Programação Orientada a Objetos com Python

tudo é um objeto.

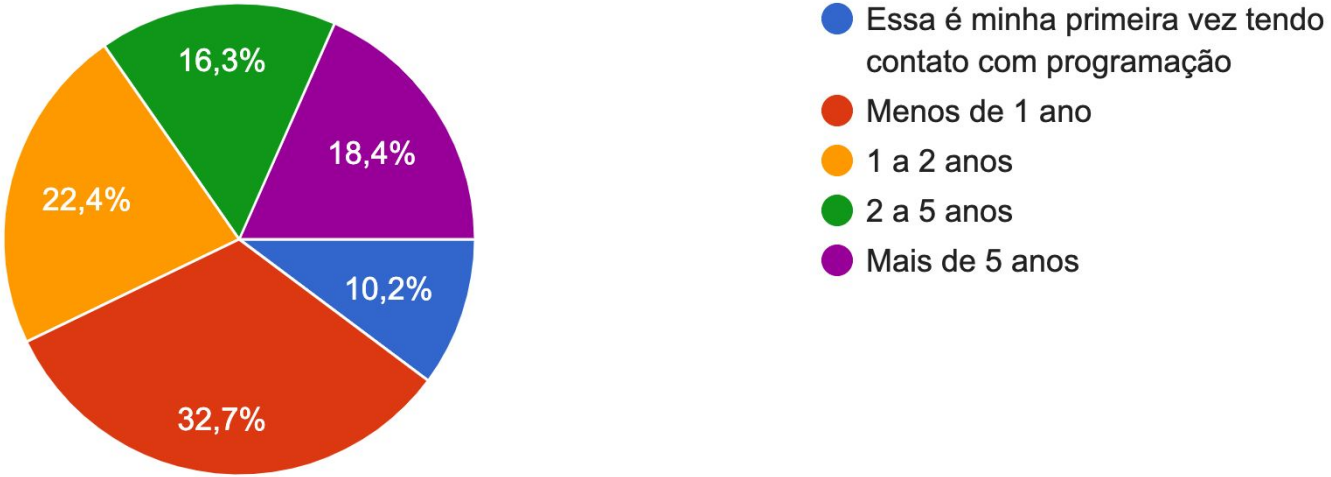
Material de apoio

https://github.com/CodarMe/curso-python/tree/main/lives/poo_python/exemplos

Quem é você?

Há quanto tempo você estuda programação?

49 respostas



sli.do/997378

Quem é você?

POO

Fundamentos

- Classes e objetos
- Abstração
- Encapsulamento
- Herança
- Polimorfismo
- Associação
  - Agregação
  - Composição
  - Dependência

POOP

Aplicando no Python

- Getters/setters
- Properties
- Classes abstratas
- Duck typing
- Métodos especiais

Classes e objetos

O que é um objeto?

Objetos na vida real

Modelagem

- Escola, turmas e alunos
- Eventos e participantes
- Cachorro, gato, …

Objetos na programação

Python e Web

- Dicionários
- Tuplas
- Requisição HTTP
- Arquivo de texto

Classe = Tipo

Mentiram para mim
sobre tipos primitivos.

Exemplos com tipos "primitivos"

Criando nossa classe: *Carro*

Classe, instância, atributos, métodos e memória.

Código

Classe Carro

- atributos: modelo, ano, fabricante, posição, velocidade, combustível
- método: acelerar

Abstração

the separation of an idea from time & place

Abstração

- Não sabemos "como", apenas "o que"
- Facilita a leitura de programas
- **Níveis de abstração**
  - Exemplos?

Encapsulamento

Encapsulamento

- Controlar o acesso à atributos e métodos
- Carro: velocidade, combustível, posição
  - acelerar
  - frear
  - abastecer

Encapsulamento: @property e @foo.setter

- Getter e Setter (get_foo, set_foo)
- Atribuições com lógica
  - @property
  - @attr.setter
- Atributos "privados"
  - _v
  - _ _v
  - dir(obj) e obj.__dict__

Exemplo

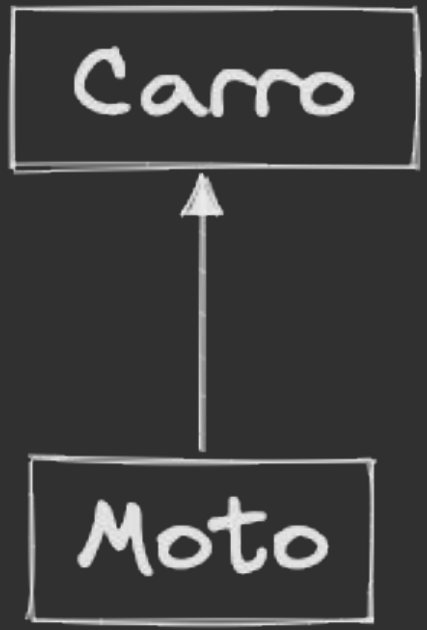
Getter e Setter

```python
@property
def parado(self):
    return self.velocidade == 0

@property
def ano(self):
    return self._ano

@ano.setter
def ano(self, valor):
    if valor > date.today().year:
        return
    self._ano = valor
```

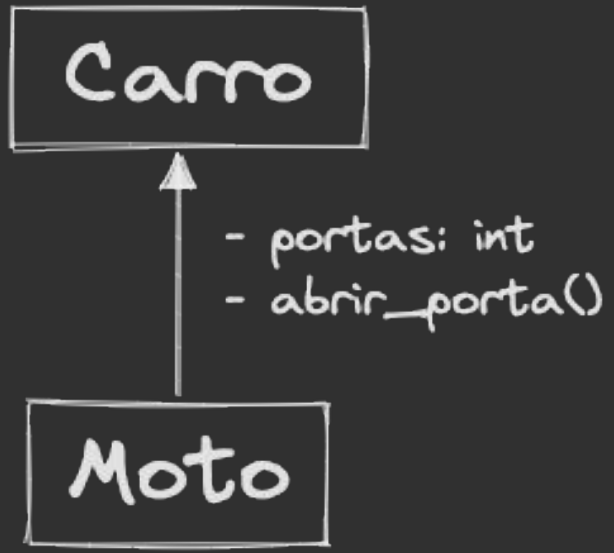
Herança

Herança

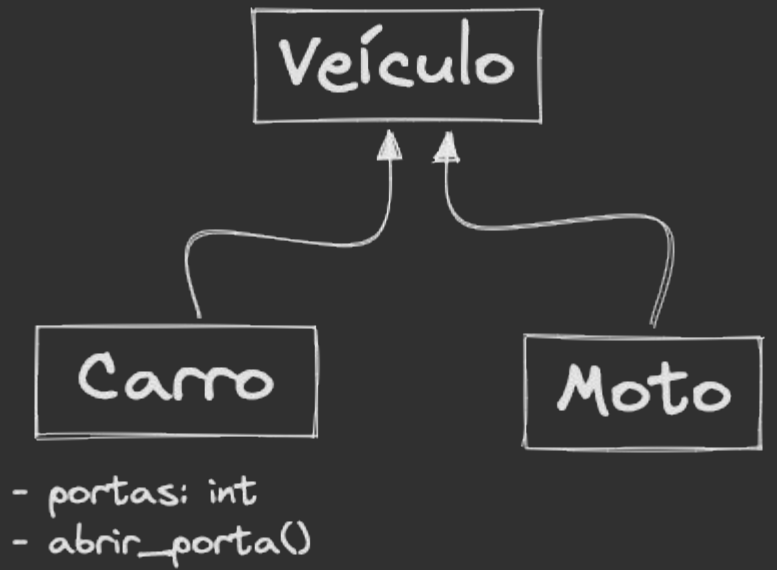
- Reaproveitar código
- Modelar relações



Herança



Herança

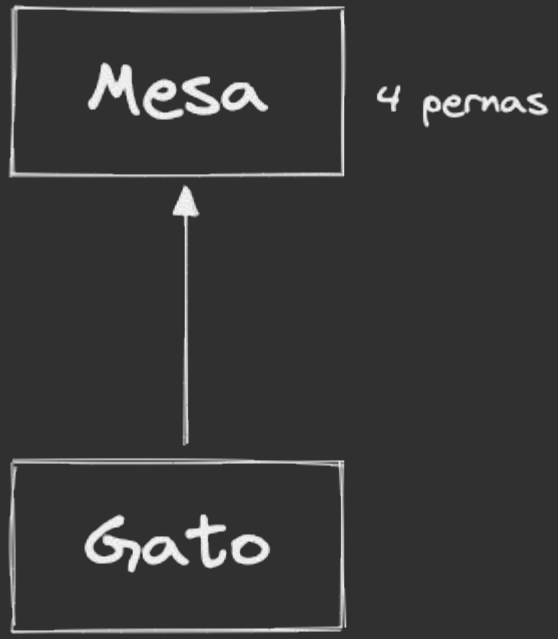


early abstraction

As abstrações devem *emergir* do seu código, e não o contrário.

early abstraction

"Se os dois têm 4 pernas…"



Polimorfismo

Exemplos

Polimorfismo

Código

- Mover arquivos (SO)
- Executar programas
- *onClick* / *onHover*
- …

Exemplos

Polimorfismo

Código

```python
class Corrida:
    def __init__(self, veiculos):
        self.veiculos = veiculos

    def iniciar(self):
        for veiculo in self.veiculos:
            veiculo.acelerar(3)
```

E se nem todo veículo tiver o método *acelerar?*

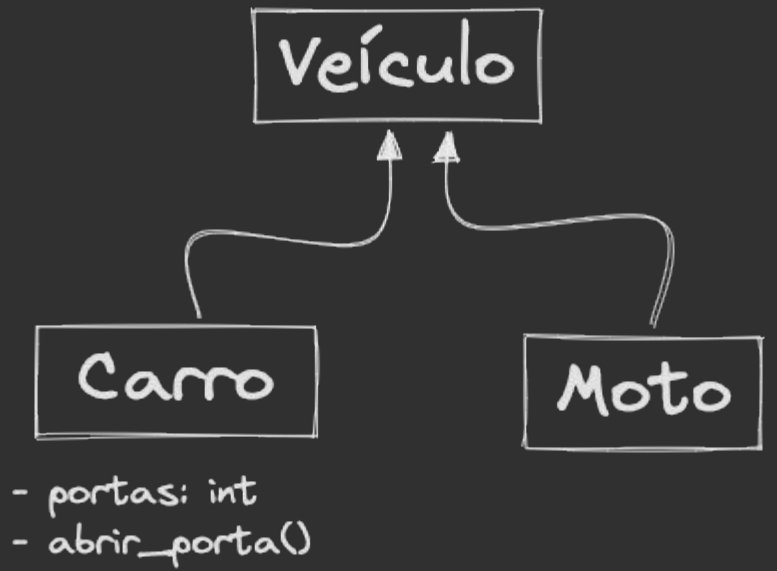
Checagem de tipo (código)

```
class Corrida:
    def __init__(self, veiculos):
        self.veiculos = veiculos

    def iniciar(self):
        for veiculo in self.veiculos:
            if not isinstance(veiculo, Veiculo):
                raise TypeError(f"{veiculo} não é uma instância de Veiculo")
            veiculo.acelerar(3)
```

Classes abstratas

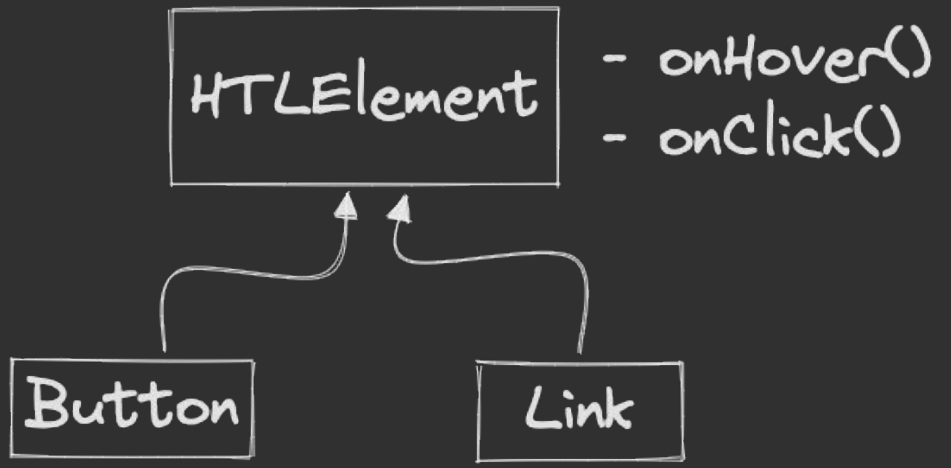
Classes abstratas

@abstractmethod



Classes abstratas

@abstractmethod



Tipagem

Tipagem

Python, JavaScript, TypeScript

- Tipagem dinâmica vs estática
- Tipagem fraca vs forte
- Executar no shell do Python
  - 4 + '7';
  - 4 * '7';
  - 2 + true;
- Executar no console do Chrome (JavaScript)

Tipagem (type hint)

```python
from typing import Iterable
from veiculos import Veiculo


def soma(a: int, b: int) -> int:
    return a + b

def divisao(a: int, b: int) -> float | int:
    return a / b

class Corrida:
    def __init__(self, veiculos: Iterable[Veiculo]) -> None:
        self.veiculos = veiculos

    def iniciar(self) -> None:
        for veiculo in self.veiculos:
            veiculo.acelerar(3)
```

Duck typing 

Duck typing 🦆

"If it walks like a duck and it quacks like a duck, then it must be a duck."

- Métodos e propriedades vs Tipos
- Mais comum em linguagens com tipagem dinâmica

Duck typing

len(obj)

```
In [41]: len([1, 2, 3])
Out[41]: 3

In [42]: len("abc")
Out[42]: 3

In [43]: len(123)
---------------------------------------------
TypeError                                 Tra
<ipython-input-43-a95e54e61f62> in <module>
----> 1 len(123)

TypeError: object of type 'int' has no len()
```

Exemplo duck typing

corrida_duck_typing.py

Exemplos da linguagem

- Sequência: __getitem__ (exemplo)
- Operações: __add__
- Iteráveis: __iter__
- Tamanho: __len__

Métodos mágicos (__*dunder methods*__)

- Implementando um Vetor 2D (código)
  - soma
  - subtração
  - multiplicação escalar
  - igualdade e identidade
  - bool?
  - __repr__
- Python Fluente

Dataclasses

Dataclasses

```python
class Carro:
    def __init__(self, modelo, ano, fabricante):
        self.modelo = modelo
        self._ano = ano
        self.fabricante = fabricante
        self.posicao = 0
        self.velocidade = 0
        self.combustivel = 0
```

```python
from dataclasses import dataclass

@dataclass
class Carro:
    modelo: str
    ano: int
    fabricante: str
    posicao = 0
    velocidade = 0
    combustivel = 0
```

Dataclasses

- __repr__
- Imutabilidade

Relacionamento entre entidades (UML)

Agregação

Agregação

- Corrida possui veículos.
- Turma possui alunos.
- E se a corrida for cancelada?
- E se a turma for cancelada?

Composição

Composição

- As partes não existem sem o todo.
  - Livro e Páginas.
  - Usuário e Alunos.
  - Eventos e Participantes.
  - Eventos e Organizadores.

Dependência

Dependência

Nem sempre é "óbvia" (associação)

```python
@dataclass
class Turma:
    pass

@dataclass
class Aluno:
    pass

@dataclass
class Curso:
    turmas: Iterable[Turma]

    def matricular(self, aluno: Aluno):
        self.turmas.append(aluno)
```

Exercício

Exemplo: Curso de Python

- Usuários
- Cursos
- Turmas
- Alunos
- Aulas
- Conteúdo

Exemplo: Curso de Python

- Esqueleto do código com enunciado