Introdução à Programação em Python

Overview Básico sobre Sintaxe, Variáveis e Loops

Objetivo: Aprender a sintaxe básica de Python e as estruturas de controle fundamentais (variáveis e loops) para criar scripts simples focados em automação de tarefas.
Público-alvo: Iniciantes em programação que desejam começar com Python.

---

1 Módulo 1: Primeiros Passos e Sintaxe Básica

1.1 1.1 O Que é Python?

- Linguagem de programação de alto nível, interpretada e de propósito geral.
- Conhecida por sua **legibilidade** e sintaxe limpa, que se assemelha ao inglês.
- Amplamente usada em Web Development, Data Science e, principalmente, **Automação**.

1.2 1.2 O Ambiente de Trabalho

- Instalação do Python (e um gerenciador de pacotes como `pip`).
- Uso de um **IDE** (Integrated Development Environment) ou editor de código (VS Code, PyCharm).
- Como executar um um script (`.py`) no terminal.

1.3 1.3 Sintaxe Fundamental

- **Comentários:** Usados para explicar o código e ignorados pelo interpretador.

```
# Este é um comentário de linha única
"""
Este é um comentário
de múltiplas linhas
(docstring)
"""
```

- **Indentação:** Em Python, a indentação (espaços em branco, geralmente 4) **define blocos de código**. É obrigatória e crucial.

```
if True:
    print("Este código está dentro do bloco")
print("Este código está fora do bloco")
```

- **Função `print()`:** Usada para exibir informações no console.

```
print("Olá, Mundo!")
```

---

2 Módulo 2: Variáveis e Tipos de Dados

2.1 2.1 Variáveis

Variáveis são nomes simbólicos que fazem referência a um valor armazenado na memória.

- **Declaração:** A declaração e atribuição são feitas com o sinal de igual (=). Python é dinamicamente tipada (não precisamos declarar o tipo).

```
nome = "Alice"
idade = 30
preco_unitario = 19.99
```

2.2 2.2 Tipos de Dados Primitivos

Tipo	Nome em Python	Descrição	Exemplo
Texto	`str` (string)	Sequência de caracteres.	`"Python é incrível"`
Número Inteiro	`int` (integer)	Números inteiros.	`10`, `-5`
Número Decimal	`float` (floating point)	Números com casas decimais.	`3.14`, `0.5`
Booleano	`bool` (boolean)	Valores lógicos: `True` ou `False`.	`True`

2.3 2.3 Operadores

- **Aritméticos:** +, -, *, /, % (módulo/resto), ** (potência).
- **Comparação:** Retornam um valor `bool`. == (igual a), != (diferente de), >, <, >=, <=.
- **Lógicos:** Usados para combinar expressões booleanas.
  - `and` (E lógico)
  - `or` (OU lógico)
  - `not` (NÃO lógico)

---

3 Módulo 3: Estruturas de Controle (Loops)

As estruturas de controle permitem que o programa tome decisões e repita ações.

3.1 3.1 Condicionais (`if`, `elif`, `else`)

Executam um bloco de código se uma condição for verdadeira.

```
hora = 14

if hora < 12:
    print("Bom dia!")
elif hora < 18:
    # Este bloco é executado se a primeira condição for False
    # e esta segunda condição for True
    print("Boa tarde!")
else:
    # Este bloco é executado se todas as condições anteriores forem False
    print("Boa noite!")
```

3.2 3.2 Loop `for`

Usado para iterar (percorrer) sequências de itens (como listas ou ranges).

```
# Itera sobre um intervalo de números (de 0 até 4)
print("Loop FOR com range:")
for i in range(5):
    print(i)

# Itera sobre itens de uma lista (abordada em módulos futuros, mas essencial para loops)
tarefas = ["Ler e-mail", "Codificar", "Testar"]
print("\nLoop FOR com lista:")
for item in tarefas:
    print(item)
```

3.3 3.3 Loop `while`

Executa um bloco de código repetidamente **enquanto** uma condição for verdadeira.

```
contador = 0

print("Loop WHILE:")
while contador < 3:
    print(f"Contando: {contador}")
    # É CRÍTICO atualizar a variável de controle para evitar loops infinitos
    contador += 1
```

---

4 Módulo 4: Automação com Scripts Simples

4.1 4.1 Input do Usuário

Usando a função `input()` para interagir com o usuário e coletar dados.

```
nome_usuario = input("Digite seu nome: ")
print(f"Olá, {nome_usuario}! Bem-vindo ao seu primeiro script.")
```

4.2 4.2 Projeto de Automação Simples: Renomeador de Arquivos (Simulado)

Este script usa a sintaxe e os loops aprendidos para simular a criação de nomes de arquivos com um padrão.

```
# Importa o módulo 'os' (Operating System), essencial para automação
# Em um script real, você usaria 'os.rename()'
import os
import time # Apenas para simular uma pausa

# 1. Definir uma variável com o prefixo
prefixo = input("Qual prefixo você quer usar nos arquivos? ")

# 2. Loop para criar 3 nomes de arquivos
for i in range(1, 4):
    # Concatenação de strings e conversão de int para str
    nome_arquivo = f"{prefixo}_{i}.txt"

    print(f"Simulando a criação/renomeação para: {nome_arquivo}")
    time.sleep(0.5) # Pausa para simular processamento

print("\nAutomação simples concluída!")
```

Exercício Final: Mini Calculadora

Desafio: Crie um script que:

1. Peça ao usuário dois números.
2. Pergunte qual operação ele deseja realizar (+, −, ∗, /).
3. Use estruturas **condicionais (`if`/`elif`)** para realizar o cálculo e exibir o resultado.

(Solução em anexo – não fornecida para encorajar a prática.)