


Python na Prática: *Fundamentos* *Essenciais para Iniciantes*



Prof. Me. Diego H. Negretto



Python na **Prática:** ***Fundamentos*** ***Essenciais*** ***para Iniciantes***

Objetivos:

Capacitar os participantes a entender e aplicar conceitos essenciais da linguagem de programação Python.

Público Alvo:

Estudantes e profissionais interessados em programação, sem necessidade de experiência prévia.

Carga horária:

12 horas - semanal, aos sábados, das 08h15 às 12h15

Início e Término:

30 de agosto a 13 de setembro de 2025




Python na **Prática:** ***Fundamentos*** ***Essenciais*** ***para Iniciantes***

Conteúdo programático

Encontro 01: Fundamentos do Python (4h)

- Visão geral do Python: História, aplicações e por que aprender Python;
- Configuração do ambiente: Instalação do Python, VS Code, Jupyter Notebook;
- Estrutura de um programa em Python;
- Variáveis, tipos de dados e operadores;
- Entrada e saída de dados (input() e print());
- Introdução a estruturas condicionais (if, elif, else);
- Estruturas de repetição: for e while;
- Mão na massa: Exercícios práticos.




Python na **Prática:** ***Fundamentos*** ***Essenciais*** ***para Iniciantes***

Conteúdo programático

Encontro 02: Funções e Manipulação de Arquivos (4h)

- Listas e tuplas: Manipulação de coleções de dados;
- Introdução a dicionários e conjuntos;
- Funções: Definição, parâmetros e retorno;
- Manipulação de arquivos (open, leitura e escrita);
- Introdução ao tratamento de exceções (*try*, *except*, *finally*);
- Boas práticas na organização do código;
- Mão na massa: Exercícios práticos.



Python na **Prática:** ***Fundamentos*** ***Essenciais*** ***para Iniciantes***

Conteúdo programático

Encontro 03: Introdução a Bibliotecas e Mini-projeto (4h)

- Uso de bibliotecas padrão do Python (*os*, *math*, *datetime*);
- Introdução ao *pandas* e *matplotlib* para manipulação de dados e visualização;
- Mão na massa: Desenvolvimento de um mini-projeto (Aplicação simples para análise de dados);
- Encerramento e discussão de próximas etapas para aprofundamento em Python.



Introdução ao Python

Strings

Strings são sequências de caracteres usadas para representar textos em Python. Elas podem ser criadas com **aspas simples ou duplas** e permitem diversas operações, como **concatenação, fatiamento e formatação**. Além disso, oferecem **métodos prontos para manipulação**, como alterar maiúsculas e minúsculas, substituir trechos ou verificar conteúdos específicos.



Introdução ao Python

Strings

```
strings.py x
Exemplos > Encontro02 > strings.py > ...
1  # Strings em Python
2  # -----
3
4  texto = "Curso Python Na Prática"
5
6  # Mostra o caractere na posição 0
7  print(texto[0])
8  # Mostra os caracteres do índice 5 até 11
9  print(texto[5:12])
10 # Mostra do índice 5 até o fim
11 print(texto[5:])
12 # Mostra do início até o índice 4
13 print(texto[:5])
14
15 # Mostra o tamanho da string (número de caracteres, incluindo espaços).
16 print(len(texto))
17
18 # Conta quantas vezes aparece o caractere "A" (maiúsculo).
19 print(texto.count("A"))
20 # Conta quantas vezes aparece o caractere "a" (minúsculo).
21 print(texto.count("a"))
22 # Conta quantos "P" existem entre os índices 5 e 11.
23 print(texto.count("P", 5, 12))
24
25 # Mostra a posição inicial da palavra "Curso"
26 print(texto.find("Curso"))
27 # Mostra a posição inicial da palavra "Python"
28 print(texto.find("Python"))
29
30 # Converte toda a string para maiúsculas.
31 print(texto.upper())
32 # Converte toda a string para minúsculas.
33 print(texto.lower())
34
35 # Deixa só a primeira letra da string em maiúscula.
36 print(texto.capitalize())
37 # Deixa a primeira letra de cada palavra em maiúscula.
38 print(texto.title())
```

Introdução ao Python

Strings

```
strings.py X
Exemplos > Encontro02 > strings.py > ...
40 # Divide a string em uma lista de palavras, usando os espaços como separador.
41 print(texto.split())
42 # Armazena essa lista de palavras na variável lista_de_palavras.
43 lista_de_palavras = texto.split()
44 # Junta as palavras da lista sem espaços
45 print(''.join(lista_de_palavras))
46
47 texto = "    CURSO PYTHON    "
48 print(texto)
49 # Remove os espaços do início e do fim.
50 print(texto.strip())
51 # Remove apenas os espaços da direita (fim da string).
52 print(texto.rstrip())
53 # Remove apenas os espaços da esquerda (início da string).
54 print(texto.lstrip())
55
56 texto = "Eu gosto de Java"
57 print (texto)
58 # Substitui "Java" por "Python"
59 novo_texto = texto.replace("Java", "Python")
60 print(novo_texto)
```




Introdução ao Python

Listas e Tuplas

Listas são **coleções mutáveis** que permitem armazenar e alterar **vários elementos em uma única variável**. Já as **tuplas** funcionam de forma semelhante, mas **são imutáveis**, sendo utilizadas quando os valores não devem ser modificados.



Introdução ao Python

Listas e Tuplas

```
listas.py X
Exemplos > Encontro02 > listas.py > ...
1 # Listas em Python
2 # -----
3
4 # Criação de uma lista vazia
5 carros = []
6 # Adicionando os elementos Ka, Fusca e Fiesta na lista
7 carros.append("Ka")
8 carros.append("Fusca")
9 carros.append("Fiesta")
10 print(carros)
11 # Adicionando o elemento 10 (inteiro) na lista
12 carros.append(10)
13 print(carros)
14
15 # Criação de uma lista com valores
16 nomes = ["João", "Maria", "José"]
17 print(nomes)
18 # Adicionando valores no final da lista.
19 nomes.append("Joaquim")
20 print(nomes)
21 # Imprimindo o primeiro elemento da lista.
22 print(nomes[0])
23
24 # Inserindo um elemento novo na lista em uma posição.
25 nomes.insert(1, "Joana")
26 print(nomes)
27
28 # Removendo o ultimo elemento da lista.
29 nomes.pop()
30 print(nomes)
31 # Removendo um elemento em uma posição específica da lista.
32 del nomes[1]
33 print(nomes)
34 # Removento um elemento da lista pelo valor.
35 nomes.remove("Maria")
36 print(nomes)
```

Introdução ao Python

Listas e Tuplas



listas.py X

Exemplos > Encontro02 > listas.py > ...

```
38 # Contando quantos elementos "João" existem na lista
39 print(nomes.count("João"))
40
41 # Invertendo a ordem dos elementos da lista
42 nomes.reverse()
43 print(nomes)
44
45 nomes.append("Ana")
46 print(nomes)
47 # Ordenando os elementos da lista
48 nomes.sort()
49 print(nomes)
50
51 # Tuplas em Python (imutáveis)
52 # -----
53 # Criando uma tupla
54 cores = ("vermelho", "azul", "verde")
55 print(cores[1])
```



Introdução ao Python

Dicionários e Conjuntos

Dicionários são estruturas que guardam pares de chave e valor, possibilitando acesso rápido e organizado aos dados. **Conjuntos**, por sua vez, são coleções de elementos únicos e sem ordem definida, úteis para eliminar duplicações.

Introdução ao Python

Dicionários e Conjuntos

```
dicionarios.py X
Exemplos > Encontro02 > dicionarios.py > ...
1  # Dicionários em Python
2  # -----
3
4  # Criando um dicionário com alguns elementos (nome, idade e profissão)
5  meu_dicionario = {"nome": "Diego", "idade": 35, "profissao": "Professor"}
6  print(meu_dicionario)
7
8  # Buscando um elemento a partir de uma chave.
9  print(meu_dicionario["nome"])
10 # Buscando um elemento a partir de uma chave com o metodo get.
11 print(meu_dicionario.get("profissao"))
12 # Removendo um elemento do dicionario a partir de uma chave.
13 print(meu_dicionario.pop("idade"))
14 print(meu_dicionario)
15
16 # Imprimindo somente as chaves do dicionario
17 print(meu_dicionario.keys())
18 # Imprimindo somente os valores do dicionario
19 print(meu_dicionario.values())
20 # Apagando todo o conteúdo do dicionario
21 meu_dicionario.clear()
22 print(meu_dicionario)
```

Introdução ao Python

Dicionários e Conjuntos

```
dicionarios.py X
Exemplos > Encontro02 > dicionarios.py > ...

24 # Exemplo de dicionarios mais complexos
25 # Criação de um dicionario com outros dicionarios dentro
26 pessoa = {
27     "nome": "Diego",
28     "idade": 35,
29     "profissao": "Professor",
30     "interesses": ["Python", "Literatura", "Games"],
31     "pet": {
32         "nome": "Mei",
33         "idade": 8,
34         "peso": "7kg"
35     }
36 }
37
38 print(pessoa)
39 print(pessoa.get("nome"))
40 print(pessoa["nome"])
41 print(pessoa.get("interesses"))
42 print(pessoa.get("interesses")[0])
43 print(pessoa["interesses"][0])
44 print(pessoa.get("pet").get("nome"))
45 print(pessoa["pet"]["nome"])
46
47 # Adicionando dado no dicionario
48 pessoa["ano_nascimento"] = 1989
49 print(pessoa)
50 # Adicionando uma lista no dicionario com chave cores_favoritas
51 pessoa["cores_favoritas"] = ["Azul", "Preto", "Verde"]
52 print(pessoa)
53 # Adicionando um dicionario dentro do dicionario com chave mãe.
54 pessoa["mae"] = {
55     "nome": "Maria",
56     "idade": 60
57 }
58 print(pessoa)
```

Introdução ao Python

Dicionários e Conjuntos

```
conjuntos.py X
Exemplos > Encontro02 > conjuntos.py > ...
1 # Conjuntos em Python
2 # -----
3
4 # Criando conjuntos
5 A = {1, 2, 3, 4}
6 B = {3, 4, 5, 6}
7 C = set() # Conjunto vazio
8
9 # Adicionar e remover elementos
10 A.add(7) # Adiciona um elemento
11 A.remove(2) # Remove o elemento 2 (erro se não existir)
12 A.discard(10) # Remove sem erro, mesmo que não exista
13 print("A:", A)
14
15 # União
16 print("União:", A.union(B)) # {1, 3, 4, 5, 6, 7}
17 print("União (|):", A | B) # forma reduzida
18
19 # Interseção
20 print("Interseção:", A.intersection(B)) # {3, 4}
21 print("Interseção (&):", A & B)
22
23 # Diferença
24 print("Diferença:", A.difference(B)) # Elementos de A que não estão em B
25 print("Diferença (-):", A - B)
26
27 # Diferença simétrica
28 print("Dif. Simétrica:", A.symmetric_difference(B)) # {1, 5, 6, 7}
29 print("Dif. Simétrica (^):", A ^ B)
30
31 # Verificação de subconjunto e superconjunto
32 print("A é subconjunto de B?", A.issubset(B))
33 print("A é superconjunto de B?", A.issuperset(B))
34
35 # Testar se não há elementos em comum
36 print("A e B são disjuntos?", A.isdisjoint(B))
```



Introdução ao Python

Funções

Funções são blocos de código reutilizáveis que executam uma tarefa específica. Elas podem receber parâmetros, que são valores usados durante sua execução, e podem retornar um resultado após o processamento.



Introdução ao Python

Funções

```
funcoes.py X
Exemplos > Encontro02 > funcoes.py > ...
1  # Funções em Python
2  # -----
3
4  # Definindo uma função chamada somar que recebe dois parâmetros
5  def somar(a, b):
6      resultado = a + b
7      return resultado # a função retorna um valor para quem a chamou.
8
9  # Fazendo a chamada da função somar passando valores.
10 meu_resultado = somar(2,2)
11 print(meu_resultado)
```

Introdução ao Python

Funções

```
funcoes.py X
Exemplos > Encontro02 > funcoes.py > ...

13 # Exemplos mais complexos
14 # Definindo uma função chamada envia_email que recebe dois parâmetros.
15 def envia_email(nome, email):
16     nome_dest = nome
17     email_dest = email
18     return (f"Email enviado para {nome_dest} - {email_dest}")
19
20 # Criação de uma lista de pessoas em que cada elemento é um dicionário com chaves email e valor.
21 pessoas = [
22     {
23         "nome" : "João",
24         "email" : "joao@email.com"
25     },
26     {
27         "nome" : "Maria",
28         "email" : "maria@email.com"
29     },
30     {
31         "nome" : "José",
32         "email" : "jose@email.com"
33     }
34 ]
35
36 # Percorrendo todos os itens da lista e chamando a função envia_email para todos.
37 for pessoa in pessoas:
38     email_enviado = envia_email(pessoa["nome"], pessoa["email"])
39     print(email_enviado)
40
41 # Imprimindo todos os índices e elementos da lista
42 for indice, pessoa in enumerate(pessoas):
43     email_enviado = envia_email(pessoa["nome"], pessoa["email"])
44     print(f"{indice} - {email_enviado}")
```



Introdução ao Python

Manipulação de Arquivos

A manipulação de arquivos em Python é feita principalmente com a **função `open()`**, que permite abrir documentos em diferentes modos. A partir dela, é possível realizar **operações de leitura** para obter conteúdo **ou escrita** para criar e modificar informações em arquivos.



Introdução ao Python

Manipulação de Arquivos

Modos comuns de abertura:

"r" → leitura (erro se o arquivo não existir)

"w" → escrita (cria ou sobrescreve o arquivo)

"a" → acrescentar conteúdo no final do arquivo

"b" → modo binário (para imagens, PDF, etc.)

"r+" → leitura e escrita no mesmo arquivo

Métodos úteis:

.read() → lê todo o conteúdo como string.

.readline() → lê apenas uma linha por vez.

.readlines() → lê todas as linhas e retorna uma lista.

.write(texto) → escreve uma string no arquivo.

.writelines(lista) → escreve várias linhas de uma vez.

Introdução ao Python

Manipulação de Arquivos

arquivos.py X

Exemplos > Encontro02 > arquivos.py > ...

```
1  # Manipulação de Arquivos em Python
2  # -----
3
4  # Escrita (sobrescreve se já existir)
5  with open("dados.txt", "w") as arquivo:
6      arquivo.write("Primeira linha\n")
7      arquivo.write("Segunda linha\n")
8
9  # Leitura de todo o conteúdo
10 with open("dados.txt", "r") as arquivo:
11     conteudo = arquivo.read()
12     print("Conteúdo do arquivo:")
13     print(conteudo)
14
15 # Leitura linha a linha
16 with open("dados.txt", "r") as arquivo:
17     for linha in arquivo:
18         print("Linha:", linha.strip())
19
20 # Acrescentar conteúdo sem apagar o existente
21 with open("dados.txt", "a") as arquivo:
22     arquivo.write("Nova linha adicionada\n")
```



Introdução ao Python

Tratamento de Exceções

O tratamento de exceções é usado para lidar com erros de forma controlada. O bloco `try` executa o código que pode gerar uma falha, o `except` trata o erro caso ele ocorra e o `finally` garante a execução de um trecho de código, independentemente de ter ocorrido erro ou não.



Introdução ao Python

Tratamento de Exceções

Exceção	Quando ocorre
<i>ZeroDivisionError</i>	Divisão por zero (1 / 0)
<i>TypeError</i>	Operação com tipos incompatíveis ("abc" + 1)
<i>ValueError</i>	Valor inválido para uma operação (int("abc"))
<i>IndexError</i>	Índice fora do intervalo de uma lista (lista[10])
<i>KeyError</i>	Acesso a chave inexistente em um dicionário (d['chave'])
<i>AttributeError</i>	Atributo não existe em um objeto (obj.alguma_coisa)
<i>NameError</i>	Variável não definida
<i>ImportError</i>	Erro ao importar um módulo
<i>FileNotFoundError</i>	Arquivo não encontrado
<i>IOError</i>	Erros de entrada/saída (geralmente leitura/gravação de arquivos)
<i>StopIteration</i>	Fim de um iterador (por exemplo, em next())
<i>RuntimeError</i>	Erro de execução genérico

Introdução ao Python

Tratamento de Exceções

```
tratamento_excecoes.py X
Exemplos > Encontro02 > tratamento_excecoes.py > ...
1  # Tratamento de Exceções em Python
2  # -----
3
4  # Exemplo de Valor inválido para uma operação (int("abc"))
5  try:
6      numero = int("abc")  # Vai gerar erro
7  except ValueError:
8      print("Erro: valor inválido!")
9  finally:
10     print("Execução finalizada.")
11
12 # Exemplo de uso com múltiplas exceções
13 try:
14     x = int("abc")
15     y = 10 / 1
16 except ValueError:
17     print("Valor inválido")
18 except ZeroDivisionError:
19     print("Divisão por zero")
20 except Exception as e:
21     print(f"Outro erro: {e}")
22 finally:
23     print("Comando executado sempre!")
```


Introdução ao Python

Mão na Massa:
Exercícios Práticos

