

PYTHON

Prof. José Matheus

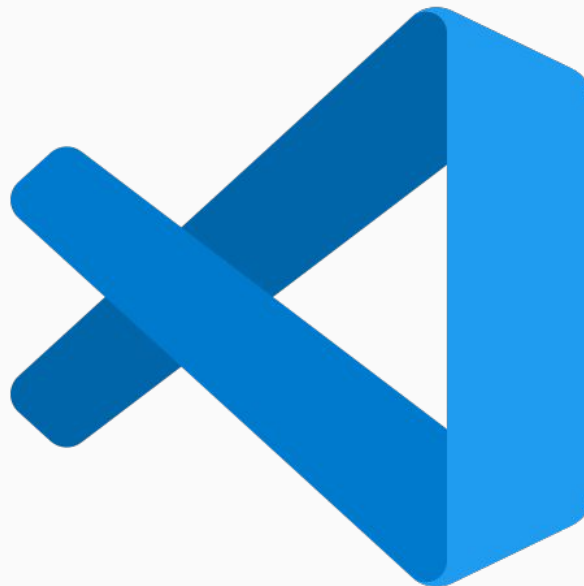
Ferramentas

Visual Studio
<https://code.visualstudio.com/>

Code:

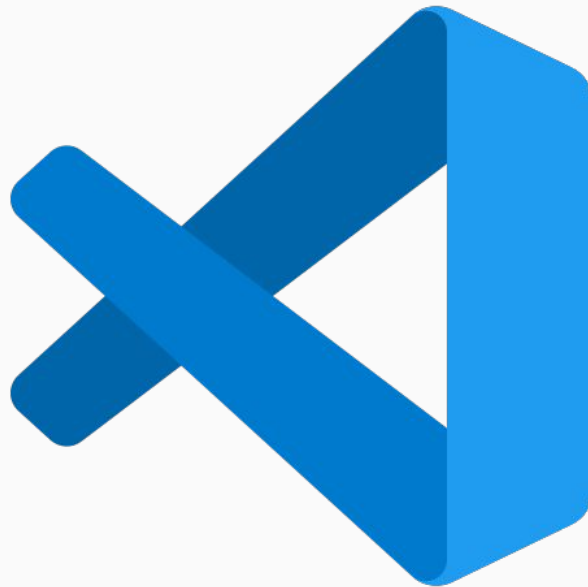
Plugins recomendados:

- Dracula official;
- Live Server;
- JavaScript Booster;
- Prettier;
- Image preview;
- ESLint;
- Indent-rainbow;
- CSS Peek;
- HTML CSS Support;
- IntelliCode;
- Material Icon Theme;
- Path Intellisense;
- SQLite Viewer;
- Django;
- isort;
- Pylance;
- Python;
- Python Debugger.



Ferramentas

- StandardJS - JavaScript Standard Style;
- TypeScript Importer;
- Typescript-Essentials;
- ES7+
React/Redux/React-Native snippets.



Ferramentas

Pycharm:

<https://www.jetbrains.com/pycharm/download/#section=windows>



Ferramentas

Postman:

<https://www.postman.com/downloads/>



Ferramentas

PostgreSQL / pgAdmin 4:

<https://www.postgresql.org/download/>

<https://www.pgadmin.org/download/pgadmin-4-windows/>



Introdução

Python é uma linguagem de programação criada pelo matemático e programador Guido van Rossum em 1991. Embora muita gente faça associação direta com cobras, por causa das serpentes píton ou pitão ou mesmo por causa de sua logo, a origem do nome se deve ao grupo humorístico britânico Monty Python.

A linguagem Python se destaca pela sua simplicidade, legibilidade e na filosofia "Zen of Python", uma coleção de 19 princípios orientadores, na forma de poema.

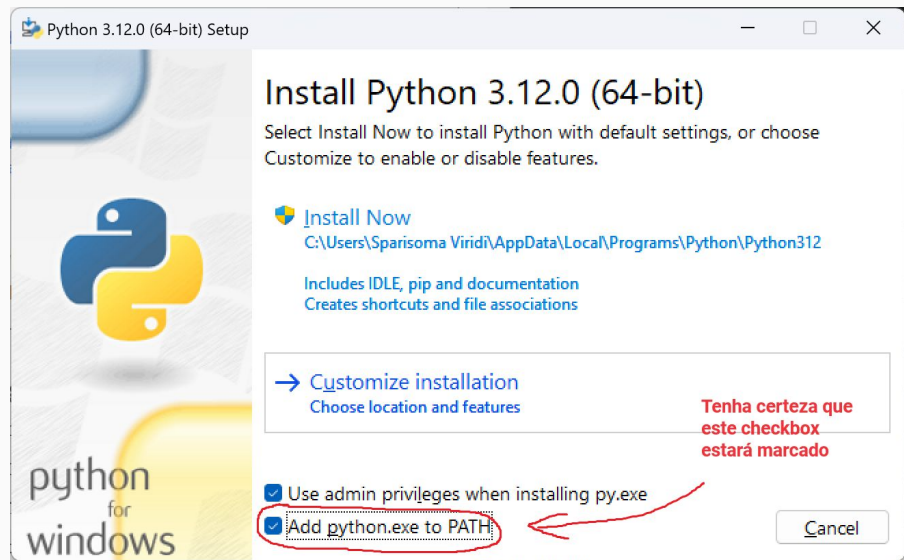
Sua popularidade é crescente devido à sua versatilidade.



Instalação do Python

Acesse o site oficial do Python (python.org) para baixar a versão mais recente (3.12.2);

Durante a instalação, marque a opção "Add Python to PATH" para facilitar o uso do Python no terminal.



Ambientes Virtuais

Ambientes virtuais são ambientes autocontidos para projetos Python, essenciais para evitar conflitos de dependências entre diferentes projetos, permitindo instalar pacotes específicos para um projeto sem afetar o ambiente global do Python.

Como criar um ambiente virtual:

- Utilize o comando `python -m venv nome_do_ambiente` no terminal;
- Ative o ambiente virtual com `source nome_do_ambiente/bin/activate` no Linux/Mac ou `nome_do_ambiente\Scripts\activate` no Windows;
- Para desativar basta digitar o comando “`deactivate`”.

Benefícios dos Ambientes Virtuais:

- **Isolamento de Dependências:** Cada projeto pode ter suas próprias versões de bibliotecas, evitando conflitos;
- **Reprodutibilidade:** Garante que outro desenvolvedor ou ambiente de produção tenha as mesmas dependências.

O prompt de comando mudará, indicando que você está agora dentro do ambiente virtual.

Gestão de Dependências com pip

“pip” é a ferramenta de gerenciamento de pacotes do Python;

- Para instalar pacotes, basta digitar o comando: `pip install nome_do_pacote`;
- Para salvar as dependências de um projeto em um arquivo, basta digitar: `pip freeze > requirements.txt`;
- E para instalar pacotes a partir de um arquivo, basta digitar: `pip install -r requirements.txt`.

Variáveis e Constantes

Variáveis armazenam dados mutáveis; constantes mantêm valores inalteráveis

A screenshot of a Python REPL window titled 'PYTHON'. The window has a dark background and three colored window control buttons (red, yellow, green) in the top right corner. The text inside the window shows three lines of code: a string 'Python' in green, the number 3.9 in red, and the number 3.1415 in red. This illustrates that Python does not raise an error when a variable is re-declared with a new value.

```
PYTHON  
  
"Python"  
  
3.9  
  
3.1415
```

Não é exibido nenhum erro ao re-declarar uma constante. O Python confia na responsabilidade do programador para seguir essas convenções.

Tipos de dados

Embora Python seja considerada uma linguagem de programação fracamente tipada, isso não significa que ela não tenha tipos. Na verdade, em Python, você pode trabalhar com tipos de dados e realizar operações específicas para cada tipo. A característica "fracamente tipada" refere-se à flexibilidade que Python oferece em relação à manipulação de tipos, especialmente quando comparada a linguagens fortemente tipadas como C++ ou Java.

Em Python, temos dois grupos de tipos de dados:

- **Tipos Primitivos:**

int, float, str, bool.

- **Tipos Compostos:**

Listas (list), Dicionários (dict).

Tipos de dados

- **Tipagem Dinâmica:** Python é uma linguagem de tipagem dinâmica, o que significa que você não precisa declarar o tipo de uma variável explicitamente. O tipo é inferido em tempo de execução.
- **Duck Typing:** Python segue o princípio de "duck typing", que se traduz como "se parece um pato e grasna como um pato, então deve ser um pato". Isso significa que o foco está nas características e comportamentos de um objeto, não no seu tipo específico.
- **Flexibilidade de Tipos:** Você pode realizar operações entre diferentes tipos de dados sem a necessidade de conversões explícitas em muitos casos. Python tentará realizar as operações da melhor maneira possível.
- **Tipos Básicos:** Python possui tipos básicos como inteiros (int), números de ponto flutuante (float), strings (str), booleanos (bool), listas (list), tuplas (tuple), conjuntos (set), e dicionários (dict).
- **Tipagem Forte nas Operações:** Apesar de ser fracamente tipada em declarações de variáveis, Python é fortemente tipada nas operações. Isso significa que você não pode realizar operações inválidas entre tipos diferentes sem converter explicitamente.

Tipos de dados

```
PYTHON

25

19.99

"João"

True

1 2 3

"chave" "valor"
```

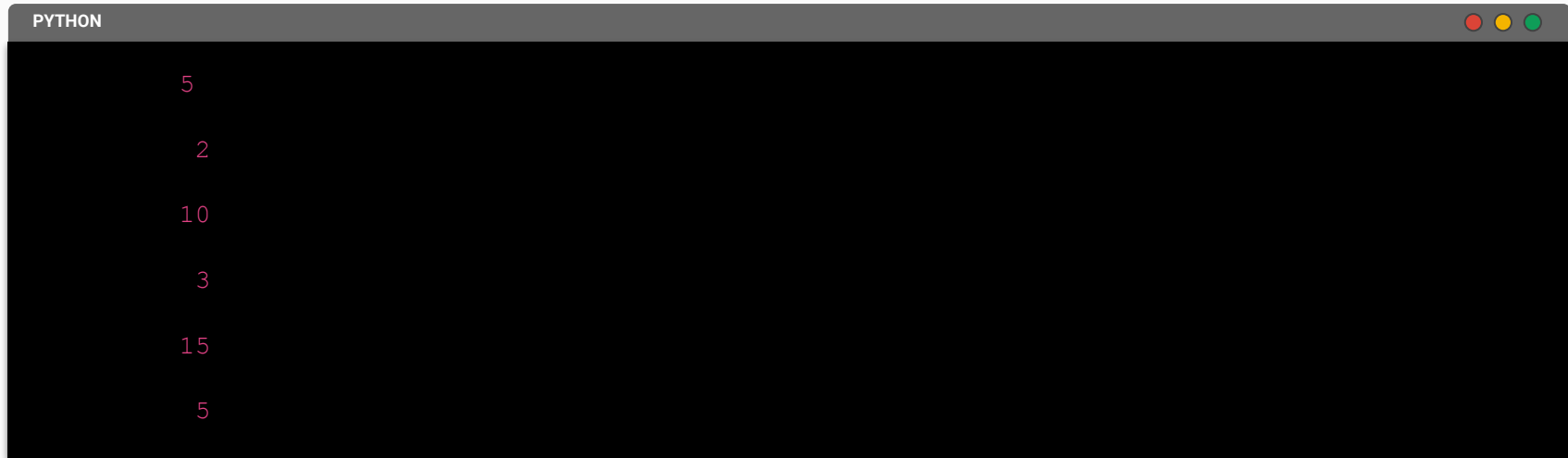
Operadores Aritméticos

Para realizar operações matemáticas, podemos utilizar os operadores aritméticos: +, -, *, /, // (divisão inteira), % (resto)



```
PYTHON
5 3
10 4
2 6
8 2
```

Operadores de Atribuição



```
PYTHON
5
    2
10
    3
15
    5
```


Entrada de Dados e Conversão de Tipos

PYTHON

```
int input "Digite sua idade: "
```

```
float input "Digite sua altura em metros: "
```

Formatação de Strings

PYTHON

```
"Alice"
```

```
30
```

```
"Olá, "          "! Você tem "  str          " anos."
```

```
"Olá, %s! Você tem %d anos."
```

```
"Olá, {}! Você tem {} anos." format
```

```
f"Olá, {nome}! Você tem {idade} anos."
```

Formatação de Strings

Em Python, existem várias maneiras de formatar strings. Aqui estão algumas das principais:

1. **Concatenação:** Simplesmente juntar strings usando o operador +.
2. **Placeholders %:** Usar o operador % para substituir valores em uma string.
3. **Método format():** Utilizar o método format() para inserir valores em uma string.
4. **F-Strings (a partir do Python 3.6):** Usar f-strings, uma forma mais moderna e legível.

Nas f-strings, você pode incluir expressões e até mesmo chamar métodos dentro das chaves.

Estruturas Condicionais

if, else, elif: Controle de fluxo baseado em condições.

PYTHON

```
if      18
    print "Você é maior de idade."

elif    18
    print "Você é menor de idade."

else
    print "Idade inválida."
```

Operadores de Comparação e Lógicos

Operadores: ==, !=, <, >, and, or, not.

PYTHON

```
if 18 and 1.60  
    print "Pode entrar no brinquedo."
```

Estruturas de Controle de Fluxo

O loop while é usado para repetir um bloco de código enquanto uma condição é verdadeira.



```
PYTHON  
  
    0  
while      5  
    print  
    1
```

Estruturas de Controle de Fluxo

O loop for é utilizado para iterar sobre sequências (como listas, tuplas, strings) ou outros objetos iteráveis.

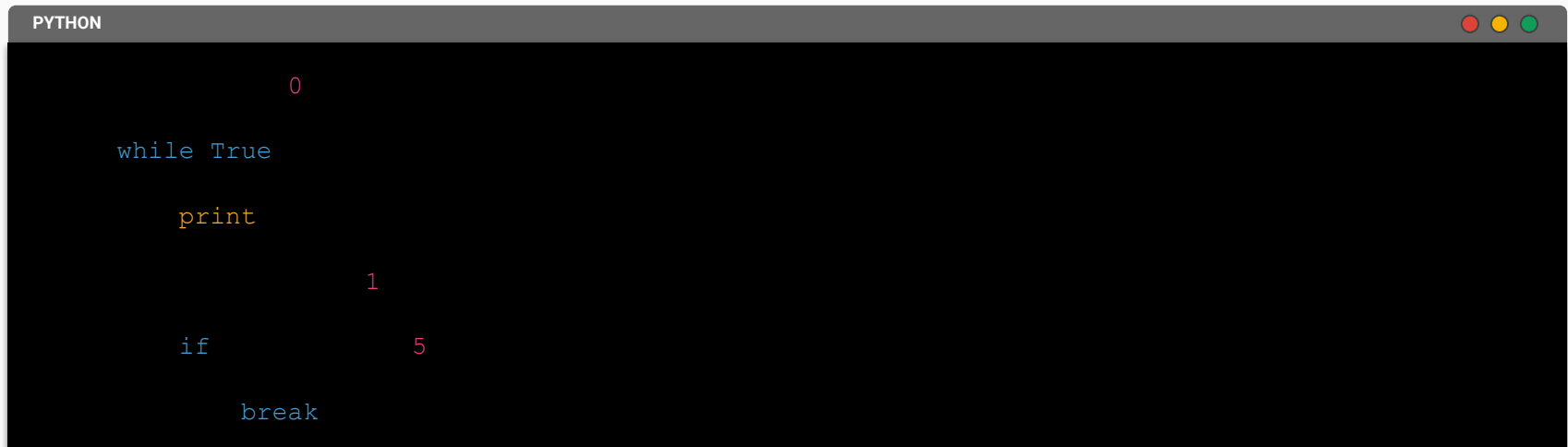
```
PYTHON
    "maçã"  "banana"  "laranja"

for      in

    print
```

Estruturas de Controle de Fluxo

`break`: É utilizado para sair imediatamente do loop, ignorando qualquer código restante no bloco.



```
PYTHON
0
while True
    print
    1
    if
        5
        break
```


Estruturas de Controle de Fluxo

continue: É utilizado para pular o restante do código no bloco e continuar com a próxima iteração do loop.

A screenshot of a Python code editor window. The window has a title bar with the word 'PYTHON' on the left and three colored window control buttons (red, yellow, green) on the right. The code is written in a dark-themed editor with syntax highlighting. It shows a 'for' loop with 'in range 5'. Inside the loop, there is an 'if' statement with a space followed by '2'. The 'if' block contains the 'continue' statement. After the 'if' block, there is a 'print' statement.

```
PYTHON

for   in range 5

    if      2

        continue

    print
```

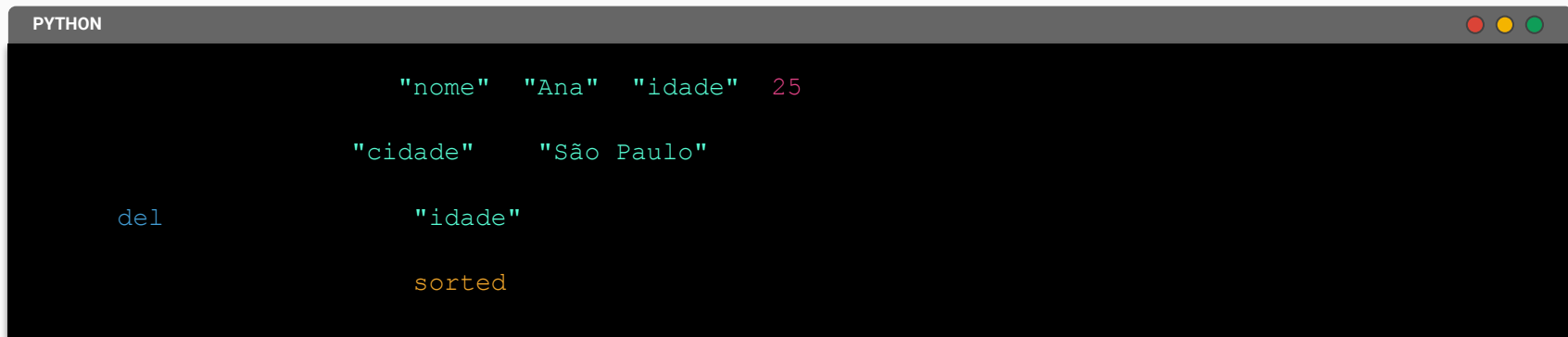
Listas

Operações Comuns: `append()`, `remove()`, `pop()`, `len()`.

```
PYTHON
1 2 3
4
2
2 # remove a partir do index
len
```

Dicionários

Operações: Adição (dic[chave] = valor), remoção (del dic[chave]), ordenação (sorted()).

A screenshot of a Python code editor window. The window has a title bar with the word 'PYTHON' on the left and three colored window control buttons (red, yellow, green) on the right. The code is written in a dark-themed editor with syntax highlighting. It shows a dictionary with keys 'nome', 'idade', and 'cidade'. The value for 'idade' is 25. The code demonstrates adding a new key-value pair, removing an existing key, and sorting the dictionary's keys.

```
PYTHON

dic = {"nome": "Ana", "idade": 25}

dic["cidade"] = "São Paulo"

del dic["idade"]

sorted(dic)
```

Funções

def para definir funções. Exemplos de funções lambda e função map().

PYTHON

```
def saudacao  
  
    return f"Olá, {nome}!"  
  
    "João"
```

Funções

- **map(função, sequência):** A função map() aplica uma função a **cada item em uma sequência** (como uma lista) e retorna um iterador contendo os resultados.
- **filter(função, sequência):** A função filter() constrói uma lista de elementos para os quais a **função dada uma condição retorna True**;
- **sorted(sequência, key=função, reverse=bool):** A função sorted() retorna **uma nova lista** contendo todos os itens da sequência na ordem classificada.

Funções

Funções lambda, também conhecidas como expressões lambda, são funções anônimas e de uma única linha em Python. Elas são definidas usando a palavra-chave lambda e são úteis quando você precisa de uma função temporária para uma operação específica.

```
PYTHON
lambda 2
print 5 # saída: 10
```

Funções

PYTHON

```
1 2 3 4 5  
  
map lambda 2  
  
print list          # saída: [1, 4, 9, 16, 25]
```

PYTHON

```
1 2 3 4 5 6 7 8 9 10  
  
filter lambda 2 0  
  
print list          # saída: [2, 4, 6, 8, 10]
```

Funções

PYTHON

```
        "nome"    "Alice"    "idade"    30    "nome"    "Bob"    "idade"    25    "nome"
"Charlie" "idade" 35

        sorted        "idade"

print

# saída: [{'nome': 'Bob', 'idade': 25}, {'nome': 'Alice', 'idade': 30}, {'nome':
'Charlie', 'idade': 35}]
```


Classes e Objetos

Orientação a Objetos: Classes e instâncias de objetos.

```
PYTHON

class Pessoa

    def __init__                # construtor

                                "Ana" 30
```

Tratamento de Exceções

try, except, finally: Lida com erros de forma controlada.

```
PYTHON

try
    10 / 0

except
    print "Divisão por zero não permitida."

finally
    print "Finalizando o bloco try-except."
```

Dúvidas?

Referências

PAIVA, Fábio A. P. de, NASCIMENTO, João M. A. do, MARTINS, Rodrigo S. e SOUZA, Givanaldo R. da. Introdução ao Python: Com aplicações de sistemas operacionais. Editora IFRN, Natal, 2020. E-book. ISBN 9786586293388.

Disponível em:

<https://memoria.ifrn.edu.br/bitstream/handle/1044/2090/EBOOK%20-%20INTRODU%C3%87%C3%83O%20A%20PYTHON%20%28EDITORIA%20IFRN%29.pdf?sequence=1&isAllowed=y>. Acesso em: 22 fev. 2024.