

Disciplina:

PYTHON PARA CIÊNCIAS DE DADOS

Professor: Leandro Lessa



APRESENTAÇÃO DO PROFESSOR

- Cientista de Dados – SEBRAE/PE.
- Professor Universitário – Desde 2020.
 - Python, Ciências de Dados, Algoritmos de ML.
 - Análise de dados, Engenharia, Segurança entre outros.
- Palestrante.
- Mestre em Informática – PUC Minas 2019
- Pós Graduado em Business Intelligence – PUC Minas 2016
- Graduação em Sistemas de Informação – PUC Minas 2014



APRESENTAÇÃO DO PROFESSOR

- Mais de 15 anos de experiência em tecnologia.
 - Desde 2012 com foco exclusivamente na área de dados.
- Gosto de músicas, seriados, filmes, jogos e animes.
- Iniciante em:
 - Trilhas
 - Blogueiro




leandrolessa.com.br
lessadatatech.com.br



**DE MÉDICO E LOUCO
CADA UM TEM UM POUCO**



ATUAÇÃO SOCIAL

- Palhaço Voluntário 
- Realizo visitas a hospitais, orfanatos e asilos
- Conhecido como Dr. Trimiliki



APRESENTAÇÃO DA DISCIPLINA

EMENTA DO CURSO

Tipos de dados. Estruturas de controle: condicional e repetição. Estruturas de dados: listas, tuplas, conjuntos, dicionários, séries e dataframes. Funções. Vetorização e matrizes numéricas. Bibliotecas de manipulação de dados, de visualização de dados e vetorização de matrizes.

OBJETIVO

FUNDAMENTOS DE PYTHON

- Compreensão e aplicação de tipos de dados.
- Utilização de estruturas de controle condicional e de repetição.
- Desenvolvimento de habilidades na criação e utilização de funções.

MANIPULAÇÃO DE DADOS

- Exploração de estruturas de dados como listas, tuplas, conjuntos, dicionários, séries e dataframes.
- Coleta e Integração de dados.
- Tratamento e correção de Dados.

APLICAÇÃO PRÁTICA DE ANÁLISE DE DADOS

- Utilização de bibliotecas para manipulação e visualização de dados.
- Desenvolvimento de habilidades práticas para análise de dados.
- Criação de visualizações e comunicar resultados de forma eficiente

AGENDA

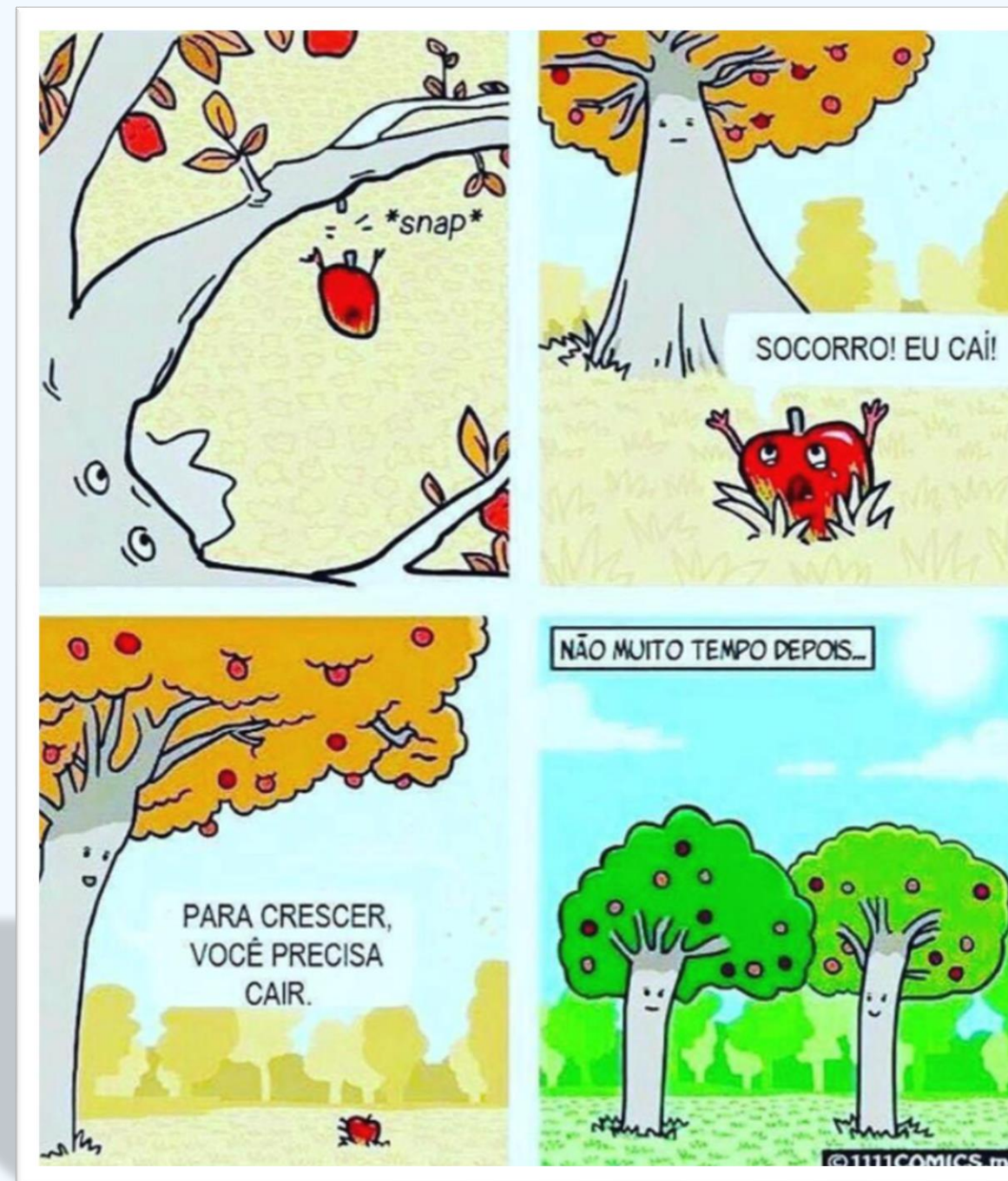
Data	Conteúdo Programado	C/H
10/04/2024	Preparação do ambiente, Fundamentos de Python e Tipos de Dados	04
17/04/2024	Estruturas de Dados, Estruturas de Controle e Funções	04
24/04/2024	Coleta, Tratamento e Manipulação de Dados	04
08/05/2024	Análise Descritiva e Exploratória de dados	04
15/05/2024	Elaboração dos Trabalhos / Apoio aos grupos	04
22/05/2024	Apresentação dos Trabalhos e entrega do relatório técnico	04
Total C/H		24

UMA BREVE REFLEXÃO

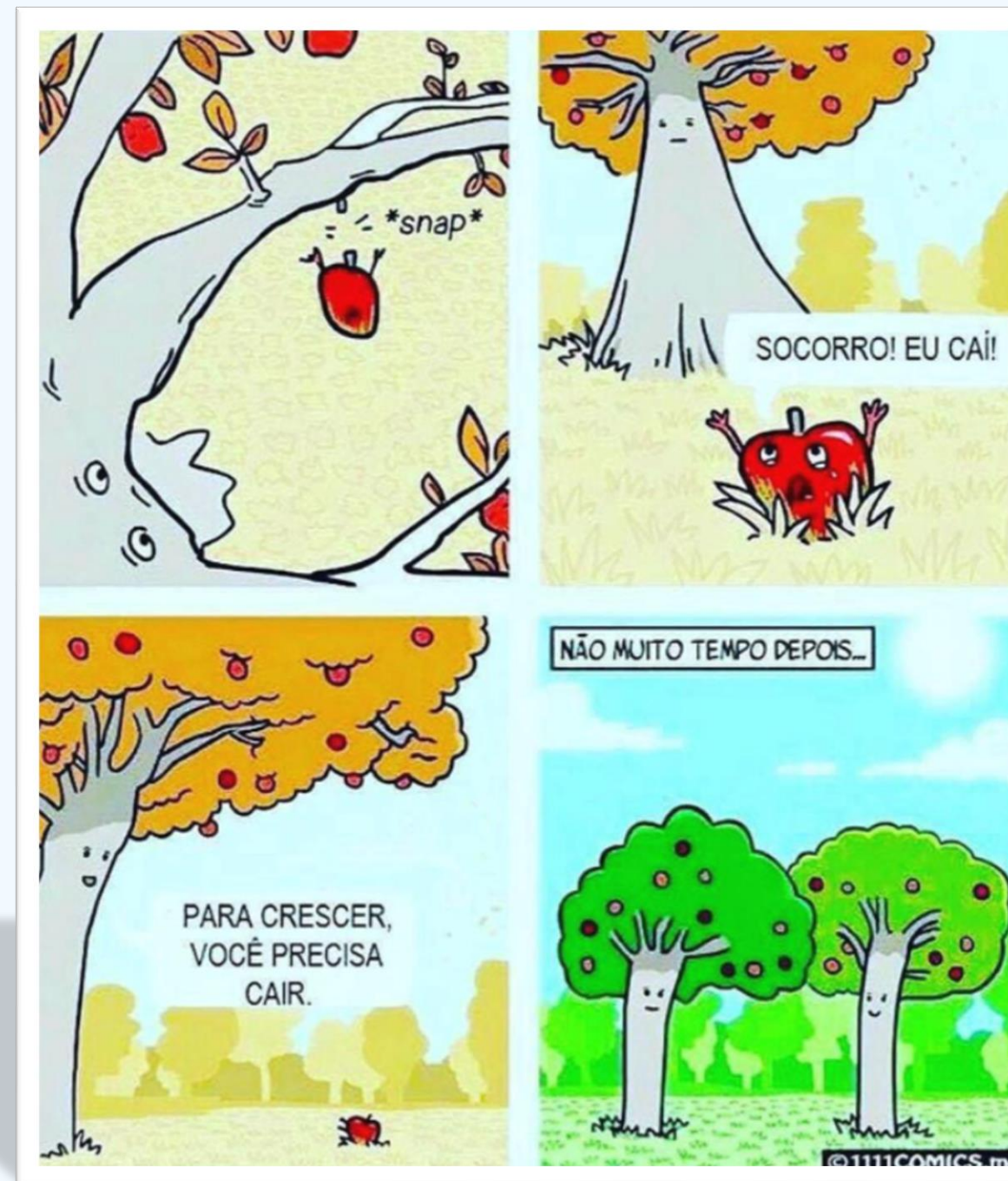


Não Desista no 1º Passo

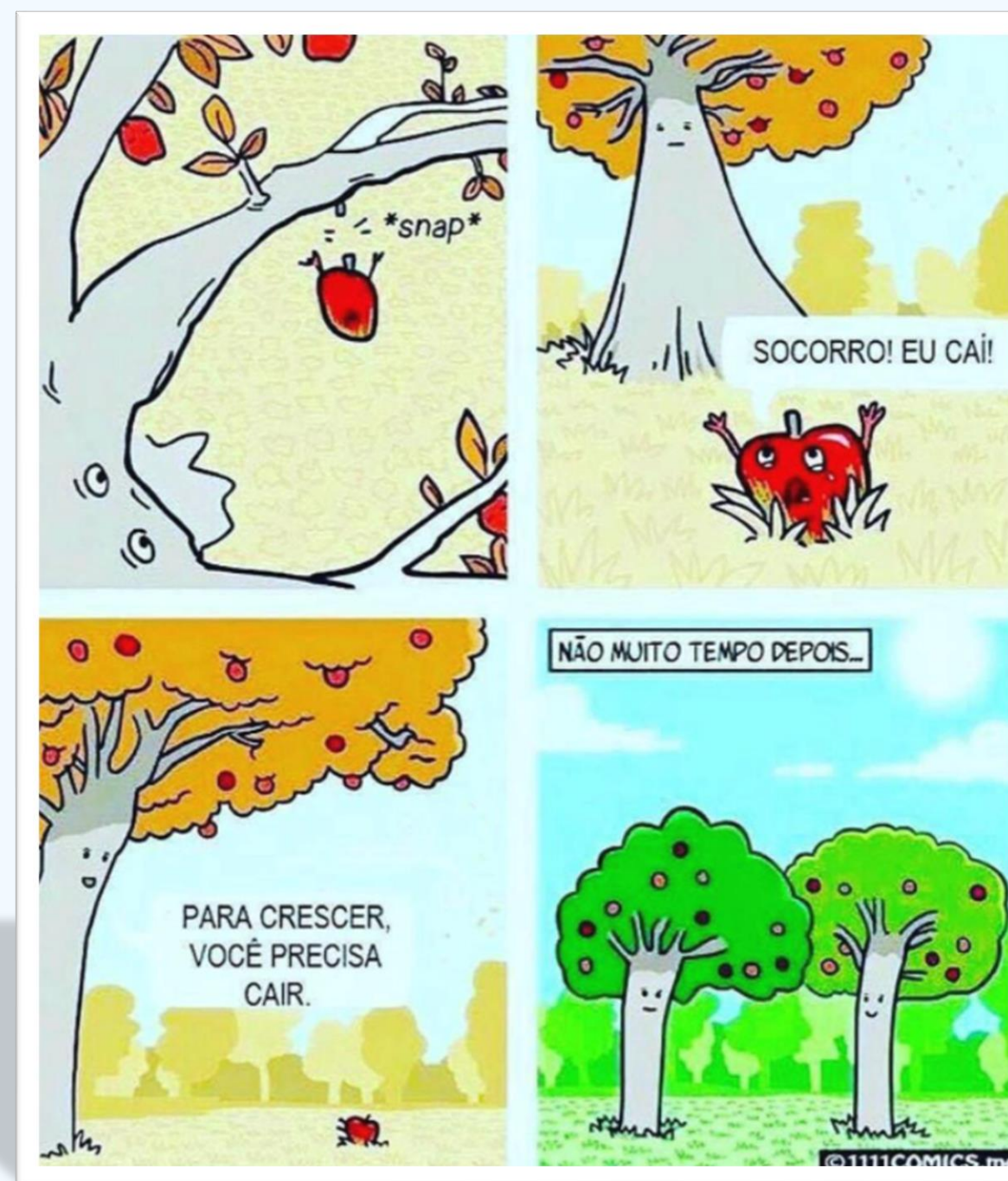




**Persevere!
Acredite nos
seus sonhos**



**Persevere!
Acredite nos
seus sonhos**



**Saia da sua zona
de conforto!**

PREPARAÇÃO DO AMBIENTE

PREPARAÇÃO DO AMBIENTE

O QUE SERÁ NECESSÁRIO?

- Instalação do Python.
- Instalação do Vscode/Jupyter notebook.
- **Vantagens:**
 - Muito utilizado no contexto profissional.
- **Desvantagem:**
 - Instalar e configurar os plugins necessários.

OPCIONAL

- Utilizar o google COLAB.
- Necessário ter uma conta do google pra realizar o acesso.
- **Vantagens:**
 - Não precisa instalar nenhum recurso adicional.
- **Desvantagens:**
 - Não utilizado no contexto profissional.

INSTALAÇÃO DO PYTHON

Download do software

- <https://www.python.org/downloads/>
- Tutorial Completo:
 - <https://leandrolessa.com.br/tutoriais/3-tutoriais-para-instalar-o-python-no-seu-sistema-operacional/>

INSTALAÇÃO DO VSCODE

Download do software

- <https://code.visualstudio.com/download>
- Tutorial Completo.
 - <https://leandrolessa.com.br/tutoriais/>

AMBIENTES VIRTUAIS

VIRTUALENV

INTRODUÇÃO

- O Virtualenv é uma ferramenta que permite criar ambientes virtuais isolados para projetos Python.
- Cada ambiente virtual possui sua própria instalação de Python e bibliotecas, independentemente do sistema operacional.

VIRTUALENV

VANTAGENS

➤ Isolamento de Ambientes

- Permite que você trabalhe em diferentes projetos Python com dependências diferentes, sem conflitos.

➤ Gerenciamento de Dependências

- Facilita a instalação e gerenciamento de bibliotecas específicas para cada projeto.

➤ Reprodução do Ambiente

- Garante que todos os colaboradores do projeto utilizem as mesmas versões de bibliotecas, facilitando a reprodução do ambiente em diferentes máquinas.

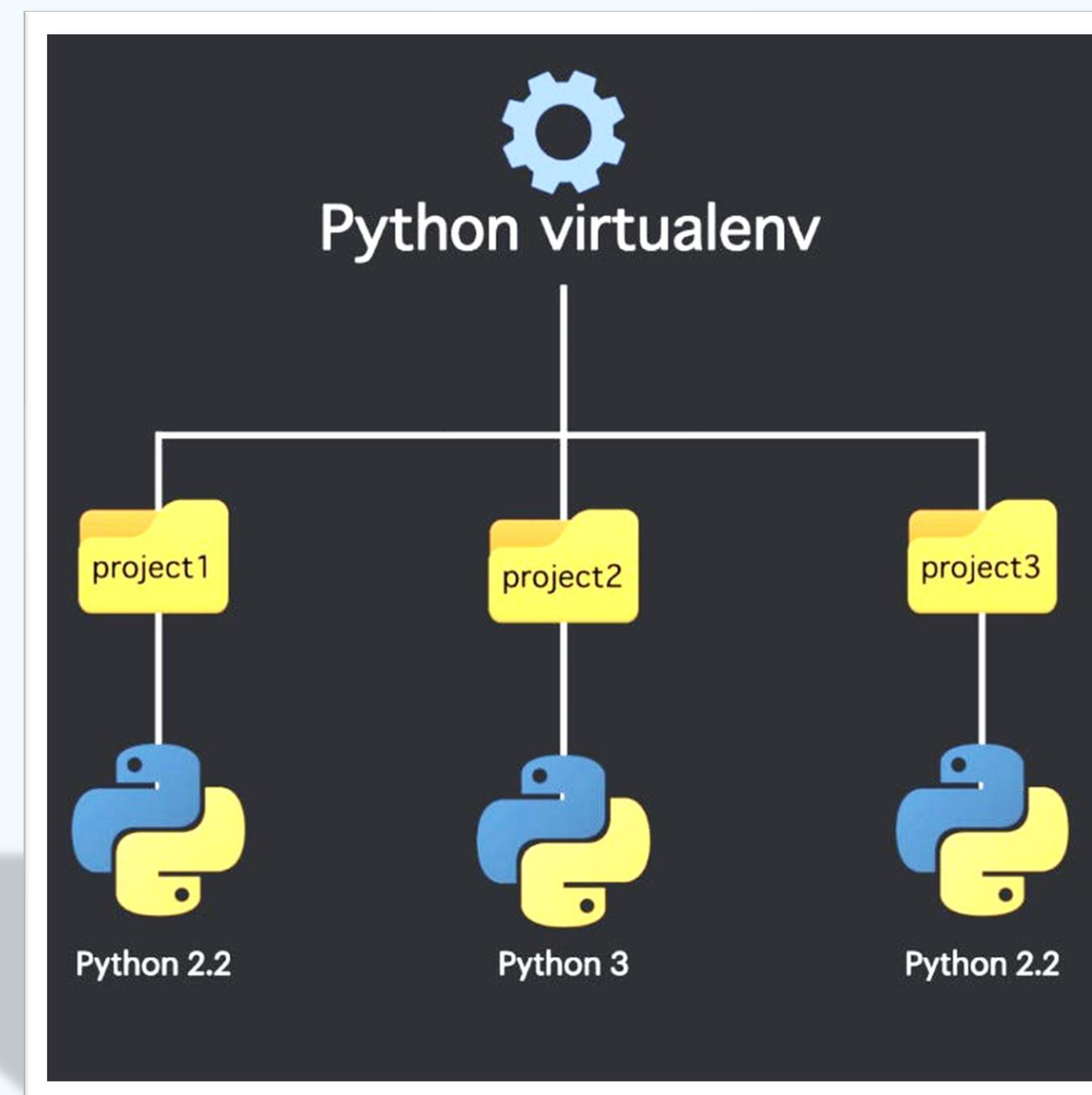
➤ Manutenção Simples

- Fácil de criar, ativar e desativar ambientes virtuais, tornando a manutenção do ambiente de desenvolvimento mais eficiente.

VIRTUALENV

APLICAÇÕES

- Com o uso de virtualenvs, podemos ter múltiplos projetos de forma independente e isolada.
 - Análise de Dados
 - Automação de Scripts
 - Coleta de Dados
 - Machine Learning
 - Processamento de NLP



FUNDAMENTOS DE PYTHON

PYTHON

VISÃO GERAL

- Python é uma linguagem de programação de alto nível e fácil aprendizado.
- Desenvolvida por Guido van Rossum e lançada em 1991.
- Possui grande comunidade de desenvolvedores, que contribuem constantemente com novas bibliotecas, módulos e ferramentas para a linguagem.

SIMPLICIDADE E VERSATILIDADE

- Sintaxe clara e concisa facilita a leitura e escrita de código.
- Versatilidade em diferentes domínios
- desde desenvolvimento web até análise de dados.

PYTHON

IMPORTÂNCIA CRESCENTE

- Python se tornou a linguagem padrão na Ciência de Dados.
- Amplamente adotada em empresas e instituições de pesquisa.

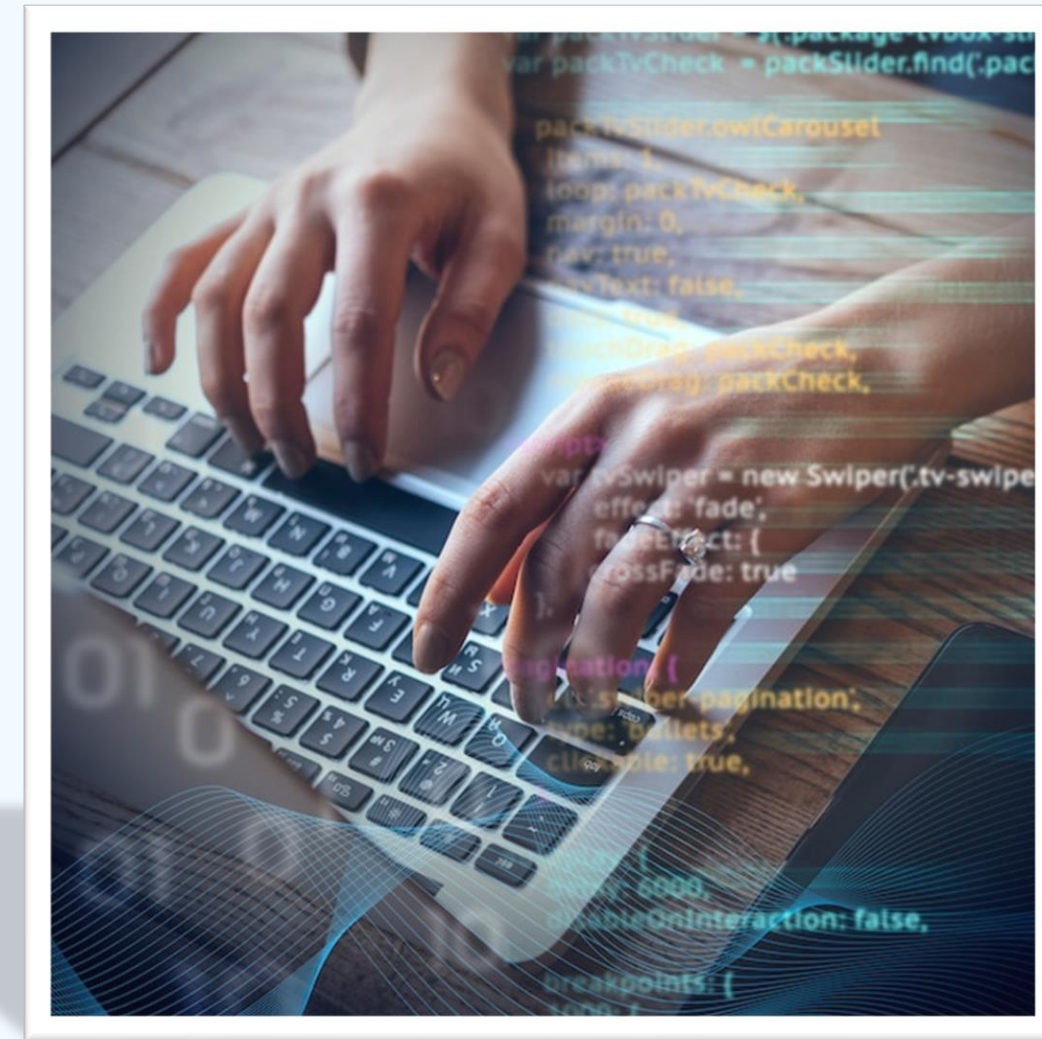
VANTAGENS

- Open Source: Gratuito e com código-fonte aberto.
- Multiplataforma: Funciona em Windows, macOS e Linux.
- Integração Fácil: Integra-se bem com outras linguagens.

PYTHON

APLICAÇÕES

- A linguagem Python pode ser aplicada em diversos contextos.
 - Análise de Dados.
 - Automatização de Tarefas.
 - Ciências de Dados.
 - Criação de Pipelines.
 - Desenvolvimento Web.
 - Desenvolvimento de Jogos.
 - Machine Learning e IA.
 - Entre outros.



FUNDAMENTOS DA SINTAXE

FUNDAMENTOS DA SINTAXE

INTRODUÇÃO

- Refere-se à estrutura e organização do código em uma linguagem de programação.
- Python é conhecido por sua sintaxe limpa e legível.

INDENTAÇÃO SIGNIFICATIVA

- Python utiliza a indentação para definir blocos de código.
- A clareza é incentivada, tornando o código mais compreensível.

VARIÁVEIS E ATRIBUIÇÃO

- A declaração de variáveis é simples e não requer especificação de tipo.
- Exemplo: idade = 25 ou nome = "Leandro".

FUNDAMENTOS DA SINTAXE

TIPOS DE DADOS

- Python abrange uma ampla gama de tipos de dados, englobando números, strings, listas, tuplas, dicionários e conjuntos.
- Esses tipos não apenas representam diferentes formas de armazenar informações, mas também ditam as operações que podem ser executadas sobre eles.

OPERADORES E EXPRESSÕES

- Python apresenta uma extensa gama de operadores, abrangendo os aritméticos, de atribuição, de comparação e lógicos.
- As expressões em Python unem valores e operadores, culminando em resultados diversos.
- Compreender a interação desses fundamentos é essencial para trabalhar fluidamente com a sintaxe Python.

TIPOS DE DADOS

STRING

- O tipo de dados de string é uma sequência de caracteres que representa um texto.
- Ela é uma das principais estruturas de dados da linguagem, sendo amplamente utilizada em diversos contextos, como manipulação de arquivos, comunicação com banco de dados e processamento de texto.

```
texto = 'Conhecendo o Python - Aula 01'  
print(texto)
```


INT

- O tipo de dados inteiro (int) representa números inteiros positivos ou negativos.
- Os Inteiros podem ser criados simplesmente digitando um número inteiro diretamente no código Python ou podem ser o resultado de operações matemáticas.

```
# Atribuição dos valores inteiros
variavel_a = 10
variavel_b = 5

# Somas das variáveis
soma = variavel_a + variavel_b
print(soma)
```

FLOAT

- O tipo de dado float, também conhecido como ponto flutuante, é utilizado para representar números que possuem casas decimais. Assim como os inteiros, os floats podem ser definidos diretamente no código ou gerados como resultado de operações matemáticas.

```
# Atribuição dos valores inteiros
variavel_a = 10.5
variavel_b = 2.75

# Somas das variáveis
soma = variavel_a + variavel_b
print(soma)
```

BOLEANO

- O tipo de dados booleano é utilizado para representar valores lógicos de verdadeiro (**True**) ou falso (**False**).
- Os valores booleanos são frequentemente utilizados em expressões condicionais e em operações de controle de fluxo, como em estruturas de decisão (if, elif, else) e em loops (while, for).

```
# Definição das permissões do usuário
tem_permissao = True

# Verificação da permissão de acesso
if tem_permissao:
    print("Acesso concedido. Bem-vindo!")
else:
    print("Desculpe, você não tem permissão para acessar este recurso.")
```

FUNÇÕES

FUNÇÕES

- Uma função em Python é um bloco de código reutilizável que executa uma tarefa específica.
- Ela recebe entradas (conhecidas como parâmetros ou argumentos), processa essas entradas e pode retornar um resultado. As funções são uma parte essencial da programação em Python.
- Permitem dividir o código em partes menores e mais gerenciáveis, promovendo a reutilização e a modularidade do código.

```
def soma(a, b):  
    resultado = a + b  
    return resultado
```

EXEMPLOS PRÁTICOS



ESTRUTURAS DE DADOS

LISTAS – []

- As listas em Python são estruturas de dados fundamentais que permitem armazenar coleções ordenadas de itens.
- Elas são mutáveis, o que significa que você pode adicionar, remover e modificar elementos após a criação da lista. As listas são muito versáteis e são amplamente utilizadas em programação Python para armazenar e manipular conjuntos de dados.
- As Listas são utilizadas em diversas situações, como por exemplo:
 - Lista de Compras.
 - Histórico de Transações Financeiras.
 - Lista de Tarefas a Fazer.
 - Dados de Estoque de Produtos.

```
# Criando uma lista de números  
numeros = [10, 20, 30, 40, 50]
```

```
# Acessando elementos da lista usando índices  
print("Primeiro elemento:", numeros[0])    # Saída: 10  
print("Último elemento:", numeros[-1])    # Saída: 50
```

TUPLAS – ()

- As tuplas em Python são estruturas de dados semelhantes às listas, porém **imutáveis**, o que significa que uma vez que uma tupla é criada, seus elementos não podem ser alterados, adicionados ou removidos.
- Isso as torna úteis para armazenar coleções de itens que não precisam ser modificadas ao longo do tempo.
- As tuplas são utilizadas em diversas situações, como por exemplo:
 - Na representação de coordenadas geográficas (latitude e longitude).
 - Na definição de cores em sistemas de design gráfico.
 - Na criação de jogos que requerem o armazenamento de posições de elementos em uma grade, entre outros.

```
# Criando uma tupla  
tupla = (1, 2, 3, 4, 5)
```

```
# Acessando elementos da tupla usando índices  
print("Primeiro elemento:", tupla[0])    # Saída: 1  
print("Último elemento:", tupla[-1])     # Saída: 5
```

DICIONÁRIOS - { }

- Os dicionários em Python são estruturas de dados que permitem armazenar pares chave-valor, onde cada valor é associado a uma chave única.
- Eles são muito úteis para representar dados estruturados e indexados por chaves significativas, facilitando o acesso e a manipulação dos dados.
- Os dicionários são utilizadas em diversas situações, como por exemplo:
 - Registro de Alunos em uma Escola.
 - Dados de Clientes em um Sistema de CRM.
 - Registro de Vendas em um E-commerce.

```
pessoa = {'nome': 'Leandro',  
          'idade': 30,  
          'cidade': 'Belo Horizonte'  
}
```

```
print(pessoa['nome']) # Saída: Leandro
```

Conjuntos - { }

- Os conjuntos em Python são estruturas de dados que representam coleções de elementos **únicos** e **não ordenados**.
- Eles são úteis para operações que envolvem testes de pertencimento, remoção de duplicatas e operações de conjunto como união, interseção e diferença.
- Os conjuntos são utilizados em diversas situações, como por exemplo:
 - Remoção de Duplicatas de uma Lista.
 - Verificação de Elementos Únicos em uma Coleção.
 - Operações de Conjunto em Listas.

```
conjunto = {1, 2, 3, 4, 5}
print(conjunto) # Saída: {1,2,3,4,5,6}
```

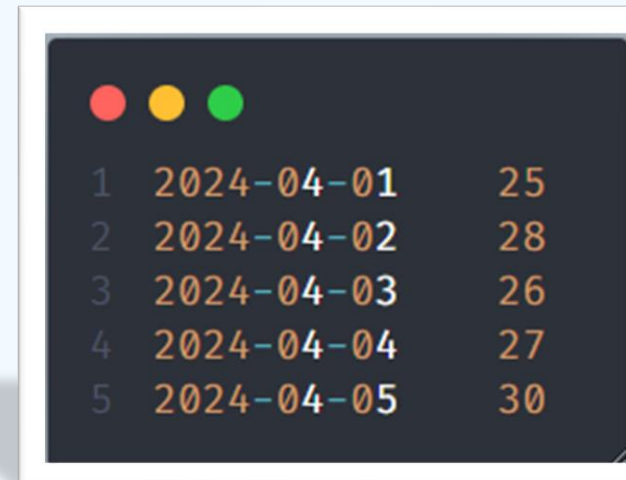
Séries

- Uma série é uma estrutura de dados unidimensional que pode conter qualquer tipo de dados, como números inteiros, números de ponto flutuante, strings, entre outros.
- Cada elemento em uma série possui um rótulo associado, chamado de índice. A série pode ser vista como uma coluna em uma planilha ou uma matriz com apenas uma linha.

```
import pandas as pd

# Criando uma série de temperaturas
temperaturas = pd.Series([25, 28, 26, 27, 30],
                          index=['2024-04-01', '2024-04-02',
                                '2024-04-03', '2024-04-04',
                                '2024-04-05'])

print(temperaturas)
```



1	2024-04-01	25
2	2024-04-02	28
3	2024-04-03	26
4	2024-04-04	27
5	2024-04-05	30

DataFrames

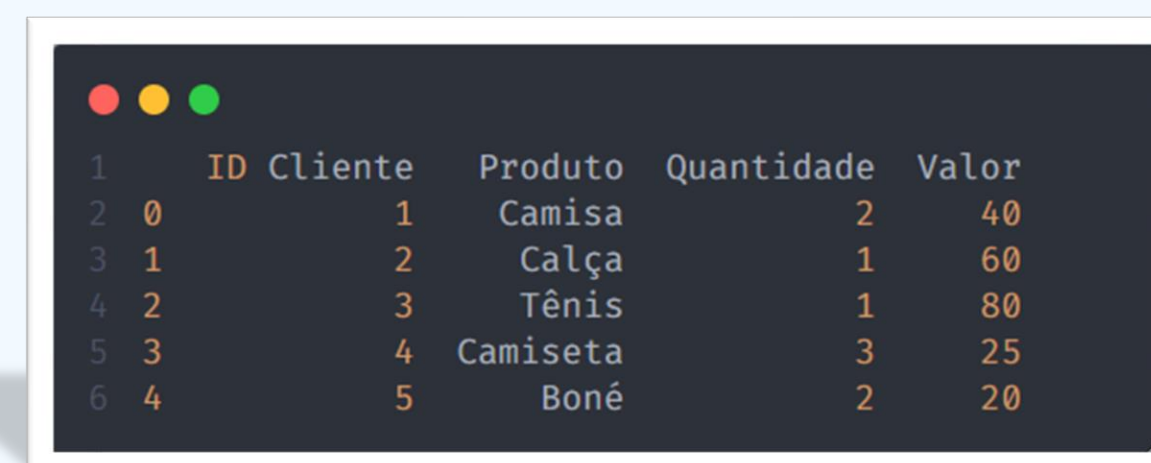
- Um DataFrame é uma estrutura de dados bidimensional semelhante a uma tabela de banco de dados ou uma planilha do Excel.
- Ele é composto por linhas e colunas, onde cada coluna pode ser de um tipo de dado diferente.
- Os DataFrames são altamente flexíveis e podem ser usados para armazenar e manipular dados heterogêneos.

```
import pandas as pd

# Criando um DataFrame de vendas
data = {
    'ID Cliente': [1, 2, 3, 4, 5],
    'Produto': ['Camisa', 'Calça', 'Tênis', 'Camiseta', 'Boné'],
    'Quantidade': [2, 1, 1, 3, 2],
    'Valor': [40, 60, 80, 25, 20]
}

vendas = pd.DataFrame(data)

print(vendas)
```



	ID Cliente	Produto	Quantidade	Valor	
2	0	1	Camisa	2	40
3	1	2	Calça	1	60
4	2	3	Tênis	1	80
5	3	4	Camiseta	3	25
6	4	5	Boné	2	20

EXEMPLOS PRÁTICOS



ESTRUTURAS DE CONTROLE

Estrutura de Controle

- As estruturas de controle são essenciais em programação para controlar o fluxo de execução do código. Duas das estruturas mais fundamentais são as estruturas **condicionais** e de **repetição**.
- **Estruturas Condicionais:**
 - Permitem que um programa tome decisões com base em condições específicas. Em Python, a estrutura condicional básica é o `if`, que pode ser estendido com `elif` e `else` para lidar com múltiplas condições.
- **Estruturas de Repetição:**
 - As estruturas de repetição permitem que um conjunto de instruções seja executado repetidamente enquanto uma condição é verdadeira. Em Python, temos os loops `for` e `while`.

Estrutura de Controle

➤ Estruturas Condicionais

```
idade = 20

if idade ≥ 18:
    print("Você pode entrar no bar.")
else:
    print("Você não tem idade suficiente para entrar no bar.")
```

➤ Estruturas de Repetição

```
lista_tarefas = ['acordar as 6:00', 'escovar os dentes', 'ler 20 minutos',
                'Fazer atividade física', 'estudar', 'almoçar', 'trabalhar']

for atividade in lista_tarefas:
    print(atividade)
```

VETORIZAÇÃO E MATRIZES

Vetorização

- É um processo de otimização de código, onde operações são aplicadas diretamente a arrays ou vetores em vez de iterar sobre cada elemento individualmente.
- A vetorização é amplamente utilizada em cenários onde a eficiência computacional é crucial, como no processamento de grandes conjuntos de dados em análise de dados.
- Exemplos:
 - Manipulação de dados.
 - Aprendizado de Máquina.
 - Processamento de Imagens.

Matrizes

- São representações de dados organizados em uma estrutura de tabela bidimensional, onde os elementos são acessados por meio de índices de linha e coluna.
- Essas matrizes podem ser implementadas usando listas aninhadas (listas de listas) ou utilizando bibliotecas especializadas, como NumPy.

- Exemplos:

- Manipulação de dados.
- Aprendizado de Máquina.
- Processamento de Imagens.

```
# criação da matriz com as vendas da semana
vendas_semana = np.array([
    [12, 10, 8, 6],
    [18, 15, 12, 10],
    [10, 8, 5, 3],
    [22, 20, 18, 16],
    [15, 12, 10, 8]
])
```

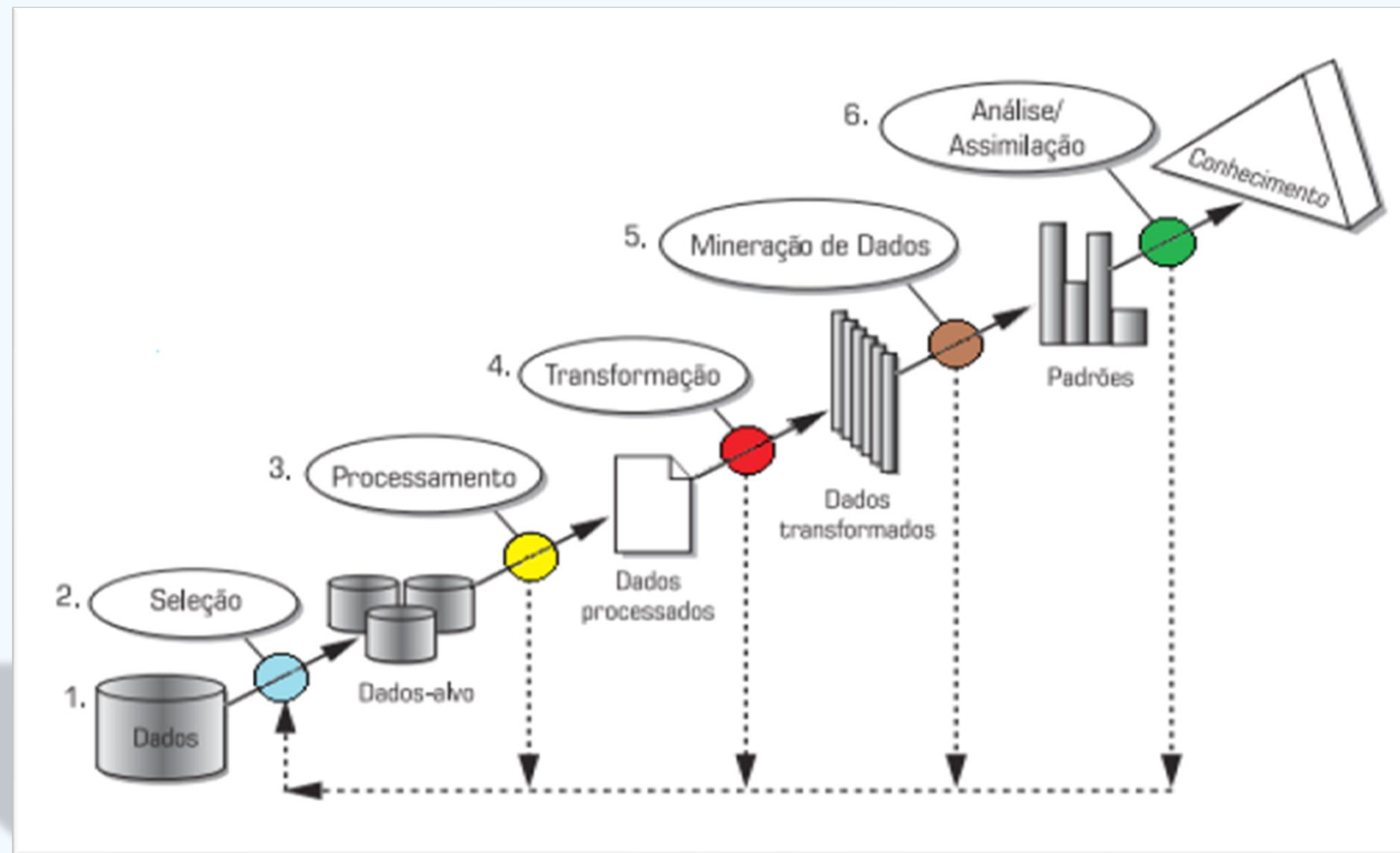
```
# As linhas representam um dia da semana e
# as colunas a quantidade de vendas de
# produtos.
```

KNOWLEDGE DISCOVERY IN DATABASES

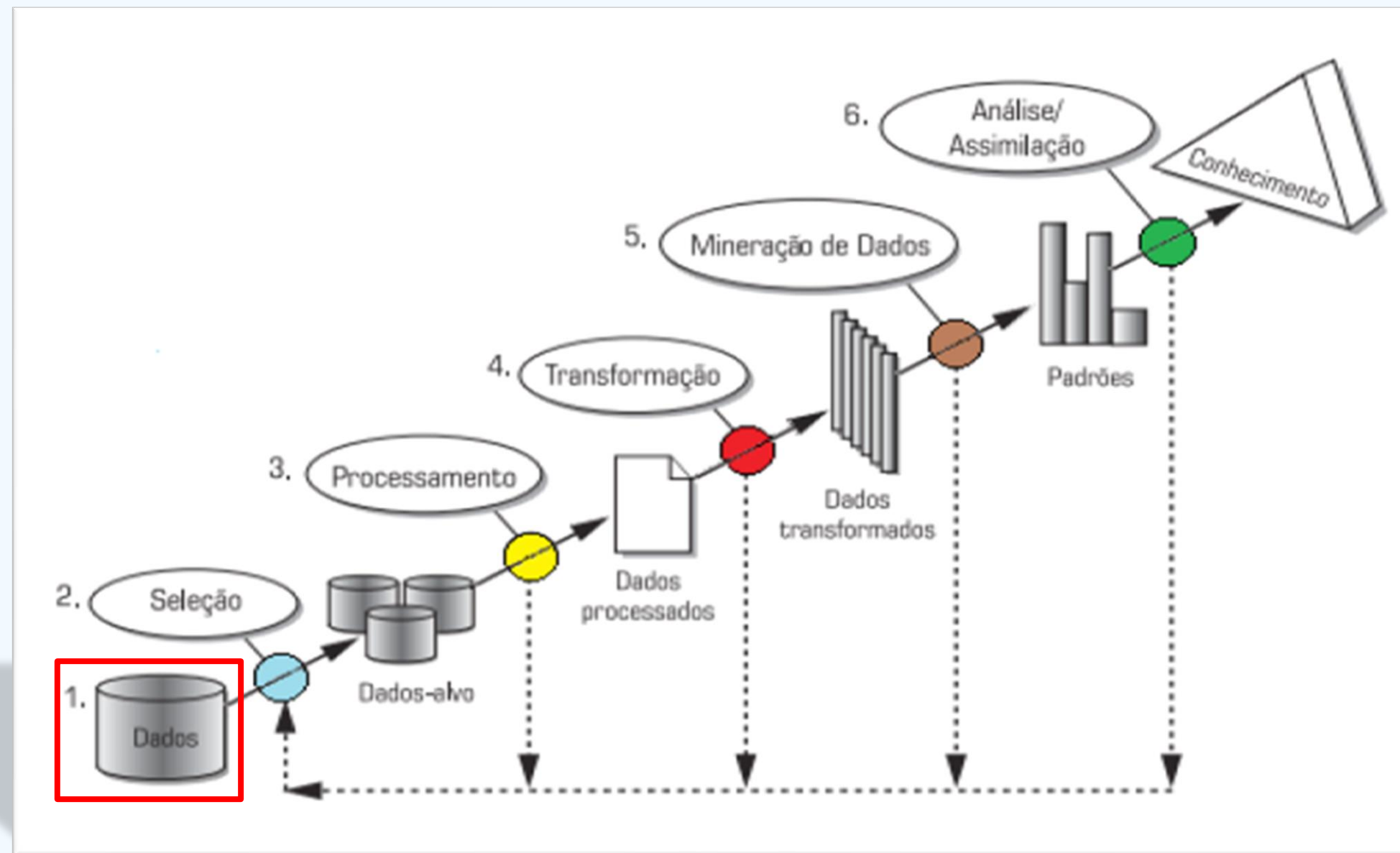
KDD

- O processo de Knowledge Discovery in Databases (KDD) é uma abordagem sistemática e iterativa para transformar dados brutos em conhecimento útil.
- O KDD mais do que uma metodologia específica. Ele representa um conjunto de etapas inter-relacionadas e técnicas aplicadas na descoberta de conhecimento a partir de grandes conjuntos de dados.
- É um processo complexo que envolve várias etapas, desde a seleção de dados até a interpretação dos resultados.

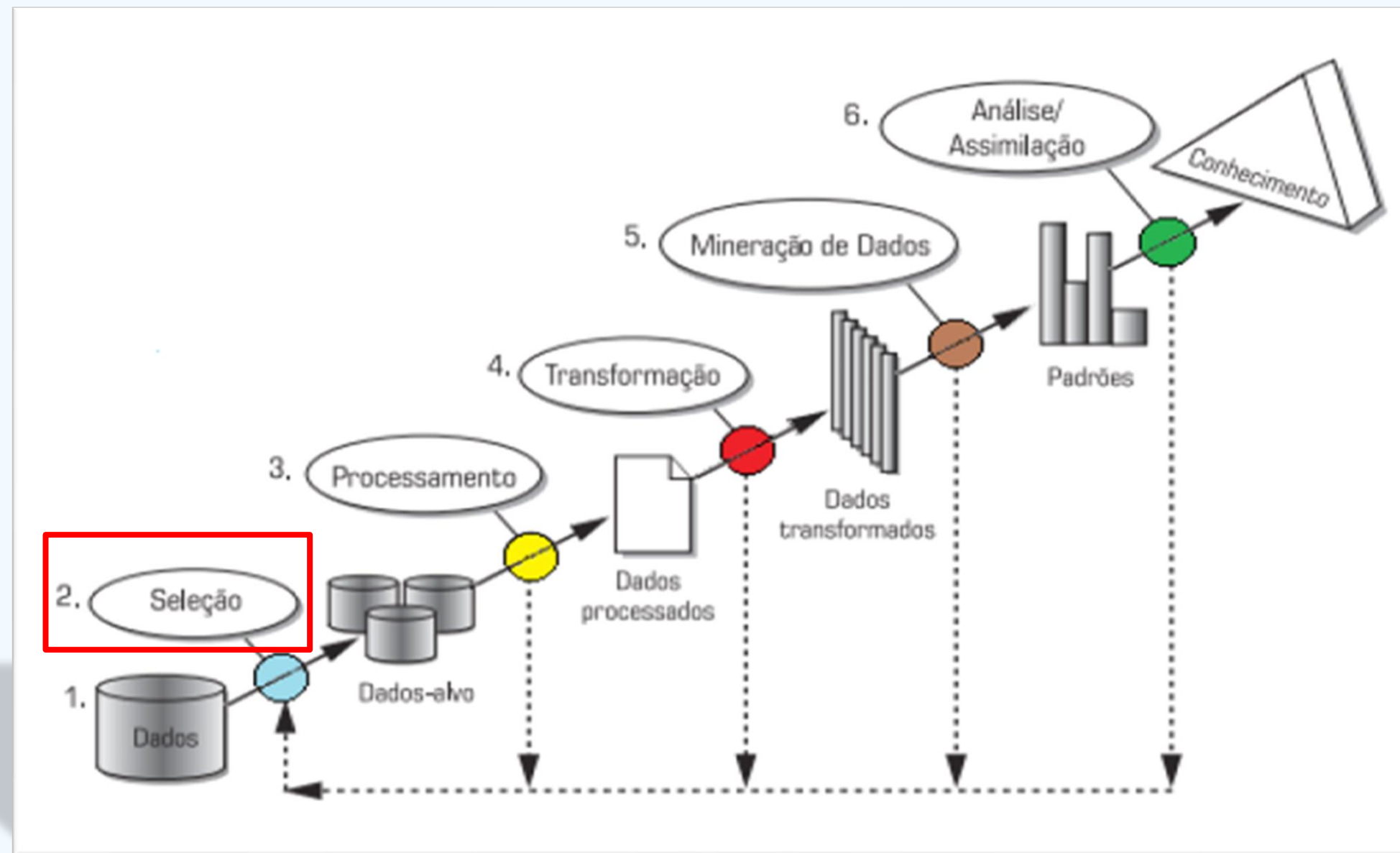
KDD



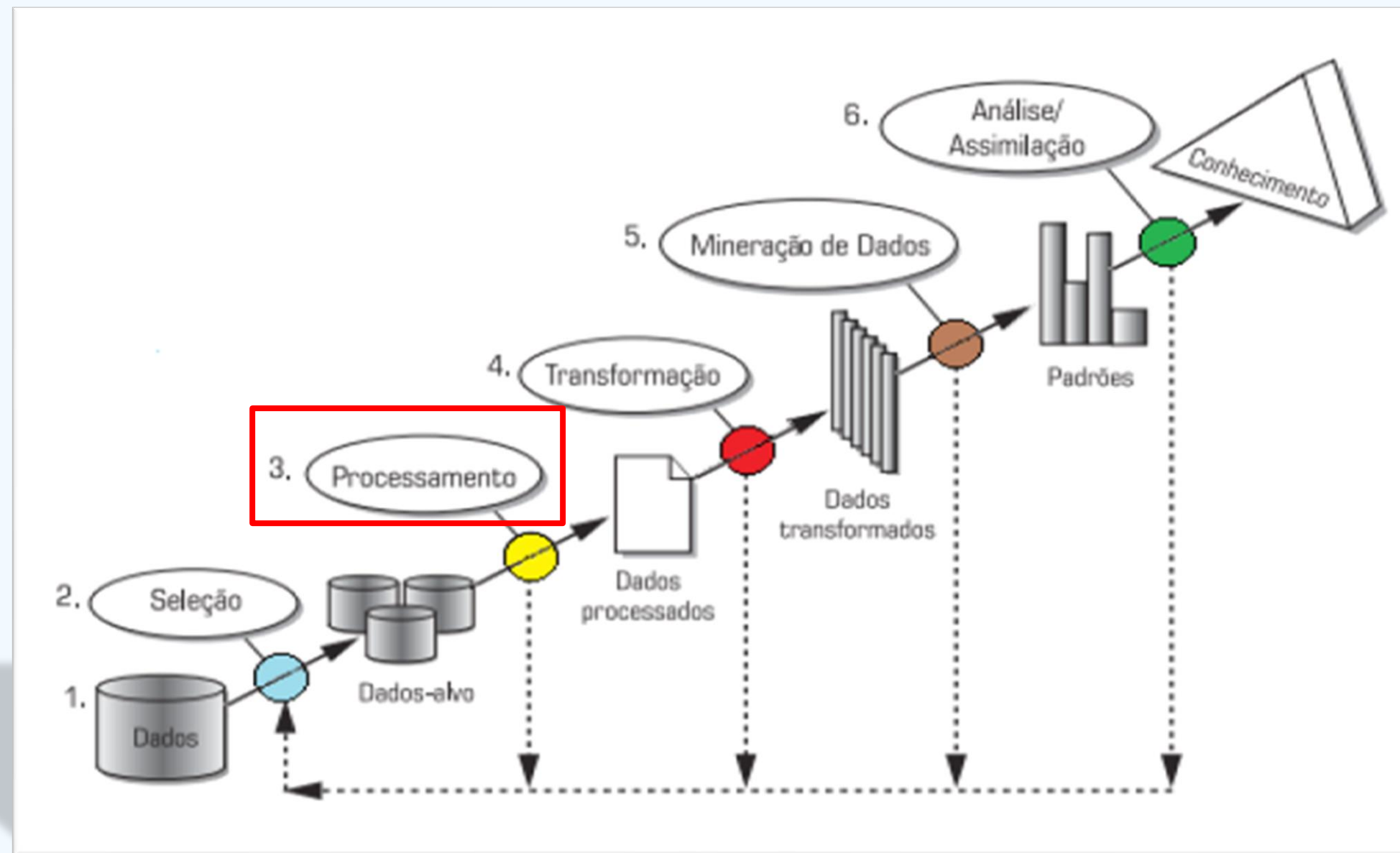
KDD



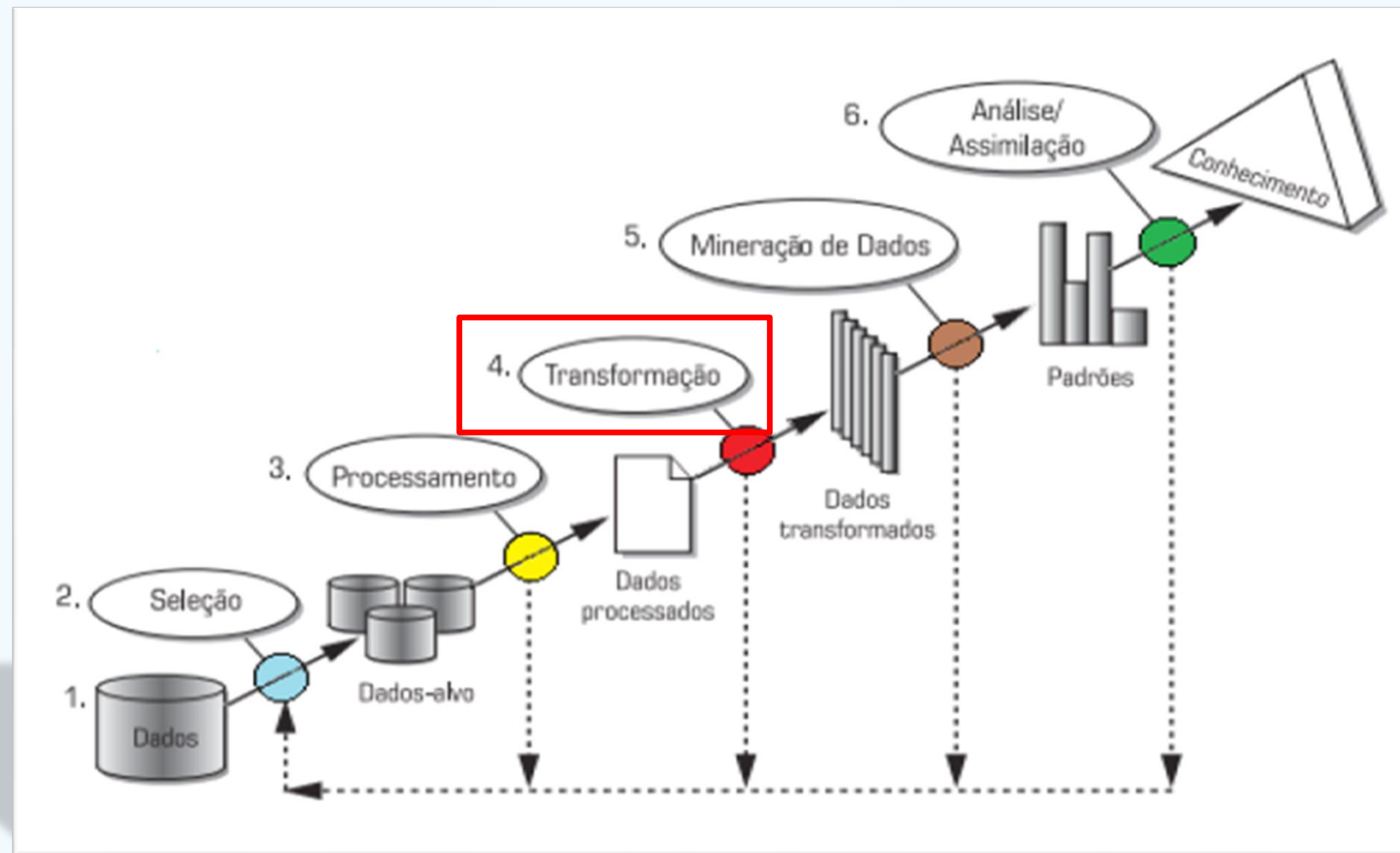
KDD



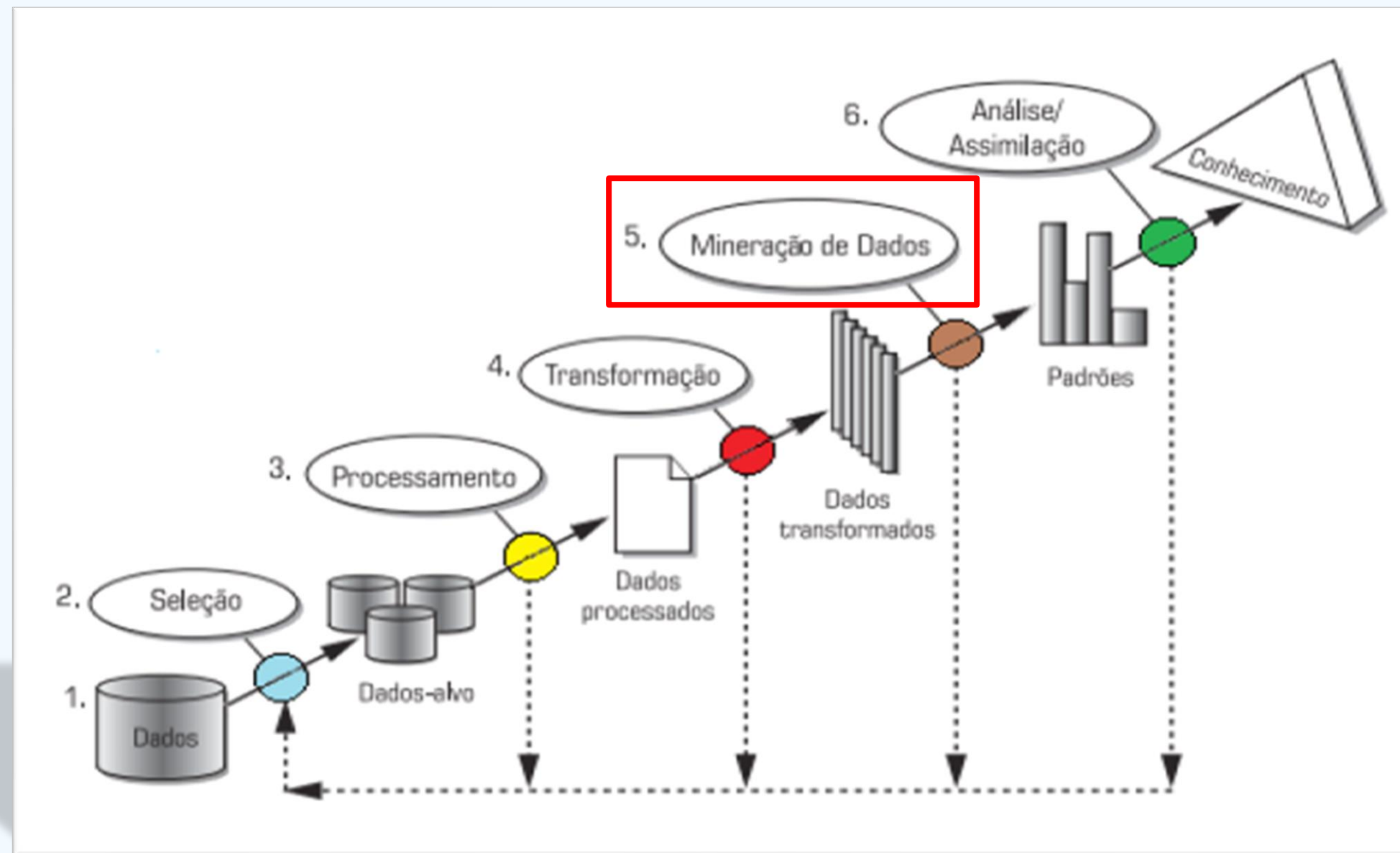
KDD



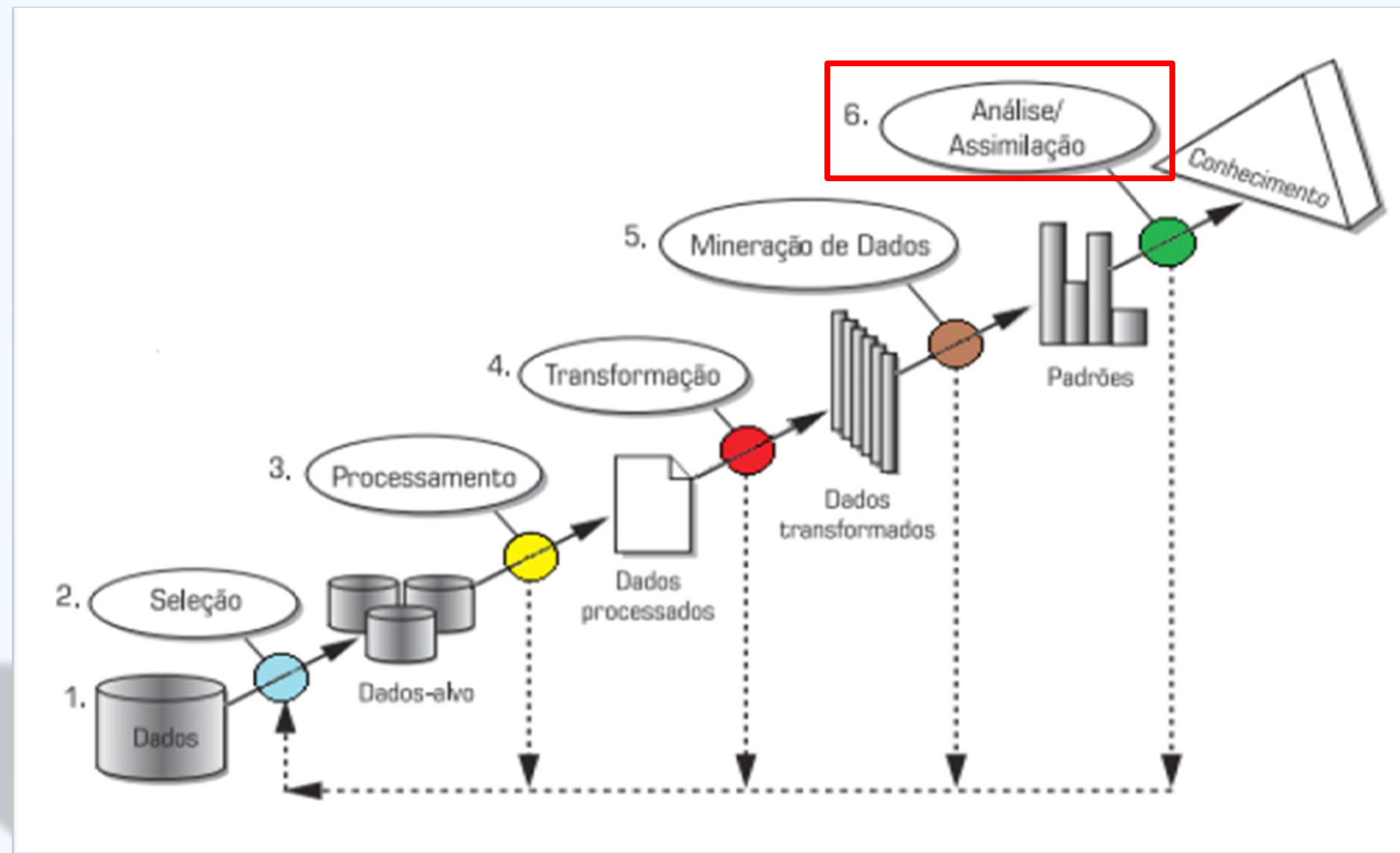
KDD



KDD



KDD

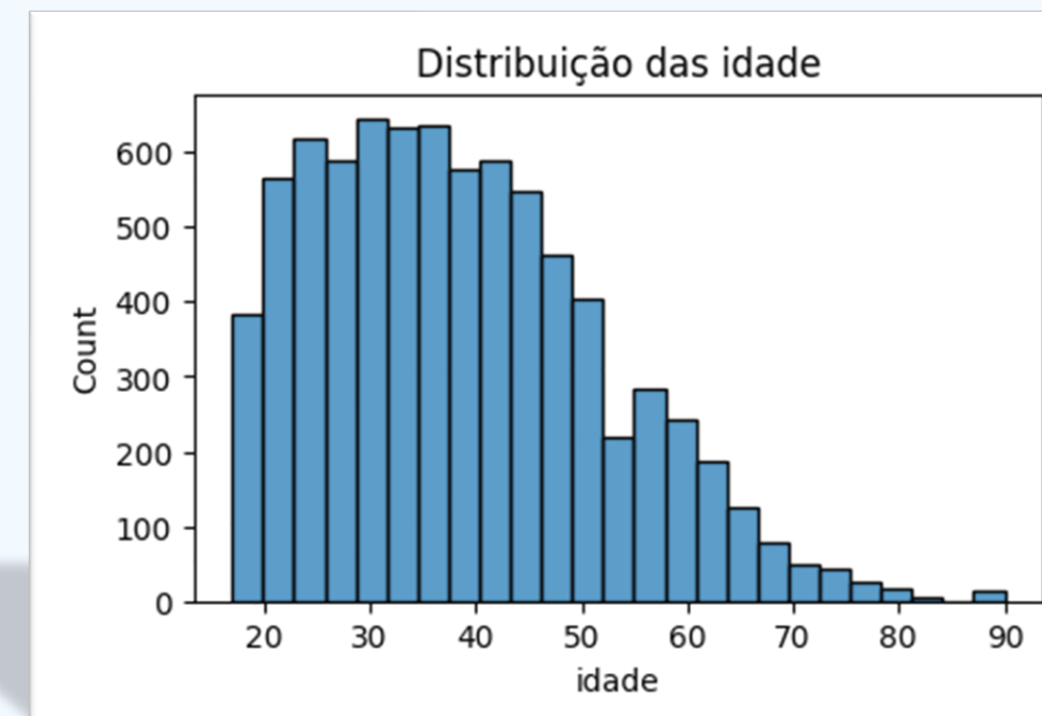


ANÁLISE DESCRITIVA

Análise Descritiva

- É a forma mais básica de análise, que busca resumir e descrever os dados disponíveis. Ela envolve a utilização de medidas estatísticas como média, mediana, moda, desvio padrão e histogramas para entender as características e padrões dos dados.

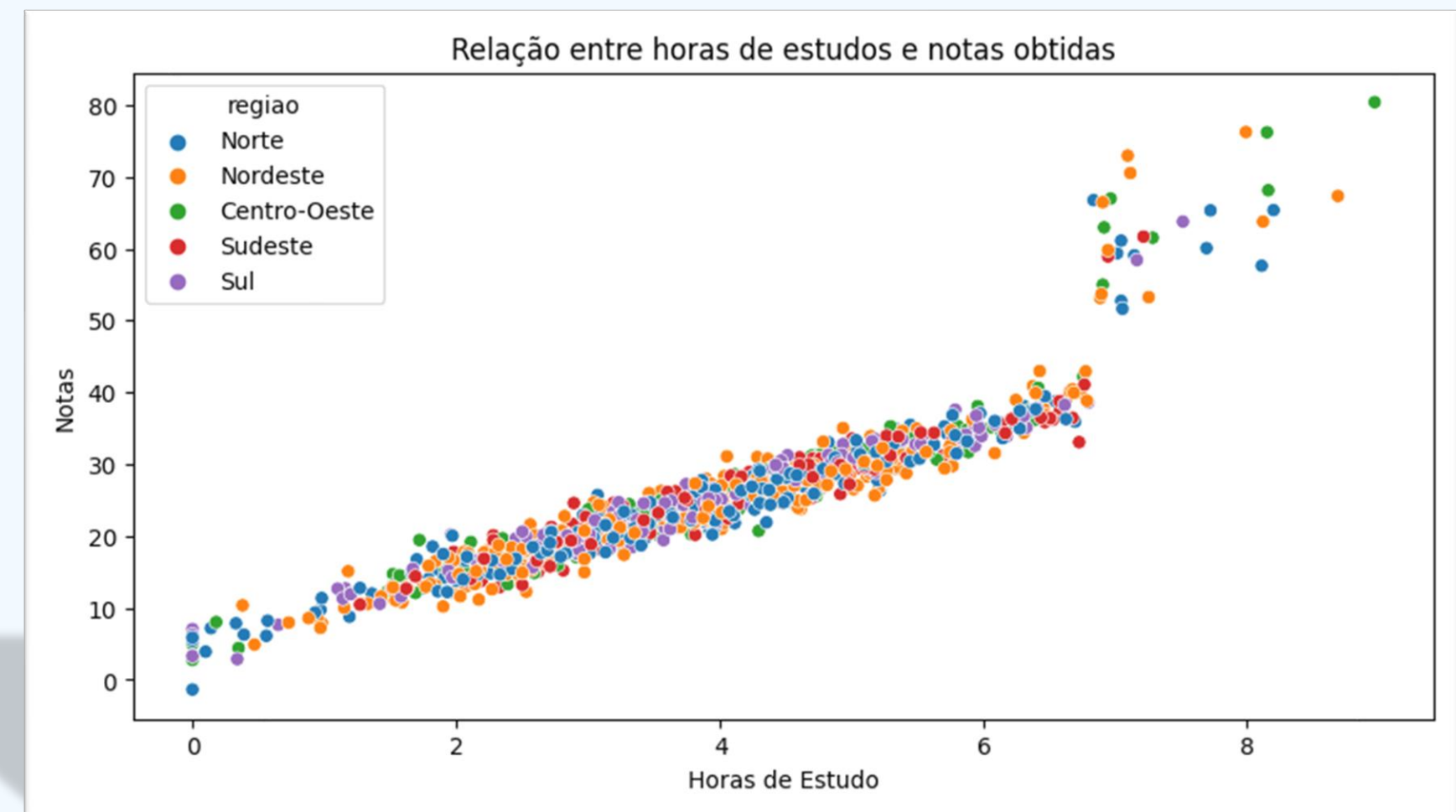
	idade	anos_estudo	qtde_filhos	salario
count	7933.00	7933.00	7931.00	7933.00
mean	38.54	10.08	2.41	3186.53
std	13.56	2.54	1.70	3062.69
min	17.00	1.00	0.00	0.00
25%	28.00	9.00	1.00	1100.00
50%	37.00	10.00	2.00	2430.00
75%	47.00	12.00	4.00	3650.00
max	90.00	16.00	5.00	19994.00



Análise Exploratória

- Visa explorar os dados de forma mais aprofundada, identificando padrões, relações e tendências ocultas.
- Isso pode ser feito através de técnicas como gráficos, análise de correlação, clusterização, análise de componentes principais e entre outros.

	Tipo de Investimento	Valor Investido
0	Ações	241009996
1	FII	125269026
2	Dólar	66010539
3	Renda Fixa	65746646



ETAPAS FUNDAMENTAIS PARA ANÁLISE DE DADOS

Etapas Fundamentais para Análise de Dados

➤ Definição do Problema/Objetivo

- Identifique claramente qual é o problema que você está tentando resolver ou qual é o objetivo da análise de dados.

➤ Coleta de Dados

- Determine quais dados são relevantes para o seu problema ou objetivo.
- Identifique as fontes de dados disponíveis e colete os dados necessários.

➤ Limpeza de Dados

- Remova dados duplicados, inconsistentes ou irrelevantes.
- Preencha dados ausentes, se necessário.
- Padronize formatos de dados, como datas e códigos.

Etapas Fundamentais para Análise de Dados

➤ Exploração de Dados

- Visualize os dados usando gráficos e tabelas para entender sua distribuição e características.
- Identifique padrões, tendências e anomalias nos dados.

➤ Preparação de Dados

- Transforme os dados conforme necessário para análises específicas.
- Crie variáveis adicionais, se úteis.
- Divida os dados em conjuntos de treinamento e teste, se aplicável.

Etapas Fundamentais para Análise de Dados

➤ Interpretação dos Resultados

- Analise os resultados da análise de dados em relação ao problema/objetivo inicial.
- Extraia conclusões e insights importantes.
- Avalie a confiabilidade dos resultados.

➤ Comunicação dos Resultados

- Apresente os resultados de forma clara e concisa, usando gráficos, tabelas e visualizações relevantes.
- Destaque os principais insights e conclusões.
- Forneça recomendações ou ações baseadas nos resultados.

Etapas Fundamentais para Análise de Dados

➤ Validação e Iteração

- Verifique se os resultados são consistentes com as expectativas e com o conhecimento do domínio.
- Se necessário, ajuste a análise de dados e repita o processo.

➤ Documentação

- Documente todas as etapas da análise de dados, incluindo os métodos utilizados, decisões tomadas e resultados obtidos.
- Mantenha a rastreabilidade dos dados e das transformações realizadas.

BIBLIOTECAS DE MANIPULAÇÃO DE DADOS

Numpy

- Fundamental para computação numérica em Python, que fornece estruturas de dados e funções eficientes para trabalhar com matrizes e arrays multidimensionais.
- Vantagens e recursos principais:
 - Computação vetorizada.
 - Operações matemáticas avançadas.
 - Manipulação de dados em larga escala.
 - Integração com outras bibliotecas.

Pandas

- Uma biblioteca poderosa para análise de dados que fornece estruturas de dados flexíveis, como DataFrame e Series, para manipular, explorar e visualizar dados de forma eficiente.
- Vantagens e recursos principais:
 - Manipulação de dados tabulares.
 - Limpeza e preparação de dados.
 - Operações de agregação e filtragem.
 - Integração com outras bibliotecas.

Matplotlib

- Uma biblioteca para criação de gráficos e visualizações em Python, que permite criar uma variedade de gráficos, desde simples até visualizações complexas e personalizadas.
- Vantagens e recursos principais:
 - Criação de gráficos estáticos e interativos
 - Personalização completa dos gráficos
 - Suporte a diversos tipos de gráficos (barras, linhas, dispersão, etc.)
 - Integração com outras bibliotecas.

Seaborn

- É uma biblioteca poderosa para análise de dados que oferece uma variedade de recursos para visualização de dados de forma eficiente e esteticamente atraente.
- Projetada para funcionar em conjunto com o Matplotlib, Seaborn simplifica o processo de criação de gráficos complexos, permitindo que os usuários criem visualizações impressionantes com apenas algumas linhas de código.
- Vantagens e recursos principais:
 - Visualizações Esteticamente Atraentes.
 - Facilidade de Uso.
 - Integração com Pandas.
 - Flexibilidade e Customização.

EXEMPLOS PRÁTICOS



Muito Obrigado!

