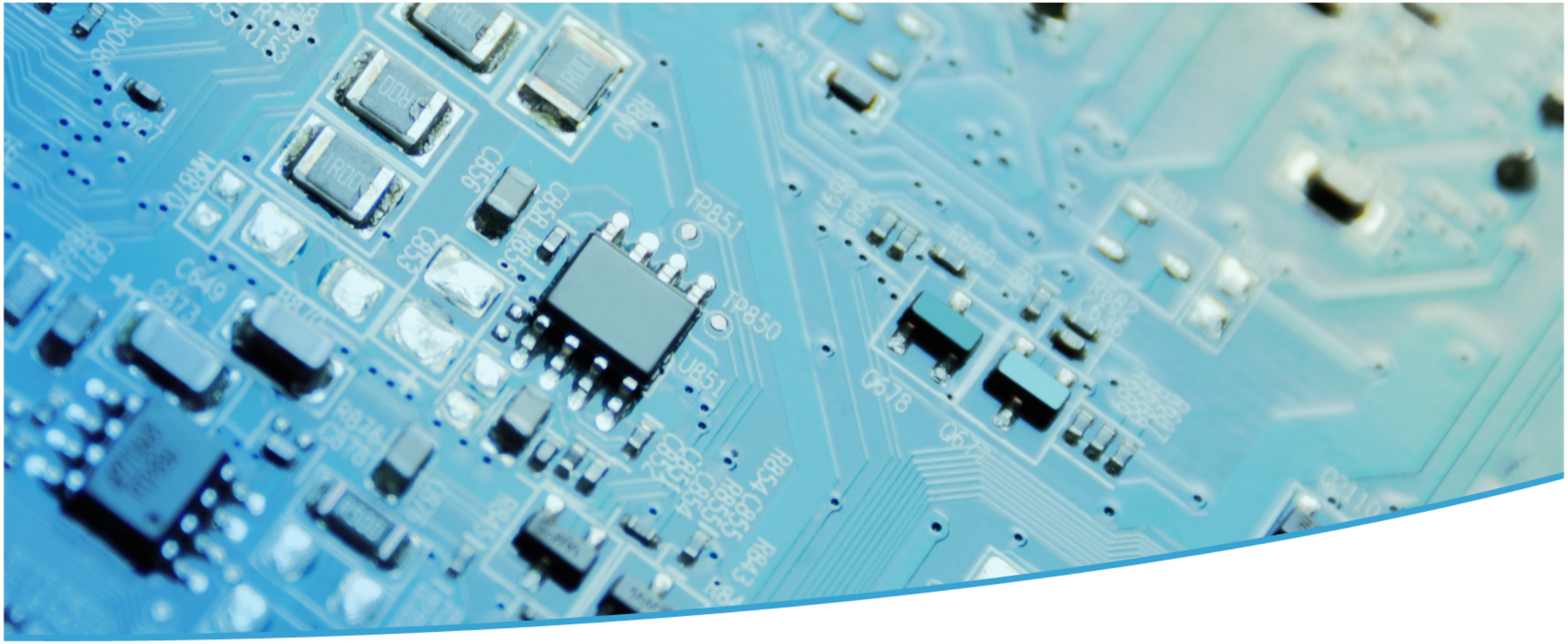
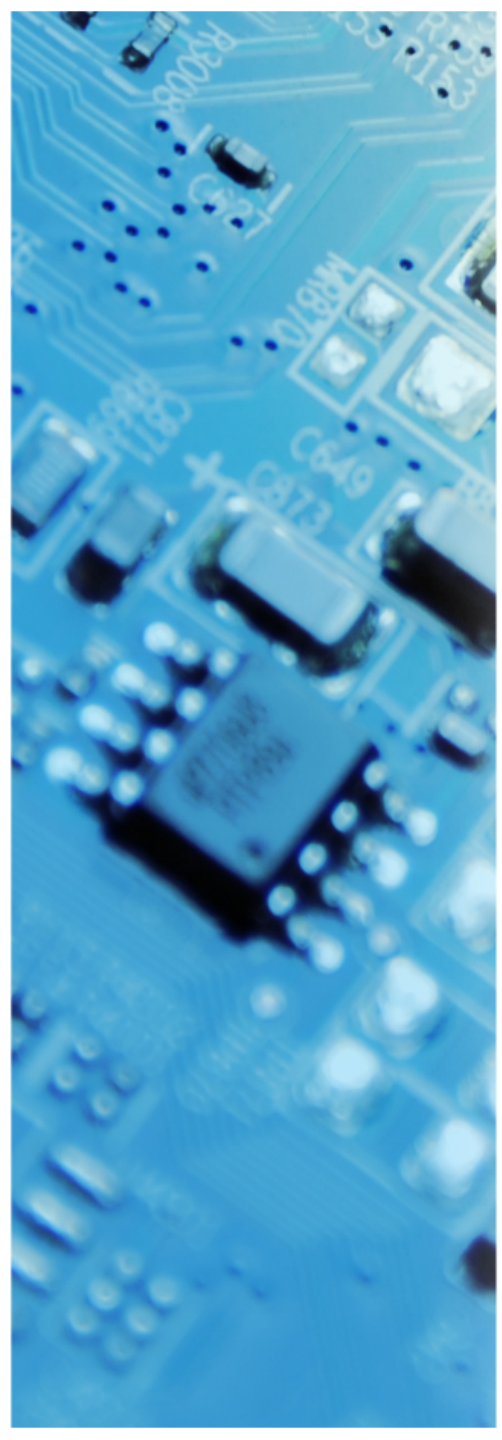
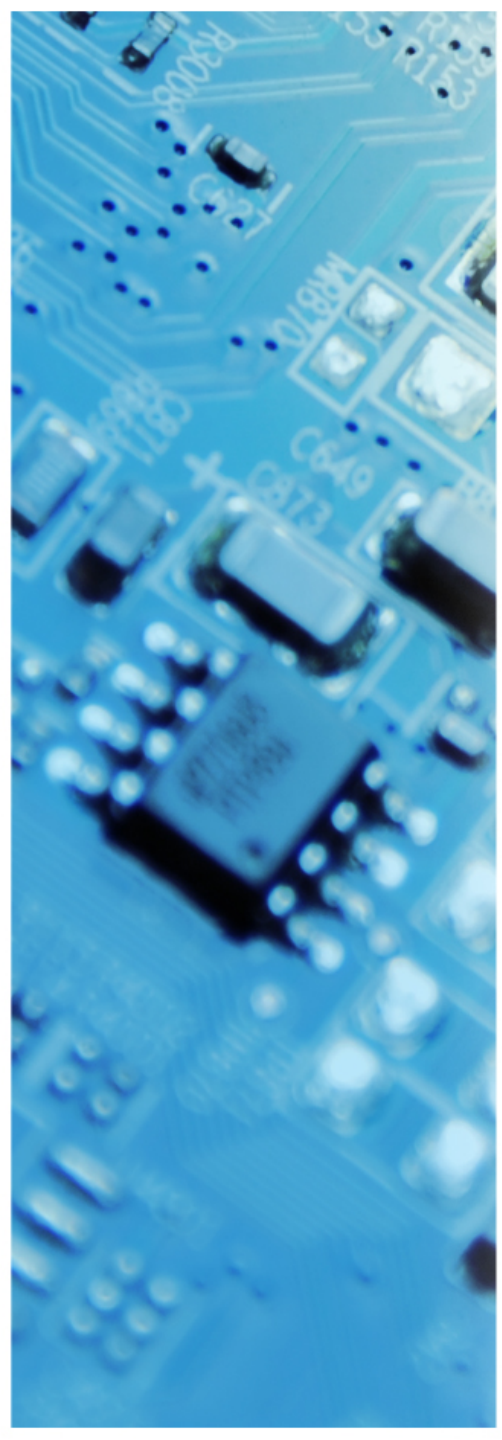


# Orientação a objetos em Python
# Classes, métodos e objetos



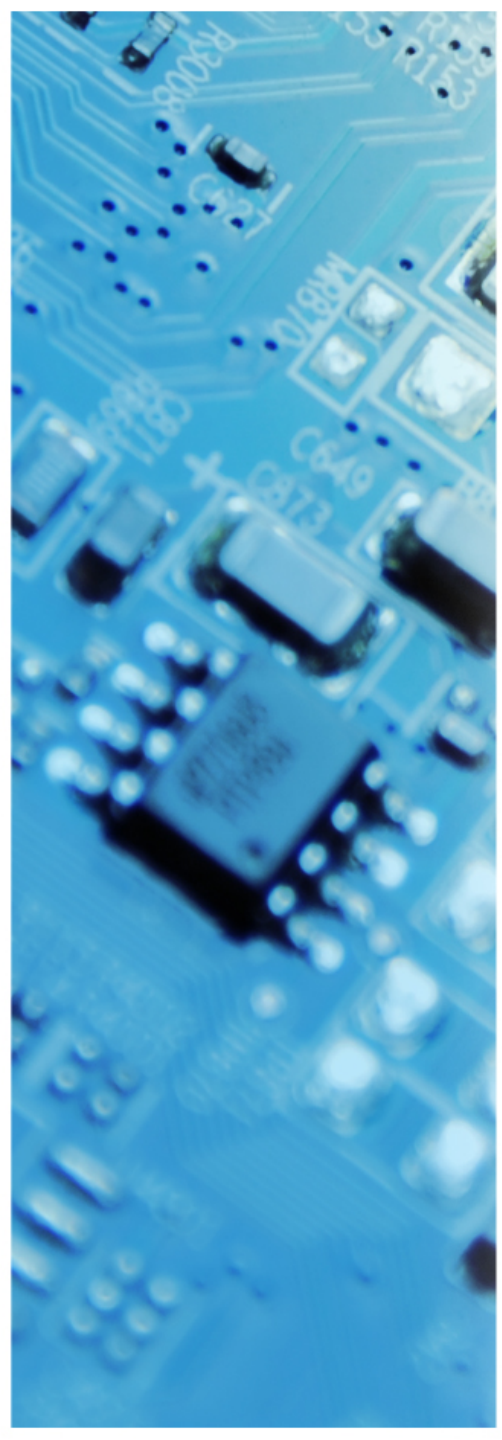
# OO - Python

- Além das funções e demais recursos, as classes são unidades de construção de aplicações disponíveis em Python.
- Classes definidas em Python seguem a maioria dos conceitos de orientação a objetos.
- A programação orientada a objetos em python é derivada do chamado modelo de dados pythônico.
- O uso de classes e objetos é algo natural, uma vez que os tipos de dados básicos são objetos.



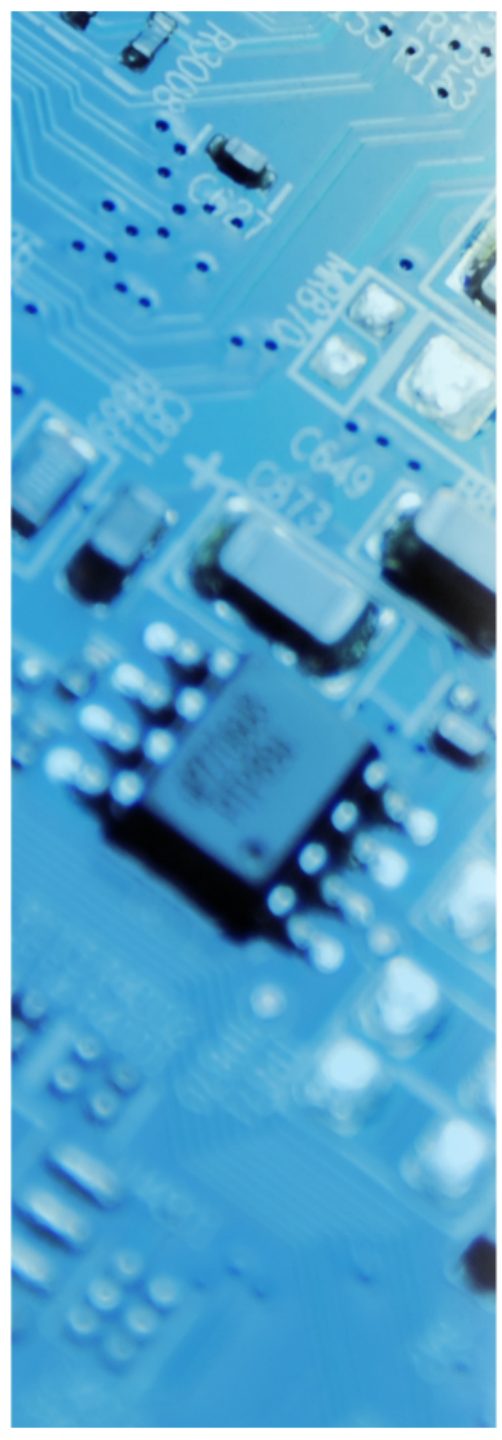
# Classes e objetos

- Conceitos OO em python:
  - Classes combinam dados (atributos) e comportamento (métodos)
  - Um objeto é uma instância de uma classe.
  - Atributos de classe são variáveis de instância em objetos.
  - Método é uma função definida em uma classe.
  - Mensagens são trocadas entre objetos.



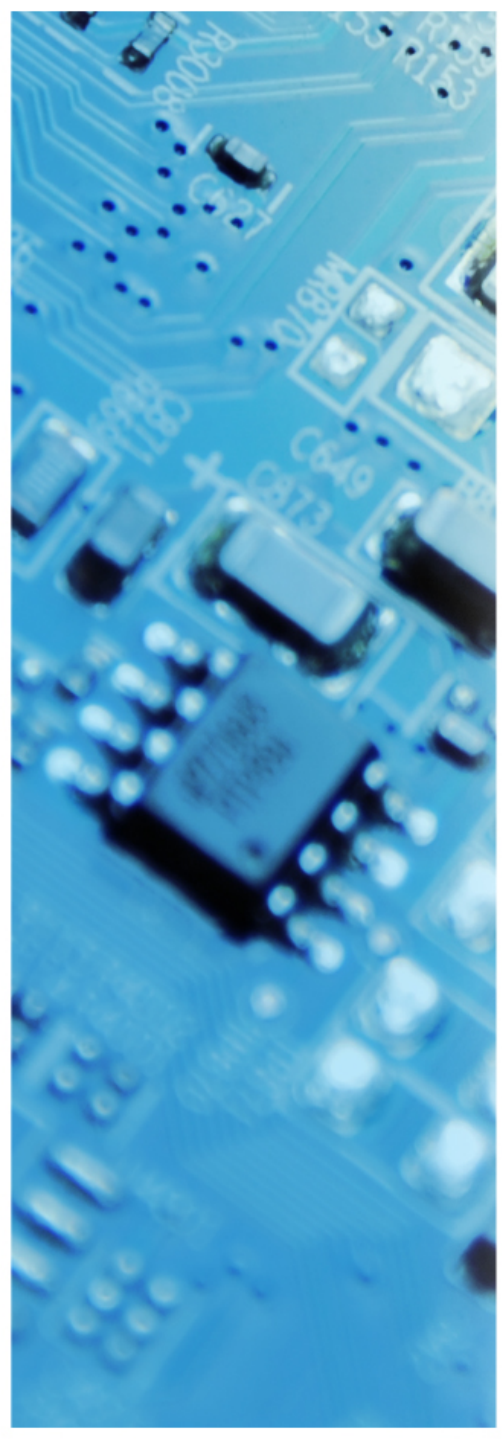
# Classes e objetos

- Classes na linguagem python podem ser usadas para:
  - Um template para criar instâncias (objetos) com as mesmas características.
  - Definir métodos que padronizam o comportamento de objetos.
  - Definir atributos que armazenam dados de objetos que possuem a mesma estrutura.
  - Padronizar o envio de mensagens para objetos da classe.



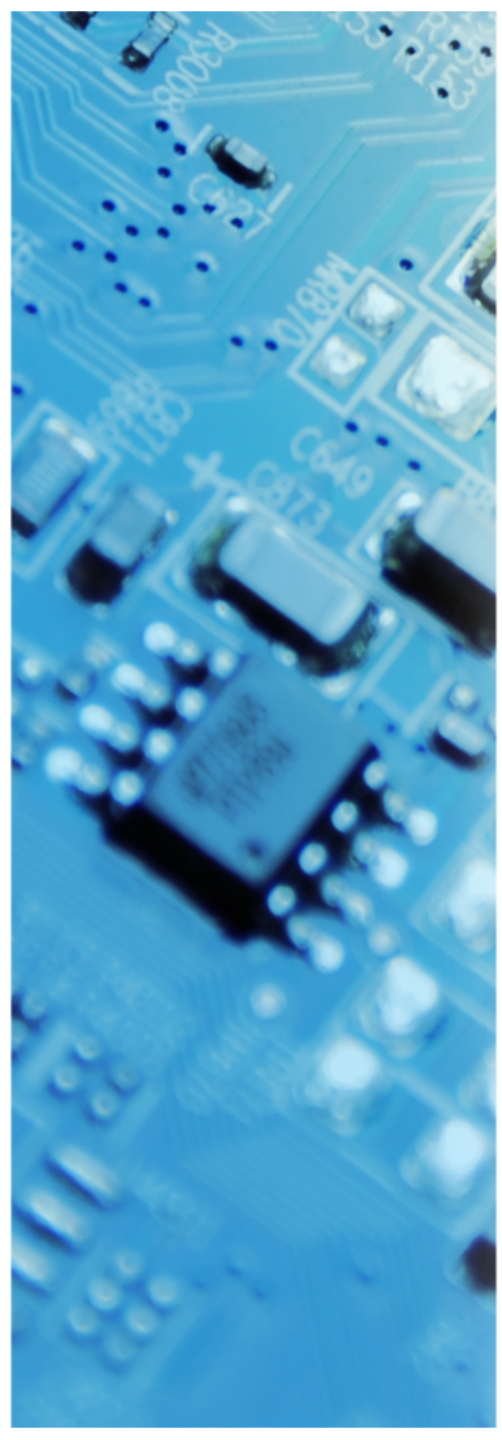
# Classes e objetos

- As seguintes orientações podem ser usadas para definir o escopo de classes:
  - A descrição da classe é simples e clara?
  - Objetos e instâncias podem ser utilizados com facilidade e em diferentes aplicações?
  - Onde os atributos da classe são utilizados?
  - Será possível alterar o valor de algum atributo diretamente na instância do objeto ou somente usando um método?



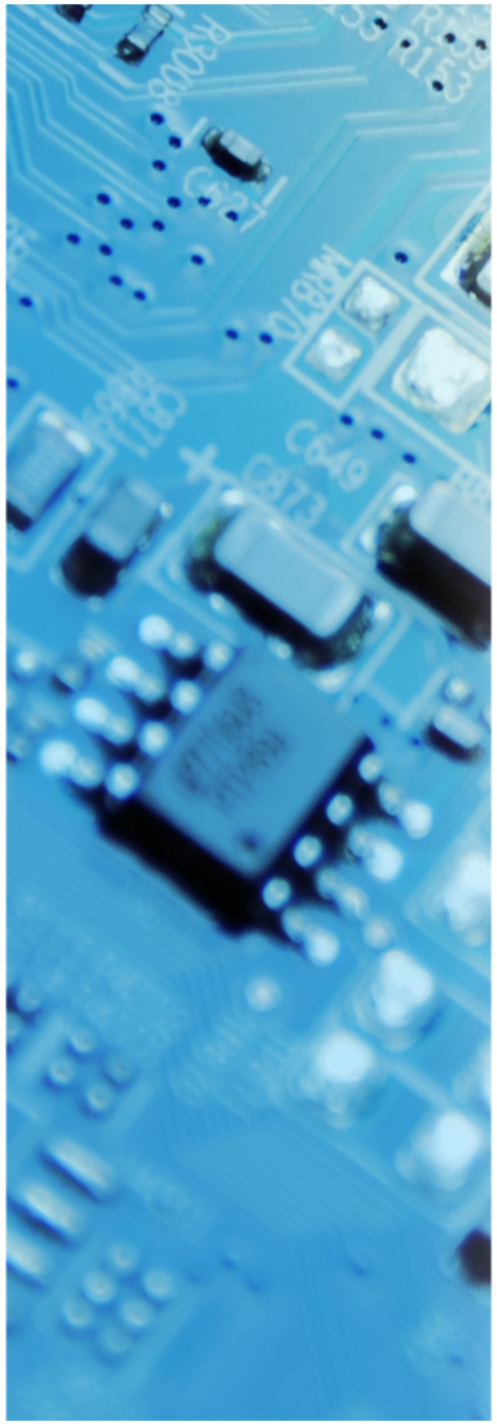
# Atributos

- Classes são definidas em python usando a palavra reservada Class.
- Atributos geralmente são criados no método construtor (__init__).
- Atributos não possuem um tipo definido, portanto, podem armazenar diferentes valores ao longo da vida de um objeto.
- Por padrão um atributo é visível e pode ser alterado diretamente no objeto, por atribuição.
- Entretanto, é possível proteger um atributo usando notação __atr



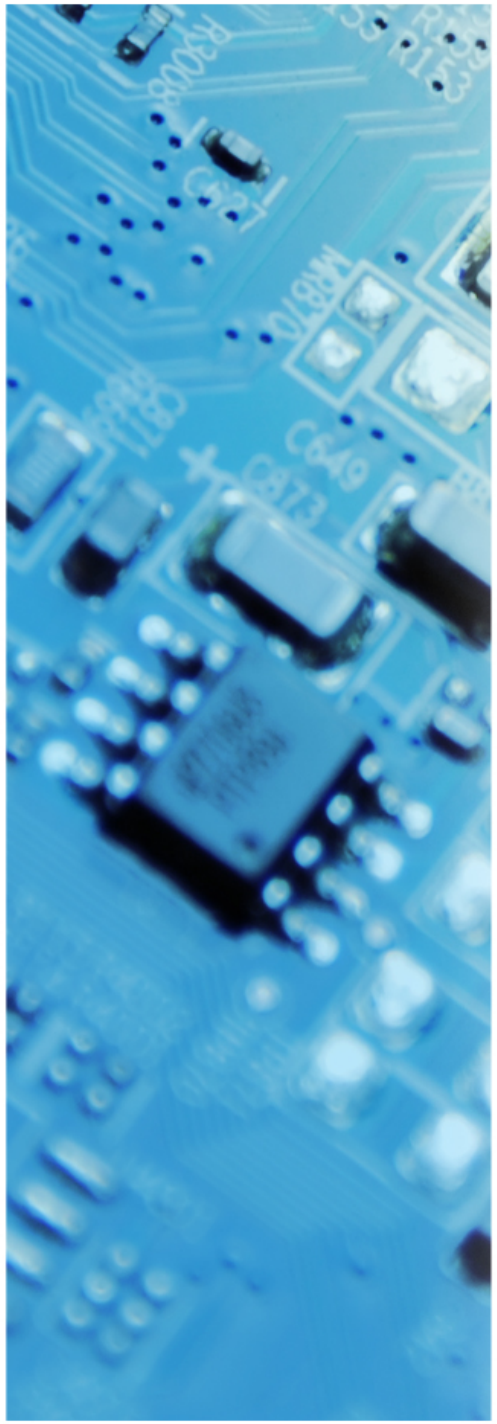
# Métodos

- Cada classe pode possuir diversos métodos que são definidos pela instrução def ..
- Assim como funções, um método pode retornar valores ou não.
- Existem os métodos especiais que possuem underline duplo antes e depois do nome do método.
- Métodos recebem como parâmetro inicial o próprio objeto (self).
- Todas as referências para propriedades devem incluir self em frente ao nome da propriedade.



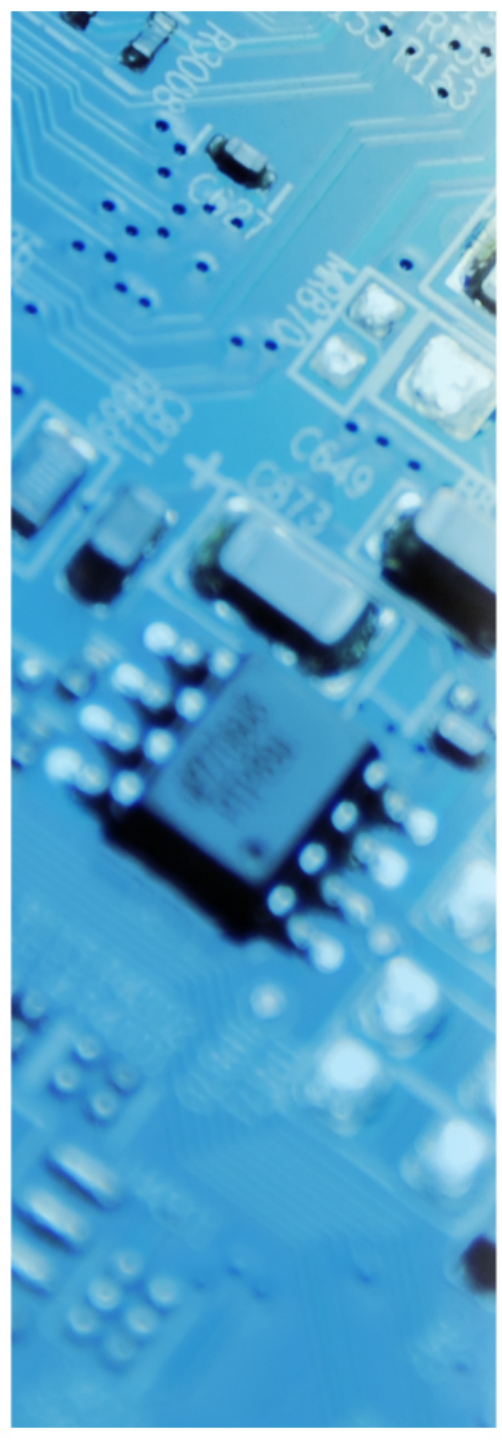
# Métodos

- A implementação de um método especial segue o modelo pythônico.
- As classes implementadas passam a responder da mesma forma que classes nativas da linguagem.
- A implementação de alguns métodos especiais, como repr, str ou get é considerada uma boa prática.
- Existem também propriedades especiais que qualquer classe possui (__name).
- Em algumas situações métodos especiais retornam valores de propriedades especiais.



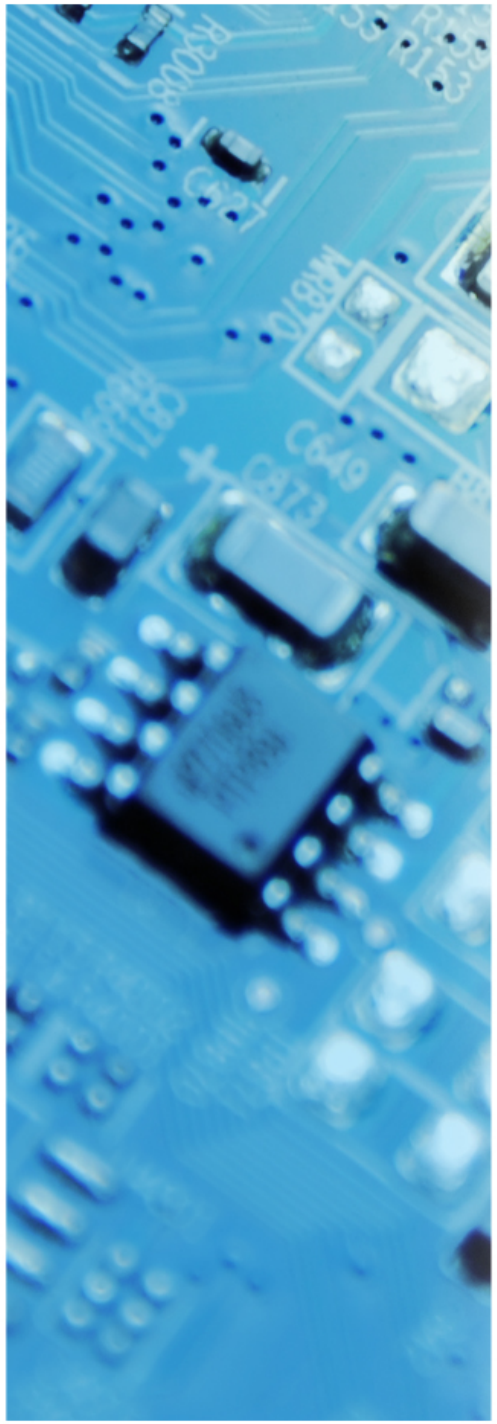
# Encapsulamento

- É possível criar atributos protegidos e métodos para garantir o encapsulamento.
- Alguns métodos podem ser definidos como propriedades (@property) para que sejam responsáveis por retornar o valor do objeto.
- Além disso o uso de @atr.setter caracteriza o método como o responsável por alterar o valor do objeto.
- Com o uso destas propriedades é possível acessar um atributo diretamente na classe, mantendo o encapsulamento – objeto.atr=1 (ex.)



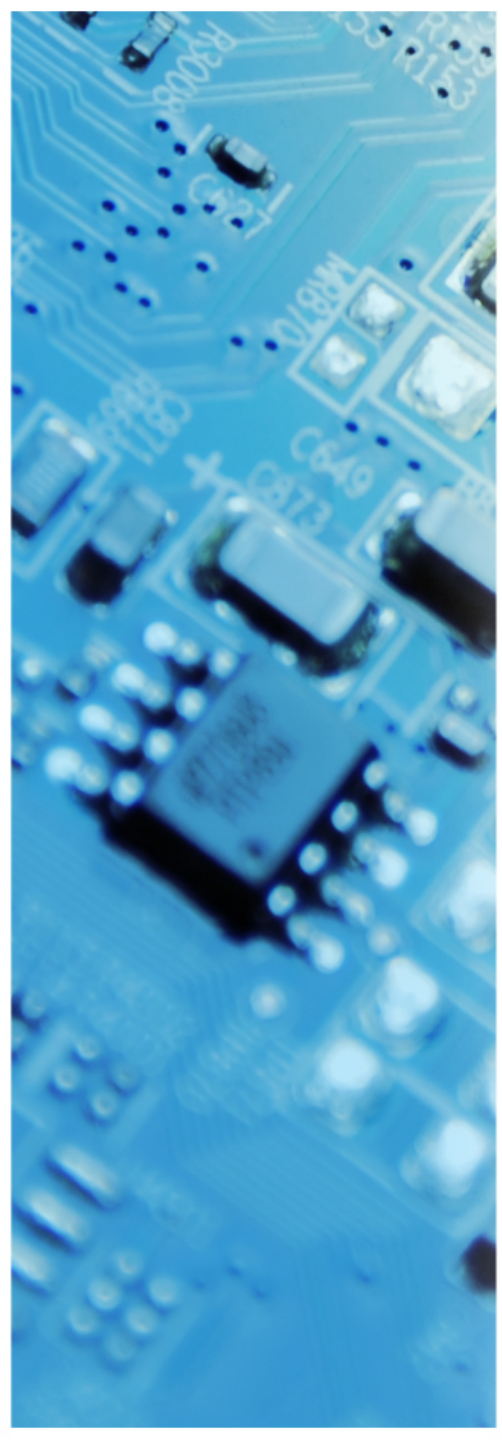
# Herança

- A herança ocorre pela especificação do nome da classe pai após o nome da classe que está sendo criada – ex. Classe(Classepai).
- São herdadas todas as propriedades e métodos da classe superior.
- É possível chamar métodos da classe pai usando super().
- O uso de super().__init__() chama o construtor da classe superior.



# Desafio

- Criar uma classe veículo para manter dados de diferentes veículos.
- Cada veículo terá placa, modelo, marca e o valor mercado.
- A classe deverá implementar todos os métodos necessários para resolver as funcionalidades abaixo.
- O programa principal deve manter uma lista de veículos.
- Um menu deve permitir ao usuário inserir, retirar ou alterar os dados de determinado veículo.
- Deve haver uma opção para que o sistema mostre os dados do veículo com maior valor de mercado



# Desafio

- O sistema deve permitir também a execução de uma rotina para aumentar o reduzir o valor de mercado de todos os automóveis.
- Neste caso será solicitado o percentual e a opção (aumento/redução) e o sistema irá executar a operação em todos os veículos.
- Para isso a classe veículo deve implementar dois métodos, um para reduzir e outro para aumentar o valor de mercado.
- Estes métodos recebem o percentual para atualização e executam a operação no referido objeto.