Programação orientada a objetos (POO)

Guilherme Arthur de Carvalho

Analista de sistemas

@decarvalhogui

Objetivo Geral

Conhecer o paradigma de programação orientada a objetos.

Pré-requisitos

- Conhecimento básico em Python.

Percurso

Etapa 1 **O que é POO**

Etapa 1

O que é POO

Paradigmas de programação

Um paradigma de programação é um estilo e programação. Não é uma linguagem (Python, Java, C, etc), e sim a forma como você soluciona os problemas através do código.

Exemplo

Problema: Beber água

Solução 1: Usar um copo para beber água.

Solução 2: Usar uma garrafa para beber água.

Alguns paradigmas

- Imperativo ou procedural
- Funcional
- Orientado a eventos

Programação orientada a objetos

O paradigma de programação orientada a objetos estrutura o código abstraindo problemas em objetos do mundo real, facilitando o entendimento do código e tornando-o mais modular e extensível. Os dois conceitos chaves para aprender POO são: **classes e objetos.**

Percurso

~~Etapa 1~~ ~~O que é POO~~

Links Úteis

- https://github.com/digitalinnovationone/trilha-python-dio

Dúvidas?

> Fórum/Artigos

> Comunidade Online (Discord)

Classes e objetos

Guilherme Arthur de Carvalho

Analista de sistemas

@decarvalhogui

Objetivo Geral

Aprender a utilizar classes e objetos com Python.

Pré-requisitos

- Conhecimento básico em Python.

Percurso

Etapa 1 **Conceito de classes e objetos**

Etapa 2 Primeiro programa com POO

Etapa 1

Conceito de classes e objetos

Classes e objetos?

Uma classe define as características e comportamentos de um objeto, porém não conseguimos usá-las diretamente. Já os objetos podemos usá-los e eles possuem as características e comportamentos que foram definidos nas classes.

Classe

```
class Cachorro:
    def __init__(self, nome, cor, acordado=True):
        self.nome = nome
        self.cor = cor
        self.acordado = acordado

    def latir(self):
        print("Auau")

    def dormir(self):
        self.acordado = False
        print("Zzzzz...")
```

Objeto

```
cao_1 = Cachorro("chappie", "amarelo", False)
cao_2 = Cachorro("Aladim", "branco e preto")

cao_1.latir()

print(cao_2.acordado)
cao_2.dormir()
print(cao_2.acordado)
```

Nosso primeiro programa POO

João tem uma bicicletaria e gostaria de registrar as vendas de suas bicicletas. Crie um programa onde João informe: **cor, modelo, ano e valor** da bicicleta vendida. Uma bicicleta pode: **buzinar, parar e correr**. Adicione esses comportamentos!

Percurso

~~Etapa 1~~ ~~O que é POO~~

~~Etapa 2~~ ~~Classes e objetos~~

Links Úteis

- https://github.com/digitalinnovationone/trilha-python-dio

Dúvidas?

> Fórum/Artigos

> Comunidade Online (Discord)

Construtores e destrutores

Guilherme Arthur de Carvalho

Analista de sistemas

@decarvalhogui

Objetivo Geral

Entender o conceito de construtor e destrutor.

Pré-requisitos

- Conhecimento básico em Python.

Percurso

Etapa 1 **Conhecendo os métodos __init__ e __del__**

Etapa 1

Conhecendo os métodos __init__ e __del__

Método construtor

O método construtor sempre é executado quando uma nova instância da classe é criada. Nesse método inicializamos o estado do nosso objeto. Para declarar o método construtor da classe, criamos um método com o nome **__init__**.

__init__

```
class Cachorro:
    def __init__(self, nome, cor, acordado=True):
        self.nome = nome
        self.cor = cor
        self.acordado = acordado
```

Método destrutor

O método destrutor sempre é executado quando uma instância (objeto) é destruída. Destrutores em Python não são tão necessários quanto em C++ porque o Pyton tem um coletor de lixo que lida com o gerenciamento de memória automaticamente. Para declarar o método destrutor da classe, criamos um método com o nome **__del__**.

__del__

```
class Cachorro:
    def __del__(self):
        print("Destruindo a instância")

c = Cachorro()
del c
```

Percurso

~~Etapa 1~~ ~~Conhecendo os métodos __init__ e __del__~~

Links Úteis

- https://github.com/digitalinnovationone/trilha-python-dio

Dúvidas?

> Fórum/Artigos

> Comunidade Online (Discord)

Herança

Guilherme Arthur de Carvalho

Analista de sistemas

@decarvalhogui

Objetivo Geral

Aprender o que é herança em POO e como podemos utilizá-la em Python.

Pré-requisitos

- Conhecimento básico em Python.

Percurso

Etapa 1	**Herança em POO**
Etapa 2	Herança simples e herança múltipla

Etapa 1

Herança em POO

O que é herança?

Em programação herança é a capacidade de uma classe filha derivar ou herdar as características e comportamentos da classe pai (base).

Benefícios da herança

- Representa bem os relacionamentos do mundo real.
- *Fornece reutilização de código, não precisamos escrever o mesmo código repetidamente. Além disso, permite adicionar mais recursos a uma classe sem modificá-la.
- *É de natureza transitiva, o que significa que, se a classe B herdar da classe A, todas as subclasses de B herdarão automaticamente da classe A.

Sintaxe da herança

```
class A:
    pass

class B(A):
    pass
```

Percurso

~~Etapa 1~~ ~~Herança em POO~~

Etapa 2 **Herança simples e herança múltipla**

Etapa 2

Herança simples e herança múltipla

Herança simples

Quando uma classe filha herda apenas uma classe pai, ela é chamada de herança simples.

Exemplo

```
class A:
    pass

class B(A):
    pass
```

Herança múltipla

Quando uma classe filha herda de várias classes pai, ela é chamada de herança múltipla.

Exemplo

```
class A:
    pass

class B:
    pass

class C(A, B):
    pass
```

Percurso

~~Etapa 1~~ ~~Herança em POO~~

~~Etapa 2~~ ~~Herança simples e herança múltipla~~

Links Úteis

- https://github.com/digitalinnovationone/trilha-python-dio

Dúvidas?

> Fórum/Artigos

> Comunidade Online (Discord)

Encapsulamento

Guilherme Arthur de Carvalho

Analista de sistemas

@decarvalhogui

Objetivo Geral

Entender o conceito de encapsulamento e como podemos aplicá-lo utilizando Python.

Pré-requisitos

- Conhecimento básico em Python.

Percurso

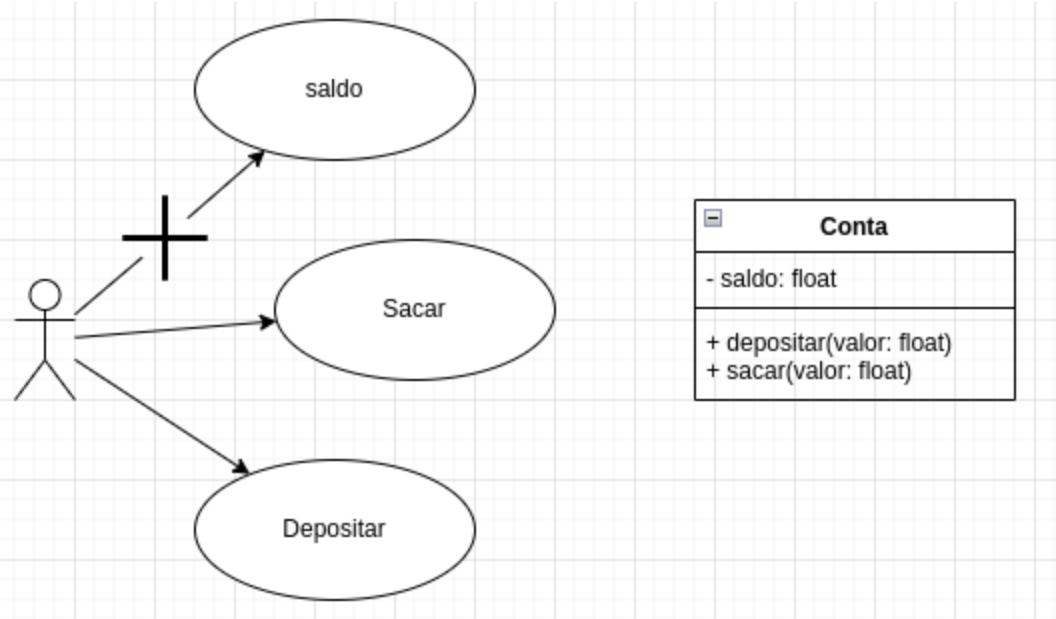
Etapa 1	**O que é encapsulamento?**
Etapa 2	Recursos públicos e privados
Etapa 3	Properties

Etapa 1

O que é encapsulamento?

Proteção de acesso

O encapsulamento é um dos conceitos fundamentais em programação orientada a objetos. Ele descreve a ideia de agrupar dados e os métodos que manipulam esses dados em uma unidade. Isso impõe restrições ao acesso direto a variáveis e métodos e pode evitar a modificação acidental de dados. Para evitar alterações acidentais, a variável de um objeto só pode ser alterada pelo método desse objeto.


saldo
Sacar
Depositar
Conta
- saldo: float
+ depositar(valor: float)
+ sacar(valor: float)

Percurso

~~Etapa 1~~	~~O que é encapsulamento?~~
Etapa 2	**Recursos públicos e privados**
Etapa 3	Properties

Etapa 2

Recursos públicos e privados

Modificadores de acesso

Em linguagens como Java e C++, existem palavras reservadas para definir o nível de acesso aos atributos e métodos da classe. Em Python não temos palavras reservadas, porém usamos convenções no nome do recurso, para definir se a variável é pública ou privada.

Definição

- Público: Pode ser acessado de fora da classe.
- Privado: Só pode ser acessado pela classe.

Público/Privado

Todos os recursos são públicos, a menos que o nome inicie com underline. Ou seja, o interpretador Python não irá garantir a proteção do recurso, mas por ser uma convenção amplamente adotada na comunidade, quando encontramos uma variável e/ou método com nome iniciado por underline, sabemos que não deveríamos manipular o seu valor diretamente, ou invocar o método fora do escopo da classe.

Exemplo

```
class Conta:
    def __init__(self, saldo=0):
        self._saldo = saldo

    def depositar(self,valor):
        pass

    def sacar(self,valor):
        pass
```

Percurso

~~Etapa 1~~	~~O que é encapsulamento?~~
~~Etapa 2~~	~~Recursos públicos e privados~~
Etapa 3	**Properties**

Etapa 3

Properties

Para que servem?

Com o property() do Python, você pode criar atributos gerenciados em suas classes. Você pode usar atributos gerenciados, também conhecidos como propriedades, quando precisar modificar sua implementação interna sem alterar a API pública da classe.

Exemplo

```
class Foo:
    def __init__(self, x=None):
        self._x = x

    @property
    def x(self):
        return self._x or 0

    @x.setter
    def x(self, value):
        _x = self._x or 0
        _value = value or 0
        self._x = _x + _value

    @x.deleter
    def x(self):
        self._x = -1

foo = Foo(10)
print(foo.x)
foo.x = 10
print(foo.x)
del foo.x
print(foo.x)
```

Percurso

~~Etapa 1~~ ~~O que é encapsulamento?~~

~~Etapa 2~~ ~~Recursos públicos e privados~~

~~Etapa 3~~ ~~Properties~~

Links Úteis

- https://github.com/digitalinnovationone/trilha-python-dio

Dúvidas?

> Fórum/Artigos

> Comunidade Online (Discord)

Polimorfismo

Guilherme Arthur de Carvalho

Analista de sistemas

@decarvalhogui

Objetivo Geral

Aprender a criar classes polimórficas com Python.

Pré-requisitos

- Conhecimento básico em Python.

Percurso

Etapa 1 **O que é polimorfismo?**

Etapa 2 Polimorfismo com herança

Etapa 1

O que é polimorfismo?

Muitas formas!

A palavra polimorfismo significa ter muitas formas. Na programação, polimorfismo significa o mesmo nome de função (mas assinaturas diferentes) sendo usado para tipos diferentes.

Exemplo

```
len("python")
len([10, 20, 30])
```

Percurso

~~Etapa 1~~ ~~O que é polimorfismo?~~

Etapa 2 **Polimorfismo com herança**

Etapa 2

Polimorfismo com herança

Mesmo método com comportamento diferente

Na herança, a classe filha herda os métodos da classe pai. No entanto, é possível modificar um método em uma classe filha herdada da classe pai. Isso é particularmente útil nos casos em que o método herdado da classe pai não se encaixa perfeitamente na classe filha.

Exemplo

```
class Passaro:
    def voar(self): pass

class Pardal(Passaro):
    def voar(self):
        print("Pardal voa")

class Avestruz(Passaro):
    def voar(self):
        print("Avestruz não voa")

def plano_de_voo(passaro):
    passaro.voar()

plano_de_voo(Pardal())
plano_de_voo(Avestruz())
```

Percurso

~~Etapa 1~~ ~~O que é polimorfismo?~~

~~Etapa 2~~ ~~Polimorfismo com herança~~

Links Úteis

- https://github.com/digitalinnovationone/trilha-python-dio

Dúvidas?

> Fórum/Artigos

> Comunidade Online (Discord)

Variáveis de classe e variáveis de instância

Guilherme Arthur de Carvalho

Analista de sistemas

@decarvalhogui

Objetivo Geral

Entender as diferenças entre variáveis de classe e variáveis de instância.

Pré-requisitos

- Conhecimento básico em Python.

Percurso

Etapa 1 **O que são e como utilizamos**

Etapa 1

O que são e quando utilizamos

Atributos do objeto

Todos os objetos nascem com o mesmo número de atributos de classe e de instância. Atributos de instância são diferentes para cada objeto (cada objeto tem uma cópia), já os atributos de classe são compartilhados entre os objetos.

Exemplo

```
class Estudante:
    escola = "DIO"

    def __init__(self, nome, numero):
        self.nome = nome
        self.numero = numero

    def __str__(self):
        return f"{self.nome} ({self.numero}) - {self.escola}"

gui = Estudante("Guilherme", 56451)
gi = Estudante("Giovanna", 17323)
```

Percurso

~~Etapa 1~~ ~~O que são e como utilizamos~~

Links Úteis

- https://github.com/digitalinnovationone/trilha-python-dio

Dúvidas?

> Fórum/Artigos

> Comunidade Online (Discord)

Métodos de classe e métodos estáticos

Guilherme Arthur de Carvalho

Analista de sistemas

@decarvalhogui

Objetivo Geral

Entender as diferenças entre métodos de classe e métodos estáticos.

Pré-requisitos

- Conhecimento básico em Python.

Percurso

Etapa 1 **O que são e como utilizamos**

Etapa 1

O que são e quando utilizamos

Métodos de classe

Métodos de classe estão ligados à classe e não ao objeto. Eles têm acesso ao estado da classe, pois recebem um parâmetro que aponta para a classe e não para a instância do objeto.

Métodos estáticos

Um método estático não recebe um primeiro argumento explícito. Ele também é um método vinculado à classe e não ao objeto da classe. Este método não pode acessar ou modificar o estado da classe. Ele está presente em uma classe porque faz sentido que o método esteja presente na classe.

Métodos de classe x métodos estáticos

- Um método de classe recebe um primeiro parâmetro que aponta para a classe, enquanto um método estático não.
- Um método de classe pode acessar ou modificar o estado da classe enquanto um método estático não pode acessá-lo ou modificá-lo.

Quanto utilizar método de classe ou estático

- Geralmente usamos o método de classe para criar métodos de fábrica.
- Geralmente usamos métodos estáticos para criar funções utilitárias.

Hands On!

“Falar é fácil. Mostre-me o código!”

Linus Torvalds

Percurso

~~Etapa 1~~ ~~O que são e como utilizamos~~

Links Úteis

- https://github.com/digitalinnovationone/trilha-python-dio

Dúvidas?

> Fórum/Artigos

> Comunidade Online (Discord)

Classes abstratas

Guilherme Arthur de Carvalho

Analista de sistemas

@decarvalhogui

Objetivo Geral

Aprender o conceito de contrato e como podemos utilizar classes abstratas em Python para implementá-los.

Pré-requisitos

- Conhecimento básico em Python.

Percurso

Etapa 1 **O que são interfaces?**

Etapa 2 Criando classes abstratas com o módulo abc

Etapa 1

O que são interfaces?

Importante!

Interfaces definem o que uma classe deve fazer e não como.

Python tem interface?

O conceito de interface é definir um contrato, onde são declarados os métodos (o que deve ser feito) e suas respectivas assinaturas. Em Python utilizamos classes abstratas para criar contratos. Classes abstratas não podem ser instanciadas.

Percurso

~~Etapa 1~~ ~~O que são interfaces?~~

Etapa 2 **Criando classes abstratas com o módulo abc**

Etapa 2

Criando classes abstratas com o módulo abc

ABC

Por padrão, o Python não fornece classes abstratas. O Python vem com um módulo que fornece a base para definir as classes abstratas, e o nome do módulo é ABC. O ABC funciona decorando métodos da classe base como abstratos e, em seguida, registrando classes concretas como implementações da base abstrata. Um método se torna abstrato quando decorado com **@abstractmethod**.

Hands On!

“Falar é fácil. Mostre-me o código!”

Linus Torvalds

Percurso

~~Etapa 1~~ ~~O que é contratos?~~

~~Etapa 2~~ ~~Criando classes abstratas com o módulo abc~~

Links Úteis

- https://github.com/digitalinnovationone/trilha-python-dio
- https://docs.python.org/pt-br/3/library/abc.html

Dúvidas?

> Fórum/Artigos

> Comunidade Online (Discord)