

Aprendendo Python

Um Guia Prático para Iniciantes na Programação



Título: Fundamentos de Programação em Python

Autor: Patrick Yuri

Algoritmos

Um **algoritmo** é uma sequência finita de passos bem definidos que, quando executados, resolvem um problema específico. Pense em um algoritmo como uma receita de bolo: cada passo deve ser seguido na ordem correta para obter o resultado desejado. **Exemplo:**

- **Entrada (input):** Coleta de dados fornecidos pelo usuário ou por um sistema.
- **Processamento:** Manipulação dos dados (cálculos, validações, filtragem).
- **Saída (print):** Apresentação do resultado final para o usuário.

Regra: Todo algoritmo deve ter um início, um fim e passos bem definidos entre eles.

Variáveis e Tipos de Dados

No Python, a tipagem é dinâmica: a variável assume automaticamente o tipo do valor que recebe. **Tipos de dados comuns em Python:**

Tipo (Conceito)	Nome no Python	Exemplo Prático
Texto	str	"Olá Mundo"
Inteiro	int	25
Real (Decimal)	float	1.75
Booleano	bool	True ou False
Lista (Mutável)	list	[1, 2, 3]
Tupla (Imutável)	tuple	(10, 20)
Dicionário	dict	{"id": 1, "nome": "Ana"}

Table 1: Tipos de dados fundamentais em Python

Nota: A tipagem dinâmica significa que não é necessário declarar explicitamente o tipo da variável — Python identifica automaticamente.

Python é case-sensitive (sensível a letras maiúsculas e minúsculas).

Isso significa que o interpretador trata identificadores com letras diferentes como entidades completamente distintas.

Exemplos:

Se você definir variáveis com variações de maiúsculas, o Python criará objetos diferentes na memória:

- escola = "ETEC"
- Escola = "FATEC"
- ESCOLA = "Ensino Técnico"

Operadores

Operadores Aritméticos:

Operador	Descrição
+	Soma
-	Subtração
*	Multiplicação
/	Divisão
//	Divisão Inteira
%	Resto (Módulo)
**	Potência

Table 2: Operadores aritméticos

Operadores de Comparação:

Operador	Descrição
<code>==</code>	Igual
<code>!=</code>	Diferente
<code>></code>	Maior
<code><</code>	Menor
<code>>=</code>	Maior ou igual
<code><=</code>	Menor ou igual

Table 3: Operadores relacionais

Operadores de Atribuição:

Operador	Descrição
<code>=</code>	Atribui valor simples
<code>+=</code>	Soma ao valor atual (ex: <code>x += 1</code>)
<code>-=</code>	Subtrai do valor atual
<code>*=</code>	Multiplica o valor atual
<code>/=</code>	Divide o valor atual

Table 4: Operadores de atribuição

Funções Essenciais

Manipulação e Inspeção

Python possui diversas funções built-in (internas) que facilitam a programação.

Algumas importantes são:

- **print()** # Exibe dados na tela
- **input()** # Recebe dados do teclado (retorna sempre como texto/str)
- **len()** # Retorna o tamanho/comprimento de um objeto
- **sum()** # Soma elementos de uma coleção numérica
- **type()** # Retorna o tipo da variável

Exemplo:

```
nome = input("Digite seu nome: ")
print(f"Olá, {nome}!")
print(f"Seu nome tem {len(nome)} caracteres")
print(f"O tipo da variável é: {type(nome)}")
```

Estruturas de Decisão (Condicionais)

Estruturas que controlam o fluxo do programa baseadas em condições:

- **Simples:** Apenas um bloco if
- **Composta:** Blocos if, elif e else
- **Aninhada:** Uso de if dentro de if para múltiplas verificações encadeadas

Exemplo:

```
nota = 7.5

if nota >= 9:
    print("Excelente!")
elif nota >= 7:
    print("Aprovado")
else:
    print("Recuperação")
```

Saída: Aprovado

Regra: A indentação é obrigatória — todos os comandos dentro de cada bloco devem estar alinhados.

Operadores Lógicos

Os operadores lógicos combinam condições.

- **AND** - Retorna `True` se ambas as condições forem verdadeiras.
- **OR** - Retorna `True` se pelo menos uma das condições for verdadeira.
- **NOT** - Inverte o valor de uma condição.

Exemplo:

```
if idade >= 18 or tem_autorizacao:  
    print("Pelo menos uma condição foi atendida")
```

Tabela Verdade

Operador and (E)

Resulta em `True` apenas se TODOS forem verdadeiros.

Condição A	Condição B	Resultado
True	True	True
True	False	False
False	True	False
False	False	False

Table 5: Tabela verdade do operador AND

Operador or (OU)

Resulta em True se PELO MENOS UM for verdadeiro.

Condição A	Condição B	Resultado
True	True	True
True	False	True
False	True	True
False	False	False

Table 6: Tabela verdade do operador OR

Exemplo prático:

```
idade = 20
tem_autorizacao = True
```

Usando AND

```
if idade >= 18 and tem_autorizacao:
    print("Acesso permitido")
```

Usando OR

```
if idade >= 18 or tem_autorizacao:
    print("Pelo menos uma condição foi atendida")
```

Loops (Laços de Repetição)

As estruturas de repetição permitem executar um bloco de código várias vezes. São fundamentais para evitar repetição de código manual e para processar coleções de dados.

Loop **for**

Utilizado quando sabemos previamente quantas vezes queremos repetir o código ou quando precisamos percorrer uma coleção de itens (como uma lista ou os caracteres de um texto).

```
# Repete 3 vezes (os valores de i serão 0, 1 e 2)

for i in range(3):
    print(f"Repetição número {i}")
```

Saída:

```
Repetição número 0
Repetição número 1
Repetição número 2
```

Loop **while** (Enquanto)

Utilizado quando não sabemos o número exato de repetições, mas temos uma condição que deve permanecer verdadeira para que o loop continue rodando.

```
contador = 0

while contador < 2:
    print(f"Contador está em: {contador}")
    contador += 1 # Importante para evitar um loop infinito!
```

Saída:

```
Contador está em: 0
Contador está em: 1
```

E o do... while?

Diferente de linguagens como C, Java ou JavaScript, o Python não possui uma estrutura do while nativa. No `do while`, o código é executado pelo menos uma vez antes de verificar a condição. Em Python, simulamos esse comportamento criando um loop infinito (`while True:`) e usando o comando `break` para interrompê-lo por dentro.

Exemplo:

```
while True:  
    # Este bloco vai executar obrigatoriamente a primeira vez  
    senha = input("Digite a senha: ")  
  
    if senha == "123":  
        print("Acesso liberado!")  
        break      # Quebra o loop e sai da repetição  
    else:  
        print("Senha incorreta, tente novamente.")
```

Exercícios Práticos

1. Acesso o Google Colabe e crie um algoritmo que peça dois números ao usuário e imprima a soma, a subtração, a multiplicação e a divisão deles.
2. Crie um algoritmo que peça a idade do usuário e diga se ele é maior ou menor de idade.
3. Crie um algoritmo que imprima os números de 1 a 10 usando um loop .
4. Crie um algoritmo que peça um número ao usuário e imprima a tabuada desse número (de 1 a 10) usando um loop .

Boas Práticas

- Use nomes de variáveis descritivos. Ex: **cor = 'Azul'** ou **animal = 'Arara'**
- Comente seu código para torná-lo mais fácil de entender.
Utilize o hashtag para comentar
- Use indentação consistente para melhorar a legibilidade.
- Divida seu código em funções menores e reutilizáveis.
- Teste seu código frequentemente para garantir que ele esteja funcionando corretamente.

Referências

Documentação Oficial Python: <https://docs.python.org>

Tutorial W3Schools - Python: <https://www.w3schools.com/python/>

Real Python - Python Tutorials: <https://realpython.com>

ALVES, William Pereira. Programação python: aprenda de forma rápida. São Paulo: Expressa, 2021. E-book. Disponível em:
<https://app.minhabiblioteca.com.br/books/9786558110149>.

Professor Patrick Yuri - [Github](#) | [LinkedIn](#)

Atualizado em: Fevereiro de 2026

Curso: Técnico de Ciências de Dados