



Geek University

Evolua seu lado geek!

www.geekuniversity.com.br



Principais Componentes da Programação Orientada a Objetos





Principais Componentes da Programação Orientada a Objetos

Depois de termos revisado o básico da orientação a objetos, chegou a hora de explorarmos seus principais componentes:

- Encapsulamento;
- Polimorfismo;
- Herança;
- Abstração;
- Composição;



Principais Componentes da Programação Orientada a Objetos

Encapsulamento:

Quando falamos de encapsulamento queremos dizer que:

- O comportamento de um objeto permanece oculto para o mundo externo, ou os objetos mantêm suas informações de estado (atributos) como privadas;
- Os clientes não podem alterar o estado interno dos objetos atuando diretamente em seus atributos; em vez disso estas alterações devem ser realizadas através dos métodos;
- Em Python, o conceito de encapsulamento não é implícito pois não existem palavras reservadas como **public**, **private** e **protected** necessárias para tratar este conceito. Entretanto, por convenção “tornamos” a acessibilidade privada usando o prefixo `__` (dunder/ duplo underline) no nome do atributo ou método.



Principais Componentes da Programação Orientada a Objetos

Encapsulamento:

Quando falamos de encapsulamento queremos dizer que:

- O comportamento de um objeto permanece oculto para o mundo externo, ou os objetos mantêm suas informações de estado (atributos) como privadas;
- Os clientes não podem alterar o estado interno dos objetos atuando diretamente em seus atributos; em vez disso estas alterações devem ser realizadas através dos métodos;
- Em Python, o conceito de encapsulamento não é implícito pois não existem palavras reservadas como **public**, **private** e **protected** necessarias para tratar este conceito. Entretanto, por convenção “tornamos” a acessibilidade privada usando o prefixo `__` (dunder/ duplo underline) no nome do atributo ou método.

```
class Pessoa:
    def __init__(self, nome):
        self.__nome = nome
        self.__nascimento = datetime.now()
```



Principais Componentes da Programação Orientada a Objetos

Polimorfismo:

O polimorfismo pode ser de dois tipos:

- Um objeto oferece diferentes implementações de um método de acordo com os parâmetros de entrada;
- A mesma interface (implementação) pode ser usada por objetos de tipos diferentes;



Principais Componentes da Programação Orientada a Objetos

Polimorfismo:

O polimorfismo pode ser de dois tipos:

- Um objeto oferece diferentes implementações de um método de acordo com os parâmetros de entrada;
- A mesma interface (implementação) pode ser usada por objetos de tipos diferentes;

```
class Curso:
    def __init__(self, nome='Curso Padrão', carga_horaria=45):
        self.__nome = nome
        self.__carga_horaria = carga_horaria

curso1 = Curso()
curso2 = Curso(nome='Padrões de Projetos em Python')
curso3 = Curso(nome='Orquestração de Containers com Kubernetes', carga_horaria=23)

print(curso1.__dict__)
print(curso2.__dict__)
print(curso3.__dict__)
```

```
nome = 'Geek University'
tupla = (1, 2, 3, 4, 5)
lista = [1, 2, 3, 4, 5]

print(nome[:4], tupla[:4], lista[:4])
```



Principais Componentes da Programação Orientada a Objetos

Herança:

Usamos herança para darmos mais flexibilidade às nossas classes e o poder de reutilização, além de:

- A herança indica que uma classe deriva (estende) sua funcionalidade da classe-pai;
- A herança permite que reutilizemos características e funcionalidades definidas na classe pai;
- A herança cria hierarquias por meio do relacionamento entre objetos de diferentes classes.

Em Python, diferente de outras linguagens, há suporte para herança múltipla, ou seja, podemos herdar de várias classes.



Principais Componentes da Programação Orientada a Objetos

Herança:

Usamos herança para darmos mais flexibilidade às nossas classes e o poder de reutilização, além de:

- A herança indica que uma classe deriva (estende) sua funcionalidade da classe-pai;
- A herança permite que reutilizemos características e funcionalidades definidas na classe pai;
- A herança cria hierarquias por meio do relacionamento entre objetos de diferentes classes.

Em Python, diferente de outras linguagens, há suporte para herança múltipla, ou seja, podemos herdar de várias classes.

```
class Pessoa:
    def __init__(self, nome):
        self.__nome = nome

    def andar(self):
        print('Estou andando...')
```

```
class Aluno(Pessoa):
    def __init__(self, nome, matricula):
        super().__init__(nome)
        self.__matricula = matricula

felicity = Aluno('Felicity', 12345)

felicity.andar()
```



Principais Componentes da Programação Orientada a Objetos

Abstração:

Ao tratarmos de abstração estamos falando em simplificação...

- A abstração oferece uma interface (implementação) simples aos clientes, por meio da qual eles podem interagir com os elementos do programa e utilizar os métodos que foram definidos.
- Abstraindo (simplificando) a complexidade do programa os clientes não precisam conhecer as implementações internas, bastando apenas executá-las.



Principais Componentes da Programação Orientada a Objetos

Abstração:

Ao tratarmos de abstração estamos falando em simplificação...

- A abstração oferece uma interface (implementação) simples aos clientes, por meio da qual eles podem interagir com os elementos do programa e utilizar os métodos que foram definidos.
- Abstraindo (simplificando) a complexidade do programa os clientes não precisam conhecer as implementações internas, bastando apenas executá-las.

```
def gerar_fibonacci(qtd):  
    if qtd <= 0:  
        print('A quantidade deve ser maior que 0')  
    else:  
        print(f'A sequência Fibonacci para {qtd} termo(s) é: ')  
        contador = 0  
        aux1, aux2 = 0, 1  
        while contador < qtd:  
            print(aux1)  
            proximo = aux1 + aux2  
            aux1 = aux2  
            aux2 = proximo  
            contador += 1  
  
gerar_fibonacci(7)
```



Principais Componentes da Programação Orientada a Objetos

Composição:

- A composição nos ajuda a combinar objetos ou classes em estruturas de dados ou implementações mais complexas;
- Na composição, um objeto é usado para chamar/executar métodos de outras classes, disponibilizando assim, as funcionalidades básicas sem o uso de herança;



Principais Componentes da Programação Orientada a Objetos

Composição:

- A composição nos ajuda a combinar objetos ou classes em estruturas de dados ou implementações mais complexas;
- Na composição, um objeto é usado para chamar/executar métodos de outras classes, disponibilizando assim, as funcionalidades básicas sem o uso de herança;

```
class Motor:
    def ligar(self):
        print('Motor ligado...')

class Pneu:
    def enxer(self):
        print('Pneu cheio...')
```

```
class Carro:
    def __init__(self, modelo):
        self.__modelo = modelo
        self.__motor = Motor()
        self.__pneu1 = Pneu()
        self.__pneu2 = Pneu()
        self.__pneu3 = Pneu()
        self.__pneu4 = Pneu()

fusca = Carro('Fusca')
fusca._Carro__motor.ligar()
```



Geek University

Evolua seu lado geek!

www.geekuniversity.com.br