Os objetos do tipo sequência são como coleções versáteis que podem armazenar vários valores. Eles servem para organizar dados em uma ordem específica e são indexados por números inteiros não negativos.
O primeiro elemento da sequência é acessado pelo índice 0, o segundo, pelo índice 1, e assim por diante, até o último elemento, que está na posição n - 1, onde n representa a capacidade de armazenamento da sequência.

Operação	**Resultado**
x in s	True caso um item de s seja igual a x, senão false.
s + t	Concatenação de s e t.
n * s	Adiciona s a si mesmo n vezes.
s[i]	Acessa o valor guardado na posição i da sequência.
s[i:j]	Acessa os valores da posição i até j.
s[i:j:k]	Acessa os valores da posição i até j, com passo k.
len(s)	Comprimento de s.
min(s)	Menor valor de s.
max(s)	Maior valor de s.

s.count(x)	Número total de ocorrência de x em s.

Listas - são estruturas de dados em Python conhecidas por sua mutabilidade, o que significa que você pode adicionar ou remover elementos conforme necessário;
São estruturas indexadas, ou seja, cada elemento tem uma posição, começando em 0
Tuplas - As tuplas são estruturas de dados pertencentes ao grupo de objetos do tipo sequência em Python;
A principal distinção entre listas e tuplas é o fato de que as listas são mutáveis, permitindo a atribuição de valores a posições específicas, enquanto as tuplas são objetos imutáveis.

Você pode criar tuplas em Python de três maneiras:

1. Usando um par de parênteses para denotar uma tupla vazia: **tupla1 = ()**.
2. Usando um par de parênteses e elementos separados por vírgulas: **tupla2 = ('a', 'b', 'c')**.
3. Usando o construtor de tipo **tuple()**.

Os objetos do tipo “set” habilitam operações de conjuntos, como união, interseção, diferença e muito mais;
Essa estrutura é especialemente útil para realizar testes de associação e eliminar valores duplicados em um sequência;
Podemos adicionar um novo elemento a um conjunto usando add(valor) e remover elementos com remove(valor)

Criar objeto do tipo ser:

1. Usando par de chaves e elementos separados por vírgulas, por exemplo: set1 = {‘a’, ‘b’, ‘c’}.
2. Usando construtor de tipo ser(iterable) com um objeto iterável, como uma lista, uma tupla ou mesmo uma sequência de caracteres (string).

Chaves e valores
Dados que estabelcem um relação entre chaves e valores são conhecidas como objetos do tipo mapping;
Em python, o principal objeto que atende a essa propriedade é o dicionário, representado pelo tipo dict;
Dicionários são mutáveis, podemos modificar o valor associado a um chave existente ou criar novas chaves;
Poemos criar dicionários em Python de várias maneiras:

1. Usando um par de chaves para denotar um dicionário vazio: dicionario 1 = {}

2. Usando pares de elementos na forma “chave: valor” separados por vírgulas: dicionario2 = {‘um’: 1, ‘dois’: 2, ‘três’: 3}
3. Usando o construtor do tipo dict().

NumPy - desenvolvida para suportar a computação científica com Python;
oferece uma ampla gama de funcionalidade, incluindo arrays mutidimensionais e funções sofisticadas;
Fornece ferramentas para integração com código em C/C++ e Fortran, bem como recursos essenciais de álgebra linear, transformada de Fourier de números aleatórios;
A biblioteca NumPy é particularmente valiosa para cientistas de dados e desenvolvedores de soluções de inteligência artificial, permitindo lidar eficientemente com matrizes de dados complexas e realizar operações avançadas.

Se você estiver interessado em aprender mais sobre o NumPy, é interessante verificar a documentação completa dessa biblioteca em numpy.

Uma linguagem de programação é considerada orientada a objetos quando incorpora os princípios de abstração e suporta o uso de encapsulamento, herança e poliformismo

Classe vs Objeto	
Classe: Pessoa	**Objeto** 1: Pessoa 1
Atributos (Dados):	**Atributos** (Dados):
Nome:	**Nome**: João
Idade:	**Idade**: 30
Gênero:	**Gênero**: Masculino
Métodos (Comportamentos): **Cumprimentar**: Saúda como “Olá, meu nome é. **Aniversário**: Aumenta a idade em 1.	**Métodos** (Comportamentos): **Cumprimentar**: Saúda como “Olá, meu nome é João. **Aniversário**: Aumenta a idade em 1.

Componente principais de uma classe
Atributos: São os dados que representam o estado objeto, como nome e idade.
Métodos: definem o comportamento do objeto, sendo as ações que ele pode executar, como cumprimentar ou fazer login.
Encapsulamento: Combina atributos e métodos em uma entidade, permitindo controlar o acesso a atributos por meio de métodos.

Herança: Permite que uma classe herde atributos e métodos de outra, promovendo o reuso de código e a organização hierárquica, como na relação entre as classes pessoa, funcionário e cliente.
Polimorfismo: Refere-se à capacidade de várias classes responderem de forma diferente à mesma mensagem, graças à herança e às respostas específicas de cada classe às mensagens.

A herança é um dos pilares fundamentais do programação orientada a objetos;
Ela permite que uma classe (a classe-filha) herde características e comportamentos de outra classe (a classe-pai);
Em Python, essa técnica é amplamente suportada e flexível, permitindo que uma classe-filha herde de múltiplas classes-pai, um conceito conhecido como herança múltipla

Benefícios da herança

1. **Reutilização de código:** a herança permite que você reutilize o código existente, aproveitando a estrutura e a funcionalidade de classes-pai em suas subclasses.
2. **Extensibilidade:** você pode estender ou adicionar comportamentos específicos às classes-filhas sem modificar as classes-pai, mantendo a coesão e a organização do código.
3. **Hierarquia de classes:** é possível criar uma hierarquia de classes na qual classes-filhas podem herdar características comuns de classes-pai e, por sua vez, serem herdadas por outras classes.

Existem duas abordagens principais para organizar o código em Python: usando funções ou classes para encapsular funcionalidades e dividindo o código em vários arquivos .py para modularizar a solução;

O ideal é combinar essa técnicas, criando módulos separados em arquivos independentes;

Mas afinal, o que são módulos? São componentes de código que servem como bibliotecas ou conjuntos de funções em Python

Em python, frequentemente ouvimos falar tanto de módulos quanto de bibliotecas;

A relação entre eles é que, na prática, um módulo pode ser considerado uma biblioteca de códigos

Tipos de módulos:

Módulos built-in: embutidos no interpretador;

--------math;os;svs;Random;datetime;re;collections-------

Módulos de terceiros: cridos por terceiros e disponibilizados via PyPI.

São criados e mantidos por desenvolvedores externos à comunidade oficial do Python;

Ampliam a funcionalidade de Python em diversas áreas;

Instalação é feita usando o gerenciador de pacotes padrão, pip. Exemplo: pip install requests;

Gerenciador dependências é essencial à medidas que projetos crescem; o uso de um arquivo requirements.txt facilita a instalação de todas as dependências em um único comando pip;

Conhecer as licenças dos módulos de terceiros é importante, pois eles podem varias de código aberto a proprietário, e a qualidade da manutenção pode variar;

Frequentemente possuem comunidades ativas de desenvolvedores e documentação rica.

Confira, a seguir, alguns pontos importantes sobre módulos de terceiros:

1. Ampliam a funcionalidade do Python em diversas áreas, como na manipulação de dados, gráficos, interfaces gráficas, integração com bancos de dados e aprendizado de máquina.
2. A instalação é feita usando o gerenciador de pacotes padrão: pip. Exemplo: **pip install requests**.
3. Gerenciar dependências é essencial à medida que projetos crescem. O uso de um arquivo requirements.txt facilita a instalação de todas as dependências em um único comando pip.
4. Ambientes virtuais isolam projetos Python para evitar conflitos entre diferentes versões de módulos de terceiros.
5. Conhecer as licenças dos módulos de terceiros é importante, pois eles podem variar de código aberto a proprietário, e a qualidade da manutenção pode mudar.
6. Módulos de terceiros geralmente possuem comunidades ativas de desenvolvedores e documentação rica, fornecendo suporte e recursos valiosos.

Módulos próprios: criados pelo desenvolvedor

O Matplotlib é uma das bibliotecas de visualização mais populares em Python;

Oferece uma ampla gama de recursos para criar gráficos e visualizações de dados de maneira flexível e personalizável;

Frequentemente usado para criar gráficos estáticos, gráficos interativos e até mesmo animações;

1. Para aprender mais detalhes sobre aplicações do Python, especialmente quanto ao Matplotlib, sugiro a leitura do artigo *Introdução à estilometria com Python*. Neste texto, apresentam-se análises estilométricas, que dizem respeito ao estudo quantitativo do estilo literário por meio de métodos de leitura distante computacional. Para acessar o material sugerido, clique no link a seguir.

O Pandas é uma das principais bibliotecas do Python e é considerada a ferramenta principal para análise de dados.

Principais recursos do Pandas:

- Ler, manipular, agregar e plotar dados
- Resumir estatísticas como média, soma, contagem, máximo e mínimo
- Carregar dados de diferentes bases de dados e formatos de dados
- Integrar conjuntos de dados de forma intuitiva
- Segmentar registros dentro de um DataFrame
- Definir suas próprias funções do Python

O nome Pandas deriva do termo “dados de painel” (panel data), um termo econométrico utilizado para se referir a conjuntos de dados estruturados multidimensionais.