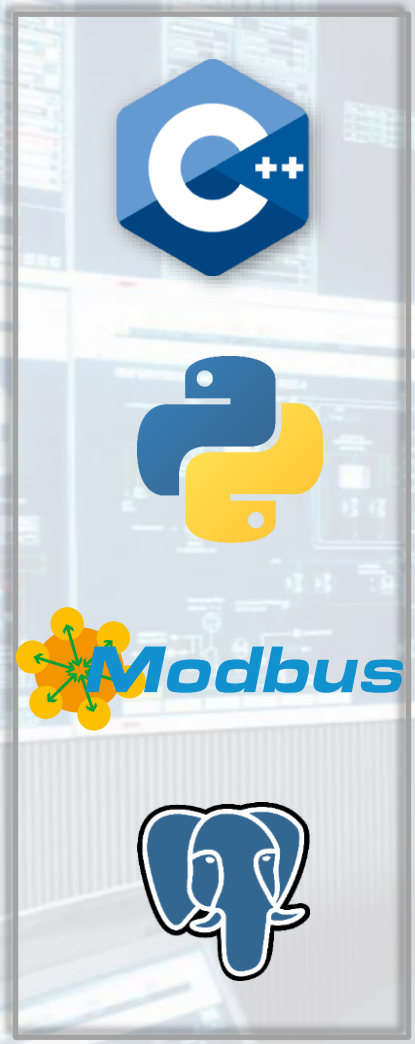


Informática Industrial

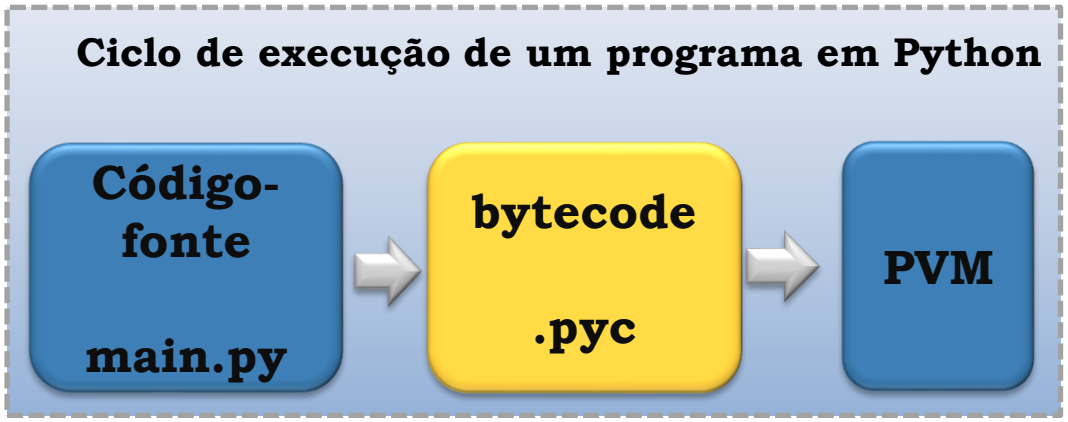
Introdução ao Python: Aspectos básicos e ambiente de programação

Prof. Guilherme Márcio Soares, Dr. Eng.
guilherme.marcio@ufjf.edu.br

ufjf
Universidade Federal de Juiz de Fora

Python

- Linguagem de **alto nível** (mais próxima à linguagem humana) lançada em 1991 por Guido van Rossum.
- É **interpretada**. Desta forma precisa de um interpretador para executar os programas.



Python

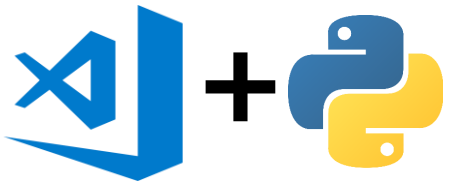
- Permite o desenvolvimento de aplicações com **menos linhas de código**;
- **Fracamente tipada**;
- A **separação dos blocos de código** é realizada por meio de **indentação e espaços** ao invés do uso de delimitadores como chaves e pontuações. **Usar IDEs !!**
- **Ecossistema** grande e **crescente**.

```
        def perm(l):                        NOVA LINHA
INDENT      if len(l) <= 1:                 NOVA LINHA
INDENT        return[1]                     NOVA LINHA
DEDENT      r = [ ]                         NOVA LINHA
            for i in range(len(l)):         NOVA LINHA
INDENT          s = l[:i] + l[i+1:]         NOVA LINHA
                p = perm(s)                 NOVA LINHA
DEDENT      for x in p:                     NOVA LINHA
INDENT          r.append(l[i:i+1]+x)        NOVA LINHA
DEDENT      return r
```

Configuração do Ambiente de Desenvolvimento

❑Passos:

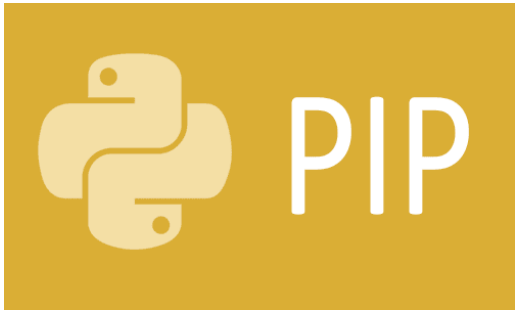
1. Verifique se o interpretador está instalado (**python --version** no terminal)
2. Instale a extensão Python no VSCode.
3. Execute um programa teste:



```
#Programa para a soma de dois números
a = input("Digite o primeiro operando: ")
b = input("Digite o segundo operando: ")
print("Resultado da soma: ", float(a) + float(b))
```

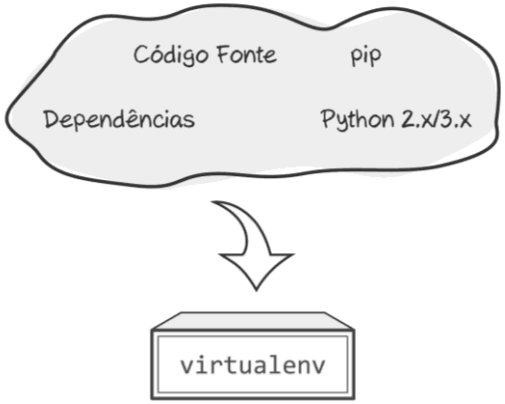
Instalação de bibliotecas e Virtual Environment

- A utilização de bibliotecas em Python é muito facilitada utilizando um sistema de gerenciamento de pacotes.
- Um dos gerenciadores mais utilizados é o **pip**, que permite a instalação de bibliotecas de maneira muito simplificada.
- A partir **da versão 3.4** do Python, o pip já foi incluído no instalador.



Instalação de bibliotecas e Virtual Environment

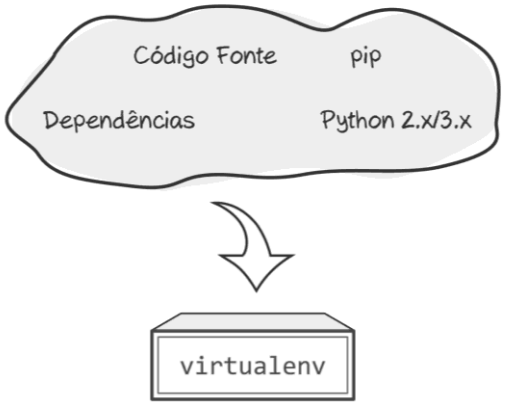
- Uma das grandes **vantagens** do desenvolvimento em **Python** é o seu conjunto **extenso de bibliotecas**.
- No entanto, a **instalação global** de bibliotecas **não é uma boa prática** na maior parte das vezes, pois em muitos casos, não existe **compatibilidade** de todos os programas com todas as **versões das bibliotecas**.
- Para evitar este problema, pode-se utilizar o **virtualenv**: uma ferramenta que permite o desenvolvimento de soluções em um **ambiente isolado**.



Fonte: https://pythonacademy.com.br/blog/python-e-virtualenv-como-programar-em-ambientes-virtuais

Instalação de bibliotecas e Virtual Environment

- Ao criar um **virtualenv**, será criada uma cópia separada de todos os diretórios necessários para a execução de um programa em Python:
  - As **bibliotecas comuns** do Python (standard library);
  - O gerenciador de pacotes **pip**;
  - O **interpretador** (Python 2.x/3.x);
  - As **dependências** que estiverem no diretório site-packages;
  - **Código-fonte**;



Fonte: https://pythonacademy.com.br/blog/python-e-virtualenv-como-programar-em-ambientes-virtuais

Instalação de bibliotecas e Virtual Environment

- Instalação:

  pip install virtualenv

- Inicialização de um novo ambiente:

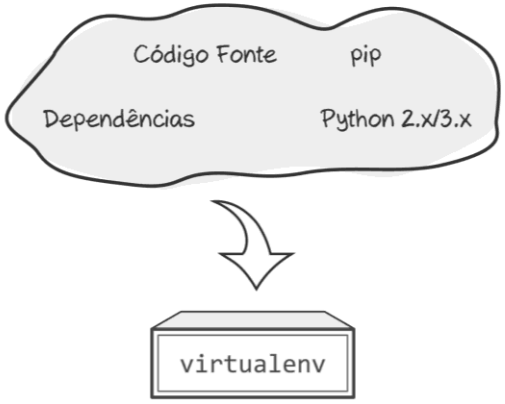
  python –m virtualenv [opções] <nome_da_pasta>

- Ativação

  <nome_da_pasta>\Scripts\activate

- Desativação

  <nome_da_pasta>\Scripts\deactivate



Fonte: https://pythonacademy.com.br/blog/python-e-virtualenv-como-programar-em-ambientes-virtuais

Instalação de bibliotecas e Virtual Environment

❑ Utilização de bibliotecas:

1. Instale a biblioteca através do comando **pip**;
2. Importe a biblioteca no código-fonte;

❑ Exemplo de bibliotecas:

- **Numpy**[1]: operação com vetores e matrizes;
- **Matplotlib**[2]: criação de gráficos e visualizações.

[1]https://numpy.org/doc/stable/

[2]https://matplotlib.org/

Coleções

❑ Coleções são contêineres para armazenamento de dados. Existem vários tipos de contêineres diferentes:

- **Listas (list)** : estruturas similares aos vetores do C++, mas com muito mais recursos. Podem armazenar dados de tipos diferentes nas posições. O acesso aos elementos é realizado por meio de índices.
- **Dicionários (dict)** : estrutura capaz de armazenar dados através da indexação por **chaves únicas**. Os elementos são alocados de acordo com a lógica **chave-valor**.
- **Tuplas (tuple)** : estruturas similares às listas, no entanto são **imutáveis**.

❑ Cada um dos contêineres listados são **classes** com diferentes métodos e atributos.

Coleções

❑ Coleções ordenadas:

- **Pilha** (pode ser implementada com o container **list**): container ordenado em que o primeiro elemento que entra é o último que sai.
- **Fila (deque)** : container ordenado em que o primeiro elemento que entra é o primeiro que sai.

❑ Cada um dos contêineres listados são **classes** com diferentes métodos e atributos.

Condicionais

- As condicionais em Python seguem a mesma lógica do C++, diferindo somente pela sintaxe.

```
if expressão_booleana:
        expressões
else:
        expressões
```

Cuidado com a indentação !

Loops

- O loop **while** segue a mesma lógica do C++, isto é, enquanto uma expressão booleana for verdadeira, executa um determinado bloco de código.

- O **loop** for possui uma lógica um pouco diferente. Sua execução está relacionada à varredura de uma sequência, como *strings*, listas, *tuplas*, etc.

```
for x in  sequencia:

    expressões
```

Cuidado com a indentação !

Desafio

- Utilize um dicionário para armazenar os parâmetros de um inversor de frequência.

- A estrutura do dicionário deverá ser:

  - Chave principal: número do parâmetro
  - Valor principal: outro dicionário contendo as informações

- Após criar o dicionário, crie um código para imprimir todos os parâmetros armazenados utilizando loops.

Parâm.	Descrição	Faixa de Valores	Ajuste de Fábrica	Ajuste do Usuário	Propr.	Grupos	Pág.
P0000	Acesso aos Parâmetros	0 a 9999	0				5-2
P0001	Referência Velocidade	0 a 65535			ro	READ	17-1
P0002	Velocidade de Saída (Motor)	0 a 65535			ro	READ	17-1
P0003	Corrente do Motor	0,0 a 200,0 A			ro	READ	17-1
P0004	Tensão Barram. CC (Ud)	0 a 2000 V			ro	READ	17-1
P0005	Frequência de Saída (Motor)	0,0 a 500,0 Hz			ro	READ	17-2
P0006	Estado do Inversor	0 = Ready (Pronto) 1 = Run (Execução) 2 = Subtensão 3 = Falha 4 = Autoajuste 5 = Configuração 6 = Frenagem CC 7 = Estado Dormir			ro	READ	17-2
P0007	Tensão de Saída	0 a 2000 V			ro	READ	17-3
P0009	Torque no Motor	-1000,0 a 1000,0 %			ro	READ	17-3
P0010	Potência de Saída	0,0 a 6553,5 kW			ro	READ	17-4
P0011	Fator de Potência	-1,00 a 1,00			ro	READ	17-4

Dica:

```
for key, value in a_dict.items():
    print(key, '->', value)
```