

PYTHON:

AMOR À PRIMEIRA LINHA



IGOR LUNA

Sumário

1. Introdução ao Python: História – pág. 1
 2. Preparando Seu Ambiente de Programação – pág. 2
 3. Primeiro Código: Printando pela Primeira Vez – pág. 3
 4. Variáveis: Guardando Informações – pág. 4
 5. Tipos de Dados: Números, Texto e Lógicos – pág. 5
 6. Operadores: Fazendo Contas e Comparações – pág. 6
 7. Manipulação de Strings: Textos na Prática – pág. 7
 8. 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
-

01

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.



02

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/>

03

**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!
```



04

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.

```
1  # Aqui estamos criando uma variável chamada 'nome' e atribuindo a ela o valor "Ana".
2  nome = "Ana"
3
4  # Nesta linha, criamos a variável 'idade' e guardamos nela o valor 25.
5  idade = 25
6
7  # Por fim, a variável 'salario' recebe o valor 2500.75.
8  salario = 2500.75
```



05

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
```

06

Operadores: Fazendo Contas e Comparações

Operadores

Fazendo contas e comparações

Operadores permitem realizar operações matemáticas e lógicas:

- Matemáticos: +, -, *, /, //, %, **

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

- Comparação: ==, !=, >, <, >=, <=

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

07

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!
```



08

Entrada de Dados: Interação com o Usuário

Entrada de dados

Interação com o usuário

Use `input()` para receber informações digitadas.

```
1  # Solicita ao usuário que digite seu nome e armazena o valor na variável 'nome'
2  nome = input("Digite seu nome: ")
3  # Exibe o valor armazenado na variável 'nome'
4  print(nome)
5
6  # Solicita ao usuário que digite sua idade e armazena o valor na variável 'idade'
7  idade = input("Digite sua idade: ")
8  # Exibe o valor armazenado na variável 'idade'
9  print(idade)
10
11 # Solicita ao usuário que digite seu salário e armazena o valor na variável 'salario'
12 salario = input("Digite seu salário: ")
13 # Exibe o valor armazenado na variável 'salario'
14 print(salario)
```

Saída:

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

09

Estruturas Condicionais: If, Elif, Else

Estruturas condicionais

If, Elif, Else

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

```
1  idade = 16 # Define a variável 'idade' com o valor 16
2
3  if idade >= 18: # Verifica se a idade é maior ou igual a 18
4      print("Pode dirigir!") # Se for, exibe que pode dirigir
5  elif idade >= 16: # Se não, verifica se a idade é maior ou igual a 16
6      print("Pode fazer autoescola teórica.") # Se for, exibe que pode fazer autoescola teórica
7  else:
8      print("Ainda não pode dirigir.") # Se nenhuma condição anterior for satisfeita, exibe que ainda não pode dirigir
```



10

**Laços de
Repetição:
While, for**

Laços de repetição

While, For

While: repete até a condição ser falsa.

```
1  # Inicializa a variável contador com o valor 1
2  contador = 1
3
4  # Enquanto o valor de contador for menor ou igual a 5, execute o bloco abaixo
5  while contador <= 5:
6      # Exibe o valor atual de contador na tela
7      print(f"Contando: {contador}")
8      # Incrementa o valor de contador em 1
9      contador += 1
```

Saída:

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

For: percorre elementos de listas ou intervalos.

```
11 # Cria uma lista chamada frutas com três elementos
12 frutas = ["maçã", "banana", "uva"]
13
14 # Para cada elemento da lista frutas, execute o bloco abaixo
15 for fruta in frutas:
16     # Exibe o nome da fruta atual na tela
17     print(f"Fruta: {fruta}")
```

Saída:

```
Fruta: maçã
Fruta: banana
Fruta: uva
```



11

Manipulando listas: Coleções de dados

Manipulando listas

Coleções de dados

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

```
1  compras = ["leite", "pão", "café"] # Cria uma lista chamada 'compras' com três itens iniciais
2  compras.append("queijo") # Adiciona o item "queijo" ao final da lista
3  compras.remove("pão")    # Remove o item "pão" da lista
4
5  for item in compras: # Percorre cada elemento da lista 'compras'
6      print(f"Item: {item}") # Exibe o item atual da lista
```

Saída:

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

12

Dicionários: Dados com Chave e Valor

Dicionários

Dados com chave e valor

Dicionários armazenam dados que se relacionam como “chave: valor”.

```
1  # Cria um dicionário chamado 'usuario' com três pares chave-valor
2  usuario = {
3      "nome": "Luana",      # A chave 'nome' armazena o valor 'Luana'
4      "idade": 29,          # A chave 'idade' armazena o valor 29
5      "cidade": "Fortaleza" # A chave 'cidade' armazena o valor 'Fortaleza'
6  }
7
8  # Acessa os valores das chaves 'nome' e 'cidade' no dicionário e imprime uma mensagem formatada
9  print(f"{usuario['nome']} mora em {usuario['cidade']}.")
```

Saída:

```
Luana mora em Fortaleza.
```



13

Funções: Reaproveitando Códigos

Funções

Reaproveitando códigos

Funções organizam o código e evitam repetição.

```
1  # Definição de uma função chamada 'saudacao' que recebe um parâmetro chamado 'nome'
2  def saudacao(nome):
3      # Esta linha imprime uma mensagem de saudação usando o valor recebido no parâmetro 'nome'
4      print(f"Olá, {nome}! Seja bem-vindo.")
5
6  # Aqui estamos chamando (executando) a função 'saudacao' e passando o argumento "Carlos"
7  saudacao("Carlos")
```

Saída:

```
Olá, Carlos! Seja bem-vindo.
```



14

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/>

