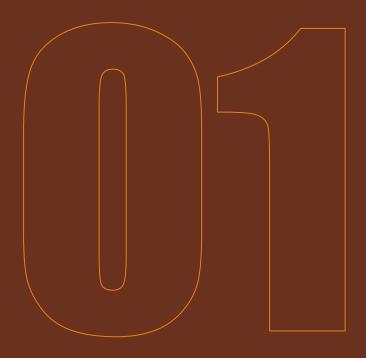


Sumário

- 1. Introdução ao Python: História pág. 1
- Preparando Seu Ambiente de Programação pág. 2
- 3. Primeiro Código: Printando pela Primeira Vez pág. 3
- 4. Variáveis: Guardando Informações pág. 4
- Tipos de Dados: Números, Texto e Lógicos pág.
- 6. Operadores: Fazendo Contas e Comparações pág. 6
- Manipulação de Strings: Textos na Prática pág.
 7
- Entrada de Dados: Interação com o Usuário pág. 8
- 9. Estruturas Condicionais: If, Elif, Else pág. 9
- 10. Laços de Repetição: While, For pág. 10
- 11. Manipulando Listas: Coleções de Dados pág.11
- 12. Dicionários: Dados com Chave e Valor pág. 12
- 13. Funções: Reaproveitando Códigos pág. 13
- 14. Conclusão: Agradecimentos pág. 14



Introdução ao Python: História

Introdução ao Python

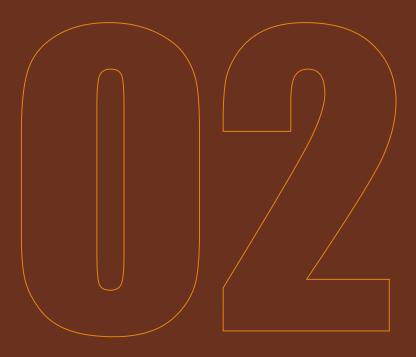
História

Python é uma linguagem de programação criada por Guido van Rossum em 1991. Popular por sua simplicidade e sintaxe clara, é gratuita, open-source e multiplataforma. Seu nome homenageia o grupo de humor Monty Python, refletindo um espírito acessível.

Versátil, Python é usada em áreas como desenvolvimento web, automação, análise de dados, inteligência artificial e educação. Empresas como Google, Netflix e Spotify a utilizam, e ela é muito escolhida por quem começa a programar ou trabalha com dados.

Se você quer aprender a programar, automatizar tarefas ou trabalhar com dados e IA, Python é uma excelente porta de entrada para a programação.





Preparando seu ambiente de programação

Preparando seu ambiente de programação

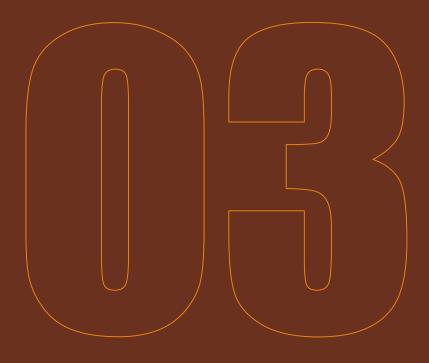
IDE(Ambiente de desenvolvimento Integrado)

Antes de começar a programar em Python, é importante configurar seu ambiente para escrever, executar e testar seus códigos de forma prática. A primeira etapa é instalar o Python em seu computador: acesse o site oficial python.org e baixe a versão mais recente compatível com o seu sistema operacional (Windows, macOS ou Linux). Durante a instalação no Windows, marque a opção "Add Python to PATH" para conseguir executar o Python de qualquer pasta pelo terminal.

Depois, você precisará de um editor de código ou IDE (Ambiente de Desenvolvimento Integrado). Para iniciantes, o VS Code é uma ótima escolha: leve, gratuito e com suporte para extensões que ajudam no aprendizado. Outra opção simples é o IDLE, que já vem instalado junto com o Python e permite escrever e rodar códigos imediatamente.

VSCode: https://code.visualstudio.com/download

Python: https://www.python.org/downloads/



Primeiro código: Printando pela primeira vez

Primeiro Código

Printando pela primeira vez

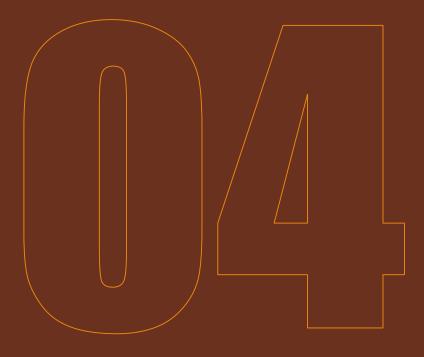
Em Python, a função print() é a porta de entrada para quem quer ver resultados aparecendo na tela. É com ela que você consegue exibir mensagens, valores de variáveis e até cálculos.

```
1 print("Olá, mundo!") # Imprime a mensagem "Olá, mundo!" na tela
```

Saída:

Olá, mundo!





Variáveis: Guardando Informações

Variáveis

Guardando informações

Variáveis são como "caixas" que armazenam valores para usar depois. Em Python, você não precisa declarar o tipo antes: o próprio Python entende o que você quer.

```
# Aqui estamos criando uma variável chamada 'nome' e atribuindo a ela o valor "Ana".
nome = "Ana"

# Nesta linha, criamos a variável 'idade' e guardamos nela o valor 25.
idade = 25

# Por fim, a variável 'salario' recebe o valor 2500.75.
salario = 2500.75
```





Tipos de Dados: Números, Texto e Lógicos

Tipos de dados

Números, textos e lógicos

Python trabalha principalmente com:

•int (inteiros): Usados para representar quantidades inteiras, como 1, -10 ou 2025.

```
# idade recebe o valor 30 (int: número inteiro)
idade = 30
```

•float (decimais):Usados quando precisamos de valores fracionários, como 3.14 ou -0.5.

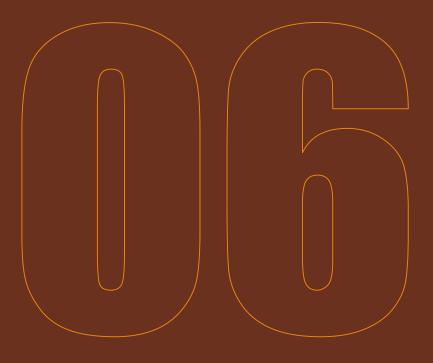
```
7 # salario recebe o valor 2500.50 (float: número de ponto flutuante)
8 salario = 2500.50
```

•str (textos):Usadas para armazenar cadeias de caracteres, como "Olá", "Python" ou "1234" (mesmo que contenha números, se estiver entre aspas, é string).

```
4 # nome recebe o valor "João" (str: string)
5 nome = "João"
```

•bool (verdadeiro ou falso):Permite representar condições lógicas. Só existem dois valores possíveis: True ou False.

```
10 # ativo recebe o valor True (bool: booleano)
11 ativo = True
```



Operadores: Fazendo Contas e Comparações

Operadores

Fazendo contas e comparações

Operadores permitem realizar operações matemáticas e lógicas:

•Matemáticos: +, -, *, /, //, %, **

```
a = 10
 4
 5
     b = 5
 6
     c = a + b # a + b = 15
     d = a - b # a - b = 5
 8
     e = a * b # a * b = 50
     f = a / b # a / b = 2.0
 9
       = a // b # a // b = 2
10
     h = a % b # a % b = 0
11
12
                     ** b = 100000
```

•Comparação: ==, !=, >, <, >=, <=

```
17
     a = 10
18
       = 5
19
               b # a == b = False
20
                 #
     d = a != b
                   a != b = True
21
                 #
                     < b = False
           < b
                   a
22
                   a > b = True
                 #
         a > b
23
         a <= b #
                   a <= b = False
24
                 #
                   a >= b = True
```

•Lógicos: and, or, not

```
29    a = True

30    b = False

31    c = a and b # a and b = False

32    d = a or b # a or b = True

33    e = not a # not a = False

34    f = not b # not b = True
```



Manipulação de Strings: Textos na Prática

Manipulação de Strings

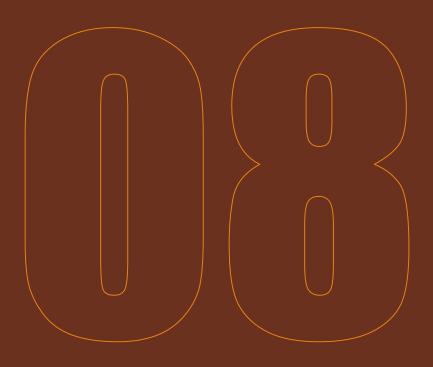
Textos na pratica

Strings permitem várias operações, como:

- Mudar para maiúsculas/minúsculas.
- Dividir textos.
- Substituir palavras.

```
1 mensagem = "bom dia, mundo!"
2 print(mensagem.title())  # Bom Dia, Mundo!
3 print(mensagem.upper())  # BOM DIA, MUNDO!
4 print(mensagem.replace("dia", "tarde")) # bom tarde, mundo!
```





Entrada de Dados: Interação com o Usuário

Entrada de dados

Interação com o usuário

Use input() para receber informações digitadas.

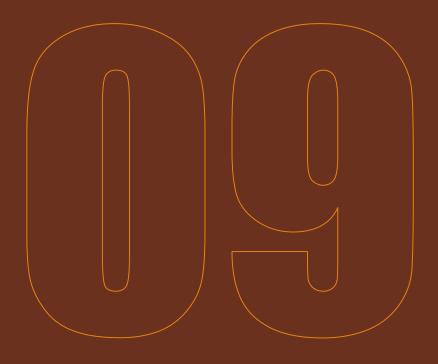
```
# Solicita ao usuário que digite seu nome e armazena o valor na variável 'nome'
nome = input("Digite seu nome: ")
# Exibe o valor armazenado na variável 'nome'
print(nome)

# Solicita ao usuário que digite sua idade e armazena o valor na variável 'idade'
idade = input("Digite sua idade: ")
# Exibe o valor armazenado na variável 'idade'
print(idade)

# Solicita ao usuário que digite seu salário e armazena o valor na variável 'salario'
salario = input("Digite seu salário: ")
# Exibe o valor armazenado na variável 'salario'
print(salario)
```

Saída:

```
Digite seu nome: Heitor
Heitor
Digite sua idade: 27
27
Digite seu salário: 1900
1900
```



Estruturas Condicionais: If, Elif, Else

Estruturas condicionais

If, Elif, Else

Use if, elif, else para que o programa execute ações diferentes conforme as condições.

```
idade = 16 # Define a variável 'idade' com o valor 16

if idade >= 18: # Verifica se a idade é maior ou igual a 18

print("Pode dirigir!") # Se for, exibe que pode dirigir

lif idade >= 16: # Se não, verifica se a idade é maior ou igual a 16

print("Pode fazer autoescola teórica.") # Se for, exibe que pode fazer autoescola teórica

relse:

print("Ainda não pode dirigir.") # Se nenhuma condição anterior for satisfeita, exibe que ainda não pode dirigir
```





Laços de Repetição: While, for

Laços de repetição

While, For

While: repete até a condição ser falsa.

```
# Inicializa a variável contador com o valor 1
contador = 1

# Enquanto o valor de contador for menor ou igual a 5, execute o bloco abaixo
while contador <= 5:
# Exibe o valor atual de contador na tela
print(f"Contando: {contador}")
# Incrementa o valor de contador em 1
contador += 1</pre>
```

Saída:

```
Contando: 1
Contando: 2
Contando: 3
Contando: 4
Contando: 5
```

For: percorre elementos de listas ou intervalos.

```
# Cria uma lista chamada frutas com três elementos
frutas = ["maçã", "banana", "uva"]

# Para cada elemento da lista frutas, execute o bloco abaixo
for fruta in frutas:
# Exibe o nome da fruta atual na tela
print(f"Fruta: {fruta}")
```

Saída:

Fruta: maçã

Fruta: banana

Fruta: uva





Manipulando listas: Coleções de dados

Manipulando listas

Coleções de dados

Listas guardam vários itens em uma única variável, podendo misturar tipos.

```
compras = ["leite", "pão", "café"] # Cria uma lista chamada 'compras' com três itens iniciais
compras.append("queijo") # Adiciona o item "queijo" ao final da lista
compras.remove("pão") # Remove o item "pão" da lista

for item in compras: # Percorre cada elemento da lista 'compras'
print(f"Item: {item}") # Exibe o item atual da lista
```

Saída:

```
Item: leite
Item: café
Item: queijo
```



Dicionários: Dados com Chave e Valor

Dicionários

Dados com chave e valor

Dicionários armazenam dados que se relacionam como "chave: valor".

```
# Cria um dicionário chamado 'usuario' com três pares chave-valor
usuario = {
    "nome": "Luana",  # A chave 'nome' armazena o valor 'Luana'
    "idade": 29,  # A chave 'idade' armazena o valor 29
    "cidade": "Fortaleza" # A chave 'cidade' armazena o valor 'Fortaleza'
}

# Acessa os valores das chaves 'nome' e 'cidade' no dicionário e imprime uma mensagem formatada
print(f"{usuario['nome']} mora em {usuario['cidade']}.")
```

Saída:

Luana mora em Fortaleza.





Funções: Reaproveitando Códigos

Funções

Reaproveitando códigos

Funções organizam o código e evitam repetição.

```
# Definição de uma função chamada 'saudacao' que recebe um parâmetro chamado 'nome'

def saudacao(nome):

# Esta linha imprime uma mensagem de saudação usando o valor recebido no parâmetro 'nome'

print(f"Olá, {nome}! Seja bem-vindo.")

# Aqui estamos chamando (executando) a função 'saudacao' e passando o argumento "Carlos"

saudacao("Carlos")
```

Saída:

Olá, Carlos! Seja bem-vindo.





Conclusão: Agradecimentos

Conclusão

Agradecimentos



Quero agradecer a você, leitor ou leitora, por dedicar seu tempo e interesse para aprender Python com este material. Espero que este eBook tenha sido útil para tornar seus primeiros passos na programação mais leves e claros.

A programação pode parecer desafiadora no começo, mas lembre-se: cada linha de código escrita é um aprendizado, e cada erro, uma oportunidade de melhorar. Obrigado por confiar neste conteúdo para fazer parte da sua jornada!

Desejo muito sucesso nos seus estudos e projetos. Continue explorando, praticando e acreditando no seu potencial!

-Igor Luna

GitHuB: https://github.com/igorluna06

Linkedin: https://www.linkedin.com/in/igor-luna-87b813303/

