Fundamentos do Python

Python é uma das linguagens de programação mais populares e queridas do mundo, e por um bom motivo: sua **sintaxe simples e legível** a torna incrivelmente fácil de aprender, mas poderosa o suficiente para aplicações complexas, desde desenvolvimento web até ciência de dados e inteligência artificial.

Se você quer começar a programar, Python é uma excelente escolha. Vamos cobrir os pilares fundamentais que você precisa conhecer.

1. O "Olá, Mundo!" e a Sintaxe Básica

A tradição ao aprender uma nova linguagem é fazer o computador dizer "Olá, Mundo!". Em Python, é apenas uma linha:

Python
```
print("Olá, Mundo!")
```

Pontos-chave da Sintaxe:

- **Legibilidade:** O código Python é feito para parecer com a língua inglesa, usando palavras como `if`, `for`, `while`, `and`, `or`, `not`.
- **Sem ponto e vírgula:** Diferente de muitas linguagens (como Java ou C#), você não precisa terminar as linhas com `;`.
- **Indentação é crucial:** Este é o ponto mais importante! Python usa espaços em branco (geralmente 4 espaços) no início de uma linha para definir blocos de código (como o que está dentro de um `if` ou de uma função). A indentação não é opcional, ela faz parte da sintaxe.

2. Variáveis e Tipos de Dados

Pense em **variáveis** como caixas etiquetadas onde você armazena informações (dados). Em Python, você não precisa declarar o tipo de dado que a caixa vai guardar; a linguagem descobre isso sozinha.

Os tipos de dados básicos mais comuns são:

- **String (`str`):** Texto, sempre entre aspas (`"` ou `'`).

Python
```
nome = "Ana"
```

- **Inteiro (`int`):** Números inteiros, sem casas decimais.

Python
```
idade = 25
```

- **Float (`float`):** Números com casas decimais (use o ponto `.` como separador).

Python
```
altura = 1.65
```

- **Booleano (`bool`):** Representa valores de Verdadeiro ou Falso.

Python
```
e_estudante = True
```

3. Operadores

Operadores são os símbolos que realizam ações com suas variáveis e valores.

- **Aritméticos:** Para matemática.
  - `+` (adição), `-` (subtração)
  - `*` (multiplicação), `/` (divisão)
  - `**` (exponenciação, ex: `2**3` é 8)
- **Comparação:** Para comparar valores; o resultado é sempre um Booleano (`True` ou `False`).
  - `==` (igual a)
  - `!=` (diferente de)
  - `>` (maior que), `<` (menor que)
  - `>=` (maior ou igual a), `<=` (menor ou igual a)
- **Lógicos:** Para combinar valores Booleanos.
  - `and` (ambos precisam ser `True`)
  - `or` (apenas um precisa ser `True`)
  - `not` (inverte o valor, `not True` é `False`)

4. Estruturas de Controle: Tomando Decisões

Seu código precisa tomar decisões. Para isso, usamos as estruturas condicionais `if`, `elif` (abreviação de "else if") e `else`.

Lembre-se: o bloco de código "dentro" do `if` deve estar **indentado**.

Python
```
idade = 20

if idade < 18:
    print("Você é menor de idade.")
elif idade == 18:
    print("Você tem exatamente 18 anos.")
else:
    print("Você é maior de idade.")
```

5. Estruturas de Repetição: Loops

Loops permitem que você execute o mesmo bloco de código várias vezes.

Loop `for`

Usado para "iterar" (ou seja, passar por cada item) de uma sequência, como uma lista.

Python
```
# Uma lista de frutas
frutas = ["maçã", "banana", "uva"]

for fruta in frutas:
    print(f"Eu gosto de {fruta}")
```

Loop `while`

Usado para repetir um bloco de código *enquanto* uma condição for verdadeira (`True`).

Python
```
contador = 0

while contador < 5:
```

```
    print(f"O número é {contador}")
    contador = contador + 1  # Incrementa o contador
```

6. Estruturas de Dados: Coleções

Além das variáveis simples, você precisará agrupar dados. As duas estruturas mais fundamentais são:

- **Listas (`list`):** Uma coleção ordenada e mutável de itens. Usa colchetes `[]`.

Python
```
#         0        1        2
cores = ["azul", "verde", "vermelho"]
print(cores[0]) # Acessa o primeiro item (índice 0), imprime "azul"
cores.append("amarelo") # Adiciona um item ao final
```

- **Dicionários (`dict`):** Uma coleção não ordenada de pares `chave: valor`. Usa chaves `{}`. É ótimo para organizar dados que têm um rótulo.

Python
```
pessoa = {
    "nome": "Carlos",
    "idade": 30,
    "cidade": "São Paulo"
}
print(pessoa["nome"]) # Acessa o valor da chave "nome", imprime
"Carlos"
```

7. Funções

Funções são blocos de código reutilizáveis que realizam uma tarefa específica. Você "define" uma função com a palavra-chave `def` e pode "chamá-la" pelo nome quando precisar dela.

Isso evita que você escreva o mesmo código várias vezes.

Python

```
# Definindo a função
def saudacao(nome):
    """Uma função simples que cumprimenta o usuário."""
    print(f"Olá, {nome}! Bem-vindo(a).")

# Chamando (usando) a função
saudacao("Beatriz")
saudacao("João")
```

Próximos Passos

Estes são os blocos de construção de qualquer programa Python. A partir daqui, a jornada envolve combinar esses conceitos para criar programas mais complexos e explorar as vastas bibliotecas que o Python oferece.