

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



Prof. Me. Diego H. Negretto

## Quem vos fala ...



Prof. Me. Diego H. Negretto  
*diegonegretto@fho.edu.br*

- **Mestre em Ciência da Computação** pela UNESP (Rio Claro)
  - Graduado em Sistemas de Informação pela FHO | Uniararas.
- **Professor** nos cursos de Sistemas de Informação e Engenharia da Computação (FHO | Uniararas) desde 2016.
- **Áreas de Atuação / Interesse:**  
*Machine Learning / Data Science / Python*



# **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





# Introdução ao Python

O que é o Python?

**Python** é uma linguagem de programação amplamente usada, **interpretada, orientada a objeto** e de **alto nível com semântica dinâmica**, usada para programação de uso geral.

O nome da linguagem de programação Python vem de uma antiga série de comédia da BBC chamada **Monty Python's Flying Circus**.





# Introdução ao Python

Quem criou a linguagem Python?

Python foi criada por **Guido van Rossum**, nascido em 1956 em Haarlem, na Holanda.

A linguagem se espalhou pelo mundo e é resultado do trabalho contínuo de milhares de programadores, testadores, usuários e entusiastas.



*Guido van Rossum*

# Introdução ao Python

Por que Python?



**Python** é:

fácil de aprender,  
fácil de ensinar,  
fácil de usar,  
fácil de entender,  
fácil de obter.









# Introdução ao Python

## Por que Python?

Tiobe Index - Agosto 2025

<https://www.tiobe.com/tiobe-index/>



Aug 2025	Aug 2024	Change	Programming Language	Ratings	Change
1	1		 Python	26.14%	+8.10%
2	2		 C++	9.18%	-0.86%
3	3		 C	9.03%	-0.15%
4	4		 Java	8.59%	-0.56%
5	5		 C#	5.52%	-0.87%
6	6		 JavaScript	3.15%	-0.76%
7	8	▲	 Visual Basic	2.33%	+0.15%
8	9	▲	 Go	2.11%	+0.08%
9	25	▲	 Perl	2.08%	+1.17%
10	12	▲	 Delphi/Object Pascal	1.82%	+0.19%

# Introdução ao Python

## Como baixar, instalar e configurar o Python

Os usuários de Linux provavelmente já têm o Python instalado (Python é usada intensivamente por muitos componentes de SO Linux).

Os usuário do macOS provavelmente já têm uma versão do Python 2 pré-instalada no seu computador. Porém, precisarão instalar a versão Python 3.

Os usuário de Windows precisarão instalar o Python 3.

<https://www.python.org/downloads/>



# Introdução ao Python

Como baixar e configurar

Os usuários de Linux provavelmente já têm o Python

instalado. Para usuários de Windows, a instalação é feita por muitos

usuários têm uma versão do Python no seu computador. Python 3.

Instalar o Python

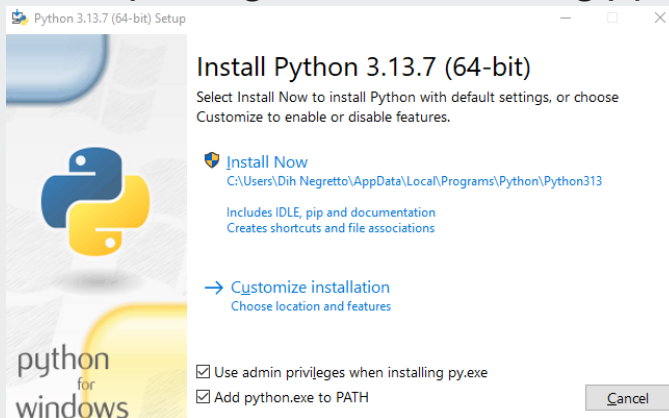
[Downloads/](#)

Jobs Community  
GO Socialize  
News Events

## IMPORTANTE!!!

No momento da instalação, selecione as opções:

*“Use admin privileges when installing py.exe”*



[Download Python 3.13.7](#)

Looking for Python with a different OS? Python for [Windows](#),  
[Linux/Unix](#), [macOS](#), [Android](#), [other](#)

Want to help test development versions of Python 3.14? [Pre-releases](#),  
[Docker images](#)



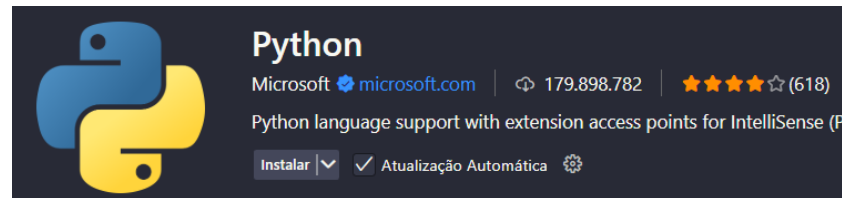
# Introdução ao Python

Como baixar, instalar e configurar o Python

Nesse curso iremos utilizar o **VsCode** com a extensão **Python**.

Note que ao instalar essa extensão, outras 3 são instaladas automaticamente:

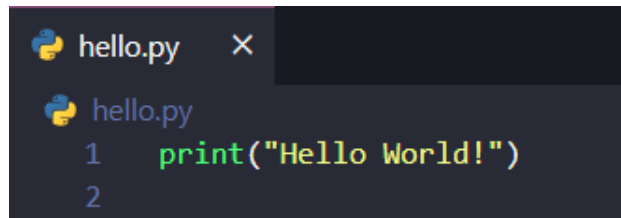
- *Pylance*
- *Python Debugger*
- *Python Environments*



# Introdução ao Python

## Hello World! em Python

Os arquivos em Python possuem a extensão `.py`



```
hello.py x
hello.py
1 print("Hello World!")
2
```





# Introdução ao Python

## A função `print()`

O nome da função (**`print()`** neste caso), juntamente com os parênteses e os argumento(s), formam a **invocação da função**.

A função **`print()`** é capaz de operar com praticamente todos os tipos de dados oferecidos pelo Python (strings, números, caracteres, valores lógicos e objetos).

```
hello.py
1  print("Hello World!")
2  print(2025)
3  print(True)
```





# Introdução ao Python

## A função *print()*

```
print.py
1  # Isso é um comentário
2  # -----
3  # Exemplos de utilização da função print()
4
5  print("Hello World!")
6
7  # \n é o símbolo especial para caractere de nova linha
8  print("Curso de Python!\nBem-vindos!")
9
10 # Escrevendo "Python" entre aspas
11 print("Curso de \"Python\"!")
12
13 # Usando vários argumentos (a função print() coloca um espaço entre
14 # os argumentos gerados por sua própria iniciativa)
15 print("Curso","de","Python!")
16
17 # O argumento end=" " faz com que a função print() coloque um espaço
18 # em branco ao final da linha ao invés de um \n (implicito)
19 print("Curso de Python.", end=" ")
20 print("FH0 Qualifica")
21
22 # O argumento sep="_" coloca um _ após cada palavra.
23 print("Curso","de","Python!",sep="_")
24
25 # Misturando os argumentos sep e end em uma mesma chamada.
26 print("Curso","de","Python", sep="_", end="!")
27 print("FH0", "Qualifica",sep="-",end="*\n")
28
29 # Repetindo a frase "Curso de Python!" cinco vezes
30 print("Curso de Python!\n" * 5)
31
32 # Python como calculadora
33 print(2+2)
34
```



# Introdução ao Python

A função *print()*

Qual a saída do seguinte comando  
*print()*?

```
print(("+" + "-"*10 + "+\n" + ("|" + " "*10 + "|\n")*5 + "+" + "-"*10 + "+\n"))
```



# Introdução ao Python

## Variáveis

Uma **variável** é um local nomeado reservado para armazenar valores na memória.

Uma variável é criada ou **inicializada automaticamente** quando você atribui um valor a ela pela primeira vez.

Cada variável deve ter um nome exclusivo – um **identificador**.

- uma sequência não vazia de caracteres;
- deve começar com o sublinhado (`_`) ou uma letra;
- não pode ser uma palavra-chave do Python.

Python é case-sensitive!

**Python é uma linguagem de tipagem dinâmica, o que significa que você não precisa declarar variáveis nela.**



# Introdução ao Python

## Variáveis - Tipagem Dinâmica

```
tipagemDinamica.py > ...
1  # Tipagem Dinâmica
2  # -----
3
4  # Declarando uma variável chama x com o valor 10
5  x = 10
6
7  # A função type() mostra o tipo de dado de uma variável
8  # Nesse momento, x é do tipo inteiro e seu conteúdo é 10
9  print(type(x))
10 print(x)
11
12 # Atribuindo a string "Maria" para a variável x
13 x = "Maria"
14
15 # Agora x é do tipo string e seu conteúdo é Maria
16 print(type(x))
17 print(x)
18
```



# Introdução ao Python

## Variáveis - Concatenação

```
variaveis.py > ...
1  # Declaração de uma variável do tipo string
2  nome_completo = "Anakin Skywalker"
3
4  # Declaração de uma variável do tipo inteiro
5  idade = 22
6
7  # Escrevendo o conteúdo da variável nome_completo concatenando
8  # com uma string.
9  print("Nome Completo:", nome_completo)
10 print("Nome Completo:" + nome_completo)
11
12 # A função a seguir causa erro, pois não é possível concatenar
13 # string e inteiro usando o +
14 # print("Idade:" + idade)
15 # Uma alternativa seria usar a ,
16 print("Idade:", idade)
17
18 # Usando f-strings para concatenação
19 print(f"{nome_completo} tem {idade} anos.")
20
```



# Introdução ao Python

## Operadores básicos

Operador	Função
+	Soma
-	Subtração
*	Multiplicação
/	Divisão
//	Divisão inteira
%	Restante (Módulo)
**	Exponenciação
==	Igualdade
!=	desigualdade
> , >= , < , <=	Maior que, Maior ou igual, Menor que, Menor ou igual



# Introdução ao Python

## Operadores básicos

```
operadores.py
1  # Operadores básicos
2  # -----
3
4  # Soma - Resultado: 4
5  print(2 + 2)
6
7  # Subtração - Resultado 2
8  print(5 - 3)
9
10 # Multiplicação - Resultado 10
11 print(5 * 2)
12
13 # Divisão - Resultado 3.0
14 print(6 / 2)
15
16 # Divisão Inteira - Resultado 2
17 print(5 // 2)
18
19 # Restante (Módulo) - Resultado 1
20 print(7 % 3)
21
22 # Exponenciação - Resultado 9
23 print(3 ** 2)
24
```



# Introdução ao Python

## A função *input()*

A função *input()* é capaz de ler os dados inseridos pelo usuário e retornar os mesmos dados para o programa em execução.

**Importante:** a função *input()* sempre retorna uma string!

```
input.py > ...  
1  # A função input()  
2  # -----  
3  
4  nome = input()  
5  print("Olá,", nome)  
6
```





# Introdução ao Python

## A função `input()`

```
input.py > ...
1  # A função input()
2  # -----
3
4  # input() com um argumento
5  nome = input("Qual seu nome?\n")
6  print("Olá,", nome)
7
8  # Conversão de tipo float
9  numero = input("Digite um número: ")
10 # print(numero ** 2) # Essa linha vai dar erro
11 print(type(numero))
12
13 numero = float(numero)
14 print(numero ** 2)
15 print(type(numero))
16
17 # Conversão de tipo int
18 numero_inteiro = int(input("Digite um número: "))
19 print(type(numero_inteiro))
20
21 # Conversão de tipo string
22 numero = 10
23 print(type(numero))
24 numero = str(numero)
25 print(type(numero))
```



# Introdução ao Python

## Tomada de decisões

Para tomar essas decisões, o Python oferece uma instrução especial que chamamos de **instrução condicional** (ou declaração condicional).

```
condicionais.py > ...
1  # Condicionais
2  # -----
3  condicao_verdadeira_ou_falsa = True
4  if condicao_verdadeira_ou_falsa:
5      print("Entrou no if\n")
6      #instrução_2
7      #instrução_3
8  else:
9      print("Não entrou no if\n")
10     #instrução_5
11
12  print("Instrução fora da condicional!\n")
```

# Introdução Python

Tomada

Para tomar essas decisões, o Python oferece uma instrução condicional (if/else).  
de instrução  
al).

## IMPORTANTE!!!

No Python a indentação é obrigatória!!!

```
= True  
sa:
```

```
")
```

```
12 print( "Instrução fora da condicional!\n")  
13
```



# Introdução ao Python

## Tomada de decisões

```
condicionais.py > ...
1  # Comandos if-else aninhados
2  # -----
3  x = 2
4  y = 5
5
6  if x > y:
7      if x % 2 == 0:
8          print("x é maior que y e é par!")
9      else:
10         print("x é maior que y e é ímpar!")
11 else:
12     if y % 2 == 0:
13         print("y é maior que x e é par!")
14     else:
15         print("y é maior que x e é ímpar!")
```



# Introdução ao Python

Tomada de decisões

```
condicionais.py > ...  
1  # Comando elif  
2  # -----  
3  media_final = 3.8  
4  
5  if media_final >= 5:  
6      print("Aprovado!")  
7  elif media_final >= 3:  
8      print("RE!")  
9  else:  
10     print("Reprovado!")
```

# Introdução ao Python

## Loopings (Estruturas de Repetição)

Para realizar loopings, o Python oferece duas instruções *while* e *for*.





# Introdução ao Python

## Loopings (Estruturas de Repetição)

```
LoopingWhile.py > ...
1  # Looping While
2  # -----
3
4  numero = int(input("Informe um número inteiro ou -1 para parar: "))
5
6  while numero != -1:
7      if numero%2 == 0:
8          print("Número informado é par!\n")
9      else:
10         print("Número informado é ímpar!\n")
11
12         numero = int(input("Informe um número inteiro ou -1 para parar: "))
13
14  print("Até logo!")
```



# Introdução ao Python

## Loopings (Estruturas de Repetição)

```
LoppingFor.py > ...
1  # Looping For
2  # -----
3
4  # Fazendo um looping for de 0 até 10
5  for i in range(11):
6      print(f"O valor atual de i é: {i}")
7
8  # Fazendo um looping com início e fim definidos
9  for i in range(1, 11):
10     print(f"O valor atual de i é: {i}")
11
12 # Fazendo um looping com início, fim e passo definidos
13 for i in range(1, 11, 2):
14     print(f"O valor atual de i é: {i}")
15
16 # Fazendo um looping de 10 a 1
17 for i in range(10, 0, -1):
18     print(f"O valor atual de i é: {i}")
```



# Introdução ao Python

Mão na Massa:  
Exercícios Práticos

