UFCD 10793_04_AG – Fundamentos Python

João Araújo

14-03-2024

PRR Plano de Recuperação e Resiliência
REPÚBLICA PORTUGUESA
Financiado pela União Europeia NextGenerationEU

Módulos ou bibliotecas

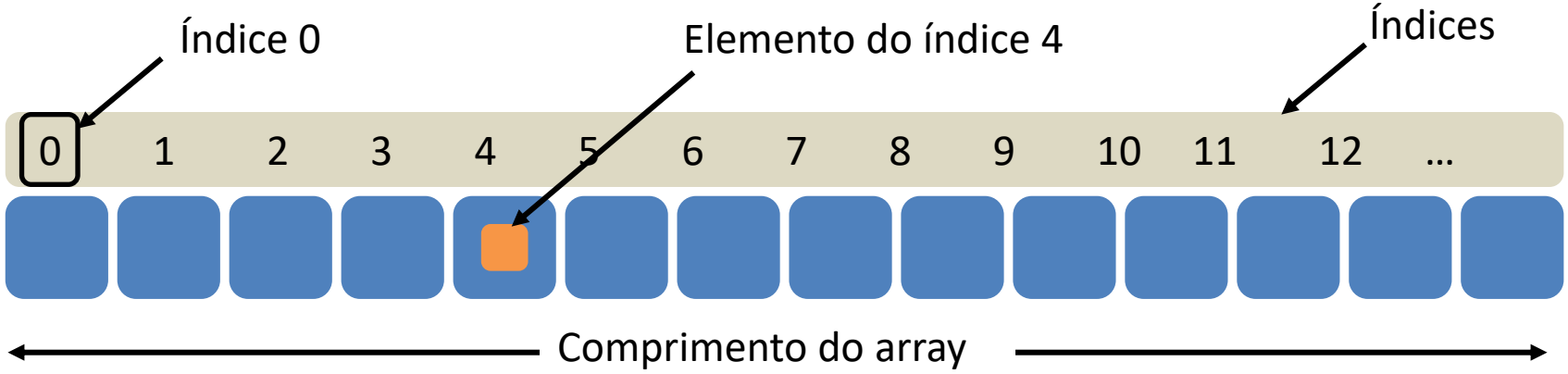
- Módulos ou bibliotecas são ficheiros que contém conjuntos de funções que podem ser usados no programa Python
- Importação
  - Utiliza-se o comando *import* para importar bibliotecas para o programa Python.
    - *import library*
  - As bibliotecas têm que estar instaladas no ambiente de trabalho e, consequentemente, no interpretador *Python*
  - É recomendado usar o comando *as* para identificar mais facilmente a biblioteca que se está a usar (ALIAS)
    - *import library as lb*

NumPy

- Biblioteca de Álgebra Linear do Python
- Fundamental para a análise de dados e Machine Learning
- Operações lógicas e matemáticas usando Matrizes e *Arrays*
- Manipular grandes quantidades de informação
- https://numpy.org/

Arrays

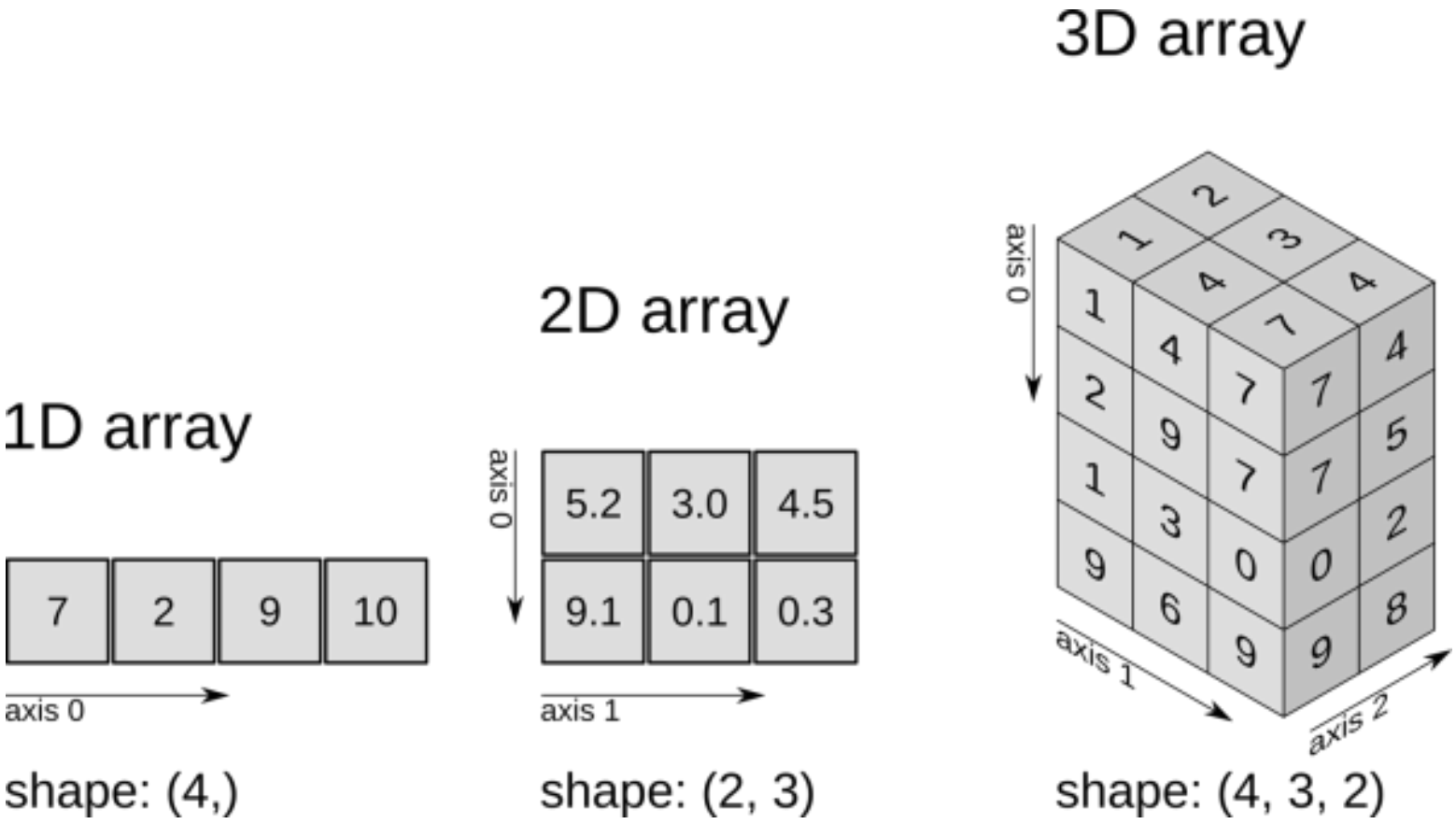
- *Arrays* são variáveis que podem conter mais do que um elemento



- Por defeito, o Python possui o tipo de dado *List,* que tem a mesma filosofia de *Arrays,* sendo possivel realizar algumas operações lógicas e matemáticas

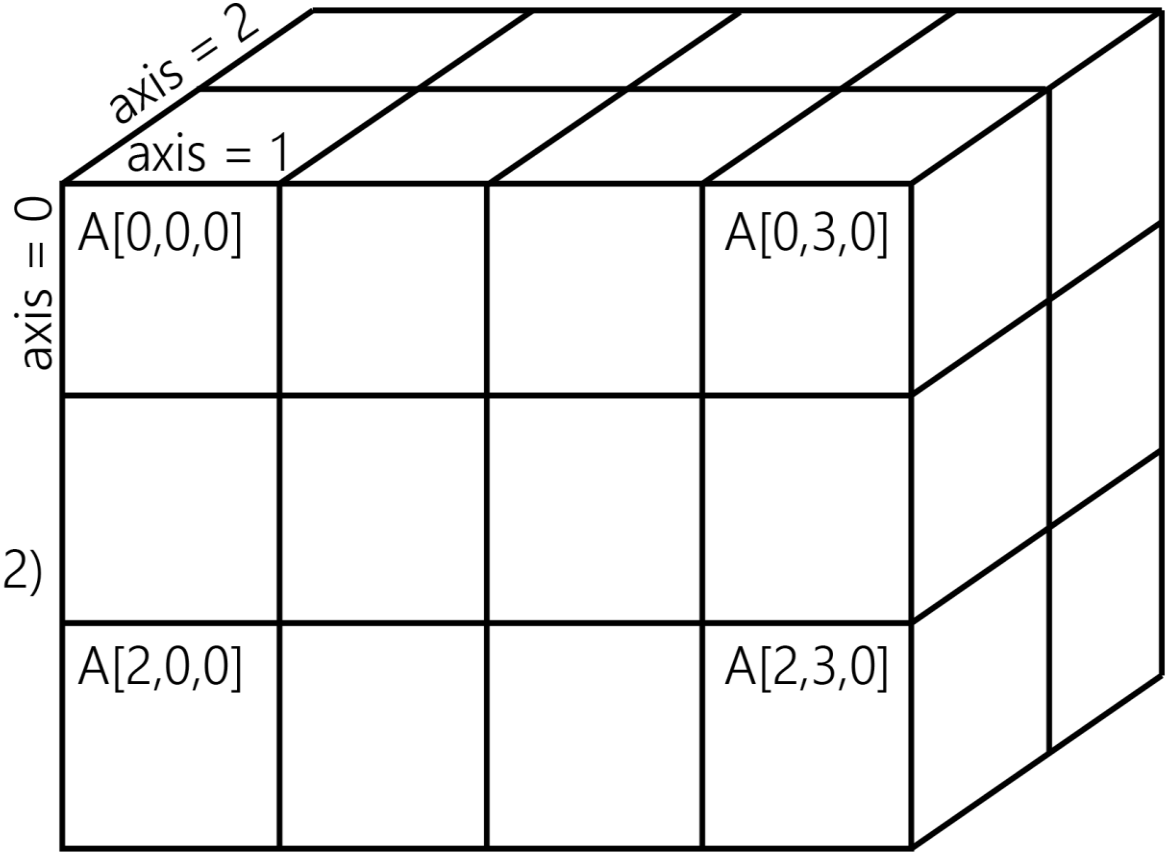
- Serve de base para o *Numpy*

Arrays


1D array
7
2
9
10
axis 0
shape: (4,)
2D array
axis 0
5.2
3.0
4.5
9.1
0.1
0.3
axis 1
shape: (2, 3)
3D array
axis 0
axis 1
axis 2
shape: (4, 3, 2)

Arrays

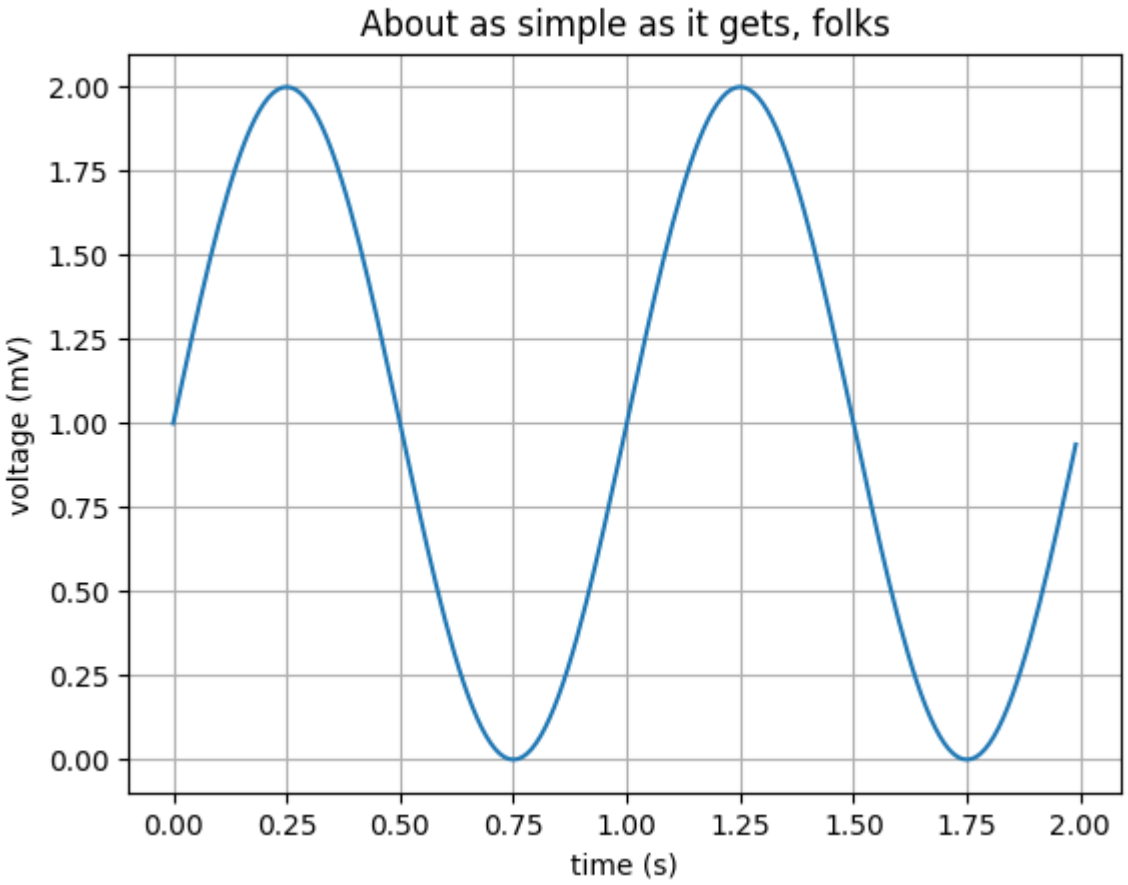
ndarray
ndim = 3
shape = (3, 4, 2)



Matplotlib

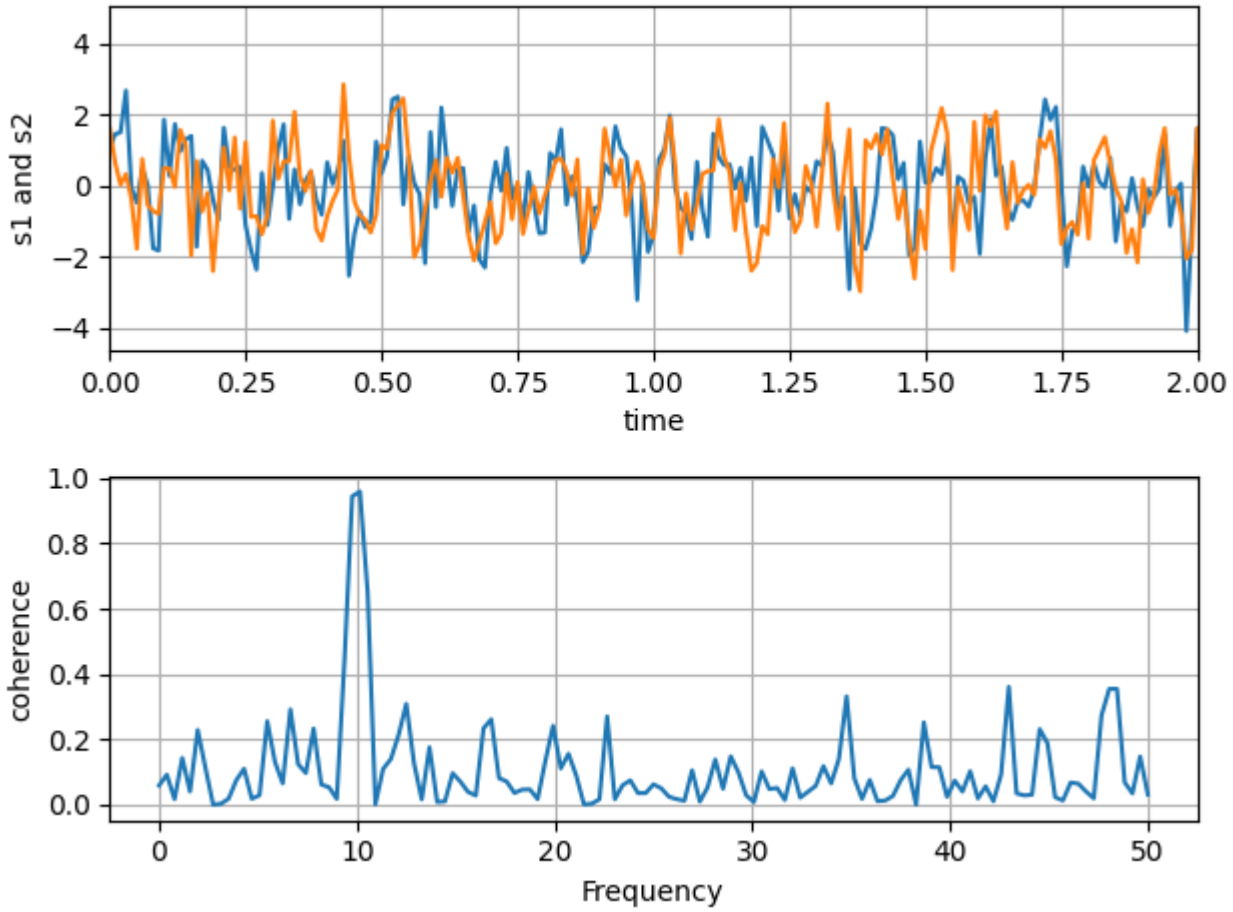
- Biblioteca para geração de gráficos
- Publicação de figuras de alta qualidade em vários formatos
- Desenho da informação a várias dimensões: 2D, 2,5D e 3D
- Gráficos de dispersão, barras e polares, histogramas, etc…
- Motor gráfico de outras bibliotecas de Python (Ex: Pandas
- https://matplotlib.org/

Matplotlib



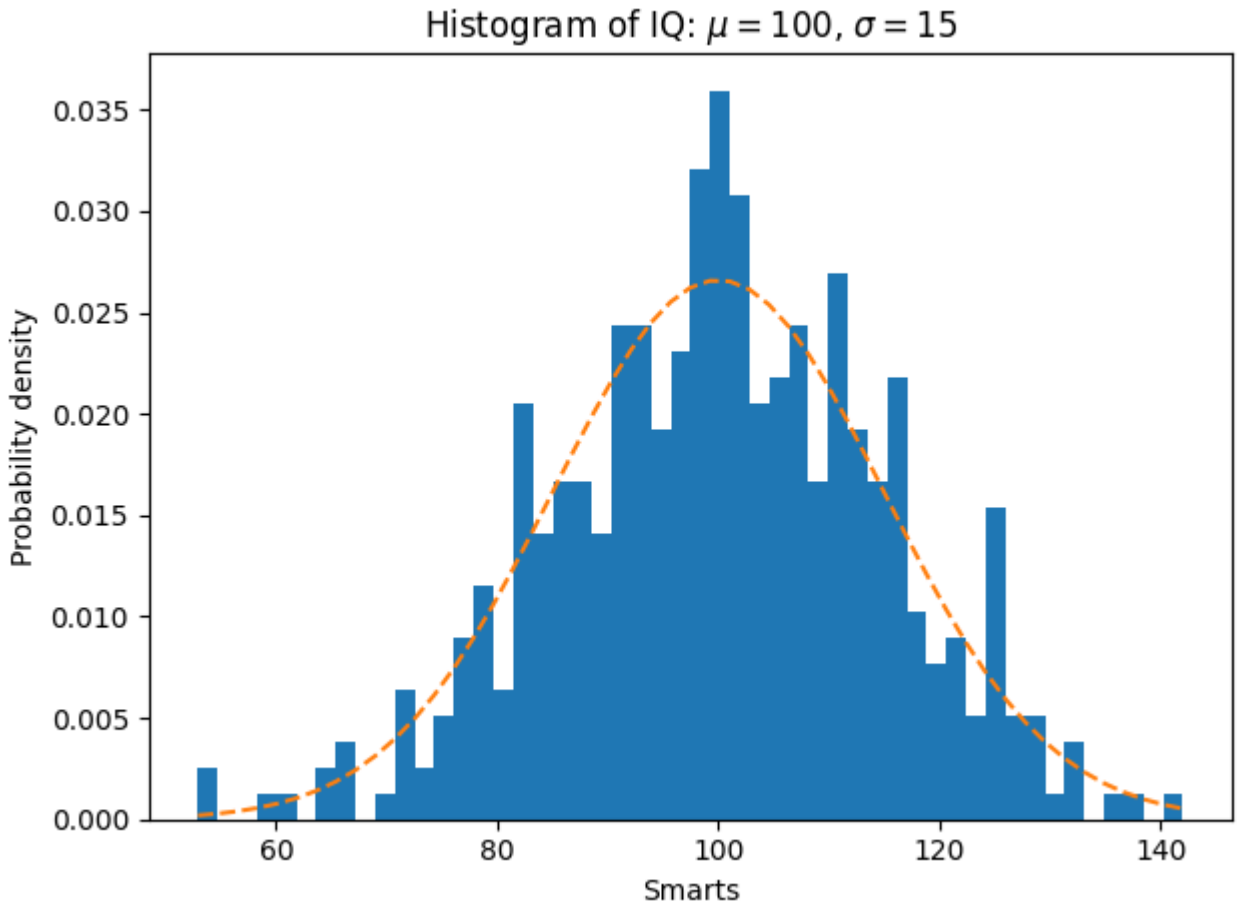
matplotlib

Matplotlib



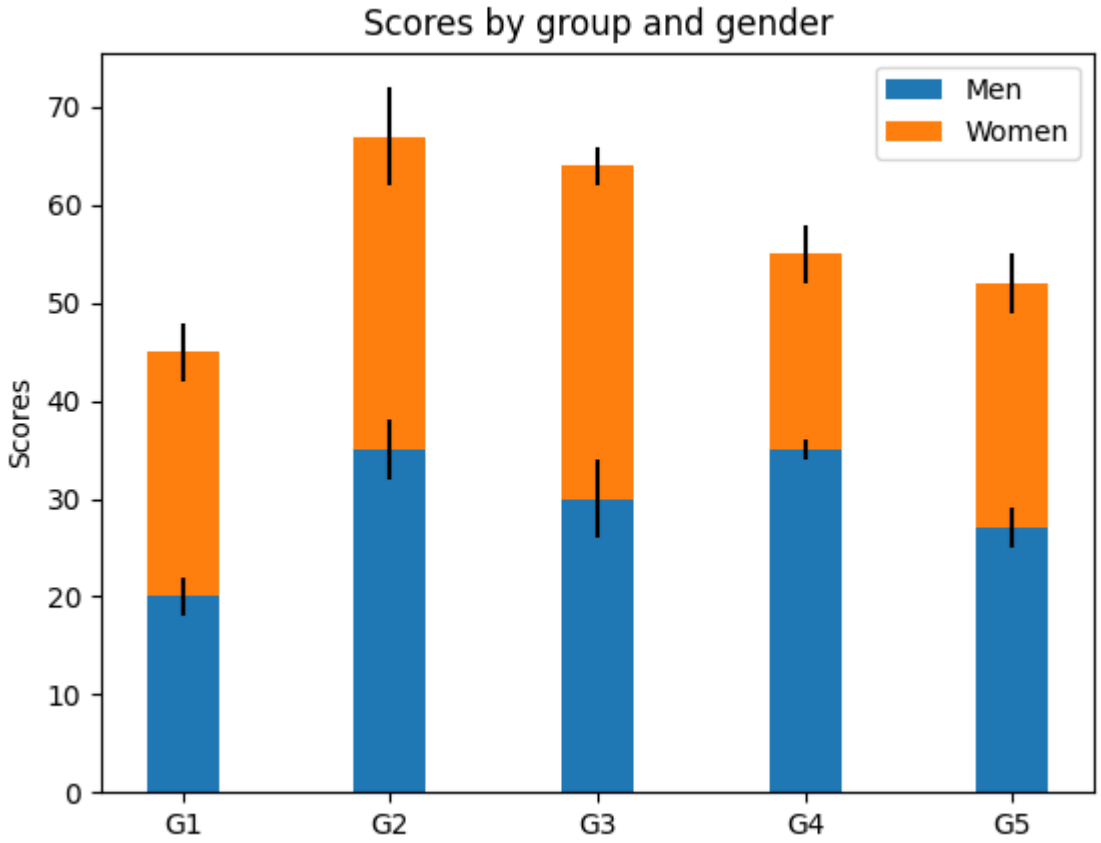
matplotlib

Matplotlib



matplotlib

Matplotlib



matplotlib

Classes e objetos

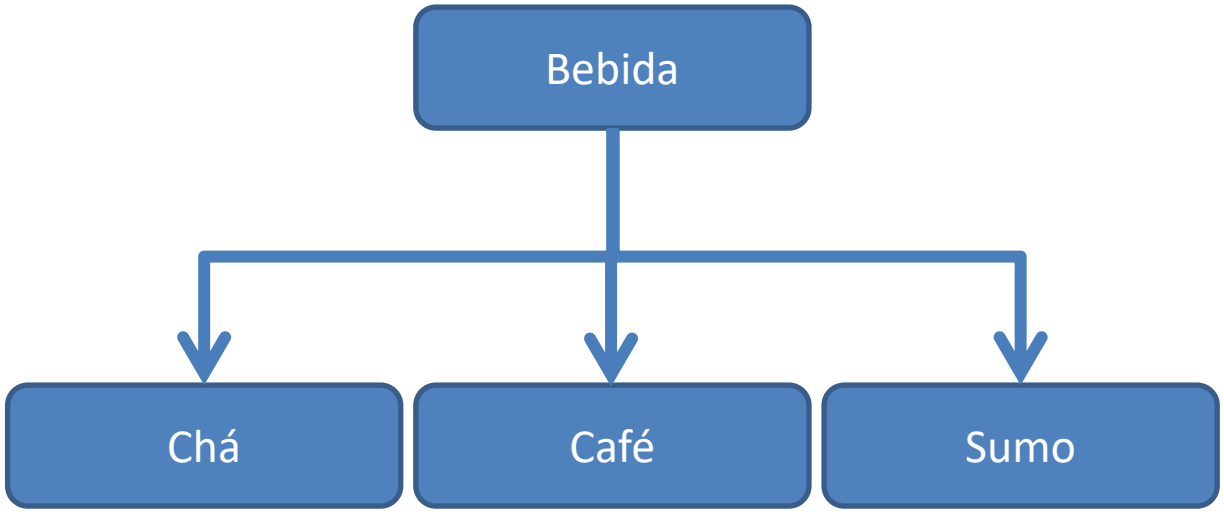
- Essenciais na Programação Orientada a Objetos(*Object Oriented Programming)*

- Em *Python* tudo se comporta como um objeto, desde às suas propriedades aos seus métodos

- Definição de classe e objeto

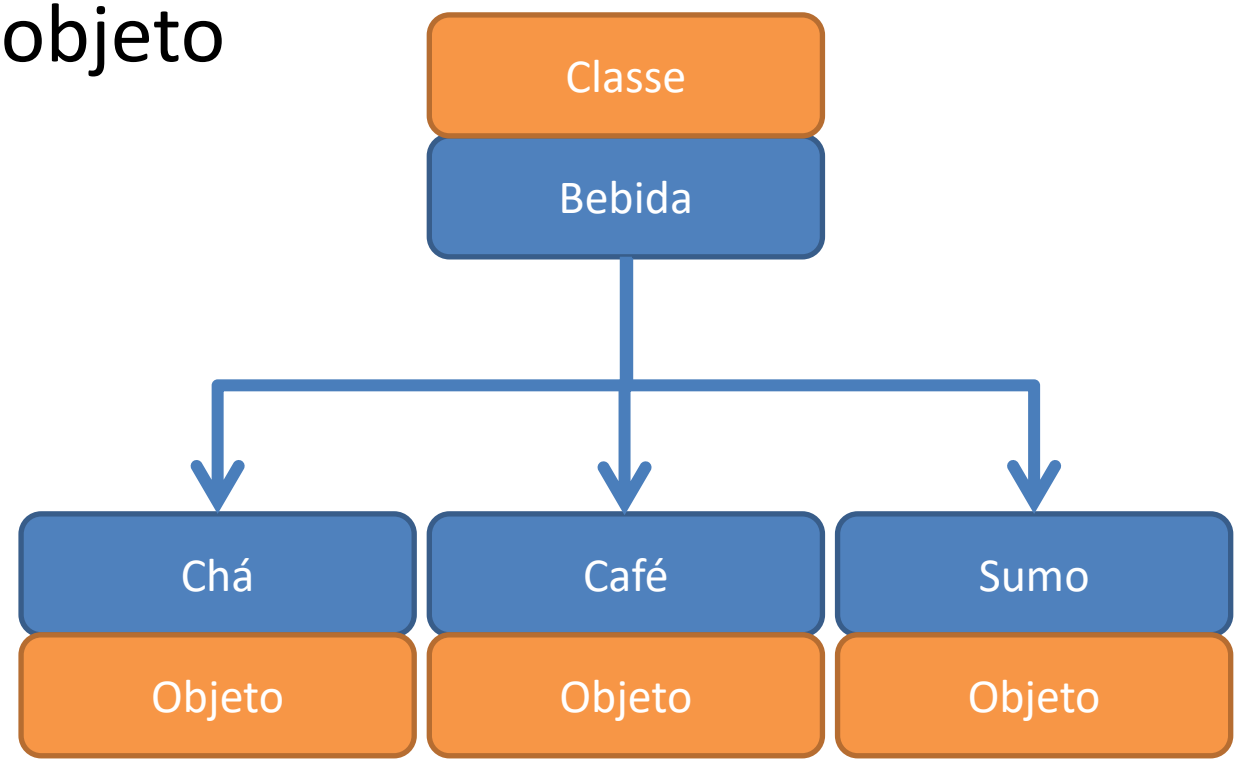
Classes e objetos

- Exemplo na vida social
  - O João vai a um café e pede uma bebida.
  - O funcionário pergunta que bebida o João quer



Classes e objetos

- Neste exemplo, uma bebida é considerada uma classe, e o tipo de bebida é considerado um objeto

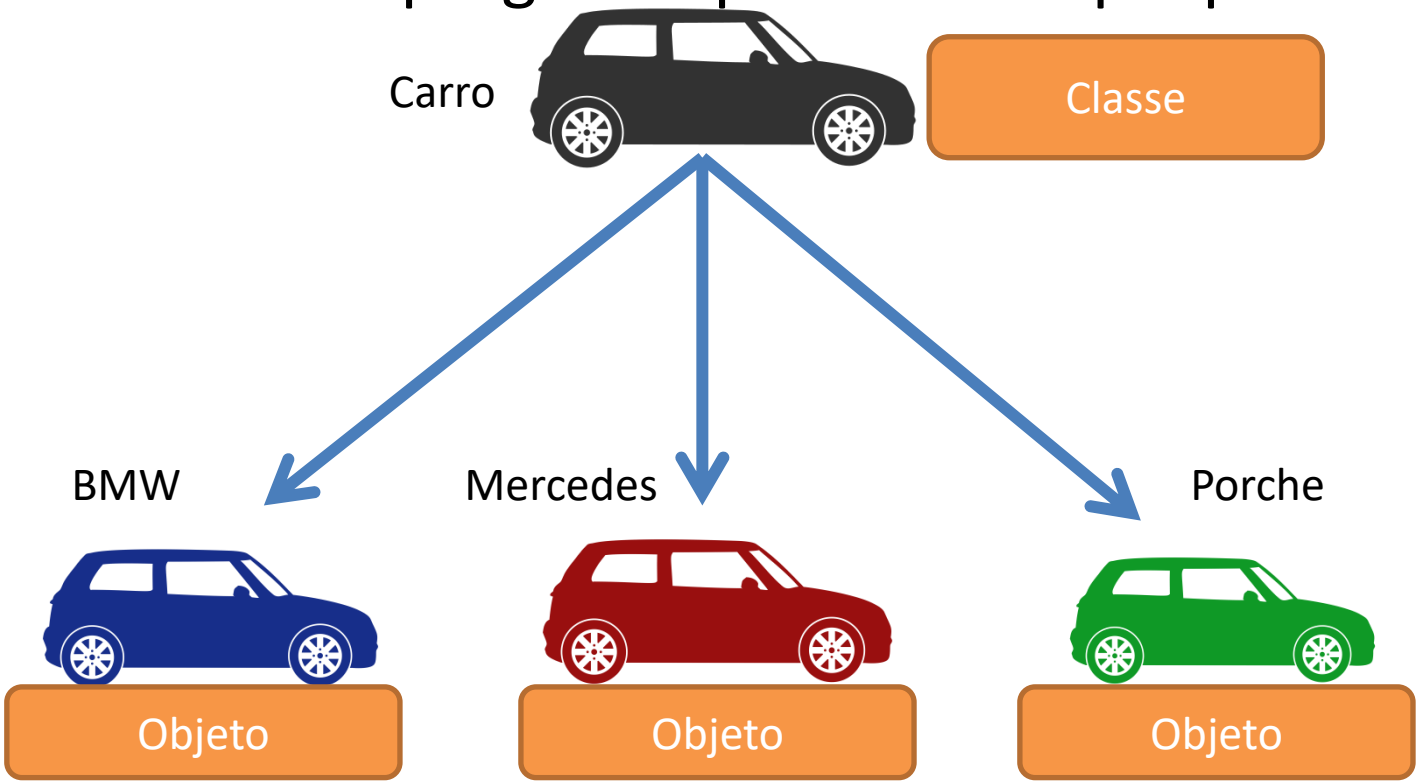


Classes e objetos

- Por definição, classe é uma entidade lógica que faz a estruturação de um objeto
  - Atributos (variáveis)
  - Métodos (Funções)
- Por outro lado, um objeto é uma coleção de atributos e métodos, estruturadas na classe que está associado.

Classes e objetos

- Outro exemplo da vida social
  - A Maria vai a um stand de automóveis para comprar um carro
  - O funcionário pergunta qual o carro que pretende



Classes e objetos: Definição de atributos e objetos

- Criar uma classe
  - class **Bebida**:
    - pass
- Criar um objeto
  - cha = **Bebida()**
  - cafe = **Bebida()**
  - sumo = **Bebida()**
- Criar atributos na classe bebida
  - class **Bebida**:
    - preco = 0
- Os objetos agora podem definir os atributos
  - cha.preco = 1.00
  - cafe.preco = 0.60
  - sumo.preco = 2.00
- Os valores são exclusivos a cada objeto criado

Classes e objetos: Definição de atributos e objetos

Construtores

- No exemplo anterior, se pretendermos inserir 10 atributos, teria que ser:
  - class **Bebidas**:
    - var1 = ...
    - var2 = ...
    - ...
    - var10 = ...
  - cha = **Bebidas()**
  - cha.var1 = ...
  - cha.var2 = ...
  - ...
  - cha.var10 = ...
- Processo trabalhoso com muitas linhas de código necessárias

Classes e objetos

- Seria mais simples definir o valor dos atributos aquando da criação do objeto
- Usar a função **__init__()** que é o **construtor** da classe. É uma função embutida que é executada sempre que a classe é inicializada,
  - class Bebidas:
    - def __init__(self, preco, cor):
      - self.preco = preco
      - self.cor = cor
    - cola = Bebidas(1.00,'preto')
    - batido = Bebidas(2.00,'branco')
- **self** – parametro que é uma referência à instância atual da classe, e é usado para aceder a todas as variáveis dentro da classe.
  - Nota: não é necessário ter o nome de self, mas necessita de ser o primeiro parâmetro da função __init__()
- preço, cor,... – numero de atributos que desejamos ser inicializados
- Cada classe tem que possuir um construtor.

Classes e objetos: Diagrama UML

- *Unified Modeling Language* (UML) é a linguagem standard para específicar visualizar, construir e documentar os modelos de um sistema de sofware

- Muito util para modelar o software e observar o seu comportamento, seguindo os procedimentos e padrões da OMG (*Object Managment Group*)

Classes e objetos

- No diagrama UML, por norma existem:
- Classes
- Subclasses
- Visibilidade de atributos. Embora em Python, considera-se sempre este elementos publicos
- Relações entre classes
- http://www.lucidchart.com

Classes e objetos: Relações entre classes

- Para criar uma classe principal, deve-se aplicar o princípio da abstração – uma classe que representa algo de forma generalista. O objetivo é esconder dados irrelevantes para a utilização do programa, e para o tornar menos complexo e eficiente.

- Herança – Existe uma classe e uma ou mais subclasses. As subclasses têm acesso aos métodos e atributos
  - Existe sempre uma **classe Pai ou Super Classe** e **uma ou mais classes Filho ou subclasses**
  - Função __init__ - inicializa os atributos da classe
  - Por defeito, as classes fiflho só têm acesso aos atributos da classe pai
  - Atrravés do método **super()** , tem-se acesso aos atributos e métodos da classe

- Associação – Associação entre classes. O Python usa dois tipos de associações:

  - Agregação: Associação entre classes em que o acesso é independente, ou seja, a existência de uma não depende da outra(s)
  - Composição: Associação entre classes em que o acesso é dependente, ou seja, a existência de uma depende da outra(s)