python

# Orientação a Objetos

PUG-MA

Helton Alves
helton99@hotmail.com
github: heltonalves99
whatsApp: (98) 98253-2197


https://github.com/nuveo/IWantToWorkAtNuveo

# O Zen do Python

**Bonito é melhor que feio.**
**Explícito é melhor que implícito.**
**Simples é melhor que complexo.**
**Complexo é melhor que complicado.**
**Plano é melhor que aninhado.**
**Esparso é melhor que denso.**
**Legibilidade conta.**
**Casos especiais não são especiais o bastante para se quebrar as regras.**
**Embora a simplicidade supere o purismo.**
**Erros nunca deveriam passar silenciosamente.**
**A menos que explicitamente silenciados.**
**Ao encarar a ambiguidade, recuse a tentação de adivinhar.**
**Deveria haver uma – e preferencialmente apenas uma – maneira óbvia de se fazer isto.**
**Embora aquela maneira possa não ser óbvia à primeira vista se você não for holandês.**
**Agora é melhor que nunca.**
**Embora nunca, seja muitas vezes melhor que pra já.**
**Se a implementação é difícil de explicar, é uma má idéia.**
**Se a implementação é fácil de explicar, pode ser uma boa idéia.**
**Namespaces são uma idéia estupenda – vamos fazer mais deles!**

# Orientação a Objetos (OO)

# Orientação a Objetos (OO)

Herança multipla, como C++
Sobrecarga de operadores, como C++
Não obriga a criar classes. Como C++
Tipagem dinâmica, como SmallTalk e Ruby

# Orientação a Objetos (OO)

O termo orientação a objetos significa organizar o mundo real como uma coleção de objetos que incorporam estrutura de dados e um conjunto de operações que manipulam estes dados.

# Orientação a Objetos (OO)

## Classes

Classes são o coração da programação orientada a objetos que tem como principio criar modelos que representem objetos do mundo real.

# Orientação a Objetos (OO)

## Classes

Classes são o coração da programação orientada a objetos que tem como principio criar modelos que representem objetos do mundo real.

# Orientação a Objetos (OO)

## Objetos

São instâncias da classe, ou seja, uma criação concreta a partir da classe.

# Orientação a Objetos (OO)

## Objetos

São instâncias da classe, ou seja, uma criação concreta a partir da classe.

# Orientação a Objetos (OO)

## Herança (multipla)

permite que uma classe herde atributos e métodos de uma classe pai.

# Orientação a Objetos (OO)

## Herança (multipla)

permite que uma classe herde atributos e métodos de uma classe pai.

# Orientação a Objetos (OO)

## Class Super()

O *super()* é utilizado entre heranças de classes, ele nos proporciona extender/subscrever métodos de uma super

# Orientação a Objetos (OO)

## Class Super()

O *super()* é utilizado entre heranças de classes, ele nos proporciona extender/subscrever métodos de uma super

# Orientação a Objetos (OO)

## Interfaces

Uma interface define métodos que devem ser desenvolvidos (contrato a ser seguido) para quem quiser implementá-la.

# Orientação a Objetos (OO)

## Interfaces

Com herança múltipla de classes e classes abstratas, você terá um comportamento parecido com interfaces.

# Orientação a Objetos (OO)

## Encapsulamento

Faz com que detalhes internos do funcionamento dos métodos de uma classe permaneçam ocultos para os objetos.

# Orientação a Objetos (OO)

## Encapsulamento

Faz com que detalhes internos do funcionamento dos métodos de uma classe permaneçam ocultos para os objetos.

# Orientação a Objetos (OO)

## Polimorfismo

Um objeto oferece diferentes implementações do método de acordo com os parâmetros de entrada.

A mesma interface pode ser usada por objetos de tipos diferentes.

# Orientação a Objetos (OO)

## Polimorfismo

Um objeto oferece diferentes implementações do método de acordo com os parâmetros de entrada.

A mesma interface pode ser usada por objetos de tipos diferentes.

# Orientação a Objetos (OO)

Características:

- Atributos dinámicos
- self explicito na declaração e implícito na chamada de métodos.
- Indentação do código

# Obrigado!

Vamos beber! o/