



# Extensão

# Python

# Introdução

Encontro 01 – Introdução

Professor Luiz Augusto Rodrigues

*luiz.a.rodrigues@cogna.com.br*



# Sumário

- Conceitos
  - Histórico
  - Características
  - Evolução
  - Futuro
- Variáveis e Entrada de Dados - String
- Variáveis e Entrada de Dados - Números



**Extensão**

**Python**

**Introdução**

**Conceitos**



Python é uma linguagem de programação de alto nível, amplamente utilizada e conhecida por sua simplicidade e legibilidade.

Criada por Guido van Rossum e lançada pela primeira vez em 1991, Python foi projetada para enfatizar a legibilidade do código, permitindo que os programadores expressem conceitos de maneira clara e concisa, com menos linhas de código do que em muitas outras linguagens.

Python foi desenvolvido no final da década de 1980 e início dos anos 1990 por Guido van Rossum, um programador holandês.

Van Rossum trabalhava no Instituto de Pesquisa Nacional para Matemática e Ciência da Computação (CWI) na Holanda, onde decidiu criar uma nova linguagem de programação que fosse simples e intuitiva, mas poderosa o suficiente para lidar com tarefas complexas.

Ele se inspirou em diversas linguagens, incluindo ABC, que ele ajudou a desenvolver anteriormente, e que influenciou Python no que diz respeito à simplicidade e facilidade de uso.



Python foi lançado como uma linguagem de código aberto, o que permitiu que desenvolvedores de todo o mundo contribuíssem para sua evolução.

Essa abordagem colaborativa foi fundamental para o crescimento e a popularidade da linguagem.

Python é uma linguagem multiparadigma, o que significa que suporta diferentes estilos de programação, incluindo programação orientada a objetos, programação imperativa e programação funcional.

Algumas das principais características do Python incluem:

## **Sintaxe Simples e Legível:**

A sintaxe do Python é clara e intuitiva, o que facilita a aprendizagem e o uso. Isso torna Python uma excelente escolha para iniciantes.

## **Interpretação e Portabilidade:**

Python é uma linguagem interpretada, o que significa que o código é executado linha por linha. Isso facilita a depuração e a portabilidade, permitindo que o Python seja executado em diferentes sistemas operacionais sem a necessidade de recompilação.

## **Bibliotecas Padrão Ricas:**

Python vem com uma vasta biblioteca padrão, que oferece módulos e funções para praticamente todas as necessidades, desde manipulação de strings até comunicação em rede.



## Comunidade Ativa e Ecossistema:

A comunidade Python é uma das maiores e mais ativas do mundo, com inúmeros recursos disponíveis, incluindo tutoriais, bibliotecas de terceiros, e fóruns de discussão.

Desde seu lançamento inicial, Python passou por várias versões e atualizações.

A primeira versão significativa foi Python 2.0, lançada em 2000, que trouxe novas funcionalidades, como a coleta de lixo para gerenciamento automático de memória.

No entanto, a versão mais significativa foi Python 3.0, lançada em 2008. Python 3 introduziu mudanças que não eram compatíveis com versões anteriores, mas que melhoraram a consistência e a eficiência da linguagem.

Apesar da transição difícil, a maioria dos desenvolvedores migrou para Python 3, e o suporte ao Python 2 foi oficialmente encerrado em 2020.

Python continua a evoluir, com atualizações regulares que introduzem novos recursos e melhorias de desempenho.

A linguagem tem se mantido relevante em diversos campos, incluindo desenvolvimento web, ciência de dados, inteligência artificial e automação.

A combinação de simplicidade, poder e uma comunidade ativa garante que Python permanecerá uma linguagem central no desenvolvimento de software por muitos anos.

Com a crescente popularidade da inteligência artificial, aprendizado de máquina e análise de dados, Python está mais popular do que nunca.

Ferramentas como TensorFlow, PyTorch e Pandas, todas escritas em Python, são exemplos da importância da linguagem no mundo atual.

Em resumo, Python é uma linguagem que evoluiu com o tempo para atender às necessidades dos desenvolvedores modernos, mantendo-se acessível para iniciantes, mas poderosa o suficiente para ser usada em projetos de grande escala.

# **Extensão**

# **Python**

# **Introdução**

## **Variáveis e Entrada de Dados - String**



Em Python, nomes de variáveis devem iniciar obrigatoriamente com uma letra, mas podem conter números e o símbolo sublinha (\_). Vejamos exemplos de nomes válidos e inválidos em Python.

Nome	Válido	Comentários
a1	Sim	Embora contenha um número, o nome a1 inicia com letra.
velocidade	Sim	Nome formado por letras.
velocidade90	Sim	Nome formado por letras e números, mas iniciado por letra.
salário_médio	Sim	O símbolo sublinha (__) é permitido e facilita a leitura de nomes grandes.
salário médio	Não	Nomes de variáveis não podem conter espaços em branco.
_b	Sim	O sublinha (__) é aceito em nomes de variáveis, mesmo no início.
1a	Não	Nomes de variáveis não podem começar com números.



A versão 3 da linguagem Python permite a utilização de acentos em nomes de variáveis, pois, por padrão, os programas são interpretados utilizando-se um conjunto de caracteres chamado UTF-8, capaz de representar praticamente todas as letras dos alfabetos conhecidos.

Variáveis têm outras propriedades além de nome e conteúdo. Uma delas é conhecida como tipo e define a natureza dos dados que a variável armazena.

Python tem vários tipos de dados, mas os mais comuns são números inteiros, números de ponto flutuante e strings. Além de poder armazenar números e letras, as variáveis em Python também armazenam valores como verdadeiro ou falso. Dizemos que essas variáveis são do tipo lógico.

Em Python, uma variável é criada e atribuída simplesmente ao nomeá-la e definir um valor para ela.

Não é necessário especificar o tipo de dado ao declarar a variável, pois Python é uma linguagem de tipagem dinâmica.

Isso significa que o tipo da variável é determinado automaticamente com base no valor atribuído a ela.

# Variáveis

Aqui está um exemplo básico:

```
idade = 25
```

```
nome = "Maria"
```

```
altura = 1.75
```

Nesse exemplo, criamos três variáveis: idade, nome, e altura.

Python automaticamente reconhece que idade é um inteiro, nome é uma string e altura é um número de ponto flutuante.

# Regras de Nomeação

---

Ao nomear variáveis em Python, é importante seguir algumas regras:

## **Nomes válidos:**

Os nomes de variáveis podem conter letras (maiúsculas e minúsculas), números e o caractere de sublinhado (\_).

No entanto, devem sempre começar com uma letra ou um sublinhado.

Por exemplo: `_variavel`, `idade`, `altura2` são nomes válidos.

## **Nomes inválidos:**

Nomes de variáveis não podem começar com números e não podem conter espaços ou caracteres especiais como !, @, #, etc.

Por exemplo: 2variavel, nome da pessoa, nome! são inválidos.

## **Palavras reservadas:**

Python possui palavras reservadas que não podem ser usadas como nomes de variáveis. Exemplos incluem if, else, while, for, try, entre outros.

# Reatribuição de Variáveis

Em Python, uma variável pode ser reatribuída a qualquer momento, e o tipo da variável pode ser alterado dinamicamente. Por exemplo:

```
numero = 10  
print(numero)  # Saída: 10  
  
numero = "dez"  
print(numero)  # Saída: dez
```

Nesse exemplo, a variável `numero` inicialmente armazena um valor inteiro, mas depois é reatribuída para armazenar uma string.



# Anotação de Tipos (Type Hinting)

Embora Python seja uma linguagem de tipagem dinâmica, a partir da versão 3.5 foi introduzido o conceito de "Type Hinting", que permite que você indique o tipo esperado de uma variável. Isso pode ser útil para documentação e para ferramentas que fazem verificação de tipos.

Exemplo de anotação de tipo:

```
idade: int = 25  
nome: str = "Maria"  
altura: float = 1.75
```

Note que as anotações de tipos são opcionais e não são forçadas pelo interpretador Python, ou seja, elas não impedem que a variável seja atribuída a um valor de outro tipo.

- A declaração de variáveis em Python é simples e flexível, graças à tipagem dinâmica da linguagem.
- Essa simplicidade, combinada com a clareza e legibilidade do código, é uma das razões pelas quais Python é tão popular entre iniciantes e desenvolvedores experientes.
- Compreender como declarar e manipular variáveis é uma habilidade fundamental que será utilizada em praticamente todos os programas que você criar em Python.

# Operadores relacionais

Operador	Operação	Símbolo matemático
==	igualdade	=
>	maior que	>
<	menor que	<
!=	diferente	≠
>=	maior ou igual	≥
<=	menor ou igual	≤

# Operadores lógicos

Operador Python	Operação
not	não
and	e
or	ou

# Variáveis String

Variáveis do tipo string armazenam cadeias de caracteres como nomes e textos em geral.

Chamamos cadeia de caracteres uma sequência de símbolos como letras, números, sinais de pontuação etc.

Exemplo: João e Maria comem pão.

Nesse caso, João é uma sequência com as letras J, o, ã, o.

Em python:

```
Frase = "João e Maria comem pão."
```

# Manipulação de Strings

---

Strings são sequências de caracteres amplamente utilizadas em Python para armazenar e manipular texto.

A linguagem oferece uma variedade de métodos e técnicas que facilitam o trabalho com strings, desde a concatenação até a formatação e análise de texto.



# Manipulação de Strings

## Criação e Declaração de Strings

Strings em Python podem ser criadas usando aspas simples ('...'), aspas duplas ("..."), ou até aspas triplas ("..." ou "...") para strings multilinhas.

```
nome = "Maria"
```

```
saudacao = 'Olá'
```

```
descricao = """Este é um exemplo  
de string multilinha."""
```

# Manipulação de Strings

## Concatenação de Strings

A concatenação é o processo de unir duas ou mais strings. Em Python, isso é feito usando o operador +.

```
primeiro_nome = "Ana"  
sobrenome = "Silva"  
nome_completo = primeiro_nome + " " + sobrenome  
print(nome_completo)  # Saída: Ana Silva
```

# Manipulação de Strings

## Repetição de Strings

Você pode repetir uma string usando o operador \*.

```
risada = "ha" * 3  
print(risada)  # Saída: hahaha
```

# Manipulação de Strings

## Acessando Caracteres e Slices

Strings em Python são indexadas, o que significa que você pode acessar caracteres individuais usando índices. Os índices começam em 0.

```
palavra = "Python"
primeira_letra = palavra[0]
ultima_letra = palavra[-1]
print(primeira_letra)    # Saída: P
print(ultima_letra)     # Saída: n
```

# Principais Métodos de Strings

Python oferece uma série de métodos embutidos para manipular strings:

**len():**

Retorna o comprimento da string.

```
tamanho = len("Python")  
print(tamanho)    # Saída: 6
```

# Principais Métodos de Strings

Python oferece uma série de métodos embutidos para manipular strings:

## **lower() e upper():**

Convertem a string para minúsculas ou maiúsculas, respectivamente.

```
texto = "Python"  
print(texto.lower())    # Saída: python  
print(texto.upper())    # Saída: PYTHON
```



# Principais Métodos de Strings

Python oferece uma série de métodos embutidos para manipular strings:

## **strip():**

Remove espaços em branco do início e do fim da string.

```
texto = "    Olá Mundo!    "  
print(texto.strip())    # Saída: Olá Mundo!
```

# Principais Métodos de Strings

Python oferece uma série de métodos embutidos para manipular strings:

## **replace():**

Substitui uma substring por outra.

```
texto = "Olá Mundo"
novo_texto = texto.replace("Mundo", "Python")
print(novo_texto)  # Saída: Olá Python
```

# Principais Métodos de Strings

Python oferece uma série de métodos embutidos para manipular strings:

## **split():**

Divide a string em uma lista, usando um delimitador.

```
texto = "um,dois,três"  
partes = texto.split(",")  
print(partes)  # Saída: ['um', 'dois', 'três']
```

# Principais Métodos de Strings

Python oferece uma série de métodos embutidos para manipular strings:

## **join():**

Junta uma lista de strings em uma única string, com um delimitador.

```
lista = ['um', 'dois', 'três']  
texto = ", ".join(lista)  
print(texto)    # Saída: um, dois, três
```

# Formatação de Strings

Python permite formatar strings de maneira flexível usando várias abordagens:

## Interpolação:

```
nome = "Ana"
idade = 30
texto = f"{nome} tem {idade} anos."
print(texto) # Saída: Ana tem 30 anos.
```

# Formatação de Strings

Python permite formatar strings de maneira flexível usando várias abordagens:

## Método format():

```
texto = "{} tem {} anos.".format(nome, idade)  
print(texto)  # Saída: Ana tem 30 anos.
```

# **Extensão**

# **Python**

# **Introdução**

**Variáveis e Entrada  
de Dados - Números**



# Operações Numéricas

---

Python oferece um suporte robusto para operações numéricas, permitindo trabalhar com diferentes tipos de números e realizar cálculos matemáticos de maneira simples e eficiente.

A linguagem suporta operações básicas como adição, subtração, multiplicação e divisão, além de funções matemáticas avançadas.



# Tipos Numéricos

---

Python lida com vários tipos de números:

## **Inteiros (int):**

Números inteiros, positivos ou negativos, sem partes decimais.

```
a = 10
```

```
b = -5
```

# Tipos Numéricos

---

Python lida com vários tipos de números:

## **Pontos Flutuantes (float):**

Números com casas decimais.

```
pi = 3.14
```

```
altura = 1.75
```

Python lida com vários tipos de números:

## Números Complexos (complex):

Números na forma  $a + bj$ , onde  $\underline{a}$  é a parte real e  $\underline{b}$  é a parte imaginária.

$$z = 2 + 3j$$

# Operações Aritméticas Básicas

Python suporta as operações aritméticas básicas:

## Adição (+):

```
soma = 10 + 5
```

```
print(soma) # Saída: 15
```

## Subtração (-):

```
diferenca = 10 - 5
```

```
print(diferenca) # Saída: 5
```

# Operações Aritméticas Básicas

Python suporta as operações aritméticas básicas:

## **Multiplicação (\*):**

```
produto = 10 * 5  
print(produto)  # Saída: 50
```

## **Divisão (/):**

```
quociente = 10 / 5  
print(quociente)  # Saída: 2.0
```

# Operações Aritméticas Básicas

Python suporta as operações aritméticas básicas:

**Divisão Inteira (//):** Retorna o quociente da divisão truncado para um inteiro.

```
divisao_inteira = 10 // 3  
print(divisao_inteira)  # Saída: 3
```

**Módulo (%):** Retorna o restante da divisão.

```
resto = 10 % 3  
print(resto)  # Saída: 1
```

# Operações Aritméticas Básicas

Python suporta as operações aritméticas básicas:

**Exponenciação (\*\*):** Eleva um número à potência de outro.

```
potencia = 2 ** 3
```

```
print(potencia) # Saída: 8
```

# Operações com Funções Matemáticas

Python fornece uma biblioteca padrão chamada `math` que inclui diversas funções matemáticas avançadas:

```
import math
```

```
# Raiz quadrada
```

```
raiz_quadrada = math.sqrt(16)
```

```
print(raiz_quadrada) # Saída: 4.0
```

```
# Valor absoluto
```

```
valor_absoluto = abs(-10)
```

```
print(valor_absoluto) # Saída: 10
```



# Operações com Funções Matemáticas

Python fornece uma biblioteca padrão chamada `math` que inclui diversas funções matemáticas avançadas:

```
import math
```

```
# Fatorial
```

```
fatorial = math.factorial(5)
```

```
print(fatorial)  # Saída: 120
```

```
# Seno, Cosseno e Tangente (em radianos)
```

```
seno = math.sin(math.pi / 2)
```

```
print(seno)  # Saída: 1.0
```

# Operadores de Atribuição

Python permite combinar operações aritméticas com atribuição usando operadores como +=, -=, \*=, /=, etc.

```
x = 10
```

```
x += 5 # Equivalente a x = x + 5
```

```
print(x) # Saída: 15
```

```
x *= 2 # Equivalente a x = x * 2
```

```
print(x) # Saída: 30
```

# Operações com Números Complexos

Python lida naturalmente com números complexos, permitindo operações como soma, multiplicação e outras.

```
z1 = 2 + 3j
```

```
z2 = 1 + 2j
```

```
soma_complexa = z1 + z2
```

```
print(soma_complexa) # Saída: (3+5j)
```

# Dúvidas?

---







Dúvidas, críticas e sugestões, entre em contato  
através do e-mail do professor:

[luiz.a.rodriques@cogna.com.br](mailto:luiz.a.rodriques@cogna.com.br)