crucial?

Para começarmos, vamos entender o que é o **Back-End**. Pensem nos aplicativos e sites que vocês usam todos os dias. A parte que vocês veem e interagem, os botões, as imagens, o texto – isso é o **Front-End**. O Back-End, por sua vez, é tudo aquilo que acontece nos "bastidores". É o servidor, o banco de dados, a lógica de negócio que permite que o Front-End funcione. É onde os dados são armazenados, processados e gerenciados.

Por que o Back-End é crucial? Basicamente, sem ele, a maioria das aplicações que conhecemos simplesmente não existiria. É o Back-End que garante que suas fotos sejam salvas no Instagram, que seus e-mails cheguem no Gmail, que suas transações bancárias sejam seguras. É a espinha dorsal de qualquer sistema robusto.

Lógica de Programação: A base de tudo

Lógica de Programação: Estrutura

A lógica de programação é o alicerce de tudo. Não importa qual linguagem vocês usem no futuro (Java, Python, Node.js, C#, etc.), os princípios lógicos serão os mesmos.

A lógica de programação nos ensina a pensar de forma estruturada para resolver problemas. Vamos explorar conceitos como:

Algoritmos: Uma sequência de passos bem definidos para resolver um problema. Pensem em uma receita de bolo: ela é um algoritmo.

Variáveis: Espaços na memória do computador para armazenar dados (números, textos, etc.).

Tipos de Dados: Classificações para os dados (inteiros, decimais, caracteres, booleanos).

Operadores: Símbolos usados para realizar operações (matemáticas, comparações, lógicas).

Estruturas Condicionais (if/else): Tomada de decisões com base em condições (se isso for verdade, faça aquilo; senão, faça outra coisa).

Estruturas de Repetição (loops - for/while): Executar um bloco de código várias vezes.

Funções/Métodos: Blocos de código reutilizáveis que realizam uma tarefa específica.

Ferramentas e ambiente de desenvolvimento

- Editor de Texto/IDE

A Espinha Dorsal de Qualquer Sistema Robusto

- Garante que os dados sejam **armazenados**, **processados** e **gerenciados** de forma segura.
- Permite a **comunicação** entre diferentes partes de um sistema.
- Viabiliza funcionalidades complexas que não seriam possíveis apenas no navegador.

- **Exemplos:** Instagram (salvar fotos), Gmail (enviar e-mails), Banco (transações).

Linguagem de Programação

- Interpretada
- Compilada

Tipos de Dados – Como guardamos os dados

Classificando as Informações

- **Inteiro (int):** Números sem casas decimais (ex: 10, -5, 1000).
- **Real/Ponto Flutuante (float/double):** Números com casas decimais (ex: 3.14, 99.99).
- **Caracter (char):** Um único caractere (ex: 'A', 'z', '@’).
- **Cadeia de Caracteres (string):** Textos, sequências de caracteres (ex: "Olá mundo", "Meu nome é...").
- **Booleano (boolean):** Valores verdadeiro (true) ou falso (false). Usado para decisões.

Identificando Variáveis!

Para cada item, diga qual tipo de dado seria mais apropriado e qual nome de variável você daria:

- Sua Altura?
- Seu nome completo?
- Se você é estudante (sim/não)?
- A quantidade de produtos no Carrinho?
- O preço de um livro?

Linguagem para Back End

Programa que Calcular a Nota Final do Aluno e fala se ele foi aprovado, reprovado ou está em recuperação.

Primeiro peça o Nome do Aluno, depois as Notas dos 04 Bimestres.

Com as Notas dos 04 Bimestre, faça a média.

Se a média for >= 7 então Aprovado

Se a média for >= 5 e < 7 então Recuperação

Se a média for < 5 então Reprovado

Quais as principais Linguagens para Back End

5 Melhores linguagens para Back End disponibilizadas no mercado.

Python

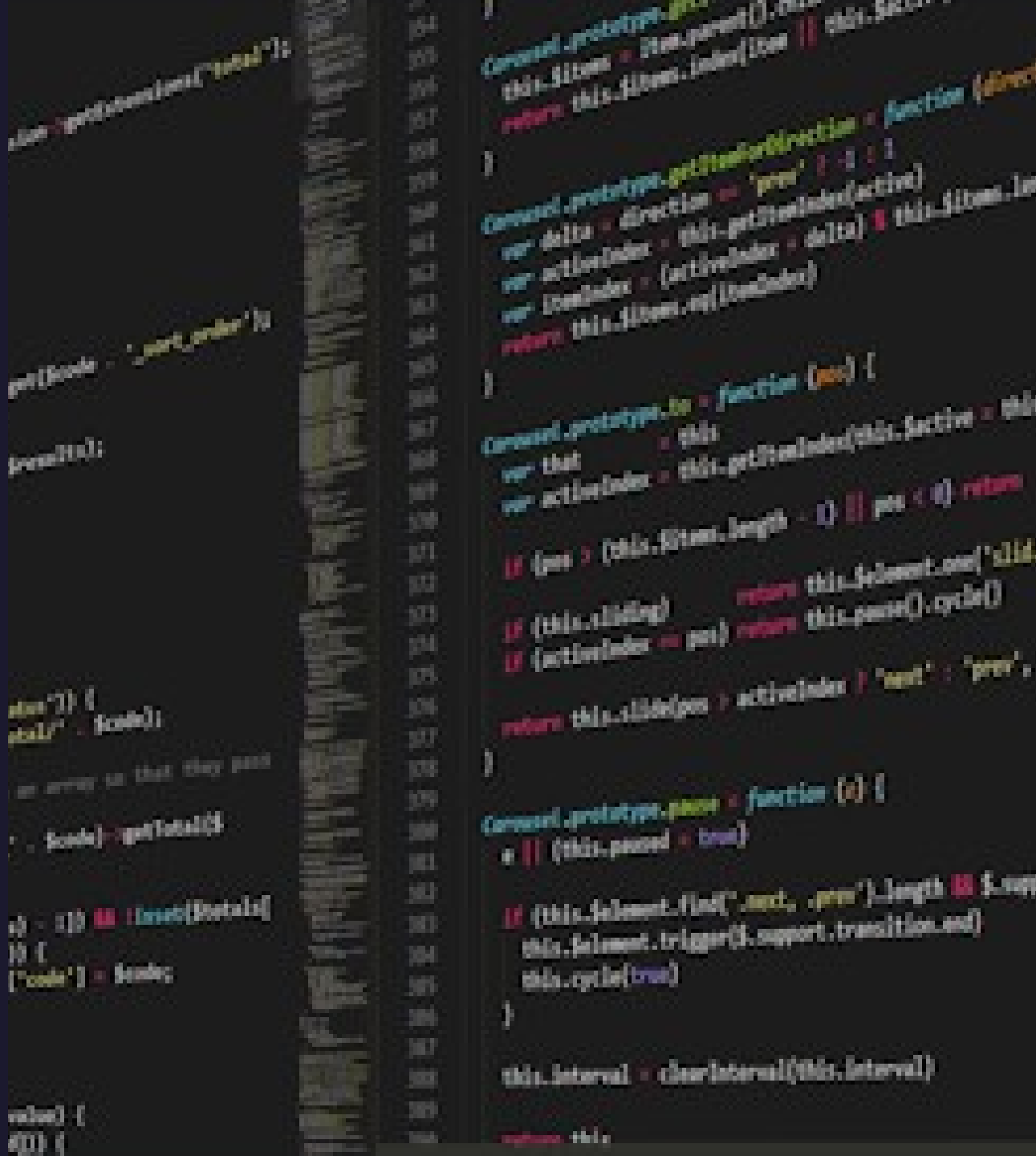
JavaSript

C#

Java

C++

PHP



Estrutura de Dados de Back End

Estrutura Condicional

IF e ELSE

PARA contador DE 1 ATÉ 5 FAÇA
    ESCREVER "Contando: " + contador

Estrutura de Repetição

FOR e WHILE

ENQUANTO (idade < 18) FAÇA
    ESCREVER "Ainda é menor de idade, esperando..."

Sempre em mente no desenolvimento

O que é Back-End e de importância vital.

A programação é uma ferramenta de criação e resolução.

A **Lógica de Programação** como o fundamento de tudo:

Algoritmos

Variáveis, Tipos de Dados, Operadores

Estruturas Condicionais (IF/ELSE)

Estruturas de Repetição (FOR/WHILE)

A importância da **prática contínua**.

Sintaxe em programação

A sintaxe de uma linguagem de programação são o conjunto de regras que definem como o código deve ser escrito e estruturado para ser válido e compreendido pelo computador e pelos humanos. Tal como a gramática em uma língua humana, a sintaxe determina a organização de palavras-chave,
símbolos, pontuação e a estrutura geral das expressões e comandos, garantindo que o código esteja correto e sem erros que impeçam sua execução.

Componentes da sintaxe:

- **Regras de formação:**
Descrevem como elementos como palavras-chave (ex: `print()`), operadores (ex: `+`, `-`), e outros símbolos devem ser combinados.
- **Estrutura:**
Define a ordem e o arranjo dos elementos para formar declarações, instruções e programas completos.

- **Validade:**
Garante que o código obedeça às regras da linguagem, evitando erros de sintaxe que impedem o computador de interpretar o código.

Exemplos:

- **Linguagem humana:** A sintaxe determina que "Maria comeu a maçã" está correto, enquanto "Maçã comeu Maria" não.

- **Programação (Python):** Para imprimir algo na tela, o comando `print()` é o uso correto da sintaxe.

Diferença entre sintaxe e semântica:

- **Sintaxe:** Refere-se à forma ou estrutura correta do código.

- **Semântica:** Diz respeito ao significado e a lógica por trás do código, ou seja, o que o código faz realmente.

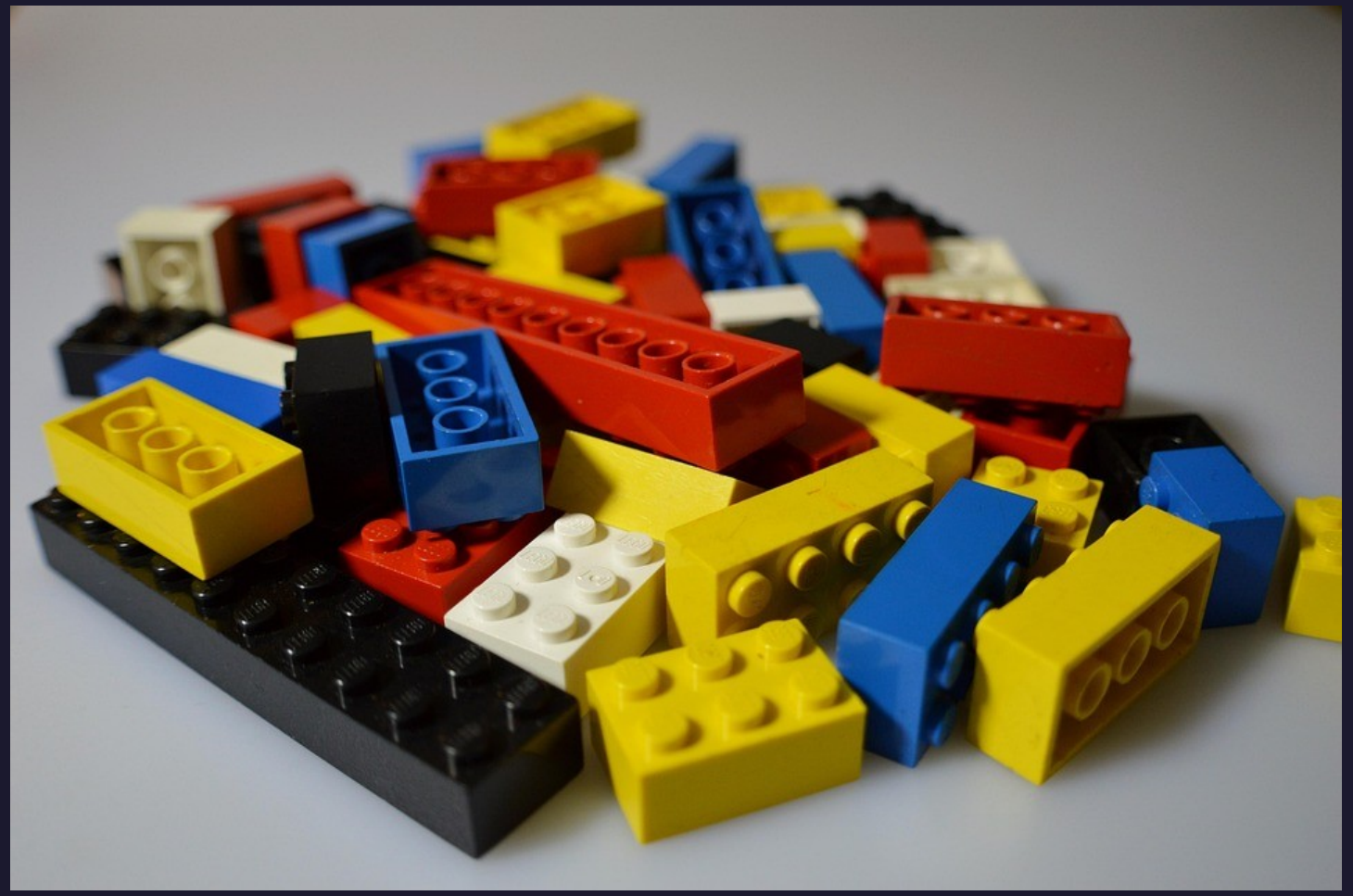
Funções nativas mais usuais no Python

Existem mais de 70 funções nativas no Python que são usadas para várias situações.

Essas funções visam simplificar o código e melhorar o aproveitamento de partes funcionais que já estão devidamente especificadas na comunidade DEV.

Como essas funções já estão registradas na comunidade e vinculadas no Python, NÃO pode ser realizada a criação de uma outra função de mesmo nome que as funções nativas.

Funções: 07 Tipos de funções existentes



Tipo: Manipulação de Listas e Sequências

Tipo: Entrada e Saída

Tipo: Conversões de Tipo

Tipo: Controle e Testes

Tipo: Strings

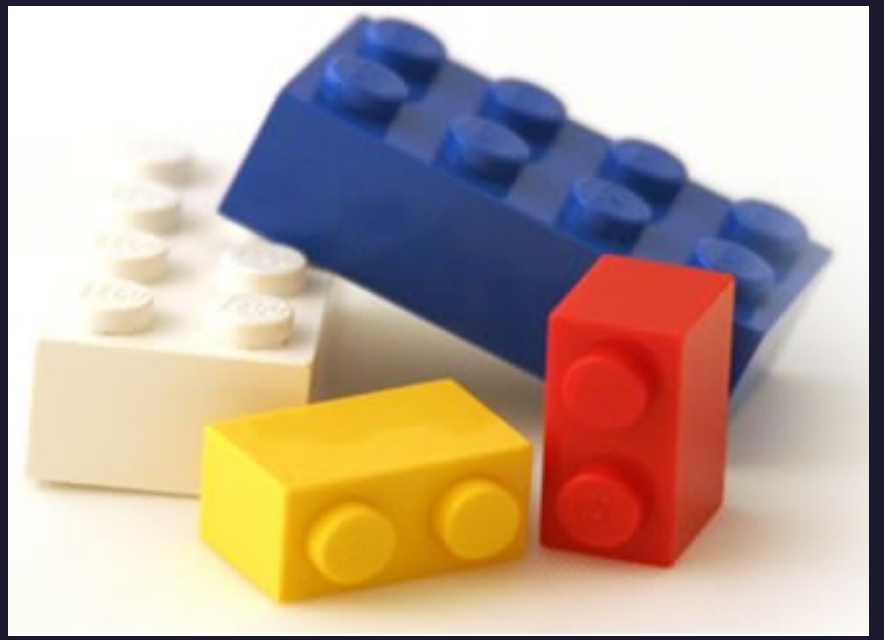
Tipo: Números

Tipo: Estruturas de Dados

Funções do Tipo Manipulação de Listas e Sequências

Manipulação de Listas e Sequências

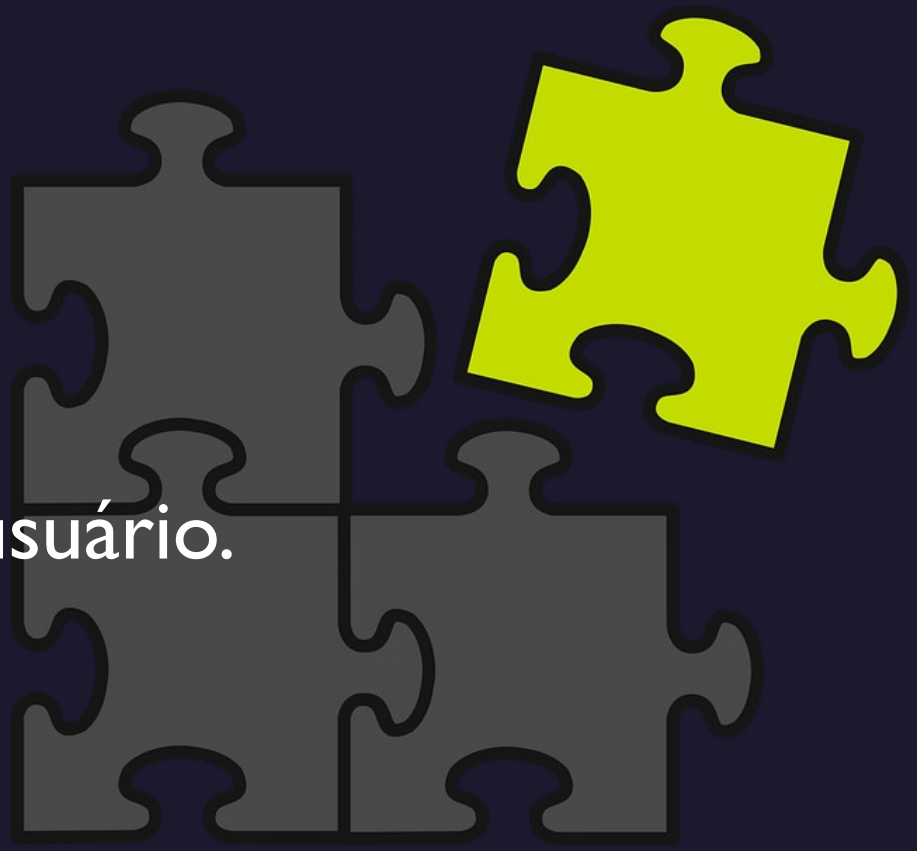
- len(obj) → retorna o tamanho (quantidade de itens).
- max(iterável) → retorna o maior valor.
- min(iterável) → retorna o menor valor.
- sum(iterável) → soma todos os elementos numéricos.
- sorted(iterável) → retorna uma lista ordenada.
- reversed(iterável) → retorna um iterador invertido.
- enumerate(iterável) → gera pares (índice, valor).
- zip(lista1, lista2, ...) → combina listas em pares ou tuplas.



Funções do Tipo Entrada e Saída

Entrada e Saída

- print() → exibe dados na tela.
- input() → recebe dados digitados pelo usuário.



Funções do Tipo Conversões de tipo

Conversões de Tipo

- int(x) → converte para inteiro.
- float(x) → converte para número decimal.
- str(x) → converte para string.
- bool(x) → converte para booleano (True ou False).
- list(x) → converte para lista.
- tuple(x) → converte para tupla.
- set(x) → converte para conjunto (sem duplicatas).
- dict() → cria ou converte para dicionário.

Funções do Tipo Controle e Testes

Controle e Testes

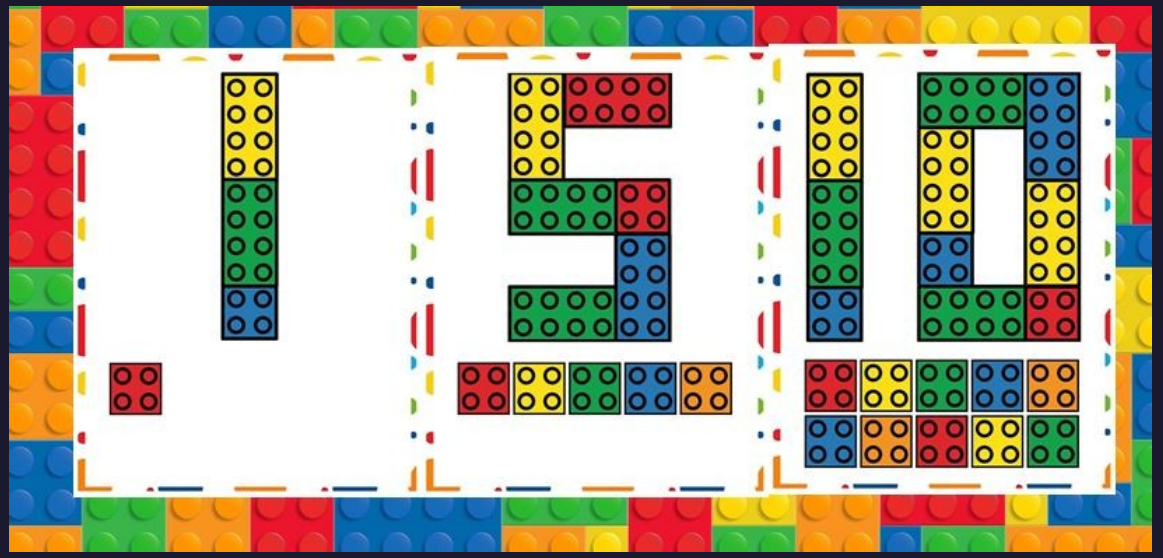
- type(x) → mostra o tipo da variável.
- isinstance(x, tipo) → testa se um objeto é de um certo tipo.
- id(x) → mostra o identificador único do objeto na memória.
- all(iterável) → retorna True se todos forem verdadeiros.
- any(iterável) → retorna True se algum for verdadeiro.

Funções do Tipo Strings

Strings

- len(texto) → tamanho da string.
- texto.upper() → tudo maiúsculo.
- texto.lower() → tudo minúsculo.
- texto.title() → primeira letra maiúscula.
- texto.strip() → remove espaços extras.
- texto.split(separador) → divide em lista.
- ' '.join(lista) → junta elementos em string.
- texto.replace("a", "b") → troca partes da string.

Funções do Tipo Números



Números

- abs(x) → valor absoluto.
- round(x, n) → arredonda com n casas decimais.
- pow(x, y) → calcula x elevado a y (igual x**y).
- divmod(x, y) → retorna (quociente, resto).

Funções do Tipo Estrutura de Dados

Estruturas de Dados

- range(início, fim, passo) → gera uma sequência de números.
- dict.keys() → retorna as chaves de um dicionário.
- dict.values() → retorna os valores.
- dict.items() → retorna pares (chave, valor).
- Append(x) → coloca X Inteiro no final da Lista.
  quando quer adicionar **um único item**.
- Extend([a, b, c]) → coloca A, depois B, e C na Lista.
  quando quer adicionar **vários itens de uma vez**.

Métodos amplamente utilizáveis

append()

- **Adiciona UM item no final da lista.**
- Esse item pode ser qualquer coisa: número, string, até outra lista.

extend()

- **Adiciona vários itens de outra lista (ou iterável) individualmente.**
- Em vez de colocar a lista inteira dentro, ele “desempacota” os elementos.

Obrigado

Wellington S. S. Silva

guitomw@outlook.com